University of the Philippines Manila
College of Arts and Sciences
Department of Physical Sciences and Mathematics

# Phenogram Generator using Frequent Structure Mining over Metabolic Pathways (PhenTor)

A special problem in partial fulfillment
of the requirements for the degree of
**Bachelor of Science in Computer Science**

Submitted by:

Lejun Christian L. Osorio

June 9, 2015

Permission is given for the following people to have access to this SP:

| | |
|---|---|
| Available to the general public | Yes |
| Available only after consultation with author/SP adviser | No |
| Available only to those bound by confidentiality agreement | No |

## ACCEPTANCE SHEET

The Special Problem entitled "Phenogram Generator using Frequent Structure Mining over Metabolic Pathways (PhenTor)" prepared and submitted by Lejun Christian L. Osorio in partial fulfillment of the requirements for the degree of Bachelor of Science in Computer Science has been examined and is recommended for acceptance.

**Geoffrey A. Solano, M.Sc.**
Adviser

**EXAMINERS:**

| | **Approved** | **Disapproved** |
|---|---|---|
| 1. Gregorio B. Baes, Ph.D. (*candidate*) | _____ | _____ |
| 2. Avegail D. Carpio, M.Sc. | _____ | _____ |
| 3. Aldrich Colin K. Co, M.Sc. (*candidate*) | _____ | _____ |
| 4. Ma. Sheila A. Magboo, M.Sc. | _____ | _____ |
| 5. Vincent Peter C. Magboo, M.D., M.Sc. | _____ | _____ |
| 6. Richard Bryann L. Chua, M.Sc. | _____ | _____ |
| 7. Bernie B. Terrado, M.Sc. (*candidate*) | _____ | _____ |

Accepted and approved as partial fulfillment of the requirements for the degree of Bachelor of Science in Computer Science.

**Ma. Sheila A. Magboo, M.Sc.**
Unit Head
Mathematical and Computing Sciences Unit
Department of Physical Sciences
and Mathematics

**Marcelina B. Lirazan, Ph.D.**
Chair
Department of Physical Sciences
and Mathematics

**Alex C. Gonzaga, Ph.D., Dr.Eng.**
Dean
College of Arts and Sciences

**Abstract**

A Phenogram generator system using FP-Graph Miner Algorithm, Jaccard similarity index and Hamming Distance, over glycolysis and citrate cycle metabolic pathways.
*Keywords*: phenogram, jaccard similarity index, hamming distance, glycolysis, citrate cycle, metabolic pathways

# Contents

**Acceptance Sheet** i

**Abstract** ii

**List of Figures** v

**I. Introduction** 1
- A. Background of the Study . . . 1
- B. Statement of the Problem . . . 2
- C. Objectives of the Study . . . 2
- D. Significance of the Project . . . 4
- E. Scope and Limitations . . . 4
- F. Assumptions . . . 5

**II. Review of Related Literature** 6

**III. Theoretical Framework** 10
- A. Kyoto Encyclopedia of Genes and Genomes . . . 10
- B. Phengoram . . . 10
- C. Metabolic Pathway . . . 10
  - 1. Glycolysis . . . 10
  - 2. Citrate Cycle . . . 11
- D. Phylogenetic Analysis . . . 11
  - 1. Clade . . . 11
  - 2. Taxon . . . 11
  - 3. Branch Length . . . 11
  - 4. Node . . . 11
- E. FP- GraphMiner . . . 12
  - 1. Bitcode of a Distinct Edge . . . 13
  - 2. Weight of a Bitcode . . . 13
  - 3. Frequency Table . . . 13
  - 4. Node . . . 13
  - 5. Clusters . . . 14
  - 6. Algorithm . . . 14
  - 7. Finding all frequent subgraphs with a given support $\alpha$ . . . 14
  - 8. Finding graphs in GD containing the Query Graph Q . . . 15
- F. Jaccard Similarity Index . . . 15
- G. Hamming Distance . . . 16
- H. Graph Similarity Tree . . . 16
  - 1. Algorithm . . . 16
  - 2. Building the GST . . . 17
- I. PHYLIP . . . 17
  - 1. TreeDist . . . 18
  - 2. DrawGram . . . 18
- J. Majority Rule Consensus Tree Algorithm . . . 18
- K. NCBI . . . 19

**IV. Design and Implementation** **20**
A. Use Cases . . . 20
B. Flow Chart . . . 20

**V. Architecture** **22**
A. System Architecture . . . 22
1. Model . . . 22
2. View . . . 22
3. Controller . . . 22
B. Techincal Architecture . . . 23

**VI. Results** **24**
A. Home Screen . . . 24
B. Selecting Input . . . 25
C. File Checking . . . 26
D. PhenTor Functions . . . 27
E. Export . . . 28
F. Online Feature . . . 30

**VII. Discussions** **33**

**VIII. Conclusions** **35**

**IX. Recommendations** **36**

**X. Bibliography** **37**

**XI. Appendix** **40**
A. Source Code . . . 40
1. Controller . . . 40
2. Model . . . 65
3. PHYLIP.DrawGram . . . 67
4. PHYLIP.Util . . . 69
5. View . . . 78

**XII. Acknowledgement** **98**

# List of Figures


1 Example of a Phenogram . . . . . 10
2 Example of a Frequency Table . . . . . 13
3 (a) Query Graph Q (b) BFS of FP-Graph for finding Q . . . . . 14
4 HeadNode of a Frequency Table . . . . . 15
5 Majority Consensus Tree Algorithm . . . . . 18
6 Top Level Use Case Diagram of PhenTor . . . . . 20
7 Flowchart . . . . . 21
8 PhenTor' Splash Screen . . . . . 24
9 PhenTor Home Screen . . . . . 24
10 Add Specie in the Dataset . . . . . 25
11 Add Specie in the Dataset . . . . . 25
12 Download File . . . . . 26
13 Download Complete . . . . . 26
14 Download Fail . . . . . 27
15 Build Phenogram Output . . . . . 27
16 Similarity of Species Output . . . . . 28
17 Reaction List Output . . . . . 28
18 Tracing Pathways using Reactions Output . . . . . 28
19 Export . . . . . 29
20 Export Folder . . . . . 29
21 Export Complete . . . . . 29
22 Online . . . . . 30
23 Online Frame . . . . . 30
24 Viewing of a Metabolic Pathway . . . . . 31
25 Donwloading File . . . . . 31
26 Download Complete . . . . . 32
27 No Internet Connection . . . . . 32
28 Sample Groups with Phylip and MCT Testing . . . . . 35

# I. Introduction

## A. Background of the Study

Building phylogenetic tree [1] is one of the things that have been in the research industry for several decades. Studying the evolutionary relationship between organisms became a big thing. Constructing different methods and algorithms were formulated to create a phylogenetic tree. Each and every one of them uses different methods to build a phylogenetic tree. The goal of building a phylogenetic tree is to create an evolutionary tree based on their morphological similarities, not based on their taxonomy.

Metabolic pathways defined as series of chemical reactions occurring within a cell. Every organism has different set of metabolic pathways. Each pathway may differ from certain specie to another. It became one of the bases for making a phylogenetic tree. Many methods exist on using metabolic pathways. Some of them use the presence or absence of a specific pathway [2]; some compares the similarity of specific pathways between species [1, 3].

Over a large database of species, the main goal is to create a more efficient way to find the similar structures (morphology) between different species. Kyoto Encyclopaedia of Genes and Genomes or KEGG [4], one of the most widely used biological databases, offers free public access on its database [http://www.kegg.jp/kegg/pathway.html]. It contains many biological data, including pathways. KEGG's data is on kgml (*.xml) format. They also show images of pathways using graph representation in jpg format. System biologist can make use of this data to create efficient ways to build a phylogenetic tree.

Frequent structure mining is one of the main topics in data mining. Finding the frequent structure over a large database, computer scientist will be to estimate on how the structure behaves. Using frequent structure mining, we can find the relationship of the sets in the system. One of the examples of a structured data is the metabolic pathways. Using frequent structure mining on metabolic pathway of different species, system biologists can estimate the relationship between the species being tested.

## B. Statement of the Problem

Using frequent structure mining algorithm, over glycolysis and citrate cycle metabolic pathways might result to different phylogeny than the previous studies about building a phenogram or phylogenetic tree.

Using similarity matrix and distance matrix are one of the obvious ways on how to get the similarity or distance between two sets, especially, on binary vectors. Metabolic pathway is a graph itself and can be represented as a binary vector, which is, absence or presence of edge in the graph with respect to the reference map of the pathway.

Existing works regarding building Phenogram using metabolic pathways didn't use Similarity Index to get the relationship of the organisms to be tested. Using Similarity Index for getting the relationship of organisms may yield to a different result than the other existing works regarding building of Phenogram or Phylogenetic Tree.

Comparing the Phenogram generated using the defined algorithm with the NCBI Taxonomy Standard using PHYLIP TreeDist and Majority Rule Consensus Tree Algorithm will be a great help for System Biologist because they doesn't need to run multiple programs to build and compare the Phenogram with NCBI Taxonomy.

KEGG offers free access for pathways including Glycolysis and Citrate Cycle in extensible mark-up language format (*.xml). Visualization of pathways can be rendered manually on it's website. [http://www.kegg.jp/kegg/pathway.html]

Given the availability of data, and the knowledge about metabolic pathways and frequent structure mining we can find:

1. finds similar structures between metabolic pathways
2. identify similar structures across the dataset
3. cluster similar species across the dataset

## C. Objectives of the Study

To create PhenTor [Phenogram Generator], a tool for making Phenogram using Glycolysis and Citrate Cycle metabolic pathways as basis, which have the following functions

to use:

1. Viewing Glycolysis and Citrate Cycle metabolic pathways from KEGG
2. Downloading KEGG data files (kgml in xml file format) of Glycolysis and Citrate Cycle metabolic pathways
3. to provide support for choosing between two indices for summarizing the input
    (a) using Hamming Distance
    (b) using Jaccard Similarity Index
4. to generate Phenogram using the input which is chosen by the user
5. to compare Phenogram output with NCBI taxonomy using:
    (a) PHYLIP TreeDist
    (b) Majority Rule Consensus Tree Algorithm
6. to list the frequent reactions between metabolic pathways (Glycolysis and Citrate Cycle) given a certain threshold
7. to list the species having a certain set of reactions
8. to provide support for export:
    (a) allows export in text file format (*.txt) of the following outputs:
        i. grouping of Organisms
        ii. Organism Similarities
        iii. Organisms containing certain set of reactions
        iv. Set of Reactions across the dataset of algorithms having a certain threshold
    (b) allow export to image (*.png) of phenogram generated

## D. Significance of the Project

As the study of the evolutionary relationship between organisms became known, creating efficient and accurate methods became one of the things system biologist study. Many methods exist for building phylogenetic tree [1]. Phylogenetic tree inferred from those methods are not necessarily the same.

Using frequent structure mining and similarity indices, the tool will try to cluster the organisms chosen by the user, and try to build a phenogram for the said organisms. Using the phenogram generated, relationship across the dataset can be inferred by the user. The user can view the closest organism based on the Glycolysis and Citrate Cycle metabolic pathway of a certain organism.

Having a system that can generate Phenogram and then compare it to NCBI Taxonomy standards can provide System Biologist and other users who want to build phenogram, shorter time for building and comparing because they don't need to run different programs at the same time.

However, a fast system that is not readily accessible may not really serve its purpose. And downloadable software answers this problem as it will make the system available for the public and they have access to the software in their own homes.

Even though there are already many existing systems that are reliable in building a Phenogram or Phylogenetic Tree, using Similarity Index over Glycolysis and Citrate Cycle metabolic pathway might yield to a different result than the other existing systems.

Different tests can be done in the generated phenogram to see how accurate the algorithm being used by the tool.

## E. Scope and Limitations

1. The organisms that can only be tested are those where glycolysis and citrate cycle metabolic pathways exists.

2. User will view instructions and guidelines.

3. User will download the kgml files of Glycolysis and Citrate Cycle metabolic path-

ways provided in KEGG

4. User will not make changes on the downloaded kgml files of the Glycolysis and Citrate Cycle metabolic pathways from KEGG.

5. User will follow the defined format for the required input file in tracing of species with certain sets of reactions

6. Proposed system will only use the defined algorithm for building Phenogram.

7. to use the online feature of downloading data from KEGG, the user must have a stable internet connection.

8. Species to be tested is only limited to aerobic organisms, where Glycolysis and Citrate Cycle exists.

9. the tool will only create a phylogeny based on what is on the algorithm being proposed. The user will be the one to test if the phylogeny generated is right or wrong.

## F. Assumptions

In order the tool being proposed to work perfectly, the following assumptions will be considered:

1. The KGML files of Glycolysis and Citrate Cycle metabolic pathways are not corrupted.

2. Glycolysis and Citrate Cycle Metabolic Pathways must come from KEGG file format.

3. Internet Connection for online features.

4. the user must have Java Runtime Environment to run the application.

# II. Review of Related Literature

System biologists developed a new way of classifying the relationship between species, not based on their taxonomy, but based on their "morphological similarities" (Weston et. al [5]). They developed Phylogenetic Analysis, which means "inferring or estimating" (Brinkman et. al [6]) of the evolutionary relationship between species.

Phylogenetic was first introduced by Willi Hennig in his paper back in 1936 (Schimtt [7]). After his introduction of Phylogenetic, System biologists find ways on how to develop the Phylogenetic Tree or the evolutionary tree of species. Each and every phylogenetic tree may differ from one another, because of the different methods and datasets that they are using. The inferred phylogenetic tree of each method is tested to see how it differs from the known taxonomy in the National Center for Biotechnology Information database, simply put, NCBI database. Brinkman et. al [6] list down some ways to check how the derived phylogenetic tree from the used method differs from the taxonomical relationships in the NCBI database. The use of Phylogenetic Software like PHYLIP, PAUP and PUZZLE is the known way to get the similarity and distance of the derived tree to the NCBI taxonomy. One of the commonly used tools is PHYLIP. It has over 30 executable programs to use. PHYLIP unlike other is a console-based application.

Some of Phylogenetic Analysis uses molecular sequence (Felsenstein [8]) and Metabolic Pathways (Forst et. al [9], Liao et. al [2], Heymans-Singh [1] ,Solano et. al [3]).

The abundance of DNA/RNA sequence data currently available for a variety of organisms has led to phylogenetic inference based on these data [1]. Most studies using molecular sequence uses Ribosomal RNA 16S sequence, since this sequence exist in all organisms and are highly conserved (Maidak et. al [10], Buyser et. al [11]). Using single gene sequence is not sufficient for phylogenetic analysis, Fitz et. al and Li et. al [12, 13] uses the whole genome as basis for phylogenetic analysis. However, only a limited number of genome sequences are available to date, and hence the results of such techniques may not be representative of the whole picture [1].

Although, Schuster et. al [14] stated that the structure of the metabolic pathways alone is insufficient for comprehensive and credible results, many researchers still use this

as one of the basis for constructing phylogenetic tree.

Baldan et. al [15] stated that the first thing to do is to choose on how to represent metabolic pathways. Most of the time, researcher uses binary vectors to represent the presence or absence of the homologs. Matteo Pellegrini [16] shows another way other than the binary vectors, which is the use of Logical Analysis on Schematics Representation.

There exist many methods on using metabolic pathways as basis for deriving phylogenetic tree [1, 2, 3, 9]).

Forst et. al [9] uses the combination of metabolic pathway and sequence information of the species. By computing distances of pathways with different weighting parameters, and by comparing the yielded phylogenies, one can draw conclusions about the robustness or versatility of relationships between pathway representations.

Liao et al [2] uses the presence and absence of metabolic pathways in organisms. It was presented as a boolean vector. Based on this methodology and using some specific distance measures on these profiles, pairwise comparisons of a set of completed genomes are performed, and phylogenetic trees are constructed using hierarchical clustering. The results provide a perspective on the relationship among organisms that is different from conventional phylogenetic trees based on 16s rRNA.

Heymans-Singh [1] uses the Enzyme Commission numbers of the enzymes present in the pathways to determine the similarity in a pairwise comparison. They used Glycolysis and Citrate Cycle pathways, as well as the Carbohydrate and Lipid metabolic networks, on a varying number of organisms.

Solano et al [3] uses Glycolysis , a metabolic pathway which exist in all species, as a basis of comparison on each species. They represent Glycolysis Metabolic Pathway as a hypergraph. Their algorithm uses FP-Graph Miner to get the frequent subgraphs patterns between the formed hypergraphs of the glycolysis metabolic pathway of each species. There are some online database of metabolic pathways, which has a free-for-all access. Two of the well-known are the KEGG or Kyoto Encyclopaedia of Genes and Genomes (KEGG) [4] and the BioCyc.org [17].

H. Dinari and H. Naderi [18] creates a survey on frequent subgraph pattern min-

ing methods. They listed some frequent subgraph mining algorithms like gSpan [19], MARGIN [20], FSMA [21] and FP-Graph Miner [22].

Yan et. al's gSpan algorithm builds a new lexicographic order among graphs, and map each graph to a unique minimum Depth-First Search (DFS) code as it's canonical label. Based on this lexicographic order gSpan adopts the depth-first search strategy to mine frequent connected subgraphs efficiently [19].

Thomas et. al's MARGIN algorithm mines the maximal frequent subgraphs while pruning the lattice space considerably. It explores a much smaller space by visiting lattice around the *f-cut* nodes. MARGIN reduces the number of isomorphism computations which is the kernel of all frequent subgraph mining problems [20].

Wu et. al's FSMA or Frequent Subgraph Mining Algorithm uses incidence matrix to represent the labelled graphs and to detect their isomorphism. Starting from the frequent edges from the graph database, the algorithm searches the frequent subgraphs by adding frequent edges progressively. By normalizing the incidence matrix of the graph, the algorithm can effectively reduce the computational cost on verifying the isomorphism of the subgraphs [21].

Vijayalakshmi et. al [22] represents the edges of each graphs as bitcode with bitsize $= n$, wherein $n$ = number of graphs. A bitcode value of Edge $E$ at position $k$ is equal to 1 if graph $K$ contains edge $E$, otherwise, 0. After getting the bitcodes of each edges, they will be sorted out first as clusters, depending on the number of 1's in the bitcode and then as nodes in each clusters depending on their decimal value (e.g *10001 = 17*). The node with a decimal value $2^{n-1}$ is the frequent subgraph over the whole database.

Paul Jaccard introduced an easy and direct approach on how to get the similarity between two graphs [23]. Jaccard Similarity Index ($JSI$ )for sets A and B is defined as :

$$JSI = \frac{(A \cap B)}{(A \cup B)}$$

Jaccard similarity index can easily be applied to any *categorical data*, where the data attributes are not numerical, but rather represent the presence or absence of a property [23]. Raymond et. al [24] uses Jaccard Index to compare the reference clustering to the

evaluated clustering from their work.

# III. Theoretical Framework

## A. Kyoto Encyclopedia of Genes and Genomes

The Kyoto Encyclopedia of Genes and Genomes (KEGG) is a public repository of experimental data that integrates genomic, chemical and systemic functional information of different organisms. It offers free access on its data for metabolic pathways. They support the viewing of all metabolic pathways for an organisms, and the viewing of all the species having the specific pathway. KEGG database can be accessed at their site, www.kegg.jp .

## B. Phengoram

A Phenogram is a diagram depicting taxonomic relationships among organisms based on overall similarity of many characteristics without regard to evolutionary history or assumed significance of specific characters

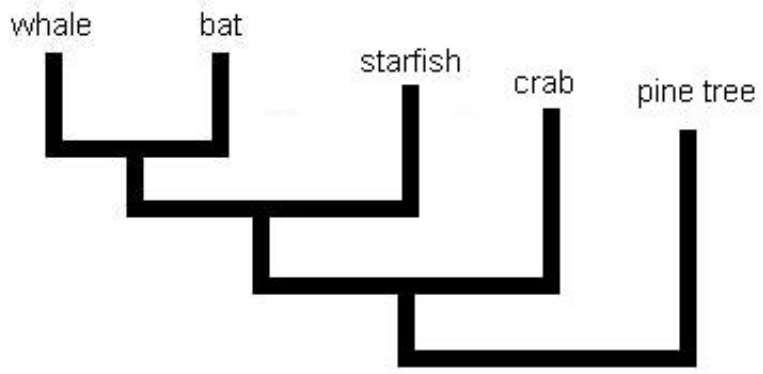


Figure 1: Example of a Phenogram

## C. Metabolic Pathway

Metabolic Pathway is a series of chemical reactions occurring within a cell.

### 1. Glycolysis

Glycolysis is the almost universal pathway that converts glucose into pyruvate

### 2. Citrate Cycle

The Citrate cycle, also called the citric acid cycle or krebs cycle, is a fundamental metabolic pathway involving eight enzymes essential for energy production through aerobic respiration, and, like glycolysis, arose early in evolution.

## D. Phylogenetic Analysis

Phylogenetic Analysis is the inferring or estimating the evolutionary relationship between organisms [6]. It was first introduced by Willi Hennig [5, 7].

Brinkman et. al defined the following terms for understanding Phylogenetic Analysis [6]:

### 1. Clade

Clades are groups of organisms or genes that include the most recent common ancestor of all of its members and all of the descendants of that most recent common ancestor.

### 2. Taxon

A taxon is any named group of organisms but not necessarily a clade.

### 3. Branch Length

Branch Length correspong to divergence.

### 4. Node

A node is a bifurcating branch point.

Brinkman et. al [6] also defined 7 "default" assumptions in the model inherited in the phylogenetic analysis [6]:

1. The sequence is correct and originates from the specified source.

2. The sequence are homologous (i.e., are all descended in some way from a shared ancestral sequence).

3. Each position in a sequence alignment is homologous with every other in that alignment.

4. Each of the multiple sequences included in a common analysis has a common phylogenetic history with the others (e.g., there are no mixtures of nuclear and organellar sequences).

5. The sampling of taxa is adequate to resolve the problem of interest.

6. Sequence variation among the samples is representative of the broader group of interest.

7. The sequence variability in the sample contains phylogenetic signal adequate to resolve the problem of interest.

A straightforward phylogenetic analysis consists of four steps [6]:

1. Alignment (both building the data model and extracting a phylogenetic dataset)

2. Determining the substitution model

3. Tree building

4. Tree evaluation

## E. FP- GraphMiner

FP-GraphMiner, introduced by Vijayalakshmi et. al [22], is an algorithm for frequent pattern mining over a large database. Vijayalakshmi et. al [22] defined the following terms as:

### 1. Bitcode of a Distinct Edge

The Bitcode of a Distinct Edge $DE_i$, is a $k$ length bit string, each bit corresponding to a graph in $GD$, consisting of 1's in the position of the graphs in which the edge is present and 0's if it is absent. The Bitcode gives information about the graphs in which the distinct edge is present.

### 2. Weight of a Bitcode

The weight of a BitCode of an edge $DE_i$, denoted as $WT(DE_i)$, is the count of 1's in it.

### 3. Frequency Table

A frequency table $FT$ is defined as a collection of distinct edges of $k$ graphs in $GD$ in decreasing order with respect to the binary encoding of the bitcodes.

| Sl No | Distinct Edges <$DE_i$,BitCode($DE_i$)> | Sl No | Distinct Edges <$DE_i$,BitCode($DE_i$)> |
|---|---|---|---|
| 1 | <ab, 11111> | 11 | <ad, 11001> |
| 2 | <ac,11111> | 12 | <dg, 10101> |
| 3 | <bc, 11111> | 13 | <be, 11000> |
| 4 | <bd,11111> | 14 | <gh, 10100> |
| 5 | <df,11111> | 15 | <bh, 10010> |
| 6 | <de,11101> | 16 | <dh, 10010> |
| 7 | <ef, 11101> | 17 | <ce, 10001> |
| 8 | <eg,11101> | 18 | <hi, 10000> |
| 9 | <fg,11101> | 19 | <eh, 00100> |
| 10 | <cd,11011> | 20 | <fh, 00100> |

*Node* *Cluster*

Figure 2: Example of a Frequency Table

### 4. Node

Edges wtih the same BitCode.

## 5. Clusters

Nodes with the same parity

## 6. Algorithm

1. Create the corresponding bitcode of each edges, with size $n$ , where $n$ = *Number of Graphs in the set*

2. Create the Frequency Table by sorting the bitcodes by cluster in decreasing order, and then, sorting the clusters, by nodes, in decreasing order.

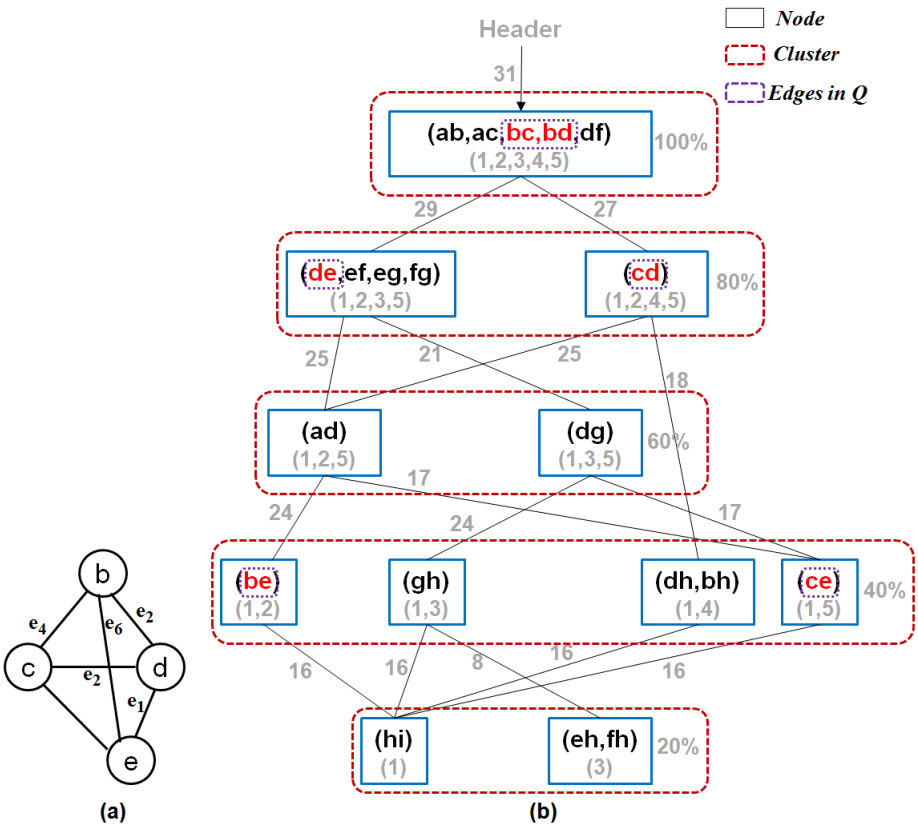


Figure 3: (a) Query Graph Q (b) BFS of FP-Graph for finding Q

## 7. Finding all frequent subgraphs with a given support $\alpha$

The FP-Graph Miner performs DFS walks starting from the Cluster having the specified support to the HeaderNode to obtain all frequent subgraphs.

### 8. Finding graphs in GD containing the Query Graph Q

Given a query graph Q, the graphs in GD containing it can be easily identified by performing a Breadth First Search BFS starting from the HeaderNode till all the edges of Q are obtained. The BitCodes of these Nodes are collected into an array BFS(Q). An AND operation on the BitCodes in BFS(Q) gives a BitCode and the position of 1's in the resulting BitCode shows the graphs containing Q.

The HeadNode with a bitcode without 0 bit, most of the time, the topmost cluster in the frequency table, is defined as the pattern or subgraph that exists in most, if not all, graphs in the database.

| <ab, 11111> |
|---|
| <ac,11111> |
| <bc, 11111> |
| <bd,11111> |
| <df,11111> |

Figure 4: HeadNode of a Frequency Table

## F. Jaccard Similarity Index

Jaccard Similarity Index as defined by Paul Jaccard [25], is the quotient of the intersection of the sets and the union of the the sets.

$$JSI = \frac{(A \cap B)}{(A \cup B)}$$

Jaccard Similarity Index is one of the most common similarity index [24], since it is an easy and direct way of getting the similarity of two sets [23]. It can also be used to get the distance between two sets. The Jaccard Distance (JD )is defined as:

$$JD = 1 - \frac{(A \cap B)}{(A \cup B)}$$

Jaccard can be applied to any categorical data, where the data attributes are not numerical but rather representation of presence or absence of a property. Jaccard Index

value only takes from [0, 1]. 0, if the two (2) have no common element. 1, if the sets are identical.

## G. Hamming Distance

The Hamming distance between two strings of equal length is the number of positions at which the corresponding symbols are different. In application to binary vectors, it is defined as the number of bit flips between two binary vectors.

## H. Graph Similarity Tree

Graph Similarity Tree (GST) (Solano et. al [26]), as the name implies, creates a tree of similarity between graphs.

### 1. Algorithm

Let GD be a set of graphs G1, G2, G3, , Gn, n = number of graphs

1. Create FP-table using GD
2. Summare FP-table using index matrix
3. Normalize index matrix (if possible)
4. Build GST

Solano et. al uses two ways on getting the index matrix for summarizing GD: (!) Jaccard Similarity Index, and (2) Hamming Distance

1. Jaccard index is the quotient between the intersection and the union of the two sets.

$$JSI = \frac{(A \cap B)}{(A \cup B)}$$

applying JSI in FP-Table: in every pair column i and column j in FP-Table, which refers to $G_i$ and $G_j$ in GD,

$$JSI[i,j] = JSI[j,i] = \frac{(i \cap j)}{(i \cup j)}$$

2. Hamming distance is the number of bit flips between two binary vectors. Every column in the FP-Table is a binary vector, so getting Hamming Distance is an easy thing.

   Let HD‘ be the hamming distance between columng i and column j, and e number of rows, which is the number of distinct edges, in FP-Table.

$$HD[i,j] = HD[j,i] = \frac{(e - HD‘)}{(e)}$$

**2. Building the GST**

1. For every $G_i$ in GD that will be inserted in tree T, find the node j in T with highest index value in ith row (only lower triangle involved)
2. if j doesnt have a sibling (i.e. initial case), make i sibling of j
3. Find graph k in tree T, wherein index[j,i] >index[j,k]. Make i, and subtree (sibling of k) be siblings. And k, their predecessor
4. if index[k,j] >index[k,i], make j right child of the sibling of k, if k is a right child, otherwise, left child
5. if index[k,j] <index[k,i], make i right child of the sibling of k, if k is a right child, otherwise, left child

# I. PHYLIP

The Phylogeny Inference Package [PhylIP] provides modules that aid in phylogenetic analysis, from basic alignment, building and visualizing the tree, and comparing the similiarity score between trees. PHYLIP is a command-line program and does not have a point-and-click interface. It consists of about 30 programs that cover most aspects of Phylogenetic Analysis.

### 1. TreeDist

The TreeDist program provides sound comparison for both metric and symmetric trees.

### 2. DrawGram

The DrawGram program provides drawing of Phenogram using their defined format in text file format.

## J. Majority Rule Consensus Tree Algorithm

Majority Rule Consensus Tree Algorithm [27] provides the consensus tree for two or more trees with equal labels. If a cluster exist in $\geqslant N/2$ , N = number of trees, that cluster will be included in the Majority Rule Consensus Tree.

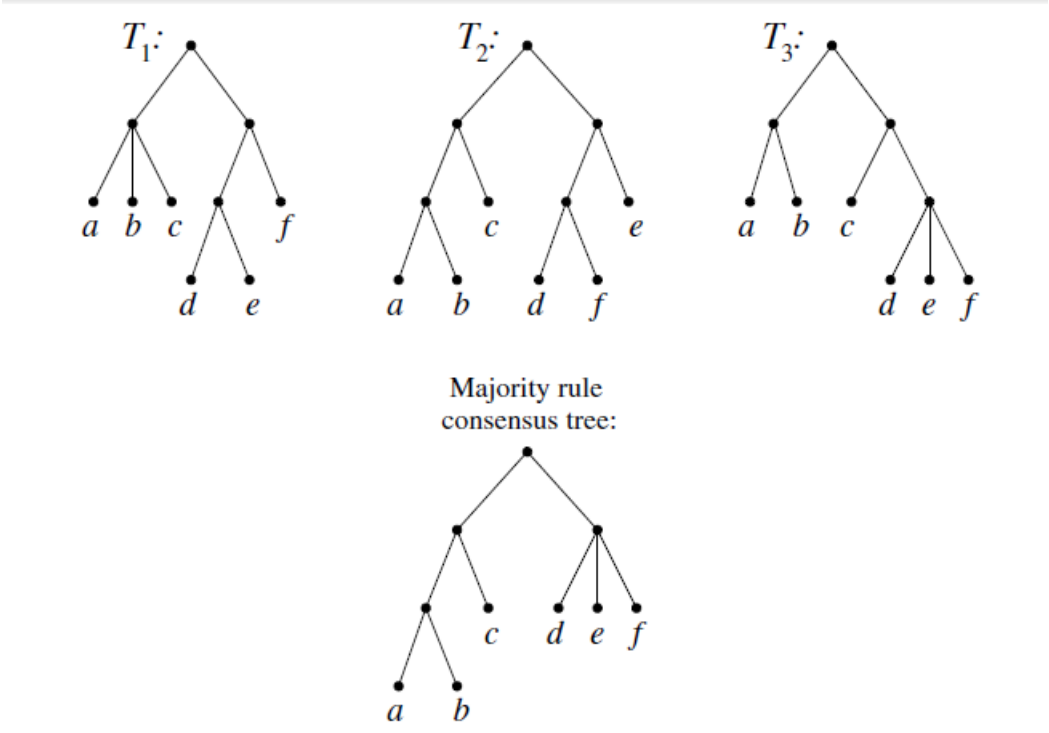


Figure 5: Majority Consensus Tree Algorithm

## K. NCBI

The National Center for Biotechnology Information (NCBI) advances science and health by providing access to biomedical and genomic information. It contains the known standard for evolutionary relationship between organisms using their genomic information.

# IV. Design and Implementation

## A. Use Cases

The user can update the data list, view metabolic pathways in kegg, download KEGG data files (*.xml) from KEGG and build a phylogenetic tree.

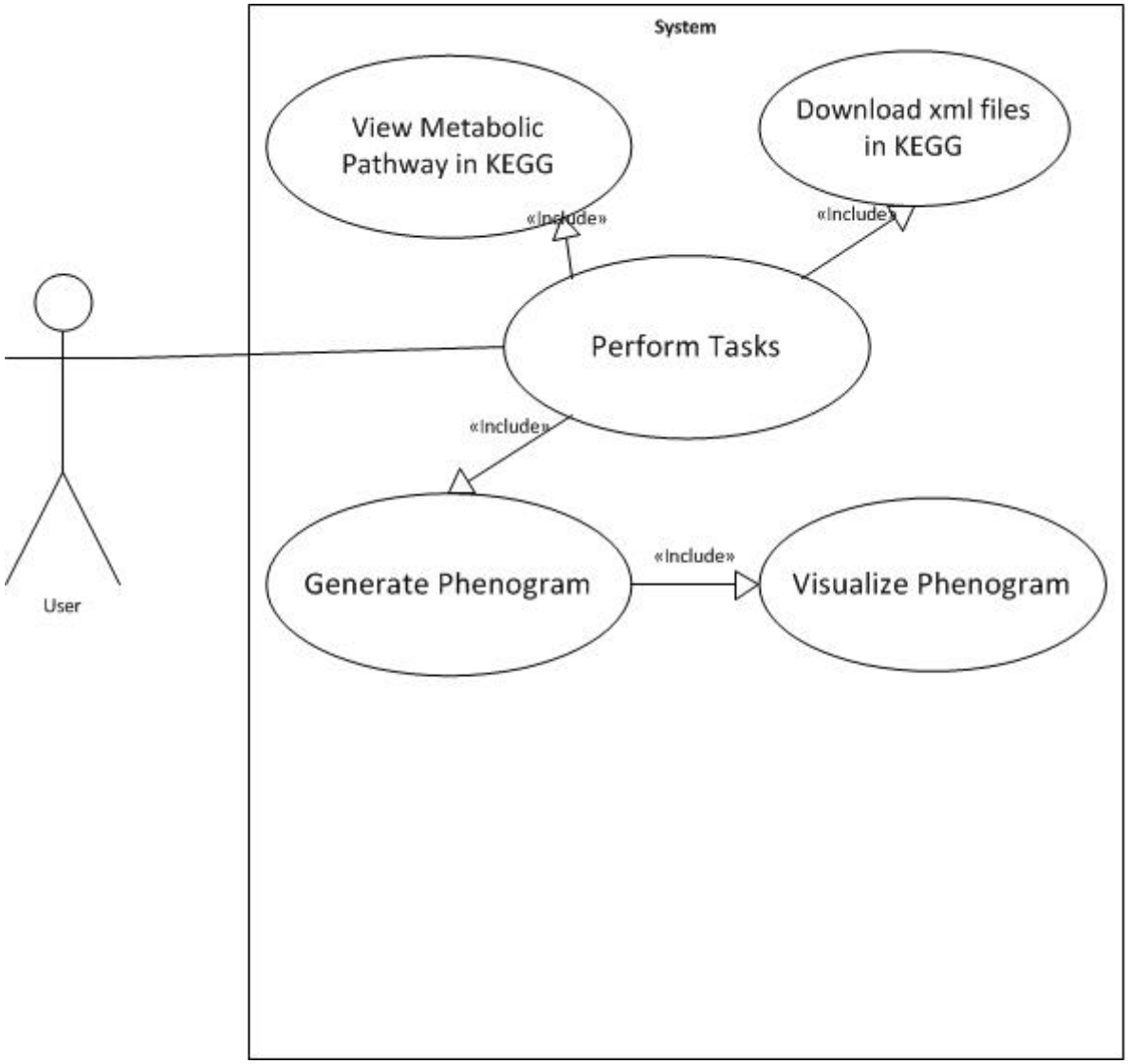


Figure 6: Top Level Use Case Diagram of PhenTor

## B. Flow Chart

The user may use the online feature for viewing JPEG of glycolysis and citrate cycle metabolic pathways of species, and download xml file format of those metabolic pathways. The offline features of the application consists of (1) Building Phenogram, (2)

Getting similarities between organisms, (3) trace frequent reactions accross the dataset and (4) trace organisms using reactions. After running the application with the desired functionality, the user can now export the output for specific functionalities in their defined format.

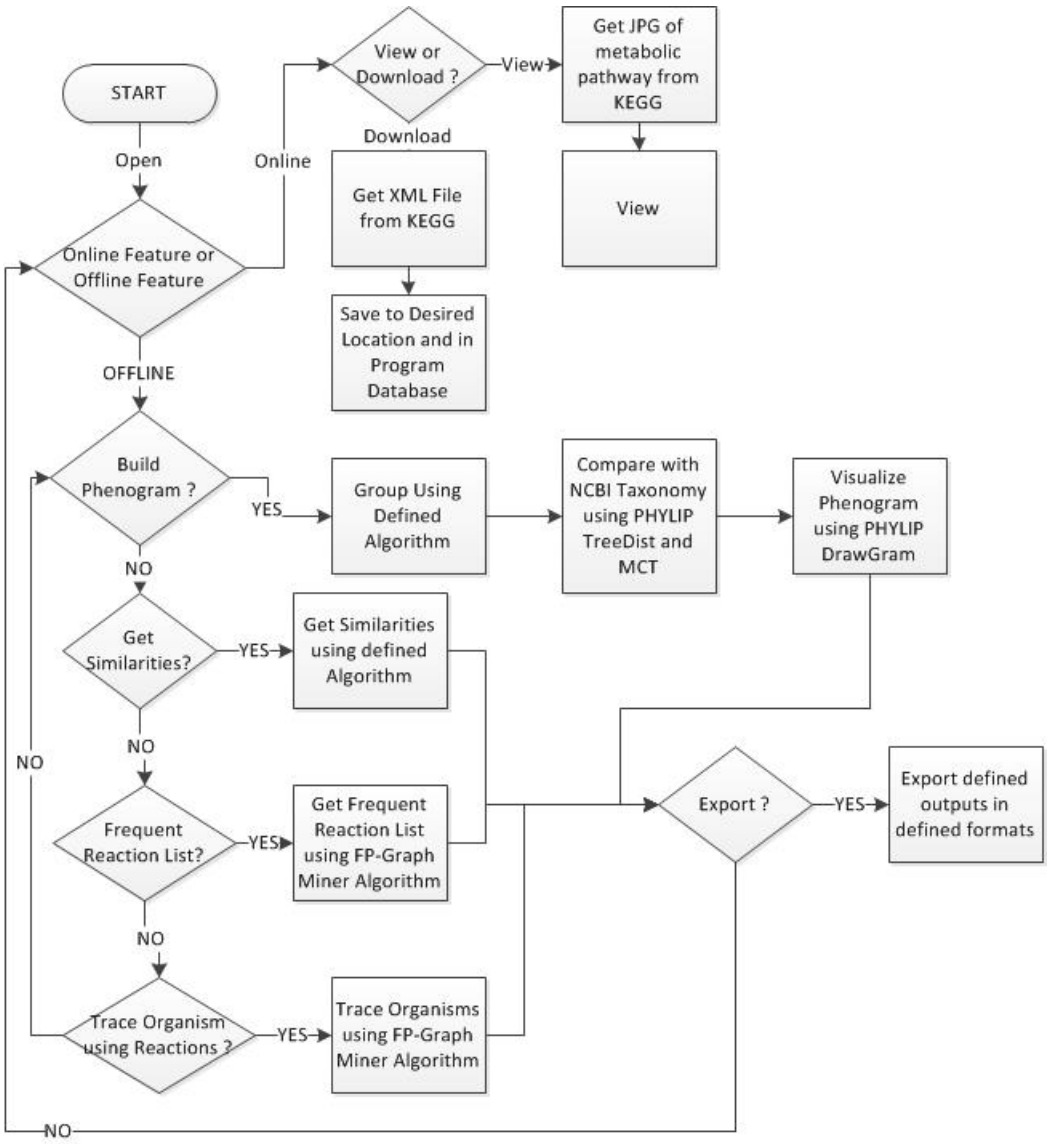


Figure 7: Flowchart

# V. Architecture

## A. System Architecture

The application was written in JAVA. The PHYLIP TreeDist is implemented in the in the application using the Runtime.Environment of JAVA. It also implements the DrawGram function of PHYLIP. The application follow the MVC standart (Model-View-Controller).

### 1. Model

This section contains the core models for the system. This module represents the glycolysis and citrate cycle metabolic pathways graphical representation of the xml files from KEGG. The glycolysis and citrate cycle metabolic pathways of each species in kgml files from KEGG will be converted into the desired representation in the Model. Each entry correspond to a vertex, and each reaction corresponds to a directed edge, where every edge (rection) contains 2 vertices. If the reaction is reversible, then the edge is a two way edge, otherwise, one-way edge.

### 2. View

This section makes the bridge between the system and the user. It shows the tool and has the help function for the user to understand how to use the system.

### 3. Controller

This module contains all the processes or the framework the system. This contains the functions for the system to work. This modules is reponsible for the validation of the input, reading input from KEGG if used in online mode, checking if the required files are in the right folder if used in offline mode. It also handles the conversion of xml files from KEGG to the desired format for the system, which is in the Model Module. It has the function of converting the kgml/ xml files into the required representation for the system. After converting the xml to graph representation, it will then validate if the

data is normalizable or not. The user can choose between 2 ways to summarize the input data, which is needed to generate a phenogram.

1. using Jaccard Similarity Index
2. using Hamming Distance

.

It also contains the main algorithm to be used for building phenogram, which is the GraphSimilarityTree Algorithm.

## B. Techincal Architecture

The system will be implemented as a stand-alone application using Java. Considering the space to be taken by the algorithm, the following hardware specifications are suggested to ensure fast processing:

1. 1.8 GHz clock speed, at least
2. 1.0 GB of memory, at least
3. 10 GB storage capacity, at least, should a larger database be needed
4. it is advised that speed of the storage be 1800 rotations-per-minute (rpm), at least

To ensure a smooth flow, the software requires:

1. 32-bit operating system
2. Microsoft Windows 2000/XP/Vista/7, Linux
3. JRE (Java Runtime Environment)
4. Reliable internet connection (for online feature)

# VI. Results

## A. Home Screen

At once, the splash screen will display as seen in Figure 8 and will open the home screen afterwards. The home screen of PhenTor application is shown in Figure 9. Upon opening of the application, the user can now try and make us of the functionalities of the application.

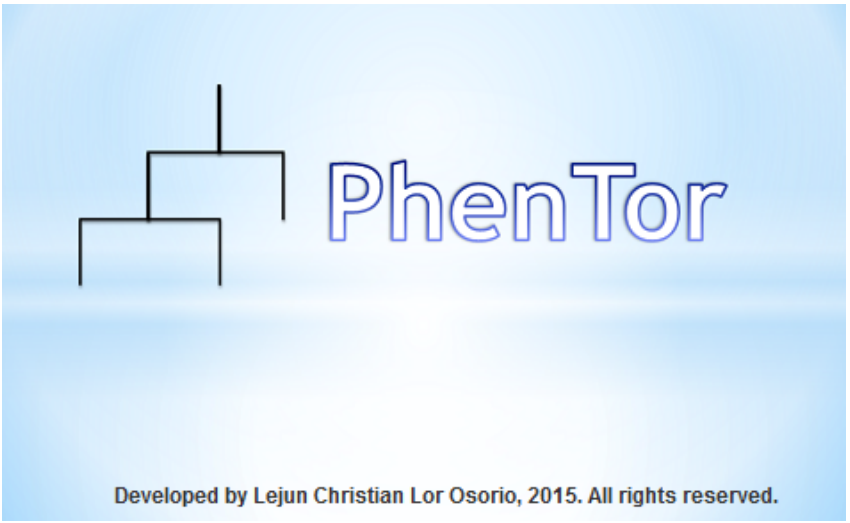


Figure 8: PhenTor' Splash Screen

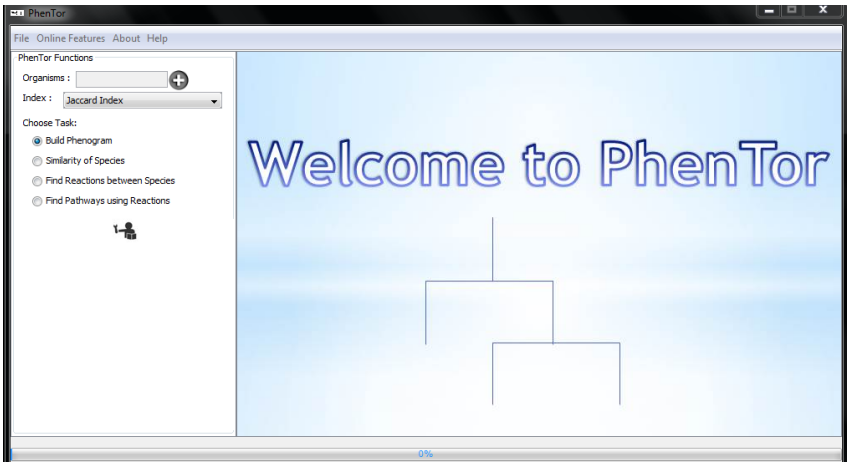


Figure 9: PhenTor Home Screen

## B. Selecting Input

The application will show a new window for the user to choose the species to test as seen in 10. The user can manually input the keyword (Specie Scientific Name or the equivalent KEGG Code Name) for the organism so that it can be added to the dataset.

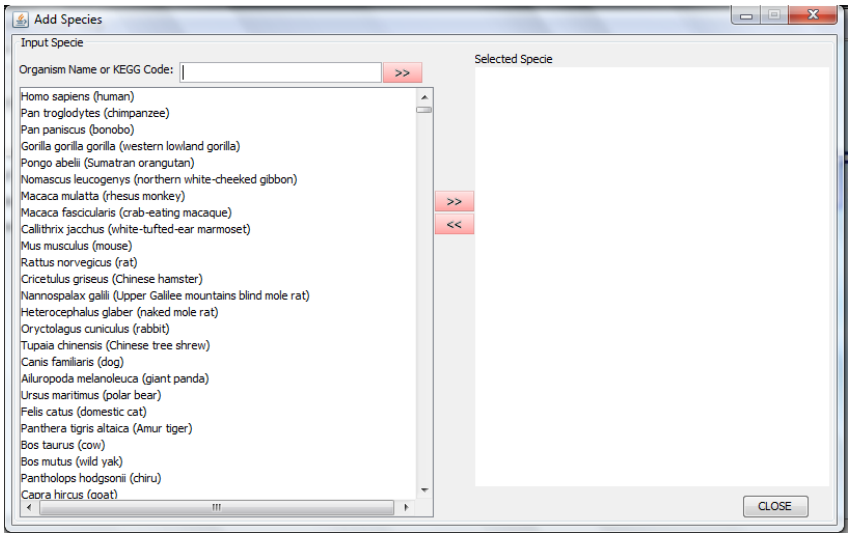


Figure 10: Add Specie in the Dataset

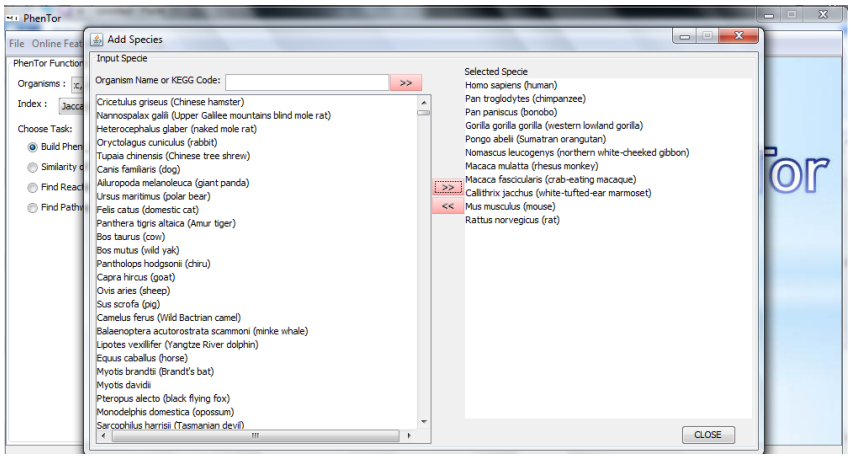


Figure 11: Add Specie in the Dataset

## C. File Checking

After entering the dataset and choosing the index to be used, the application will check if all the files needed for building phenogram exists. If not, as seen in 12, the application will try to download the non-existing file from KEGG. The application will notify the user if the download has been completed (Figure 13), or not as seen in (Figure 14).

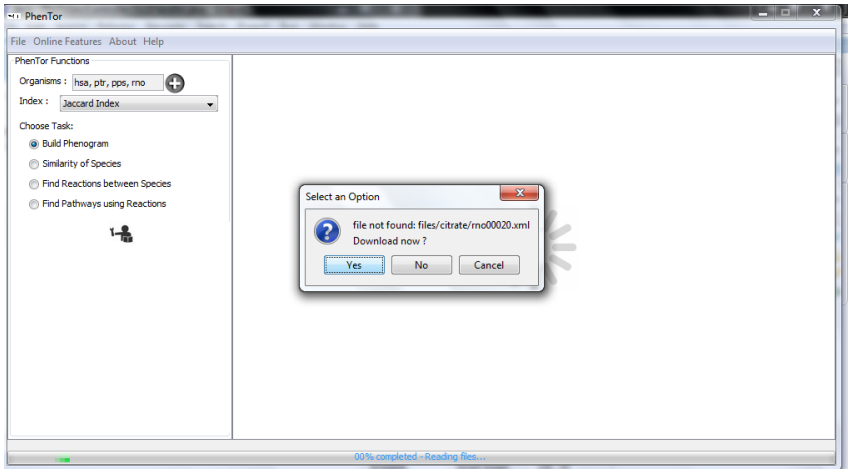


Figure 12: Download File

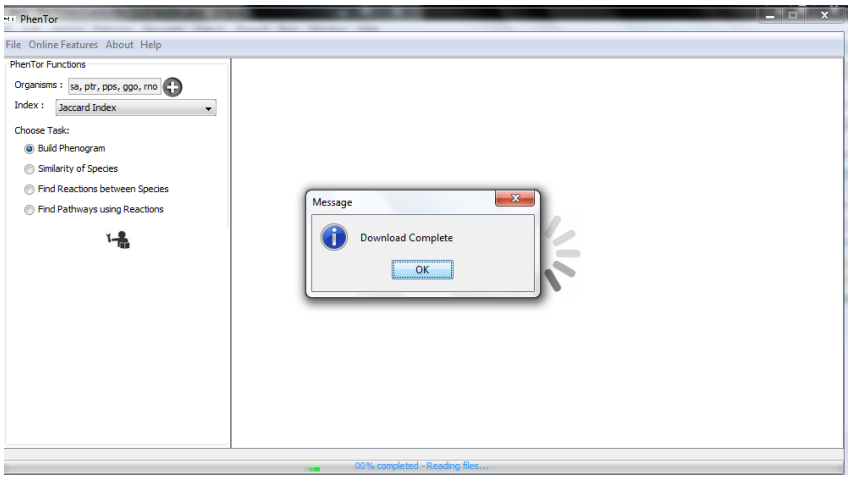


Figure 13: Download Complete

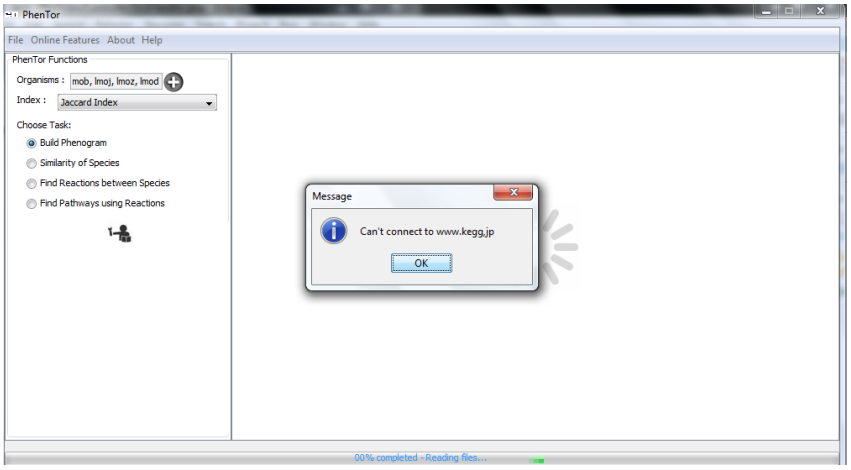


Figure 14: Download Fail

## D. PhenTor Functions

After clicking the run button for the selected functionality of PhenTor, the application will view the specific output for each functionalities.

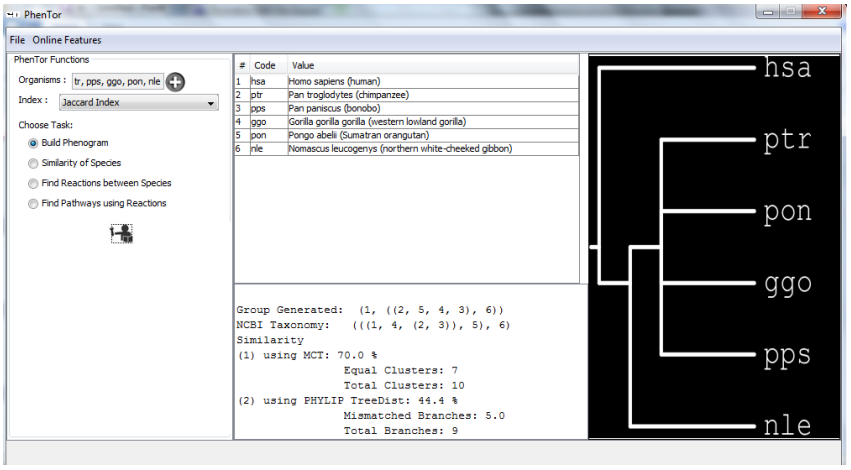


Figure 15: Build Phenogram Output

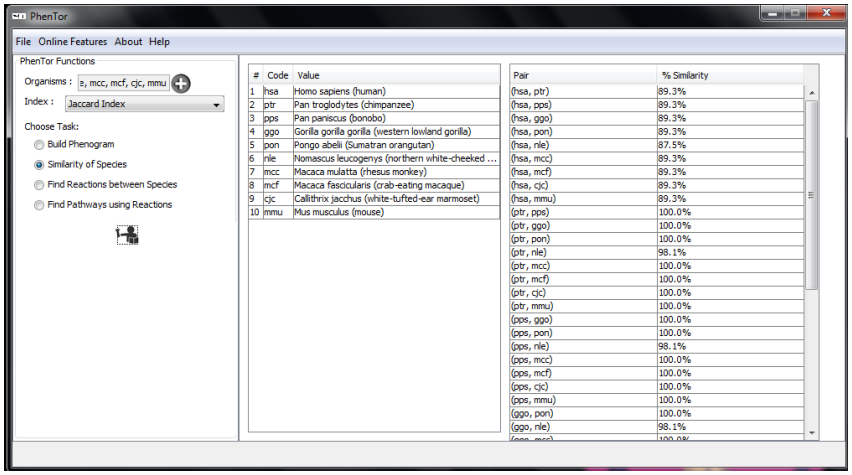


Figure 16: Similarity of Species Output

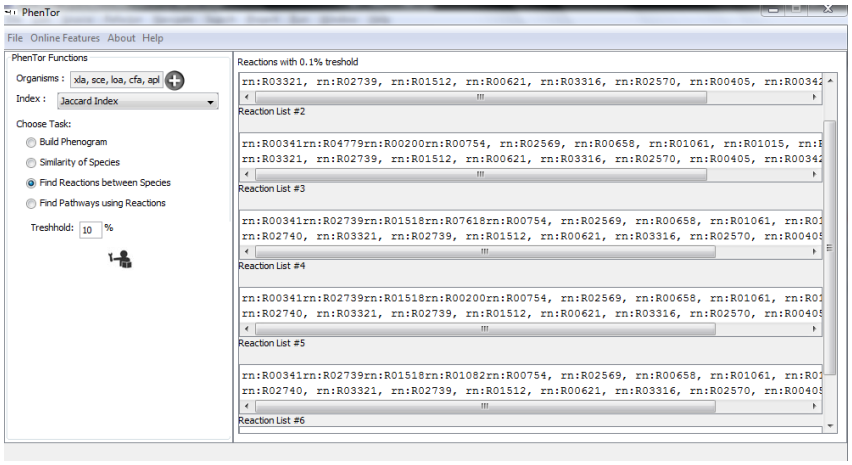


Figure 17: Reaction List Output

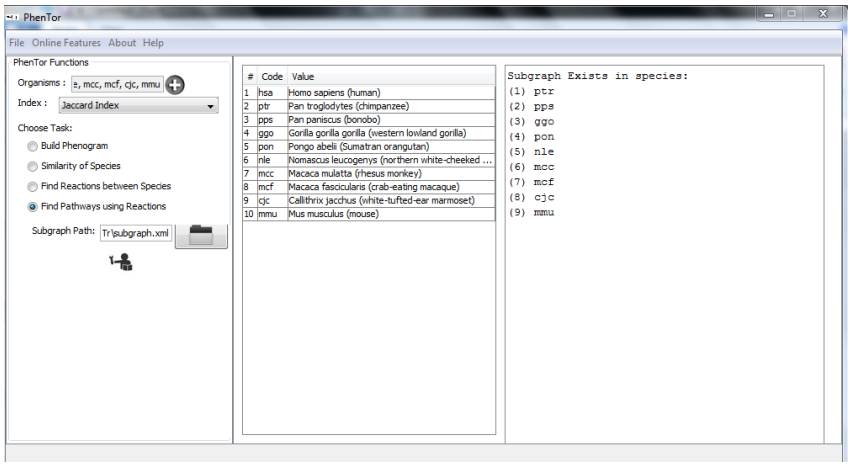


Figure 18: Tracing Pathways using Reactions Output

## E. Export

The user can also export the current output to its defined format.

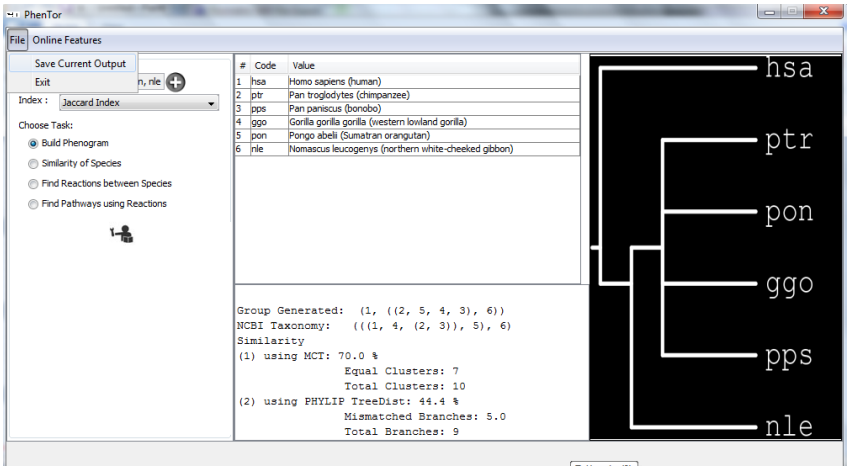


Figure 19: Export

The user can choose the destination folder

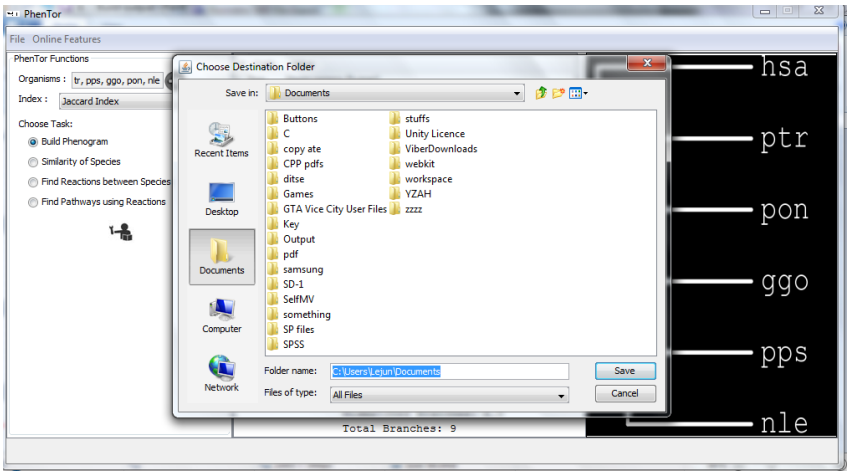


Figure 20: Export Folder

The application will notify the user if the exporting of files has been completed

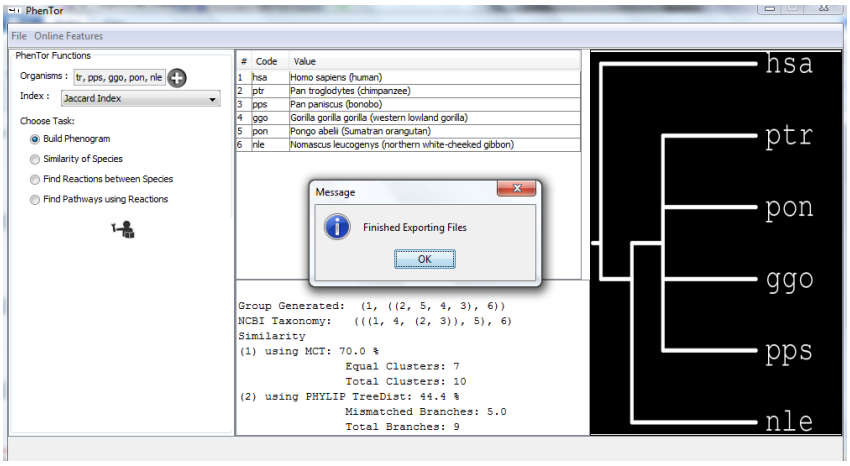


Figure 21: Export Complete

## F. Online Feature

The user can view the image representation of glycolysis and citrate cycle metabolic pathway from KEGG.

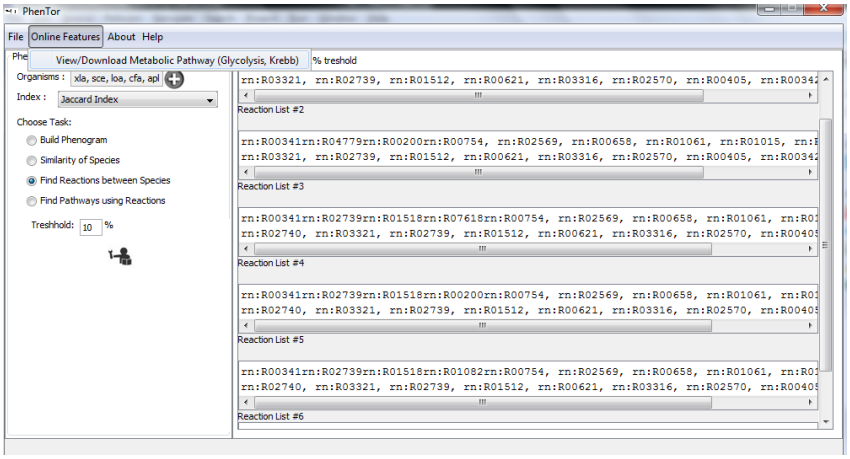


Figure 22: Online

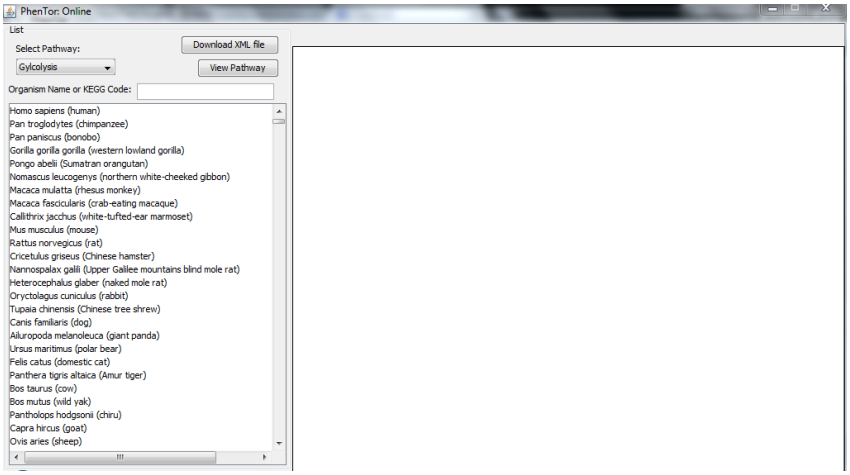


Figure 23: Online Frame

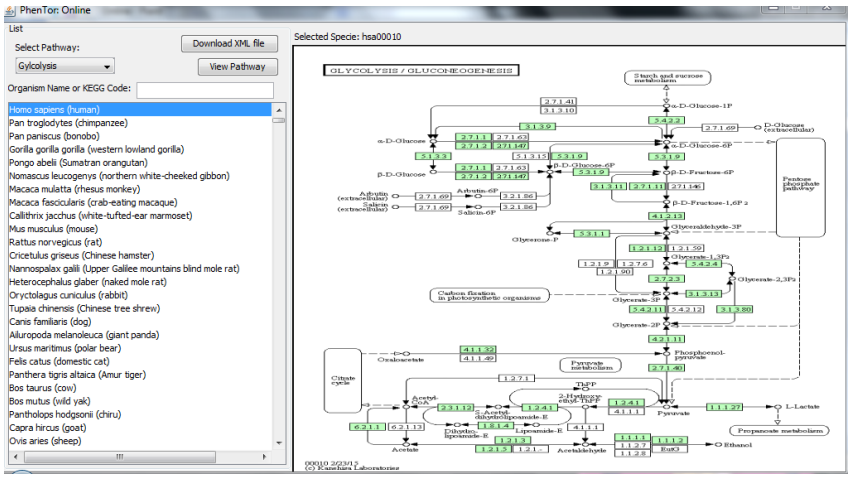


Figure 24: Viewing of a Metabolic Pathway

The user can also download XML Files of glycolysis and citrate cycle metabolic pathway from KEGG.

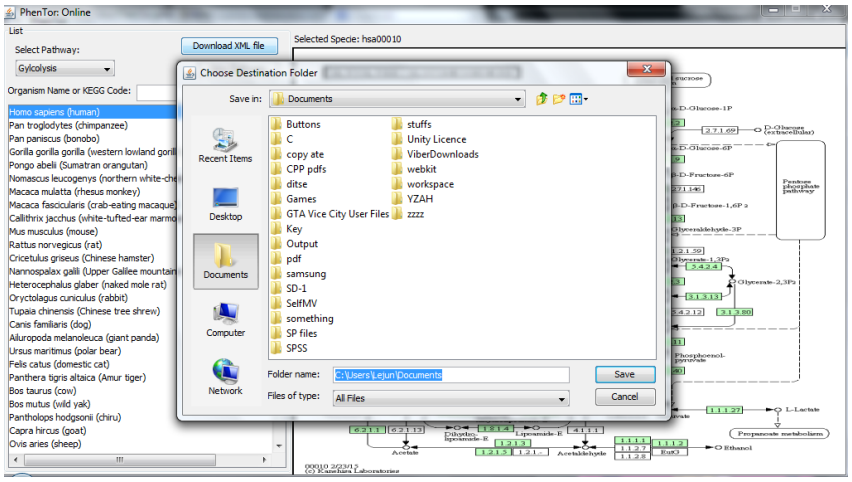


Figure 25: Donwloading File

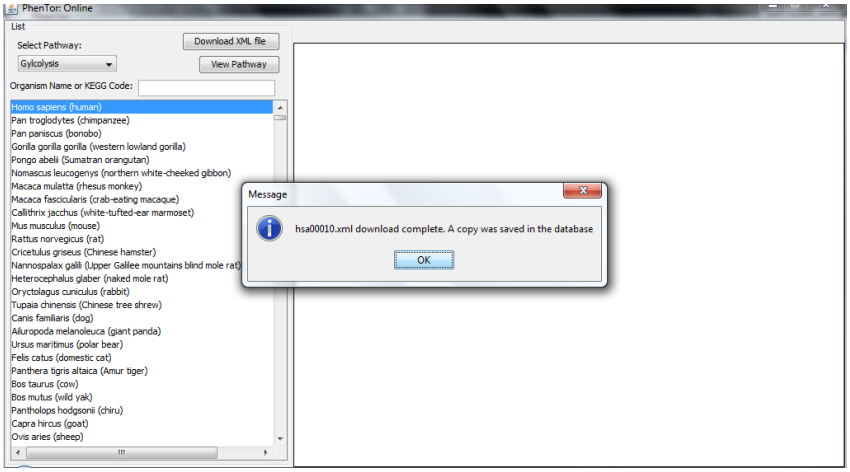


Figure 26: Download Complete

The application will notify the user if there is no internet connection before using the online feature

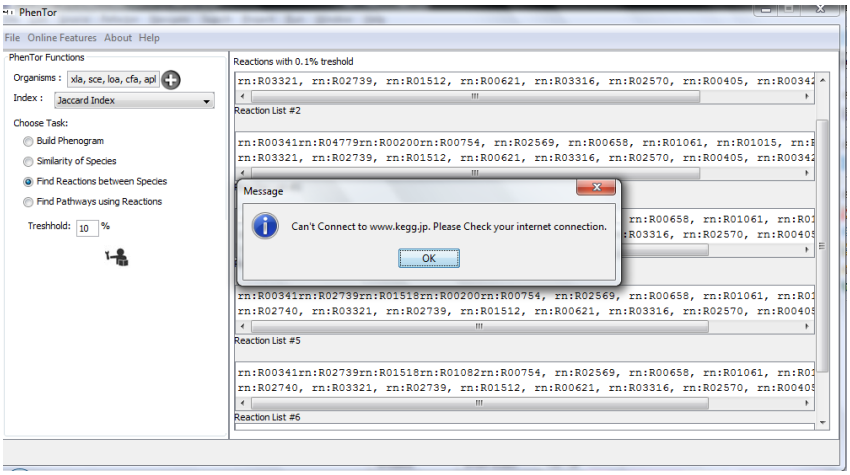


Figure 27: No Internet Connection

# VII. Discussions

The Phenogram Generator (PhenTor) is a stand-alone application that aims to classify organisms using their Glycolysis and Citrate Cycle metabolic pathways. It lets the user to select the dataset and the index to be used by the application.

The user can select from the following functionalities: (1) build a Phenogram, (2) get tthe similarities between species across the dataset, (3) trace the frequent reactions across the dataset, and (4) trace the organisms having certain set of reactions.

In building a phenogram, the xml files of glycolysis and citrate cycle metabolic pathways of the data input from the user will be converted into graphs. Using FP-GraphMiner, the application will get the FP-Table from the set of graphs generated from the data input. After that, using the chosen index by the user, the application will summarize the FP-Table into a 2D-Array similarity matrix. Where SimilarityMatrix[i,j] is equal to similarity between Specie i and j. Using the Graph Similarity Tree (GST) Algorithm, the species will then be classified using Similarity Matrix. The group generated from GST will then be compared from NCBI Taxonomy using PHYLIP's TreeDist program and Majority Rule Consensus Tree (MCT) Algorithm. The Phenogram will be displayed as image using PHYLIP's DrawGram application.

In getting the similarites between species, it will follow the same steps of building a phenogram, from reading the xml files of glycolysis and citrate cycle metabolic pathways to summarizing FP-table into Similarity Matrix. The Similarity Matrix is the matrix of similarities between species.

In Tracing the frequent reactions across the dataset and tracing the organisms having certain set of reactions, they will follow the FP-Graph Miner for tracing graphs using subgraphs, and frequent structure mining.

The Application is written in Java. PHYLIP's TreeDist program was implemented using Runtime.getRuntime() function of Java.

The user can also use the online feature of PhenTor, which are viewing images of glycolysis and citrate cycle metabolic pathways, and downloading xml files of the glycolysis and citrate cycle metabolic pathways.

The main issue encountered during the development of the algorithm to be used in the application. Using the defined algorithm, series of tests is done to check if the algorithm works the same with the theory behind it.

Another issue was getting the data from KEGG, which is must be downloaded manually. KEGG offers many metabolic pathways, including Glycolysis and Citrate Cycle, for over 3000 species.

# VIII. Conclusions

PhenTor has been developed to allow the users to download the software freely. It is made available to public so that, System Biologist can easily generate Phenogram using Jaccard index or Hamming Distance.

Using frequent subgraph mining and similarity index, relationship between graphs can easily be obtained.

The average accuracy of the system based on the average of all tests, 50 sets of 5 species, is 94 %. The total mismatches is 47, over the total branches, 374. This accuracy is based on the comparison of the Phenogram generated using the defined algorithm, with the NCBI Taxonomy.

| Test Data | % match | | | |
|---|---|---|---|---|
| | Phylip - JSI | Phylip - HD | MCT - JSI | MCT - HD |
| aaa, abb, acc, adi, adk | 100 | 100 | 89 | 89 |
| afi, afl, afn, afo, aga | 57 | 86 | 67 | 67 |
| afo, ahy, apa, pps, vsa | 86 | 86 | 78 | 78 |
| aap, aav, abad, abaz, abj | 86 | 86 | 78 | 78 |
| aho, ami, amr, asg, ash | 57 | 57 | 67 | 67 |

Figure 28: Sample Groups with Phylip and MCT Testing

# IX. Recommendations

PhenTor can be improved by cutting the running time of the algorithms used by the application. There are some researches regarding cutting of BFS, which is one of the algorithms used in the application.

The implementation of PHYLIP's TreeDist program was made possible by using Runtime function. Implementing it using the source code given by PHYLIP might make the application more platform independent.

Saving the current work of the user will be helpful when the application closed accidentally.

# X. Bibliography


[1] M. Heymans and A. Singh, "Deriving phylogenetic trees from similarity analysis of metabolic pathways," *Bioinformatics - Oxford University Press*, vol. 19:1, pp. i138 – i146, 2003.

[2] L. Liao, S. Kim, and J. Tomb, "Genome comparison based on profiles of metabolic pathways," *Sixth International Conference on Knowledge-Based Intelligent Information Engineering Systems (KES 2002).*

[3] G. Solano, E. G. III, M. C. Carillo, J. Clemente, and H. Adorna, "Building phylogenetic trees from frequent subgraph mining techniques on reaction hypergraphs," *Proceedings of the 5th IEEE International Conference on Information, Intelligence, Systems, and Applications*, 2014.

[4] M. Kanehisa, S. Goto, Y. Sato, M. Furumichi, and M. Tanabe, "Kegg for the integration and interpretation of large-scale molecular datasets," *Nucleic Acids Research - Oxford University Press*, 2011.

[5] P. Weston and M. D. Crisp, "Introduction to phylogenetic analysis," **INTERNET**, 1995.

[6] F. S. L. Brinkman and D. D. Leipe, "Phylogenetic analysis," *Bioinformatics: A Practical Guide to the Analysis of Genes and Proteins, 2nd Edition*, 2001.

[7] M. Schmitt, "Willi hennig, the cautious revolutioniser," *Willi Hennig Symposium on Phylogenetics and Evolution*, 2009.

[8] J. Felsenstein, "Phylogenies from molecular sequences," *Anuual Rev. Genet*, vol. 22, pp. 521–565, 1998.

[9] C. Frost and K. Schulten, "Phylogenetic analysis of metabolic pathways," *Journal of Molecular Evolution*, vol. 52, pp. 471–489, 2001.

[10] B. Maidak, J. Cole, T. Lilburn, C. Parker, P. Saxman, R. Farris, G. Garrity, G. Olsen, T. Schmidt, and J. Tiedje, "The rdp - ii (ribosomal database project)," *Nucleic Acids Research*, vol. 29, pp. 173–174, 2001.

[11] M. de Buyser, A. Morvan, S. Aubert, F. Dilasser, and N. E. Solh, "Evaluation of a ribosomal rna gene probe for the identification of species and subspecies within the genus staphylococcus," *J.Gen. Microbiol.*, vol. 138, pp. 889–899, 1992.

[12] S. Fitz-Gibbon and C. House, "Whole genome-based phylogenetic analysis of free-living microorganisms," *Nucleic Acids Research*, vol. 22, pp. 4218 – 4222, 1999.

[13] M. Li, J. H. Badger, X. Chen, S. Kwong, P. Kearney, and H. Zhang, "An information-based sequence distance and its application to whole mitochondrial genome phylogeny," *Bioinformatics*, vol. 17, pp. 149–154, 2001.

[14] S. Schuster, D. Fell, and T. Dandekar, "A general definition of metabolic pathways useful for systematic organization and analysis of complex metabolic networks," *Nature Biotechnology*, vol. 18, 2000.

[15] P. Baldan, N. Cocco, and M. Simeoni, "Comparison of metabolic pathways by considering potential fluxes," *Proceedings of the Petri Net Conference*, 2012.

[16] M. Pellegrini, "Using phylogenetic profiles to predict functional relationships,"

[17] R. Caspi, H. Foerster, C. Fulcher, R. Hopkinson, J. Ingraham, P. Kalpa, M. Krummenacker, S. Paley, J. Pick, S. Rhee, C. Tissier, P. Zhang, and P. Karp, "Metacyc : a multiorganism database of metabolic pathways and enzymes," *Nucleic Acid Research*, vol. 34, 2006.

[18] H. Dinari and H. Naderi, "A survey of frequent subgraphs and subtree mining methods," *International Journal of Computer Science and Business Informatics*, vol. 14, 2014.

[19] Yan, Xifeng, and J. Han, "gspan: Graph-based substructure pattern mining," *Proceedings International Conference on Data Mining, IEEE*, pp. 721–724, 2002.

[20] Thomas, L. T, S. Valluri, and K. Karlapalem, "Margin: Maximal frequent subgraph mining," *IEEE Sixth International Conference on Data Mining (ICDM)*, pp. 1097–1101, 2006.

[21] J. Wu and L. Chen, "A fast frequent subgraph mining algorithm," *9th International Conference for Young Computer Scientist, IEEE*, pp. 82–87, 2008.

[22] R. Vijayalakshmi, R. Nadarajan, J. Roddick, and M. Thilaga, "Fp-graphminer, a fast frequent patterm mining algorithm for network graphs," *Journal of Graphs Algorithms and Applications*, vol. 15, pp. 753–776, 2011.

[23] S. E. Schaeffer, "Graph algorithm," *Computer Science Review 1*, pp. 27–64, 2007.

[24] J. Raymond, C. Bankley, and P.Willet, "Comparison of chemical clustering methods using graph- and fingerprint-based similarity measures," *Journal of Molecular Graphics and Modeling*, pp. 421–433, 2003.

[25] P. Jaccard, "The distribution of the flora in the alpine zone," *Bulletin del la Socit Vaudoisedes Sciences Naturelles*, vol. 37, pp. 241–272, 1901.

[26] G. Solano, L. Osorio, M. C. Carillo, and H. Adorna, "Using jaccard index and hamming distance in making graph similarity tree," *unpublished*, 2015.

[27] J. Jansson, C. Shen, and W.-K. Sung, "An optimal algorithm for building the majority rule consensus tree," *RECOMB 2013, LNBI 7821*, pp. 88–99, 2013.

# XI. Appendix

## A. Source Code

### 1. Controller

1. BuildPhenogramGroup.java

```
package com.Controller;
import java.util.ArrayList;

import com.Model.Graph;


public class BuildPhenogramGroup {

        private final static int O = -1;
        private final static int C = -2;

        ArrayList<Graph> G;
        ArrayList<String> pathwayList;
        double[][] SimilarityMatrix;
        double max;
        int index1 , highestIndex;
        int fpsizeR, fpsizeC;

        int index;

        ArrayList<Integer> group;

        public BuildPhenogramGroup(ArrayList<Graph> G){
                this.G = G;
        }

        public BuildPhenogramGroup(ArrayList<Graph> G, int index){
                this.G = G;
                this.index = index;
        }

        public BuildPhenogramGroup(double[][]
                        SimilarityMatrix, ArrayList<Graph> G){
                fpsizeC = G.size();
                this.SimilarityMatrix = SimilarityMatrix;
                this.G = G;
        }

    public void FPGraph(){
        fpsizeC = G.size();

        FPGraphMiner fpg = new FPGraphMiner(G);
        fpg.FPTable();
        SimilarityMatrix = (new SimilarityMatrix(
                        fpg.FPTable, fpg.counts, G)).getSimilarityMatrix(index);

        generateGroup();
    }

    public void generateGroup(){
                simulate();
        }

    public ArrayList<Integer> simulate(){
        group = new ArrayList<Integer>();
        if(G.size() == 1){
                //only one graph to group
                group.add(O);
                group.add(1);
                group.add(C);
                return group;
        }

        group.add(O);
                group.add(1);
                group.add(2);
                group.add(C);

                for(int i = 2 ; i < fpsizeC ; i++){
                        getMax(i);
                        int ind1 = getInd1(highestIndex);
                        int ind2 = getInd2(highestIndex);
                        int n = 0;
                        for(int j = ind1 ; j < ind2 ; j++){
                                if(group.get(j) != highestIndex
                                                && group.get(j) != O && group.get(j) != C){
                                        n = group.get(j);
                                        break;
                                }
                        }

                        if(max > SimilarityMatrix[n-1][highestIndex-1]){
                                add(O, group.indexOf(highestIndex));
                                add((i+1), group.indexOf(highestIndex) + 1);
```

```
                                add(C, group.indexOf(i+1) + 1);
                        }else if(max < SimilarityMatrix[n-1][highestIndex-1]){
                                boolean found = false;
                                while(!found){
                                        int prev = ind1;
                                        int countC = 0;
                                        for(int j = ind1-1 ; j >=0 ; j--){
                                                if(group.get(j) == C)
                                                countC++;
                                        if(group.get(j) == O){
                                                countC--;
                                                if(countC < 0){
                                                        ind1 = j;
                                                        break;
                                                }
                                        }
                                        }

                                        int prev2 = ind2;
                                        int countO = 0;
                                        for(int j = ind2+1 ; j < group.size() ; j++){
                                                if(group.get(j) == O)
                                                        countO++;
                                                if(group.get(j) == C){
                                                        countO--;
                                                        if(countO < 0){
                                                                ind2 = j;
                                                                break;
                                                        }
                                                }
                                        }

                                        int tn = -1 ;
                                        if(ind1 == prev && ind2 == prev2){
                                                add(O, 1);
                                                group.add((i+1));
                                                group.add(C);
                                                found = true;
                                                break;
                                        }

                                        if(prev-1 == ind1){//check right;
                                                tn = getMaxG(prev2, ind2);
                                        }else if(prev2+1 == ind2){
                                                tn = getMaxG(ind1, prev-1);
                                        }

                                        if(tn <= 0){
                                                add(O, 1);
                                                group.add((i+1));
                                                group.add(C);
                                                found = true;
                                                break;
                                        } else if(max > SimilarityMatrix[tn-1][highestIndex-1]){
                                                add(O, prev);
                                                add((i+1), prev2 + 2);
                                                add(C, group.indexOf(i+1) + 1);
                                                found = true;
                                        }
                                }
                        }else{
                                add((i+1), group.indexOf(highestIndex) +1);
                        }
                }

        return group;

}

public int getMaxG(int ind1, int ind2){
        int n = -1;
        for(int i = ind1 + 1; i <= ind2 ; i++){
                if(group.get(i) == O || group.get(i) == C)
                        continue;
                if(n < SimilarityMatrix[group.get(i) -1][highestIndex-1]){
                        n = group.get(i);
                }
        }
        return n;
}

public int getInd1(int num){
        int index = group.indexOf(num);
        int countC = 0;
        for(int i = index-1 ; i >=0 ; i--){
                if(group.get(i) == -2)
                        countC++;
                if(group.get(i) == -1){
                        countC--;
                        if(countC < 0)
                                return i;
                }
        }
        return -1;
}

public void add(int num, int index){

        ArrayList<Integer> temp = new ArrayList<Integer>();
```

```
    for(int i = 0; i < group.size() ; i++){
                    if(i == index)
                            temp.add(num);
                    temp.add(group.get(i));
            }

            group = temp;

    }

public int getInd2(int num){
    int index = group.indexOf(num);
    int countO = 0;
    for(int i = index+1 ; i < group.size() ; i++){
            if(group.get(i) == -1)
                    countO++;
            if(group.get(i) == -2){
                    countO--;
                    if(countO < 0)
                            return i;
            }
    }
    return -1;
}

public void getMax(int num){
    max = 0;
    for(int i = 0 ; i < num ; i++){

            double n = 0;
            n = SimilarityMatrix[num][i];

            if(n > max){
                    max = n;
                    highestIndex = i+1;
            }
    }
}

public String getGroup(){
    String str = "";

    for(int i = 0 ; i < group.size() ; i++){
            if(group.get(i) == O){
                    if((i-1) > 0 && ((group.get(i -1)
                                    != C && group.get(i -1) != O) ||
                                    group.get(i-1) == C))
                            str += ", ";
                    str += "(";
            }else if(group.get(i) == C)
                    str += ")";
            else{
                    if(group.get(i -1) == C || (group.get(i -1)
                                    != C && group.get(i -1) != O))
                            str+= ", ";
                    str += group.get(i) +"";
            }
    }

    return str;
}

public String getGroup_(){
    String str = "";

    for(int i = 0 ; i < group.size() ; i++){
            if(group.get(i) == O){
                    if((i-1) > 0 && ((group.get(i -1) != C
                                    && group.get(i -1) != O) ||
                                    group.get(i-1) == C))
                            str += ", ";
                    str += "(";
            }else if(group.get(i) == C)
                    str += ")";
            else{
                    if(group.get(i -1) == C || (group.get(i -1)
                                    != C && group.get(i -1) != O))
                            str+= ", ";
                    str += G.get(group.get(i)-1).getName() +"";
            }
    }

    return str;
}

public String getGroupV(){
    String str = "";

    for(int i = 0 ; i < group.size() ; i++){
            if(group.get(i) == O){
                    if((i-1) > 0 && ((group.get(i -1) != C
                                    && group.get(i -1) != O) ||
                                    group.get(i-1) == C))
                            str += ", ";
                    str += "(";
            }else if(group.get(i) == C)
                    str += ")";
            else{
                    if(group.get(i -1) == C || (group.get(i -1) != C
```

```
                                        && group.get(i -1) != O))
                                str+= ", ";
                        String temp = G.get(group.get(i)-1).getName();
                        temp = GUIHandler.options.get(0).get(
                                        GUIHandler.options.get(1).indexOf(temp));
                        String[] stra = temp.split("\\(");
                                temp = stra[0].trim();
                        if(stra.length > 1){
                                temp += " - " + stra[1].substring(0, stra[1].length() - 1);
                        }
                                str += temp;
                }
        }

        return str;
    }

    public double[][] getSimilarity(){
        return SimilarityMatrix;
    }
}
```

2. Comparison.java

```
package com.Controller;

import java.util.ArrayList;

public class Comparison {

        ArrayList<ArrayList<Integer>> clusterAlg;
        ArrayList<ArrayList<Integer>> clusterNCBI;

        public Comparison(){

        }

        public double MCTCompare(String s1, String s2){
                clusterAlg = getCluster(s1);
                clusterNCBI = getCluster(s2);
                int count = countEqual(clusterAlg, clusterNCBI);

                double similarity = (double) count/clusterNCBI.size();
                similarity = round(similarity);
                return similarity*100;
        }

        public ArrayList<ArrayList<Integer>> getCluster(String s){
                ArrayList<ArrayList<Integer>> tempCluster =
                                new ArrayList<ArrayList<Integer>>();
                ArrayList<ArrayList<Integer>> cluster =
                                new ArrayList<ArrayList<Integer>>();
                for(int i = 0 ; i < s.length() ; i++){
                        if(s.charAt(i) == '('){
                                tempCluster.add(new ArrayList<Integer>());
                        }else if(s.charAt(i) == ',' || s.charAt(i) == ' '){
                                continue;
                        }else if(s.charAt(i) == ')'){
                                ArrayList<Integer> c = tempCluster.get(tempCluster.size() - 1);
                                tempCluster.remove(tempCluster.size() - 1);
                                c = sortList(c);
                                cluster.add(c);
                        }else{//number
                                String str = s.charAt(i) + "";
                                i++;
                                while(s.charAt(i) != ' ' && s.charAt(i) != ','
                                                && s.charAt(i) != '(' && s.charAt(i) != ')'){
                                        str += s.charAt(i);
                                        i++;
                                }
                                i--;
                                int num = Integer.parseInt(str);
                                for(ArrayList<Integer> aai : tempCluster)
                                        aai.add(num);
                                ArrayList<Integer> temp = new ArrayList<Integer>();
                                temp.add(num);
                                cluster.add(temp);
                        }
                }

                return cluster;
        }

        public ArrayList<Integer> sortList(ArrayList<Integer> list){
                for(int i = 0 ; i < list.size() - 1 ; i++){
                        for(int j = i+1 ; j < list.size() ; j++){
                                if(list.get(j) < list.get(i)){
                                        int temp = list.get(i);
                                        list.set(i, list.get(j));
                                        list.set(j, temp);
                                }
                        }
                }

                return list;
        }

        public int countEqual(ArrayList<ArrayList<Integer>>
```

```
        clusterA, ArrayList<ArrayList<Integer>> clusterB){
                int count = 0;
                boolean[] found = new boolean[clusterB.size()];

                for(int i = 0 ; i < clusterB.size() ; i++)
                        found[i] = false;

                for(int i = 0 ; i < clusterA.size() ; i++){
                        for(int j = 0 ; j < clusterB.size() ; j++){
                                if(clusterA.get(i).size() == clusterB.get(j).size()){
                                        boolean isthere = true;
                                        for(int k = 0 ; k < clusterA.get(i).size() ; k++){
                                                if(clusterA.get(i).get(k) != clusterB.get(j).get(k)){
                                                        isthere = false;
                                                        break;
                                                }
                                        }

                                        if(isthere)
                                                count++;

                                }
                        }
                }

                return count;
        }

        public double round(double num){
                String s = num + "";

                if(s.length() > 5){ // 0.xyz
                        String s2 = s.charAt(5) + "";
                        int n = Integer.parseInt(s2);
                        if(n >= 5){
                                String s3 = s.substring(2, 5) + "";
                                int n2 = Integer.parseInt(s3);
                                n2++;
                                String s4 = s.substring(0, 2) + (n2 + "");
                                num = Double.parseDouble(s4);
                        }else{
                                String s4 = s.substring(0, 5);
                                num = Double.parseDouble(s4);
                        }
                }

                return num;
        }
}
```

3. CreateTempFile.java

```
package com.Controller;

import java.io.Closeable;
import java.io.File;
import java.io.FileNotFoundException;
import java.io.FileOutputStream;
import java.io.IOException;
import java.io.InputStream;
import java.io.OutputStream;
import java.net.MalformedURLException;
import java.net.URI;
import java.net.URISyntaxException;
import java.net.URL;
import java.security.CodeSource;
import java.security.ProtectionDomain;
import java.util.zip.ZipEntry;
import java.util.zip.ZipException;
import java.util.zip.ZipFile;

public class CreateTempFile {


  public URI convert(String fileP)
                throws URISyntaxException,
                       ZipException,
                       IOException
        {
            final URI uri;
            final URI exe;

          //  uri = getJarDir(com.View.PhenTor.class);

            uri = getJarURI();
            exe = getFile(uri, fileP);

            System.out.println(exe.getPath());
            return exe;
        }

            private static URI getJarURI()
            throws URISyntaxException
        {
            final ProtectionDomain domain;
            final CodeSource       source;
            final URL              url;
            final URI              uri;
```

```
        domain = com.View.PhenTor.class.getProtectionDomain();
        System.out.println(domain.toString());
        source = domain.getCodeSource();
        System.out.println(source.toString());
        url    = source.getLocation();
        System.out.println(url.toString());
        uri    = url.toURI();
        System.out.println(uri.toString());
        return (uri);
    }

        public static URI getJarDir(Class aclass) throws URISyntaxException {
            URL url;
            String extURL;      //  url.toExternalForm();

            // get an url
            try {
                url = aclass.getProtectionDomain().getCodeSource().getLocation();
            } catch (SecurityException ex) {
                url = aclass.getResource(aclass.getSimpleName() + ".class");
            }

            // convert to external form
            extURL = url.toExternalForm();

            // prune for various cases
            if (extURL.endsWith(".jar"))   // from getCodeSource
                extURL = extURL.substring(0, extURL.lastIndexOf("/"));
            else {  // from getResource
                String suffix = "/"+(aclass.getName()).replace(".", "/")+".class";
                extURL = extURL.replace(suffix, "");
                if (extURL.startsWith("jar:") && extURL.endsWith(".jar!"))
                    extURL = extURL.substring(4, extURL.lastIndexOf("/"));
            }

            return url.toURI();
        }

    private static URI getFile(final URI    where,
                               final String fileName)
        throws ZipException,
               IOException
    {
        final File location;
        final URI  fileURI;

        location = new File(where);

        // not in a JAR, just return the path on disk
        if(location.isDirectory())
        {
            fileURI = URI.create(where.toString() + fileName);
        }
        else
        {
            final ZipFile zipFile;

            zipFile = new ZipFile(location);

            try
            {
                fileURI = extract(zipFile, fileName);
            }
            finally
            {
                zipFile.close();
            }
        }

        return (fileURI);
    }

    private static URI extract(final ZipFile zipFile,
                               final String  fileName)
        throws IOException
    {
        final File          tempFile;
        final ZipEntry      entry;
        final InputStream   zipStream;
        OutputStream        fileStream;

        tempFile = File.createTempFile(fileName,
                       Long.toString(System.currentTimeMillis()));
        tempFile.deleteOnExit();
        entry    = zipFile.getEntry(fileName);

        if(entry == null)
        {
            throw new FileNotFoundException("cannot find file: "
        + fileName + " in archive: " + zipFile.getName());
        }

        zipStream  = zipFile.getInputStream(entry);
        fileStream = null;

        try
        {
            final byte[] buf;
```

```
                int           i;

                fileStream = new FileOutputStream(tempFile);
                buf        = new byte[1024];
                i          = 0;

                while((i = zipStream.read(buf)) != -1)
                {
                    fileStream.write(buf, 0, i);
                }
            }
            finally
            {
                close(zipStream);
                close(fileStream);
            }

            return (tempFile.toURI());
        }

        private static void close(final Closeable stream)
        {
            if(stream != null)
            {
                try
                {
                    stream.close();
                }
                catch(final IOException ex)
                {
                    ex.printStackTrace();
                }
            }
        }

}
```

4. DMPParser.java

```
package com.Controller;

import java.io.BufferedReader;
import java.io.BufferedWriter;
import java.io.File;
import java.io.FileReader;
import java.io.FileWriter;
import java.io.IOException;
import java.io.InputStreamReader;
import java.net.URL;
import java.util.Scanner;
import java.util.StringTokenizer;

public class DMPParser {

        Scanner scanner;
        String filePath;
        int index = 0;

        private final int cutter = 1000;

        public DMPParser(){}

        public DMPParser(String filePath){
                this.filePath = filePath;
        }

        public void cutFiles() throws IOException{
                File file = new File(filePath + ".dmp");

                if (!file.exists()) {
                        file.createNewFile();
                } else {
                        System.out.println("Reading " + filePath + " .... ");
                        double start = System.currentTimeMillis()/1000;
                        scanner = new Scanner(new FileReader(file));
                        int current = 1;
                        int ind = 1;
                        String content = "";
                        int counter = 1;
                        double start2 = System.currentTimeMillis()/1000;
                        while (scanner.hasNextLine()) {
                                String nCont = "";
                                String str = scanner.nextLine().trim();
                                StringTokenizer st = new StringTokenizer(str, "|");

                                int count = 0;


                                while(st.hasMoreTokens()){
                                        String s = st.nextToken().trim();
                                        nCont += s;
                                        count++;
                                        if(count == 2)
                                                break;
                                        else if(count == 1){
                                                current = Integer.parseInt(s);
                                                nCont += " ";
                                        }
```

```
                        }
                        if(current >= cutter*counter || !scanner.hasNextLine()){
                                if(current == cutter*counter || !scanner.hasNextLine()){
                                        content += nCont;
                                }

                                File file2 = new File("dmp/"
                                + filePath + "/" + filePath + ind + ".txt");

                                // if file doesnt exists, then create it
                                if (!file2.exists()) {
                                        file2.createNewFile();
                                }

                                FileWriter fw = new FileWriter(file2.getAbsoluteFile());
                                BufferedWriter bw = new BufferedWriter(fw);
                                bw.write(content);
                                bw.close();
                                double end2 = System.currentTimeMillis()/1000;
                                System.out.println(filePath
                                                + ind + " created! " + (end2 - start2) + " second('s)");
                                ind++;
                                content = "";

                                if(current > cutter*counter)
                                        content = nCont += "\n";
                                counter++;

                                System.out.println("Generating " + filePath + ind + " ....");
                                start2 = System.currentTimeMillis()/1000;
                        } else {
                                content += nCont;
                                content += "\n";
                        }
                }
                double end = System.currentTimeMillis()/1000;
        }
}

public int getId(String name) throws IOException{
        int id = -1;

        URL url = getClass().getResource("/dmp/names.dmp");
    BufferedReader in = new BufferedReader(
                new InputStreamReader(url.openStream()));

    String inputLine;
    while ((inputLine = in.readLine()) != null){
        String str = inputLine;

                StringTokenizer st = new StringTokenizer(str, "|");

                int count = 0;
                String name_ = "";
                while(st.hasMoreTokens()){
                        String s = st.nextToken().trim();
                        if(count == 0)
                                id = Integer.parseInt(s);
                        if(count == 1){
                                name_ = s;
                                break;
                        }
                        count++;
                }

                if(name_.equalsIgnoreCase(name))
                        return id;

    }
    in.close();

        return id;
}

public int getParent(int id) throws IOException{
        int counter = id/cutter;
        if(id % cutter != 0)
                counter++;

        URL url = getClass().getResource("/dmp/nodes/nodes"+counter+".txt");
    BufferedReader in = new BufferedReader(
                new InputStreamReader(url.openStream()));

    String inputLine;
    while ((inputLine = in.readLine()) != null){
        String str = inputLine;
        String[] cont = str.split(" ");
                if(id == Integer.parseInt(cont[0])){
                        return Integer.parseInt(cont[1]);
                }
    }

        return -1;
}

public int getParent_(int id) throws IOException{

        File file = new File("nodes.dmp");
```

```
                if (!file.exists()) {
                        file.createNewFile();
                } else {
                        scanner = new Scanner(new FileReader(file));
                        while (scanner.hasNextLine()) {
                                String str = scanner.nextLine().trim();
                                StringTokenizer st = new StringTokenizer(str, "|");

                                int count = 0;
                                boolean found = false;
                                while(st.hasMoreTokens()){
                                        String s = st.nextToken().trim();
                                        if(found && count == 1){
                                                return Integer.parseInt(s);
                                        }else if(count == 0){
                                                int id_ = Integer.parseInt(s);
                                                if(id == id_)
                                                        found = true;
                                                else
                                                        break;
                                        }

                                        count++;
                                }
                        }
                }
                return -1;
        }
}
```

5. FPGraphMiner.java

```
package com.Controller;

import java.util.ArrayList;

import com.Model.Edge;
import com.Model.Graph;


public class FPGraphMiner {

        int[][] FPTable_;
        int[][] adjMatrix;

        ArrayList<Graph> G;
        ArrayList<Edge> eList;
        ArrayList<int[]> FPTable;
        ArrayList<Integer> counts;
        ArrayList<ArrayList<Integer>> clusters;
        ArrayList<int[]> indices;
        ArrayList<ArrayList<Integer>> dfs;
        ArrayList<Integer> temp;
        ArrayList<ArrayList<Integer>> temp_;
        ArrayList<Integer> bfsIndex;

        int[] bfsArray;
        int[] edgeIndex;
        int index1 , index2;
        int fpsizeR, fpsizeC;

        public FPGraphMiner(ArrayList<Graph> G){
                this.G = G;
        }

        public void getEdgeList(){
        eList = new ArrayList<Edge>();

        eList.addAll(G.get(0).geteList());
        for(int i = 1 ; i < G.size() ; i++)
                for(Edge e : G.get(i).geteList())
                        if(!containEdge(eList, e))
                                eList.add(e);

        edgeIndex = new int[eList.size()];
        for(int i = 0 ; i < eList.size() ; i++)
                edgeIndex[i] = i;
    }

        public boolean containEdge(ArrayList<Edge> eList, Edge e){
        for(Edge e1 : eList){
                if(e1.isEqual(e, false))
                        return true;
        }
        return false;
    }

        public void FPTable(){
                getEdgeList();
                getFPTable_();
                sortFPTable_();
                groupFPTable_();
        }

        public ArrayList<int[]> getFPTable(){
                return FPTable;
        }
```

```
    public void getFPTable_(){
    fpsizeR = eList.size();
    fpsizeC = G.size();
    FPTable_ = new int[fpsizeR][fpsizeC];

    for(int i = 0 ; i < fpsizeR ; i++){
            for(int j = 0 ; j < fpsizeC ; j++){
                    if(containEdge(G.get(j).geteList(), eList.get(i))){
                            FPTable_[i][j] = 1;
                    }else{
                            FPTable_[i][j] = 0;
                    }
            }
    }
}

public void sortFPTable_(){
    for(int i = 0 ; i < fpsizeR - 1 ; i++){
            for(int j = i + 1 ; j < fpsizeR ; j++){
                    if(countOne(FPTable_[i]) < countOne(FPTable_[j])){
                            /*int temp = edgeIndex[i];
                            edgeIndex[i] = edgeIndex[j];
                            edgeIndex[j] = temp;
                            */
                            Edge etemp = eList.get(i);
                            eList.set(i, eList.get(j));
                            eList.set(j, etemp);

                            for(int k = 0 ; k < fpsizeC ; k++){
                                    int temp = FPTable_[i][k];
                                    FPTable_[i][k] = FPTable_[j][k];
                                    FPTable_[j][k] = temp;

                            }

                    }
                    else if (countOne(FPTable_[i]) == countOne(FPTable_[j])){
                            int num1 = 0;
                            for(int k = 0 ; k < fpsizeC ; k++){
                                    num1 += Math.pow(2, k)*FPTable_[i][fpsizeC - 1 - k];
                            }
                            int num2 = 0;
                            for(int k = 0 ; k < fpsizeC ; k++){
                                    num2 += Math.pow(2, k)*FPTable_[j][fpsizeC - 1 - k];
                            }

                            if(num1 < num2){
                                    /*int temp = edgeIndex[i];
                                    edgeIndex[i] = edgeIndex[j];
                                    edgeIndex[j] = temp;
                                    */
                                    Edge etemp = eList.get(i);
                                    eList.set(i, eList.get(j));
                                    eList.set(j, etemp);

                                    for(int k = 0 ; k < fpsizeC ; k++){
                                            int temp = FPTable_[i][k];
                                            FPTable_[i][k] = FPTable_[j][k];
                                            FPTable_[j][k] = temp;
                                    }
                            }
                    }
            }
    }
}

public int countOne(int[] arr){
    int count = 0;

    for(int i = 0 ; i < arr.length ; i++)
            count += arr[i];

    return count;
}

public void groupFPTable_(){
    FPTable = new ArrayList<int[]>();
    FPTable.add(FPTable_[0]);

    indices = new ArrayList<int[]>();
    int[] ind_ = new int[2];
    ind_[0] = 0;

    counts = new ArrayList<Integer>();
    counts.add(1);

    for(int i = 1 ; i < fpsizeR ; i++){
            if(!isEqualArray(FPTable.get(FPTable.size() - 1), FPTable_[i])){
                    ind_[1] = i-1;
                    indices.add(ind_);
                    ind_ = new int[2];
                    ind_[0] = i;
                    FPTable.add(FPTable_[i]);

                    counts.add(1);
        }else{
            int pos = counts.size() -1;
            int curr = counts.get(pos);
            curr++;
```

```
                        counts.set(pos, curr);
                }
        }
        ind_[1] = fpsizeR - 1;
        indices.add(ind_);
}

public boolean isEqualArray(int[] arr1, int[] arr2){
        if(arr1.length != arr2.length)
                return false;

        for(int i = 0 ; i < arr1.length ; i++){
                if(arr1[i] != arr2[i])
                        return false;
        }

        return true;
}

public void frequentSubgraphs(double treshold){

        clusters();

        int index = 0;
        if(treshold >= .50){
                for(int i = 0 ; i < clusters.size() ; i++){
                        int parity = countOne(FPTable.get(clusters.get(i).get(0)));
                        double alpha = (double) parity/ G.size();
                        if(alpha >= treshold)
                                index = i;
                        else
                                break;
                }
        }else{
                for(int i = clusters.size()/2 ; i < clusters.size() ; i++){
                        int parity = countOne(FPTable.get(clusters.get(i).get(0)));
                        double alpha = (double) parity/ G.size();
                        if(alpha >= treshold)
                                index = i;
                        else
                                break;
                }
        }

        adjacencyMatrix();
        performDFS(index);
}

public void performDFS(int index){
        dfs = new ArrayList<ArrayList<Integer>>();
        temp_ = new ArrayList<ArrayList<Integer>>();
        for(int i = 0 ; i < clusters.get(index).size() ; i++){
                temp = new ArrayList<Integer>();
                dfs(clusters.get(index).get(i));
        }
}

public void dfs(int index){
        temp.add(index);
        int[] arr = adjMatrix[index];
        boolean foundOne = false;
        for(int i = 0 ; i < index ; i++){
                if(arr[i] == 1){
                        foundOne = true;
                        ArrayList<Integer> te = new ArrayList<Integer>();
                        for(int j = 0 ; j < temp.size(); j++){
                                te.add(temp.get(j));
                        }
                                temp_.add(te);
                        dfs(i);
                }
        }

        if(!foundOne)
                dfs.add(temp);

        temp = new ArrayList<Integer>();
        if(temp_.size() != 0){
                temp.addAll(temp_.get(temp_.size() - 1));
                temp_.remove(temp_.get(temp_.size() - 1));
        }
}

public void clusters(){
        clusters = new ArrayList<ArrayList<Integer>>();
        int currentParity = countOne(FPTable.get(0));
        ArrayList<Integer> temp = new ArrayList<Integer>();
        temp.add(0);
        for(int i = 1 ; i < FPTable.size() ; i++){
                int parity = countOne(FPTable.get(i));
                if(parity != currentParity){
                        currentParity = parity;
                        clusters.add(temp);
                        temp = new ArrayList<Integer>();
                }
                temp.add(i);
        }

        clusters.add(temp);
```

```
    }

    public void adjacencyMatrix(){
        int size = FPTable.size();
        adjMatrix = new int[size][size];

        for(int i = 0 ; i < size ; i++){
                for(int j = 0 ; j < size ; j++){

                        if(i == j){
                                adjMatrix[i][j] = 0;
                                continue;
                        }else if(i > j){
                                continue;
                        }

                        if(isConnected(FPTable.get(i), FPTable.get(j))){
                                adjMatrix[i][j] = 1;
                                adjMatrix[j][i] = 1;

                                for(int k = 0 ; k < i ; k++){
                                        if(adjMatrix[i][k] == 1 && adjMatrix[k][j] == 1){
                                                adjMatrix[k][j] = 0;
                                                adjMatrix[j][k] = 0;
                                        }
                                }
                        }
                }
        }
    }

    public boolean isConnected(int[] arr1, int[] arr2){
        for(int i = 0 ; i < arr2.length ; i++){
                if(arr2[i] == 1 && arr1[i] == 0)
                        return false;
        }
        return true;
    }

    public void patternFinder(Graph subG){
        bfsIndex = new ArrayList<Integer>();

        ArrayList<Edge> edges = new ArrayList<Edge>();
        edges.addAll(subG.geteList());

        bfsArray = new int[G.size()];
        for(int i = 0 ; i < G.size() ; i++){
                bfsArray[i] = 0;
        }

        for(int i = 0 ; i < subG.geteList().size() ; i++){
                int index = getIndex(subG.geteList().get(i));
                for(int j = 0 ; j < indices.size() ; j++){
                        if(indices.get(j)[0] <= index && indices.get(j)[1] >= index){
                                if(!bfsIndex.contains(j)){
                                        bfsIndex.add(j);
                                        if(bfsIndex.size() == 1)
                                                bfsArray = FPTable.get(j);
                                        else
                                                for(int k = 0 ; k < FPTable.get(j).length ; k++){
                                                        if(bfsArray[k] == 1 && FPTable.get(j)[k] == 1)
                                                                bfsArray[k] = 1;
                                                        else
                                                                bfsArray[k] = 0;
                                                }
                                }
                        }
                }
        }
    }

    public int getIndex(Edge e){
        for(int i = 0 ; i < eList.size() ; i++){
                if(eList.get(i).isEqual(e, false)){
                        return i;
                }
        }
        return -1;
    }
}
```

6. GUIHandler.java

```
package com.Controller;

import java.io.BufferedReader;
import java.io.BufferedWriter;
import java.io.File;
import java.io.FileWriter;
import java.io.IOException;
import java.io.InputStreamReader;
import java.net.URL;
import java.util.ArrayList;
import java.util.StringTokenizer;

import javax.swing.JOptionPane;
import javax.swing.JProgressBar;
```

```
import com.Model.Edge;
import com.Model.Graph;
import com.Model.Specie;
import com.PHYLIP.DrawGram.DrawgramInterface;
import com.PHYLIP.Util.DrawPreview;

public class GUIHandler {

        private final int build = 1;
        private final int fs = 2;
        private final int fp = 3;
        private final int gs = 4;

        public static ArrayList<ArrayList<String>> options;
        int index;
        public double treshold;
        String fileSub;
        JProgressBar progressBar;

        //gsr variables
        public double[][] gsrSimMatrix;
        //build variables
        public String outGroup, ncbiGroup, phylipComp, mctComp;
        String[] data_;
        //fpr variables
        public Graph subG;
        public int[] speSub;
        //fsr variables
        public ArrayList<ArrayList<Integer>> dfs;
        public ArrayList<int[]> indices;
        public ArrayList<Edge> eList;
        public int[] edgeIndex;
        ArrayList<int[]> fptable;
        public GUIHandler(int index, JProgressBar progressBar){
                this.index = index;
                this.progressBar = progressBar;
        }

        public GUIHandler(){}

        public GUIHandler(int index){
                this.index = index;
        }

        public GUIHandler(int index, JProgressBar progressBar, String fileSub){
                this.index = index;
                this.progressBar = progressBar;
                this.fileSub = fileSub;
        }

        public GUIHandler(int index, JProgressBar progressBar, double treshold){
                this.index = index;
                this.progressBar = progressBar;
                this.treshold = treshold;
        }

        public ArrayList<ArrayList<String>> getOptions(){
                ArrayList<Specie> species = (new JSONParser_()).ParseJSON();

                options = new ArrayList<ArrayList<String>>();
                ArrayList<String> content = new ArrayList<String>();
                ArrayList<String> value = new ArrayList<String>();
                ArrayList<String> value_small = new ArrayList<String>();

                for(Specie s : species){
                        content.add(s.getContent());
                        value.add(s.getValue());

                        StringTokenizer st = new StringTokenizer(s.getContent(), "(");

                        while(st.hasMoreTokens()){
                                String s2 = st.nextToken().trim();
                                value_small.add(s2.toLowerCase());
                                break;
                        }
                }

                options.add(content);
                options.add(value);
                options.add(value_small);

                return options;
        }

        public String SimulateSelected
        (String[] data, int choice) throws Exception{
                String str = "";
                progressBar.setString
                ("00% completed - Reading files...");
                ArrayList<Graph> G = new ArrayList<Graph>();
                for(int i = 0; i < data.length; i++){
                        String filePath =
                                        "/files/glyco/"+data[i]+"00010.xml";

                        try{
                                URL kgml = getClass().
                                                getResource("/files/glyco/"+data[i]+"00010.xml");
                                InputStreamReader isr
                                = new InputStreamReader(kgml.openStream());
```

```
                BufferedReader in = new BufferedReader(isr);
        } catch (Exception e){
                e.printStackTrace();

                int dload = JOptionPane.
                                showConfirmDialog(null, "file not found: "
                +data[i]+"00010.xml\nDownload now ?");
                if(dload == 0){
                        boolean fin = connectToInternet(data[i], 10);
                        if(!fin)
                                return null;
                }else{
                        JOptionPane.showMessageDialog(null, "Process aborted");
                        return null;
                }
        }

        G.add((new XMLParser(filePath)).getGraph(false));
        filePath = "/files/citric/"+data[i]+"00020.xml";
        try{
                URL kgml = getClass().getResource(
                                "/files/citric/"+data[i]+"00020.xml");
                InputStreamReader isr =
                                new InputStreamReader(kgml.openStream());
                BufferedReader in = new BufferedReader(isr);
        } catch (Exception e){
                e.printStackTrace();

                int dload = JOptionPane.showConfirmDialog(null,
                                "file not found: "+data[i]+"00020.xml\nDownload now ?");
                if(dload == 0){
                        boolean fin = connectToInternet(data[i], 10);
                        if(!fin)
                                return null;
                }else{
                        JOptionPane.showMessageDialog(null, "Process aborted");
                        return null;
                }
        }


        Graph ccG = (new XMLParser(filePath)).getGraph(false);
        G.get(G.size() - 1).joinGraph(ccG);
        G.get(G.size() - 1).setName(data[i]);
}

//this
data_ = data;
System.out.println("to group..");

progressBar.setString("05% completed - Grouping organisms....");
BuildPhenogramGroup GP = new BuildPhenogramGroup(G, index);
GP.FPGraph();

if(choice == build){
        //str += GP.getGroup_();
        outGroup = GP.getGroup();
        str += "\nGroup Generated:  " + outGroup;
        //str += " -> " + GP.getGroupV();
        System.out.println("get parent..");

        progressBar.setString(
                        "25% completed - Getting parenthood of organisms");
        NCBITaxManager NCBI = new NCBITaxManager(G);
        NCBI.getParentHood();

        progressBar.setString(
                        "50% completed - Creating phenogram for NCBI Taxonomy...");
        BuildPhenogramGroup GP2 =
                        new BuildPhenogramGroup(NCBI.getSimMatrix(), G);
        GP2.generateGroup();

        progressBar.setString(
                        "70% completed - Generating .txt files...");
        generateIntree(GP.getGroup(), GP2.getGroup());
        generateDrawGram(GP.getGroup_(), true);
        generateDrawGram(GP.getGroup_(), false);


        progressBar.setString(
                        "75% completed - Checking PHYLIP TreeDist");
        PHYLIPTreeDist ptd = new PHYLIPTreeDist(GP2.getGroup());
        ptd.checkTree();
        System.out.println("to string..");
        ncbiGroup = GP2.getGroup();
        str += "\nNCBI Taxonomy:   " + ncbiGroup;
        str += "\nSimilarity";
        progressBar.setString("90% completed - Checking MCT Comparison");
        double mct = (new Comparison()).
                        MCTCompare(GP.getGroup(), GP2.getGroup());
        mct = round(mct);
        mctComp = mct + "";
        str += "\n(1) using MCT: " + mctComp + " %";
        double phylipTD = ptd.compare();
        phylipTD = round(phylipTD);
        phylipComp = "" + phylipTD;
        str += "\n(2) using PHYLIP TreeDist: " + phylipComp + " %";
        System.out.println("done..");
}else if(choice == gs){
```

```
                gsrSimMatrix = GP.getSimilarity();

                for(int i = 0; i < G.size() ; i++){
                        for(int j = 0 ; j < G.size() ; j++){
                                if(i >= j)
                                        continue;

                                str += "pair (" +
                                G.get(i).getName() + ", " + G.get(j).getName() + ") - ";
                                progressBar.setString(
                                                "Getting similarity.... for pair ("
                                + G.get(i).getName() + ", " + G.get(j).getName() + ")");
                                gsrSimMatrix[i][j] = 100*gsrSimMatrix[i][j];
                                if(gsrSimMatrix[i][j] != 100)
                                        gsrSimMatrix[i][j] = round(gsrSimMatrix[i][j]);
                                str += gsrSimMatrix[i][j] + "%\n";
                        }
                }
        }else if(choice == fs){
                FPGraphMiner fpg = new FPGraphMiner(G);
                fpg.FPTable();
                fptable = fpg.FPTable;
                fpg.frequentSubgraphs(treshold);
                dfs = fpg.dfs;
                eList = fpg.eList;
                indices = fpg.indices;
                edgeIndex = fpg.edgeIndex;
                str += "Subgraps with " + (treshold*100) + "% treshold\n";
                for(int j = 0 ; j < dfs.size() ; j++){
                        ArrayList<Integer> curr = dfs.get(j);
                        str += "Graph "+ (j+1) + ": ";
                        for(int i = 0 ; i < curr.size() ; i++){
                                str += curr.get(i);
                                if(i < curr.size() -1)
                                        str += " - ";
                        }
                        str += "\n";
                }

                str += "";
        }else if(choice == fp){
                progressBar.setString("reading " + fileSub+".xml file..");
                subG = (new XMLParser(fileSub)).getGraph(true);
                progressBar.setString("generating FPGraphMiner Pattern Finder");
                FPGraphMiner fpg = new FPGraphMiner(G);
                fpg.FPTable();
                fpg.patternFinder(subG);
                progressBar.setString("Finalizing");
                str += "Subgraph exist('s) in ";
                boolean first = true;
                speSub = fpg.bfsArray;
                for(int i = 0 ; i < speSub.length ; i++){
                        if(speSub[i] == 1){
                                if(first){
                                        first = false;
                                }else{
                                        str += ", ";
                                }
                                str += G.get(i).getName();
                        }
                }
                if(speSub.length == 0)
                        str += "No species found.";
                str += "\n";
        }
        return str;
}

public void generateIntree(String g1, String g2){
        try {

                String content = g1 + ";\n" + g2 + ";";

                File file = new File("intree");
                // if file doesnt exists, then create it
                if (!file.exists()) {
                        file.createNewFile();
                }

                FileWriter fw = new FileWriter(file.getAbsoluteFile());
                BufferedWriter bw = new BufferedWriter(fw);
                bw.write(content);
                bw.close();

        } catch (IOException e) {
                e.printStackTrace();
        }
}

public void generateDrawGram(String tree, boolean isDraw){
        try {

                String content = tree + ";";
                File file = new File("treedata");

                if(isDraw)
                        file = new File("treedata");
                else
                        file = new File("treedata2");
```

```
                // if file doesnt exists, then create it
                if (!file.exists()) {
                        file.createNewFile();
                }

                FileWriter fw = new FileWriter(file.getAbsoluteFile());
                BufferedWriter bw = new BufferedWriter(fw);
                bw.write(content);
                bw.close();

        } catch (IOException e) {
                e.printStackTrace();
        }
}

public double round(double num){
        String s = num + "";
        int index = s.indexOf('.');
        int prefsize = index + 3;
        if(s.length() > prefsize){ // 0.xyz
                String s2 = s.charAt(prefsize) + "";
                int n = Integer.parseInt(s2);
                if(n >= 5){
                        String s3 = s.substring(index + 1, prefsize) + "";
                        int n2 = Integer.parseInt(s3);
                        n2++;
                        String s4 = s.substring(0, index + 1) + (n2 + "");
                        num = Double.parseDouble(s4);
                }else{
                        String s4 = s.substring(0, prefsize);
                        num = Double.parseDouble(s4);
                }
        }
        return num;
}

public void exportFiles(int choice, String filePath, String fileName){
        try{
                String pretext = "";
                String str = "";
                if(choice == build){
                        DrawgramInterface dg = new DrawgramInterface();
                        dg.DrawgramRun("treedata2");

                        String title = "Preview: ";
                        String curDir = System.getProperty("user.dir");
                        curDir += "/JavaPreview.ps";

                        new DrawPreview(title, curDir, filePath, true);

                        str += "(FORMAT = Number -> KEGG Code -> Organism Name";
                        //show data
                        for(int i = 0 ; i < data_.length ; i++){
                                str += "\n" + (i+1);
                                str += " -> " + data_[i];
                                str += " -> " +
                                options.get(0).get(options.get(1).indexOf(data_[i]));
                        }

                        str     += "\n\n\n++++++++++++++++++++++++++++++++++++++++\n\n\n";

                        //output
                        str += "\nGroup Generated:  " + outGroup;
                        str += "\nNCBI Taxonomy:    " + ncbiGroup;
                        str += "\nSimilarity";
                        str += "\n(1) using MCT: " + mctComp + " %";
                        str += "\n(2) using PHYLIP TreeDist: " + phylipComp + " %";

                        pretext = "build";
                }else if(choice == gs){

                        str += "(FORMAT = KEGG Code -> Organism Name";
                        //show data
                        for(int i = 0 ; i < data_.length ; i++){
                                str += "\n" + data_[i];
                                str += " -> "
                                + options.get(0).get(options.get(1).indexOf(data_[i]));
                        }

                        str += "\n\n\n++++++++++++++++++++++++++++++++++++++++\n\n\n";

                        //output
                        for(int i = 0; i < data_.length ; i++){
                                for(int j = 0 ; j < data_.length ; j++){
                                        if(i >= j)
                                                continue;

                                        str += "pair (" + data_[i] + ", " + data_[j] + ") - ";
                                        str += gsrSimMatrix[i][j] + "%\n";
                                }
                        }

                        pretext = "gs";
                }else if(choice == fs){

                        str += "(FORMAT = KEGG Code -> Organism Name";
                        //show data
                        for(int i = 0 ; i < data_.length ; i++){
```

```
                str += "\n" + data_[i];
                str += " -> " +
                options.get(0).get(
                                options.get(1).indexOf(data_[i]));
        }

        str += "\n\n\n+++++" +
                        "+++++++++++++++" +
                        "++++++++++++++++++++\n\n\n";

        str += "\nSubgraphs with " +
        treshold + "% treshold: ";
        ArrayList<ArrayList<Edge>> recList =
                        new ArrayList<ArrayList<Edge>>();
        for(int i = 0 ; i < dfs.size() ; i++){
                String text = "";
                int size = 0;
                ArrayList<Edge> recE = new ArrayList<Edge>();
                for(int j = 0 ; j < dfs.get(i).size() ; j++){
                        int start = indices.get(dfs.get(i).get(j))[0];
                        int last = indices.get(dfs.get(i).get(j))[1];
                        for(int k = start ; k <= last ; k++){
                                int eIndex = edgeIndex[k];
                                String s = eList.get(eIndex).reaction;
                                recE.add(eList.get(eIndex));
                                if(k != start){
                                        text += ", ";
                                        if(size > 85){
                                                size = 0;
                                                text += "\n";
                                        }
                                }
                                size += s.length();
                                text += s;
                        }
                }

                str += "\n\nReaction List #" + (i+1);
                str += "\n" + text;
                recList.add(recE);
        }

        pretext = "fs";

        for(int i = 0 ; i < recList.size() ; i++){

                String content = "<?xml version=\"1.0\"?>";
                content += "\n<subgraph treshold=\""
                + treshold + "\" orgs=\"";


                ArrayList<Integer> curr = dfs.get(i);

                int ind = curr.get(0);
                int[] currAr = fptable.get(ind);
                String orgCont = "";
                for(int j = 0 ; j < currAr.length ; j++){
                        if(currAr[j] == 1){
                                if(orgCont.length() != 0)
                                        orgCont += ", ";
                                orgCont += "[" + data_[j] +
                                                "]" + options.get(0).
                                                get(options.get(1).indexOf(data_[j]));
                        }
                }
                content += orgCont + "\">";

                String num = "(";
                for(int j = 0 ; j < curr.size() ; j++){
                        num += curr.get(j);
                        if(i < curr.size() -1)
                                num += "-";
                }
                num += ")";


                ArrayList<Edge> recE = recList.get(i);
                String reactions = "";
                for(int j = 0 ; j < recE.size() ; j++){
                        Edge e = recE.get(j);
                        reactions += "\n\t<reaction id=\"" + e.id + "\" ";
                        reactions += "name=\"" + e.reaction + "\" ";
                        if(e.reversible)
                                reactions += "type=\"reversible\">";
                        else
                                reactions += "type=\"irreversible\">";

                        for(int k = 0 ; k < e.vertex1.names.size() ; k++){
                                reactions += "\n\t\t<substrate name=\""
                        + e.vertex1.names.get(k) + "\" />";
                        }
                        for(int k = 0 ; k < e.vertex2.names.size() ; k++){
                                reactions += "\n\t\t<product name=\""
                        + e.vertex2.names.get(k) + "\" />";
                        }
                        reactions += "\n\t</reaction>";
                }
                content += reactions;
                content += "\n</subgraph>";
```

```
                                        File xmlfile = new File(filePath + "/subg"
                                        + (i+1) + "_" + treshold + "-" + num + "_.xml") ;
                                        // if file doesnt exists, then create it
                                        if (!xmlfile.exists()) {
                                                xmlfile.createNewFile();
                                        }

                                        FileWriter xmlfw = new FileWriter(
                                                        xmlfile.getAbsoluteFile());
                                        BufferedWriter xmlbw = new BufferedWriter(xmlfw);
                                        xmlbw.write(content);
                                        xmlbw.close();
                                }
                        }else if(choice == fp){
                                str += "Reaction File: " + fileSub + "\n";
                                str += "Reaction exist('s) in ";
                                boolean first = true;
                                for(int i = 0 ; i < speSub.length ; i++){
                                        if(speSub[i] == 1){
                                                if(first){
                                                        first = false;
                                                }else{
                                                        str += ", ";
                                                }
                                                str += "\n(" + data_[i] + ")"
                                                + options.get(0).get(options.get(1).indexOf(data_[i]));
                                        }
                                }
                                str += "\n";

                                if(speSub.length == 0)
                                        str += "No species found.";
                                pretext = "fp";
                        }

                        File file = new File(filePath + "/" + fileName + "_" + pretext + ".txt") ;
                        // if file doesnt exists, then create it
                        if (!file.exists()) {
                                file.createNewFile();
                        }

                        FileWriter fw = new FileWriter(file.getAbsoluteFile());
                        BufferedWriter bw = new BufferedWriter(fw);
                        bw.write(str);
                        bw.close();
                } catch (IOException e) {
                        e.printStackTrace();
                }
        }


        private boolean connectToInternet(String spe, int pwayI){
                try{
                        String path = "http://rest.kegg.jp/get/"+
                spe + "000" + pwayI + "/kgml";
                System.out.println("Get XML file from " + path);

                        URL kgml = new URL(path);
                    BufferedReader in = new BufferedReader(
                                new InputStreamReader(kgml.openStream()));

                    String inputLine;
                    String content = "";
                    while ((inputLine = in.readLine()) != null){
                        content += inputLine + "\n";
                    }
                    in.close();

                        String addP = "glyco";
                        if(pwayI == 20)
                                addP = "citric";

                        String s = getClass().getResource("/files/"
                        + addP + "/"+ spe +"000" + pwayI + ".xml").toString();
                        s = s.substring(s.indexOf('C'), s.length());
                        File xmlfile2 = new File(s) ;
                        // if file doesnt exists, then create it
                        if (!xmlfile2.exists()) {
                                xmlfile2.createNewFile();
                        }
                        FileWriter xmlfw2 = new FileWriter(

                                        xmlfile2.getAbsoluteFile());
                        BufferedWriter xmlbw2 = new BufferedWriter(xmlfw2);
                        xmlbw2.write(content);
                        xmlbw2.close();

                        JOptionPane.showMessageDialog(null, "Download Complete");
                } catch(Exception e){
                        e.printStackTrace();
                        JOptionPane.showMessageDialog(null, "Can't connect to www.kegg,jp");
                        return false;
                }

                return true;
        }

}
```

7. JSONParser.java

```
package com.Controller;

import java.io.BufferedReader;
import java.io.FileNotFoundException;
import java.io.IOException;
import java.io.InputStreamReader;
import java.net.URL;
import java.util.ArrayList;
import java.util.Iterator;

import org.json.simple.JSONArray;
import org.json.simple.JSONObject;
import org.json.simple.parser.JSONParser;
import org.json.simple.parser.ParseException;

import com.Model.Specie;


public class JSONParser_ {

        private static final String filePath = "files/options.JSON";

        ArrayList<Specie> species;

        public JSONParser_(){

        }

        public ArrayList<Specie> ParseJSON(){
                species = new ArrayList<Specie>();
                try {
                        // read the json file
                        //FileReader reader = new FileReader(filePath);

                        URL fileP = getClass().getResource("/files/options.JSON");
                    BufferedReader in = new BufferedReader
                                (new InputStreamReader(fileP.openStream()));

                    String inputLine;
                    String contentJSON = "";
                    while ((inputLine = in.readLine()) != null){
                        contentJSON += inputLine + "\n";
                    }
                    in.close();

                        JSONParser jsonParser = new JSONParser();

                        JSONObject jsonObject = (JSONObject)
                                        jsonParser.parse(contentJSON);

                        // get an array from the JSON object
                        JSONObject query = (JSONObject) jsonObject.get("query");
                        JSONObject result = (JSONObject) query.get("results");
                        JSONArray select = (JSONArray) result.get("select");
                        // take the elements of the json array

                        @SuppressWarnings("rawtypes")
                        Iterator i = select.iterator();

                        // take each value from the json array separately
                        while (i.hasNext()) {
                                JSONObject inobj = (JSONObject) i.next();
                                JSONObject option = (JSONObject) inobj.get("option");
                                Object content = option.get("content");
                                if(content.toString().contains("Reference") ||
                                                content.toString().contains("personalized") ||
                                                content.toString().contains("alphabet") ||
                                                content.toString().contains("Disease"))
                                        continue;
                                Object value = option.get("value");
                                species.add(new Specie(content.toString(),
                                                value.toString()));
                        }


                } catch (FileNotFoundException ex) {
                        ex.printStackTrace();
                } catch (IOException ex) {
                        ex.printStackTrace();
                } catch (ParseException ex) {
                        ex.printStackTrace();
                } catch (NullPointerException ex) {
                        ex.printStackTrace();
                }


                return species;
        }

}
```

8. NCBITaxManager.java

```
package com.Controller;

import java.io.IOException;
```

```
import java.util.ArrayList;

import javax.swing.JOptionPane;

import com.Model.Graph;


public class NCBITaxManager {

        ArrayList<Graph> G;
        ArrayList<ArrayList<Integer>> parent;
        ArrayList<Integer> group;
        double[][] SimilarityMatrix;

        public NCBITaxManager(){}

        public NCBITaxManager(ArrayList<Graph> G){
                this.G = G;
                (new GUIHandler()).getOptions();
        }

        public ArrayList<ArrayList<Integer>>
        getParentHood() throws IOException{
                parent = new ArrayList<ArrayList<Integer>>();
                for(int i = 0 ; i < G.size() ; i++){
                        parent.add(new ArrayList<Integer>());
                        String temp = G.get(i).getName();
                        temp = GUIHandler.options.get(0).get(
                                        GUIHandler.options.get(1).indexOf(temp));
                        String[] stra = temp.split("\\(");
                        temp = stra[0].trim();
                        parent.get(i).add(getNameId(temp));
                }

                while(!isEqualAll()){
                        for(ArrayList<Integer> specie : parent){
                                if(specie.get(specie.size() - 1) == 1)
                                        continue;
                                specie.add(getParent(specie.get(specie.size() - 1)));
                        }
                }

                return parent;
        }

        public boolean isEqualAll(){
                int n = parent.get(0).get(parent.get(0).size() - 1);
                for(ArrayList<Integer> specie : parent)
                        if(specie.get(specie.size() - 1) != n)
                                return false;

                return true;
        }

        public ArrayList<ArrayList<Integer>>
        getParentHood_() throws IOException{
                parent = new ArrayList<ArrayList<Integer>>();
                for(int i = 0 ; i < G.size() ; i++){
                        parent.add(new ArrayList<Integer>());
                        String temp = G.get(i).getName();
                        temp = GUIHandler.options.get(0).
                                        get(GUIHandler.options.get(1).indexOf(temp));
                        String[] stra = temp.split("\\(");
                        temp = stra[0].trim();
                        int id = -1;
                        while((id = getNameId(temp)) == -1){
                                stra = temp.split(" ");
                                if(stra.length == 1){
                                        JOptionPane.showMessageDialog(null,
                                                        G.get(i).getName() +
                                                        " does not exist in NCBI Organism List.\n"+
                                                        "Check spelling and contact the admin for errors.");" +
                                }
                                for(int j = 0 ; j < stra.length ; j++)
                                        temp += stra[j] + " ";

                                temp = temp.trim();
                        }

                        parent.get(i).add(id);
                }

                for(int i = 0 ; i < parent.size() ; i++){
                        ArrayList<Integer> specie = parent.get(i);
                        int last = specie.get(specie.size() - 1);
                        while(last != 1){
                                int num = getParent(last);
                                specie.add(num);
                                last = num;
                                for(int j = 0 ; j < i ; j++){
                                        ArrayList<Integer> tspecie = parent.get(j);
                                        if(tspecie.contains(num)){
                                                for(int k = tspecie.indexOf(num) + 1 ;
                                                                k < tspecie.size() ; k++)
                                                        specie.add(tspecie.get(k));

                                                last = 1;
                                                break;
                                        }
```

```
                                }
                        }
                }
                return parent;
        }

        public int getParent(int id) throws IOException{
                return (new DMPParser()).getParent(id);
        }

        public int getNameId(String name){
                try {
                        return (new DMPParser()).getId(name);
                } catch (IOException e) {
                        e.printStackTrace();
                }

                return -1;
        }

        public double[][] getSimMatrix(){
                int size = G.size();
                SimilarityMatrix = new double[size][size];

                for(int i = 0 ; i < size ; i++){
                        for(int j = 0 ; j < size ; j++){
                                if(i == j){
                                        SimilarityMatrix[i][j] = 0;
                                        continue;
                                }
                                if(i > j)
                                        continue;

                                SimilarityMatrix[i][j] = getEqual(i, j);
                                SimilarityMatrix[j][i] = SimilarityMatrix[i][j];
                        }
                }

                return SimilarityMatrix;
        }

        public int getEqual(int i , int j){
                int count = 0;

                ArrayList<Integer> speI = parent.get(i);
                int sizeI = speI.size();

                ArrayList<Integer> speJ = parent.get(j);
                int sizeJ = speJ.size();

                int k = sizeI;
                if(sizeI > sizeJ){
                        k = sizeI;
                }

                for(int index = 0; index < k; index++){
                        int n1 = speI.get(sizeI - 1 - index);
                        int n2 = speJ.get(sizeJ - 1 - index);
                        if(n1 == n2){
                                count++;
                        }else
                                break;
                }

                return count;
        }
}
```

9. PHYLIPTreeDist.java

```
package com.Controller;

import java.io.File;
import java.io.FileReader;
import java.io.IOException;
import java.io.OutputStream;
import java.io.PrintStream;
import java.net.URI;
import java.net.URISyntaxException;
import java.util.Scanner;

public class PHYLIPTreeDist {

        String NCBIGroup;

        public PHYLIPTreeDist(){}

        public PHYLIPTreeDist(String NCBIGroup){
                this.NCBIGroup = NCBIGroup;
        }

        public void checkTree() throws
        IOException, URISyntaxException{
                URI exe = (new CreateTempFile()).convert("exe/treedist.exe");

                Process process = Runtime.getRuntime().exec(exe.getPath());

                OutputStream os = process.getOutputStream();
```

```
                PrintStream printStream = new PrintStream(os);
                printStream.println("R");
                printStream.flush();
                printStream.println("D");
                printStream.flush();
                printStream.println("Y");
                printStream.flush();
                printStream.println();
                printStream.flush();
                printStream.close();

                File file = new File("outfile");
                while(!file.exists())
                        file = new File("outfile");
        }

        public double compare(){
                double similarity = 0;

                try {
                        String str = getDist().trim();
                        String numS = "";
                        for(int i = str.indexOf(':')+1 ;
                                        i < str.length() ; i++){
                                if(str.charAt(i) != ' ')
                                        numS += str.charAt(i);
                        }
                        double num = Double.parseDouble(numS);
                        similarity = num/countBranches();
                        similarity = round(similarity);
                        similarity = 100 - similarity*100;
                } catch (IOException e) {
                        e.printStackTrace();
                }

                return similarity;
        }

        public String getDist() throws IOException{
                String dist = "";
                File file = new File("outfile");
                Scanner scanner;
                if (!file.exists()) {
                        file.createNewFile();
                } else {
                        scanner = new Scanner(new FileReader(file));
                        int counter = 1;
                        while (scanner.hasNextLine()) {
                                String str = scanner.nextLine().trim();
                                if(counter == 6){
                                        scanner.close();
                                        return str;
                                }
                                counter++;
                        }
                        scanner.close();
                }

                return dist;
        }

        public int countBranches(){
                int count = 0;

                for(int i = 0 ; i < NCBIGroup.length() ; i++)
                        if(NCBIGroup.charAt(i) == '(' || NCBIGroup.charAt(i) == ',')
                                count++;

                return count;
        }

        public double round(double num){
                String s = num + "";

                if(s.length() > 5){ // 0.xyz
                        String s2 = s.charAt(5) + "";
                        int n = Integer.parseInt(s2);
                        if(n >= 5){
                                String s3 = s.substring(2, 5) + "";
                                int n2 = Integer.parseInt(s3);
                                n2++;
                                String s4 = s.substring(0, 2) + (n2 + "");
                                num = Double.parseDouble(s4);
                        }else{
                                String s4 = s.substring(0, 5);
                                num = Double.parseDouble(s4);
                        }
                }

                return num;
        }
}
```

10. SimilarityMatrix.java

```
package com.Controller;

import java.util.ArrayList;
```

```
import com.Model.Graph;


public class SimilarityMatrix {

        private final int Jaccard = 0;
        private final int Hamming = 1;

        double[][] SimilarityMatrix;
        ArrayList<Graph> G;
        ArrayList<int[]> FPTable;
        ArrayList<Integer> counts;
        int GSizeint, choice;

        public SimilarityMatrix(ArrayList<int[]> FPTable,
                        ArrayList<Integer> counts, ArrayList<Graph> G){
                this.FPTable = FPTable;
                this.counts = counts;
                this.G = G;
        }


        public double[][] getSimilarityMatrix(int choice){
                int GSize = G.size();
                SimilarityMatrix = new double[GSize][GSize];

                int total = 0;
                if(choice == Hamming)
                        total = countEdges();

                for(int i = 0 ; i < GSize ; i++){
                        for(int j = 0 ; j < GSize ; j++){

                                if(i < j)
                                        continue;

                                if(i == j){
                                        SimilarityMatrix[i][j] = 0;
                                        continue;
                                }

                                double nume = 0;
                                int count = 0;

                                for(int k = 0 ; k < FPTable.size() ; k++){
                                        if(choice == Jaccard){
                                                if(FPTable.get(k)[i] == 1
                                                                && FPTable.get(k)[j] == 1)
                                                        nume += counts.get(k);
                                        } else {
                                                if(FPTable.get(k)[i] != FPTable.get(k)[j])
                                                count+= counts.get(k);
                                        }
                                }

                                double num = 0;
                                if(choice == Jaccard){
                                        num = nume/(G.get(i).geteList().size() +
                                                        G.get(j).geteList().size() - nume);
                                }else{
                                        double num1 = (total - count);
                                num = num1/total;
                                }
                                num = round(num);
                        SimilarityMatrix[i][j] = num;
                                SimilarityMatrix[j][i] = num;
                        }
                }

                return SimilarityMatrix;
        }

        public double round(double num){
                String s = num + "";

                if(s.length() > 5){ // 0.xyz
                        String s2 = s.charAt(5) + "";
                        int n = Integer.parseInt(s2);
                        if(n >= 5){
                                String s3 = s.substring(2, 5) + "";
                                int n2 = Integer.parseInt(s3);
                                n2++;
                                String s4 = s.substring(0, 2) + (n2 + "");
                                num = Double.parseDouble(s4);
                        }else{
                                String s4 = s.substring(0, 5);
                                num = Double.parseDouble(s4);
                        }
                }

                return num;
        }

        public int countEdges(){
                int total = 0;
                for(int k = 0 ; k < counts.size() ; k++){
                        total+= counts.get(k);
                }
```

```
                return total;
        }
}
```

11. TXTParser.java

```
package com.Controller;

import java.io.File;
import java.io.FileReader;
import java.io.IOException;
import java.util.ArrayList;
import java.util.Scanner;


public class TXTParser {

        Scanner scanner;
        String filePath;
        ArrayList<String[]> data;
        int index = 0;

        public TXTParser(String filePath){
                this.filePath = filePath;
        }

        public ArrayList<String[]> readFile()
                        throws IOException{
                String s = getClass().getResource("/"
                        + filePath).toString();
                s = s.substring(s.indexOf('C'), s.length());
                File file = new File(s);
                data = new ArrayList<String[]>();

                if (!file.exists()) {
                        file.createNewFile();
                } else {
                        scanner = new Scanner(new FileReader(file));
                        while (scanner.hasNextLine()) {
                                String str = scanner.nextLine().trim();
                                String[] tData = str.split(",");
                                for(int i = 0 ; i < tData.length ; i++){
                                        tData[i] = tData[i].trim();
                                }
                                data.add(tData);
                        }
                }
                return data;
        }
}
```

12. XMLParser.java

```
package com.Controller;

import java.io.IOException;
import java.util.ArrayList;

import javax.swing.JOptionPane;
import javax.xml.parsers.DocumentBuilder;
import javax.xml.parsers.DocumentBuilderFactory;
import javax.xml.parsers.ParserConfigurationException;

import org.w3c.dom.Document;
import org.w3c.dom.Element;
import org.w3c.dom.NodeList;
import org.xml.sax.InputSource;
import org.xml.sax.SAXException;

import com.Model.Edge;
import com.Model.Graph;
import com.Model.Vertex;


public class XMLParser {

        Document dom ;
        String filePath ;
        private Graph G;
        boolean isSub = false;

        public XMLParser(String filePath){
                this.filePath = filePath;
                G = null;
        }

        public Graph getGraph(boolean isSub){
                G = new Graph();
                this.isSub = isSub;
                parseXmlFile();
                return G;
        }

        private void parseXmlFile(){
                //get the factory
                DocumentBuilderFactory dbf =
                                DocumentBuilderFactory.newInstance();
```

```
        dbf.setValidating(false);
        try {
                dbf.setFeature(
                                "http://apache.org/xml/features/"+
                                                "nonvalidating/load-external-dtd",
                                false);
        } catch (ParserConfigurationException pe) {
                pe.printStackTrace();
        }

        try {
                DocumentBuilder db = dbf.newDocumentBuilder();

                if(!isSub){
                        System.out.println(filePath);
                        String s = getClass().getResource(filePath).toString();
                        //s = s.substring(s.indexOf('C'), s.length());

                        //Using factory get an instance of document builder

                        //parse using builder to get DOM representation of the XML file

                        InputSource is = new InputSource(s);
                    dom = db.parse(is);
                } else
                        dom = db.parse(filePath);
                parseDocument();
        }catch(ParserConfigurationException pce) {
                pce.printStackTrace();
        }catch(SAXException se) {
                se.printStackTrace();
        }catch(IOException ioe) {
                System.out.println("File not found: " + filePath);
                JOptionPane.showMessageDialog(null,
                                "File not found: " + filePath);
        }
}

private void parseDocument(){
        //get the root element
        Element docEle = dom.getDocumentElement();

        ArrayList<Integer> eID = new ArrayList<Integer>();
        ArrayList<String> eName = new ArrayList<String>();

        NodeList nl1 = docEle.getElementsByTagName("entry");
        if(nl1 != null && nl1.getLength() > 0) {
                for(int i = 0 ; i < nl1.getLength(); i++) {
                        Element el = (Element)nl1.item(i);
                        String name = el.getAttribute("name");
                        int id = Integer.parseInt(el.getAttribute("id"));

                        eID.add(id);
                        eName.add(name);
                }
        }


        //get a nodelist of elements
        NodeList nl = docEle.getElementsByTagName("reaction");
        if(nl != null && nl.getLength() > 0) {
                for(int i = 0 ; i < nl.getLength();i++) {

                        Element el = (Element)nl.item(i);
                        String name = el.getAttribute("name");

                        String type = el.getAttribute("type");
                        int id = Integer.parseInt(el.getAttribute("id"));

                        String ent = eName.get(eID.indexOf(id));
                        String[] entry_ = ent.split(" ");
                        ArrayList<String> entry = new ArrayList<String>();
                        for(int j = 0 ; j < entry_.length ; j++)
                                entry.add(entry_[j]);

                        boolean reversible = false;
                        if(type.equalsIgnoreCase("reversible"))
                                reversible = true;

                        NodeList nl2 = el.getElementsByTagName("substrate");
                        ArrayList<String> subs = new ArrayList<String>();
                        if(nl2 != null && nl2.getLength() > 0) {
                                for(int j = 0 ; j < nl2.getLength(); j++) {
                                        Element el2 = (Element)nl2.item(j);
                                        subs.add(el2.getAttribute("name"));
                                }
                        }

                        NodeList nl3 = el.getElementsByTagName("product");
                        ArrayList<String> prod = new ArrayList<String>();
                        if(nl3 != null && nl3.getLength() > 0) {
                                for(int j = 0 ; j < nl3.getLength(); j++) {
                                        Element el3 = (Element)nl3.item(j);
                                        prod.add(el3.getAttribute("name"));
                                }
                        }

                        if(name.contains(" ")){
                                String[] names = name.split(" ");
```

```
                                        for(int j = 0; j < names.length ; j++){
                                                for(String s : subs){
                                                        ArrayList<String> subs_ =
                                                                        new ArrayList<String>();
                                                        subs_.add(s);
                                                        Vertex v1_ = new Vertex(subs_);
                                                        for(String s1 : prod){
                                                                ArrayList<String> prod_ =
                                                                                new ArrayList<String>();
                                                                prod_.add(s1);
                                                                Vertex v2_ = new Vertex(prod_);

                                                                if(!G.vertexExist(v1_))
                                                                        G.addVertex(v1_);
                                                                if(!G.vertexExist(v2_))
                                                                        G.addVertex(v2_);

                                                                Edge e = new Edge(id, entry,
                                                                                names[j], v1_, v2_, reversible);
                                                                if(!G.edgeExist(e))
                                                                        G.addEdge(e);
                                                        }
                                                }
                                        }
                                }else{
                                        for(String s : subs){
                                                ArrayList<String> subs_ = new ArrayList<String>();
                                                subs_.add(s);
                                                Vertex v1_ = new Vertex(subs_);
                                                for(String s1 : prod){
                                                        ArrayList<String> prod_ = new ArrayList<String>();
                                                        prod_.add(s1);
                                                        Vertex v2_ = new Vertex(prod_);

                                                        if(!G.vertexExist(v1_))
                                                                G.addVertex(v1_);
                                                        if(!G.vertexExist(v2_))
                                                                G.addVertex(v2_);

                                                        Edge e = new Edge(id, entry,
                                                                        name, v1_, v2_, reversible);
                                                        if(!G.edgeExist(e))
                                                                G.addEdge(e);
                                                }
                                        }
                                }
                        }
                }
        }
}
```

# 2. Model

1. Edge.java

```
package com.Model;
import java.util.ArrayList;


public class Edge {

        public Vertex vertex1; // substrate
    public Vertex vertex2; // product
        public String reaction; // name of reaction
        public boolean reversible; // check if rev or not
        public ArrayList<String> entry;
        public int id;

    public Edge(int id, ArrayList<String> entry,
                String reaction, Vertex substrate,
                Vertex product, boolean reversible){
        vertex1 = substrate;
        vertex2 = product;
        this.reaction = reaction;
        this.reversible = reversible;
        this.entry = entry;
        this.id = id;
    }

    public boolean isEqual(Edge e, boolean isJoin){
                if(isJoin){
                        if(entry.size() != e.entry.size())
                                return false;

                        if(reaction.equalsIgnoreCase(e.reaction)){
                                for(int i = 0 ; i < entry.size() ; i++){
                                        if(entry.indexOf(e.entry.get(i)) == -1)
                                                return false;
                                }
                                return true;
                        }
                }else{
                        if(reaction.equalsIgnoreCase(e.reaction)){
                        if(vertex1.isEqual(e.vertex1)){
                                if(vertex2.isEqual(e.vertex2)){
                                        return true;
                                }
```

```
                                        }else{
                                                if(reversible || e.reversible){
                                                        if(vertex1.isEqual(e.vertex2)
                                                                        && vertex2.isEqual(e.vertex1))
                                                                return true;
                                                }
                                        }
                                }
                        }

                return false;
        }
}
```

2. Graph.java

```
package com.Model;
import java.util.ArrayList;


public class Graph {
        ArrayList<Vertex> vList;
        ArrayList<Edge> eList;
        String name;

        int[][] adjMatrix;

        public Graph(){
                vList = new ArrayList<Vertex>();
                eList = new ArrayList<Edge>();
        }

        public Graph(String name,
                        ArrayList<Vertex> vList, ArrayList<Edge> eList){
                this.vList = vList;
                this.eList = eList;
                this.name = name;
        }

        public void setName(String name){
                this.name = name;
        }

        public String getName(){
                return name;
        }

        public ArrayList<Vertex> getvList(){
                return vList;
        }

        public ArrayList<Edge> geteList(){
                return eList;
        }

        public void addVertex(Vertex v){
                vList.add(v);
        }

        public void addEdge(Edge e){
                eList.add(e);
        }

        public boolean vertexExist(Vertex v){
                for(Vertex v1 : vList){
                        if(v1.isEqual(v))
                                return true;
                }
                return false;
        }

        public boolean edgeExist(Edge e){
        for(Edge e1 : eList){
                if(e1.isEqual(e, true)){
                        return true;
                }
        }

        return false;
    }

        public void joinGraph(Graph G){
                for(Edge e : G.eList){
                        if(!edgeExist(e)){
                                eList.add(e);
                                if(!vertexExist(e.vertex1))
                                        vList.add(e.vertex1);
                                if(!vertexExist(e.vertex2))
                                        vList.add(e.vertex2);
                        }
                }
        }
}
```

3. Specie.java

```
package com.Model;
```

```
public class Specie {

        String content;
        String value;

        public Specie(String content, String value){
                this.content = content;
                this.value = value;
        }

        public String getContent(){
                return content;
        }

        public String getValue(){
                return value;
        }

}
```

4. Vertex.java

```
package com.Model;
import java.util.ArrayList;


public class Vertex {

        public ArrayList<String> names;

        public Vertex(ArrayList<String> names){
                this.names = names;
        }

        public boolean isEqual(Vertex v){
        if(names.size() != v.names.size())
                return false;

        for(int i = 0; i < names.size() ; i++){
                if(!names.get(i).equalsIgnoreCase(v.names.get(i)))
                        return false;
        }

        return true;
    }
}
```

# 3. PHYLIP.DrawGram

1. DrawGramInterface.java

```
package com.PHYLIP.DrawGram;

import java.io.BufferedReader;
import java.io.BufferedWriter;
import java.io.File;
import java.io.FileOutputStream;
import java.io.FileWriter;
import java.io.InputStream;
import java.io.InputStreamReader;
import java.io.OutputStream;
import java.net.URL;

import com.Controller.CreateTempFile;
import com.PHYLIP.Util.TestFileNames;
import com.sun.jna.Library;
import com.sun.jna.Native;

public class DrawgramInterface {
        public interface Drawgram extends Library {
            public void  drawgram(
                                String  intree,
                                String  usefont,
                                String  plotfile,
                                String  plotfileopt,
                                String  treegrows,
                                String  treestyle,
                                boolean usebranchlengths,
                                double  labelangle,
                                boolean scalebranchlength,
                                double  branchlength,
                                double  breadthdepthratio,
                                double  stemltreedratio,
                                double  chhttipspratio,
                                String  ancnodes,
                                boolean doplot,
                                String  finalplotkind);

        }

            public boolean DrawgramRun(String fileName){
                    TestFileNames test = new TestFileNames();

                    if (!test.FileAvailable(fileName, "Intree"))
                    {
```

```
                return false;
        }

        // at this point we hook into the C code
        String wherestr = "System.load";
        try
        {
                wherestr = "Native.loadLibrary";

                URL url = getClass().getResource("/font/font3");
            BufferedReader in = new BufferedReader(new InputStreamReader(url.openStream()));

            String inputLine;
            String content = "";
            while ((inputLine = in.readLine()) != null){
                content += inputLine + "\n";
            }
            File file = new File("font3");

                // if file doesnt exists, then create it
                if (!file.exists()) {
                        file.createNewFile();
                }

                FileWriter fw = new FileWriter(file.getAbsoluteFile());
                BufferedWriter bw = new BufferedWriter(fw);
                bw.write(content);
                bw.close();

                String fullPath = ExportResource("/exe/drawgram.dll");

                Drawgram Drawgram = (Drawgram) Native.loadLibrary(
                                "drawgram", Drawgram.class);
                Drawgram.drawgram(
                        fileName,
                        "Courier",
                        "plotfile.ps",
                        "wb",
                        "horizontal",
                        "phenogram",
                        true,
                        90.0,
                        true,
                        10.0,
                        0.53,
                        0.05,
                        0.333,
                        "Centered",
                        false,
                        "Postscript"
        );
                return true;
        }
        catch (UnsatisfiedLinkError e)
        {
                String mapedLibName =
                                System.mapLibraryName("drawgram");
                String libpath =
                                System.getProperty("user.dir");;
                if (mapedLibName.contains("jnilib"))
                {
                        // mac
                        libpath += "/libdrawgram.dylib";
                }
                else if (mapedLibName.contains("dll"))
                {
                        // windows
                        String s1 = getClass().
                                        getResource("/exe/drawgram.dll").toString();
                        s1 = s1.substring(s1.indexOf('C'), s1.length() - 4);
                        libpath = s1 + ".dll";
                }
                else
                {
                        // unix
                        libpath += "/libdrawgram.so";
                }
                String msg = "Drawgram library not found in : ";
                msg += libpath;
                msg += " by ";
                msg += wherestr;
                msg += ". Error msg: ";
                msg += e;
                System.out.println(msg);

        } catch (Exception e) {
                e.printStackTrace();
        }
        return false;
}

public String ExportResource(String resourceName)
                throws Exception {
        InputStream stream = null;
        OutputStream resStreamOut = null;

        String file = resourceName.
                        substring(resourceName.indexOf('/') + 1,
                                        resourceName.length());
```

```
                        file = file.substring(file.indexOf('/') + 1,
                                        file.length());
                        System.out.println("file: " + file);
                        String jarFolder;
                        try{
                                stream = com.View.PhenTor.class.
                                                getResourceAsStream(resourceName);

                                if(stream == null){
                                        throw new Exception("Cannot get resource \"" +
                                resourceName +"\" from Jar file.");
                                }
                                System.out.println("1");
                                int readBytes;
                                byte[] buffer = new byte[4096];
                                jarFolder = new File(com.View.PhenTor.class.
                                                getProtectionDomain().getCodeSource().
                                                getLocation().toURI().getPath())
                                                .getParentFile().getPath().replace('\\', '/');
                                System.out.println(jarFolder);
                                resStreamOut = new FileOutputStream(
                                                jarFolder + "/" + file);
                                System.out.println("init res");
                                while((readBytes = stream.read(buffer)) > 0){

                                        resStreamOut.write(buffer, 0, readBytes);
                                }
                                System.out.println("*2");

                        } catch (Exception e){
                                throw e;
                        } finally {
                                stream.close();
                                if(resStreamOut != null)
                                        resStreamOut.close();
                        }

                        return jarFolder + "/" + file;
                }
}
```

# 4. PHYLIP.Util

1. DisplayProgress.java

```
package com.PHYLIP.Util;
import java.awt.Font;
import java.io.FileNotFoundException;
import java.io.File;
import java.util.Scanner;

import javax.swing.JFrame;
import javax.swing.JOptionPane;
import javax.swing.JScrollPane;
import javax.swing.JTextArea;
import javax.swing.ScrollPaneConstants;

@SuppressWarnings("serial")
public class DisplayProgress extends JFrame
{
        public DisplayProgress(
                        String progressfilename, String progressfile) // constructor
        {
            super(progressfilename); // label frame
                //Font useFont = new Font("Monospaced", Font.PLAIN, 12);
        JTextArea ta = new JTextArea(200,400);
        ta.setFont(new Font("Monospaced", Font.PLAIN, 12));
        JScrollPane jsp = new JScrollPane(
                        ta,ScrollPaneConstants.VERTICAL_SCROLLBAR_AS_NEEDED,
                        ScrollPaneConstants.HORIZONTAL_SCROLLBAR_AS_NEEDED);
            try
            {

                        Scanner sf = new Scanner(new File(progressfile));
                        while (sf.hasNextLine())
                        {
                                String curline = sf.nextLine();
                                ta.append(curline);
                                ta.append("\n");
                        }
                        sf.close();
                        ta.setEditable(false);
            jsp.getViewport().add(ta);
            this.getContentPane().add(jsp);
            this.setLocation(800, 200);
            this.setSize(600, 400);
                    this.setVisible(true);

            }
                catch (FileNotFoundException e)
                {
                        String msg = "Progress file: ";
                        msg += progressfile;
                        msg += " does not exist.";
                        JOptionPane.showMessageDialog(null,
                                        msg, "Error", JOptionPane.ERROR_MESSAGE);
                }
```

```
        }
    }
```

2. DrawPreview.java

```
package com.PHYLIP.Util;
import java.awt.*;

import javax.imageio.ImageIO;
import javax.swing.*;

import com.Controller.GUIHandler;
import com.PHYLIP.Util.PlotData;
import com.PHYLIP.Util.SectionData;



import java.awt.geom.*;

import java.lang.Math;

import java.io.File;
import java.io.FileNotFoundException;
import java.io.IOException;
import java.util.Scanner;
import java.awt.geom.Line2D;
import java.awt.image.BufferedImage;

public class DrawPreview
{

 double yadd = 400;
 public Canvas canv;public DrawPreview(String plotfilename,
                    String plotfile, String path, boolean isExp) // constructor
        {
            try
            {
                PlotData curplot = ReadPlotFile(plotfile);

                    if (curplot != null)
                    {
                            GCanvas canvas=new GCanvas(curplot, true);      // create drawing canvas
                        canvas.setBackground(Color.WHITE);

                            int max_size = 0;
                            if(isExp)
                            for (int i=0; i<curplot.m_treePart.size(); i++) {
                                    SectionData section = curplot.m_treePart.get(i);

                                    for(int j = 0 ; j < section.texts.size() ; j++){
                                            LabelData text = section.texts.get(j);
                                            text.m_displayText =
                                                    GUIHandler.options.get(0).get(
                                                            GUIHandler.options.get(1).
                                                            indexOf(text.m_displayText));
                                            System.out.println("text: " + text.m_displayText);
                                            if(text.m_displayText.length() > max_size)
                                                    max_size = (int) (
                                                            text.m_displayText.length()*
                                                            (text.m_useFont.getSize2D() + 25));
                                    }
                            }

                            System.out.println("max size: " + max_size);
                            BufferedImage image=new BufferedImage(
                                            curplot.m_plotwidth+30 + max_size,
                                            curplot.m_plotheight,BufferedImage.TYPE_INT_RGB);

                            Graphics2D g2=(Graphics2D)image.getGraphics();
                            g2.setBackground(Color.WHITE);
                            canvas.paint(g2);

                        // write the captured image as a PNG

                        try {
                                if(!isExp){
                                        String s = getClass().
                                                        getResource("/" + path
                                                        + "/phenom.png").toString();
                                        s = s.substring(s.indexOf('C'), s.length());
                                        //ImageIO.write(image, "png", new File(s));
                                        ImageIO.write(image, "png",
                                                        new File("phenom.png"));
                                }else{
                                        ImageIO.write(image, "png",
                                                        new File(path+"/phenom.png"));
                                }
                                } catch (IOException e1) {
                                        e1.printStackTrace();
                                }
                        }
                }
                    catch (FileNotFoundException e)
                    {
                            String msg = "Plot file: ";
                            msg += plotfilename;
                            msg += " does not exist.";
                            JOptionPane.showMessageDialog(null,
```

```
                                msg, "Error", JOptionPane.ERROR_MESSAGE);

        }

}

public PlotData ReadPlotFile(String plotfilename)
                throws FileNotFoundException// constructor
{
        PlotData curplot = new PlotData();
        SectionData cursec = new SectionData();
        cursec.strokewidth = -1.0;
        Scanner scanfile = new Scanner(new File("plotfile.ps"));
        boolean pagefound = false;
        while (scanfile.hasNextLine())
        {
                String curline = scanfile.nextLine();
            Scanner scanline = new Scanner(curline);
            scanline.useDelimiter(" ");
            if (scanline.hasNext())
            {
                        if (pagefound)
                        {
                                if (curline.contains(" l"))
                                {
                                        // pick up line limits
                                        int index = 0;
                                        Double xbeg = 0.0;
                                        Double ybeg = 0.0;
                                        Double xend = 0.0;
                                        Double yend = 0.0;
                                        while (scanline.hasNext())
                                        {
                                                if(scanline.hasNextDouble())
                                                {
                                                        switch (index)
                                                        {
                                                                case 0: xbeg = scanline.nextDouble(); bre
                                                                case 1: ybeg = scanline.nextDouble(); bre
                                                                case 2: xend = scanline.nextDouble(); bre
                                                                case 3: yend = scanline.nextDouble(); bre
                                                        }
                                                        index+=1;
                                                }
                                                else
                                                {
                                                        scanline.next();
                                                }
                                        }

                                        // correct for inverted y value
                                        ybeg = curplot.m_plotheight - ybeg;
                                        yend = curplot.m_plotheight - yend;

                                        // make line
                                        Line2D.Double cur2d = new Line2D.Double(xbeg,ybeg,xend,yend);
                                        cursec.lines.add(cur2d);
                                }
                                else if (curline.contains(" moveto"))
                                {
                                        // Cubic Bezier curve is on two lines
                                        // the first defines the start of the curve
                                        int index = 0;
                                        Double xbeg  = 0.0;
                                        Double ybeg  = 0.0;
                                        Double xoff1 = 0.0;
                                        Double yoff1 = 0.0;
                                        Double xoff2 = 0.0;
                                        Double yoff2 = 0.0;
                                        Double xend  = 0.0;
                                        Double yend  = 0.0;
                                        while (scanline.hasNext())
                                        {
                                                if(scanline.hasNextDouble())
                                                {
                                                        switch (index)
                                                        {
                                                                case 0: xbeg = scanline.nextDouble(); bre
                                                                case 1: ybeg = scanline.nextDouble(); bre
                                                        }
                                                        index+=1;
                                                }
                                                else
                                                {
                                                        scanline.next();
                                                }
                                        }

                                        // the second defines the two off curve points and the end of the
                                        curline = scanfile.nextLine();
                                    scanline.close();
                                        scanline = new Scanner(curline);
                                        index = 0;
                                        while (scanline.hasNext())
                                        {
                                                if(scanline.hasNextDouble())
                                                {
                                                        switch (index)
                                                        {
```

```
                                                case 0: xoff1 = scanline.nextDouble(); br
                                                case 1: yoff1 = scanline.nextDouble(); br
                                                case 2: xoff2 = scanline.nextDouble(); br
                                                case 3: yoff2 = scanline.nextDouble(); br
                                                case 4: xend  = scanline.nextDouble(); br
                                                case 5: yend   = scanline.nextDouble(); b
                                        }
                                        index+=1;
                                }
                                else
                                {
                                        scanline.next();
                                }
                        }

                        // correct for inverted y value
                        ybeg  = curplot.m_plotheight - ybeg;
                        yoff1 = curplot.m_plotheight - yoff1;
                        yoff2 = curplot.m_plotheight - yoff2;
                        yend  = curplot.m_plotheight - yend;

                        // make curve
                        CubicCurve2D.Double cube2d
                        = new CubicCurve2D.Double(xbeg,ybeg,xoff1,yoff1,xoff2,yoff2,xend,
                        cursec.curves.add(cube2d);


                }

        }

        if (curline.contains("DocumentMedia:"))
        {
                // get image dimensions
                int index = 0;
                while (scanline.hasNext())
                {
                        if(scanline.hasNextInt())
                        {
                                if (index == 0)
                                {
                                        curplot.m_plotwidth = scanline.nextInt();
                                }
                                else if (index == 1)
                                {
                                        curplot.m_plotheight = scanline.nextInt();
                                }
                                else
                                {
                                        scanline.next();
                                }
                                index+= 1;
                        }
                        else
                        {
                                scanline.next();
                        }
                }
        }
        else if (curline.contains("setlinewidth"))
        {
                // get line width
                if (cursec.strokewidth > 0)
                {
                        // output previous section if it exists
                        curplot.m_treePart.add(cursec);
                }
                cursec = new SectionData();
                while (scanline.hasNext())
                {
                        if(scanline.hasNextDouble())
                        {
                                cursec.strokewidth = scanline.nextDouble();
                        }
                        else
                        {
                                scanline.next(); // throw away misc text
                        }
                }
        }
        else if (curline.contains("Page:"))
        {
                pagefound = true;
        }
        else if (curline.contains("findfont"))
        {
                // text output is on 4 lines
                Font useFont;
                Point.Double translation = new Point.Double(0,0);
                Double rotation = 0.0;
                String displayText = new String("");

                // first line has the font name and the scaling
                String fontstring = scanline.next().replace('/', ' ');
                fontstring = fontstring.trim();
                String []fontparts = fontstring.split("-");

                int fontsize = 0;
                while(scanline.hasNext())
```

```
        {
                if (scanline.hasNextDouble())
                {
                        fontsize = (int)scanline.nextDouble();
                        fontsize *= 0.8;
                }
                else
                {
                        scanline.next(); // throw away misc text
                }
        }
        String fontkind;
        if (fontparts.length == 1)
        {
                // Courier & Helvetica
                fontkind = "book";
        }
        else if (fontparts.length == 3)
        {
                // Helvetica-Narrow
                fontkind = fontparts[2].toLowerCase();
        }
        else
        {
                fontkind = fontparts[1].toLowerCase();
        }

        if (fontkind.contains("roman") ||
                fontkind.contains("book") ||
                fontkind.contains("narrow") ||
                fontkind.contains("light"))
        {
                useFont = new Font(fontparts[0], Font.PLAIN, fontsize);
        }
        else if(fontkind.contains("bolditalic") ||
                        fontkind.contains("boldoblique") ||
                        fontkind.contains("demioblique") ||
                        fontkind.contains("demiitalic"))
        {
                useFont
                = new Font(fontparts[0], Font.ITALIC+Font.BOLD, fontsize);
        }
        else if(fontkind.contains("bold") ||
                        fontkind.contains("demi") )
                {
                        useFont = new Font(fontparts[0], Font.BOLD, fontsize);
                }
        else if(fontkind.contains("italic") ||
                        fontkind.contains("oblique") ||
                        fontkind.contains("bookoblique") ||
                        fontkind.contains("lightitalic") ||
                        fontkind.contains("mediumitalic"))
                {
                        useFont
                        = new Font(fontparts[0], Font.ITALIC, fontsize);
                }
        else
        {
                useFont = new Font("SanSerif", Font.PLAIN, fontsize);

        }

        curline = scanfile.nextLine();

scanline.close();
        scanline = new Scanner(curline);
        boolean xfound = false;
        boolean yfound = false;
        while(scanline.hasNext())
        {
                if (scanline.hasNextDouble())
                {
                        if (!xfound)
                        {
                                translation.x = scanline.nextDouble();
                                xfound = true;
                        }
                        else if (!yfound)
                        {
                                translation.y = scanline.nextDouble();
                                translation.y = curplot.m_plotheight
                                                - translation.y;
                                // inverted y correction
                                yfound = true;
                        }
                        else
                        {
                                rotation = -scanline.nextDouble();
                                // inverted y correction
                        }
                }
                else
                {
                        scanline.next(); // throw away misc text
                }
        }

        // third line is a (0,0) moveto
        curline = scanfile.nextLine();
```

```
                                        scanline.close();
                                            scanline = new Scanner(curline);

                                            // last line is (name) show
                                            curline = scanfile.nextLine();
                                            if(curline.contains("("))
                                            {
                                                    displayText
                                                    = curline.substring(
                                                                    (curline.indexOf('(') + 1),
                                                                    curline.lastIndexOf(')'));
                                            }

                                            // make text block
                                            LabelData newlabel
                                            = new LabelData(useFont, translation, rotation, displayText);
                                            cursec.texts.add(newlabel);
                                    }
                            }

                            scanline.close();
                    }

                    // output final section if it exists
                    if (cursec.strokewidth > 0)
                    {
                            curplot.m_treePart.add(cursec);
                    }

                    scanfile.close();
                    return curplot;
            }
    }

    @SuppressWarnings("serial")
    class GCanvas extends Canvas // create a canvas for your graphics
    {
            private PlotData locplot;
            public int addY = 170;
            boolean isDraw;
            public GCanvas(PlotData curplot, boolean isDraw)
            {
                    locplot = curplot;
                    this.isDraw = isDraw;
            }

            public void setY(int y){
                    addY = y;
            }

            @Override
            public void paint(Graphics g) // display shapes on canvas
              {
    ;
                Graphics2D g2D=(Graphics2D) g; // cast to 2D
                g2D.setRenderingHint(RenderingHints.KEY_ANTIALIASING,
                                     RenderingHints.VALUE_ANTIALIAS_ON);
                g2D.setBackground(Color.WHITE);
                    //System.out.println("in paint");
                    for (int i=0; i<locplot.m_treePart.size(); i++)
                    {
                            SectionData section = locplot.m_treePart.get(i);
                            setStroke(g2D, section.strokewidth);

                            for (int j=0; j<section.lines.size(); j++)
                            {
                                    Line2D.Double line = section.lines.get(j);
                                    if(!isDraw){
                                            line.y1 -= addY;
                                            line.y2 -= addY;

                                            line.y1 *= 0.52;
                                            line.y2 *= 0.52;
                                    }

                                    drawLine(g2D, line);
                            }
                            for (int j=0; j<section.curves.size(); j++)
                            {
                                    //drawCubic(g2D, section.curves.get(j));
                            }
                            for (int j=0; j<section.texts.size(); j++)
                            {
                                    showText(g2D,
                                                    section.texts.get(j),
                                                    (int) (locplot.m_plotwidth*.52),
                                                    (int) (locplot.m_plotheight*.52));
                            }

                    }
              }

              public void setStroke(Graphics2D g2D, double width)
              {
                      BasicStroke stroke1
                      = new BasicStroke((float)width,
                                        java.awt.BasicStroke.CAP_ROUND,
                                        java.awt.BasicStroke.JOIN_ROUND);
```

```
                g2D.setStroke(stroke1);
        }

        public void drawLine(Graphics2D g2D,

                        double x1, double y1, double x2, double y2)
        {
                Line2D.Double line1 = new Line2D.Double(x1,y1,x2,y2);
                g2D.draw(line1);
        }

        public void drawLine(Graphics2D g2D, Line2D.Double line1)
        {
                g2D.draw(line1);
        }

        public void drawArc(Graphics2D g2D,
                        double x1, double y1, double x2, double y2, double sd, double rd, int cl)
        {
                Arc2D.Double arc1=new Arc2D.Double(x1,y1,x2,y2,sd,rd,cl);
                g2D.fill(arc1);
        }

        public void drawEllipse(Graphics2D g2D,
                        double x1,double y1,double x2,double y2)
        {
                Ellipse2D.Double oval1=new Ellipse2D.Double(x1,y1,x2,y2);
                g2D.fill(oval1);
        }

        public void drawCubic(Graphics2D g2D,
                        double x1,double y1,double x2,double y2,double x3,double y3,double x4,double y4)
        {
                CubicCurve2D.Double curve1=new CubicCurve2D.Double(x1,y1,x2,y2,x3,y3,x4,y4);
                g2D.fill(curve1);
        }

        public void drawCubic(Graphics2D g2D, CubicCurve2D.Double curve1)
        {
                g2D.draw(curve1);
        }

        public void showText(Graphics2D g2D,
                        LabelData text, int dispWidth, int dispHeight)
        {
                AffineTransform oldat = g2D.getTransform();

                // have to add rotation after translate
                // if you do it the other way the rotation
                // gets applied to the translation values
                AffineTransform newat = new AffineTransform();
                newat.translate((int)
                                (text.m_translation.x), (int)(text.m_translation.y) );
                newat.rotate(Math.toRadians(text.m_rotation));

                g2D.setTransform(newat);
                g2D.setFont(text.m_useFont);
                if(isDraw){
                        g2D.drawString(text.m_displayText,0,0);
                } else
                        g2D.drawString(text.m_displayText,-5,(addY+70)*(-1));

                //System.out.println(text.m_displayText);
                //g2D.drawString(text.m_displayText,0,0);
                // may be unnecessary, but safer
                g2D.setTransform(oldat);
        }
}
```

3. LabelData.java

```
package com.PHYLIP.Util;
import java.awt.Font;
import java.awt.Point;

public class LabelData {
        Font m_useFont;
        Point.Double m_translation;
        Double m_rotation;
        String m_displayText;
        public LabelData()
        {
                m_useFont = new Font("SanSerif", Font.PLAIN, 12);
                m_translation = new Point.Double(0,0);
                m_rotation = 0.0;
                m_displayText = new String("undefined");
        }

        public LabelData(Font useFont, Point.Double translation,
                        Double rotation, String displayText)
        {
                m_useFont = useFont;
                m_translation = translation;
                m_rotation = rotation;
                m_displayText = displayText;
        }
}
```

4. PhylipFileDialog.java

```
package com.PHYLIP.Util;
import java.awt.EventQueue;

import javax.swing.JFrame;
import javax.swing.JTextField;

import javax.swing.JPanel;
import javax.swing.JLabel;
import javax.swing.SwingConstants;
import java.awt.event.ActionListener;
import java.awt.event.ActionEvent;
import javax.swing.JButton;
import java.awt.Color;

public class PhylipFileDialog {
	private JFrame frame;
	/**
	 * @wbp.nonvisual location=991,851
	 */
	private final JLabel lblTitle = new JLabel("WARNING");
	private final JTextField txtFile = new JTextField();
	private final JLabel lblFile = new JLabel("File:");
	private final JLabel lblExists = new JLabel("already exists. ");
	private final JButton btnReplace = new JButton("Replace");
	private final JButton btnAppend = new JButton("Append");
	private final JButton btnChoose = new JButton("Choose a new file");
	private final JButton btnQuit = new JButton("Quit");

	public enum FileAction{REPLACE, APPEND, QUIT}

	public class phylipAction{
		FileAction retval;
	}

	/**
	 * Launch the application.
	 */
	public static void main(String[] args) {
		EventQueue.invokeLater(new Runnable() {
			@Override
			public void run() {
				try {
					PhylipFileDialog window = new PhylipFileDialog();
					window.frame.setVisible(true);
				} catch (Exception e) {
					e.printStackTrace();
				}
			}
		});
	}

	/**
	 * Create the application.
	 */
	public PhylipFileDialog() {
		initialize();
	}

	protected void FileActionToggle(FileAction kind) {
		 phylipAction action = new phylipAction();
		 action.retval = kind;
		 /*
		 try{
			this.finalize();
		 } catch (Exception e) {
				e.printStackTrace();
		}
		 */
	}

	/**
	 * Initialize the contents of the frame.
	 */
	private void initialize() {
		txtFile.setBounds(42, 35, 336, 28);
		txtFile.setColumns(10);
		frame = new JFrame();
		frame.setBounds(100, 100, 500, 150);
		frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
		frame.getContentPane().setLayout(null);

		JPanel panel = new JPanel();
		panel.setBounds(0, 0, 494, 23);
		frame.getContentPane().add(panel);
		lblTitle.setHorizontalAlignment(SwingConstants.CENTER);
		lblTitle.setForeground(Color.RED);

		panel.add(lblTitle);

		frame.getContentPane().add(txtFile);
		lblFile.setBounds(10, 41, 36, 16);

		frame.getContentPane().add(lblFile);
		lblExists.setBounds(384, 41, 95, 16);

		frame.getContentPane().add(lblExists);
```

```
                btnReplace.addActionListener(new ActionListener() {
                        @Override
                        public void actionPerformed(ActionEvent e) {
                                FileActionToggle(FileAction.REPLACE);
                        }
                });
                btnReplace.setBounds(10, 75, 104, 29);
                frame.getContentPane().add(btnReplace);

                btnAppend.addActionListener(new ActionListener() {
                        @Override
                        public void actionPerformed(ActionEvent e) {
                                FileActionToggle(FileAction.APPEND);
                        }
                });
                btnAppend.setBounds(122, 75, 95, 29);
                frame.getContentPane().add(btnAppend);

                btnChoose.addActionListener(new ActionListener() {
                        @Override
                        public void actionPerformed(ActionEvent e) {
                                FileActionToggle(FileAction.QUIT);
                        }
                });
                btnChoose.setBounds(218, 75, 160, 29);
                frame.getContentPane().add(btnChoose);

                btnQuit.addActionListener(new ActionListener() {
                        @Override
                        public void actionPerformed(ActionEvent e) {
                                FileActionToggle(FileAction.QUIT);
                        }
                });
                btnQuit.setBounds(384, 75, 100, 29);
                frame.getContentPane().add(btnQuit);
        }

}
```

5. PlotData.java

```
package com.PHYLIP.Util;

import java.util.ArrayList;

public class PlotData {
                public int m_plotwidth;
                public int m_plotheight;
                public ArrayList <SectionData> m_treePart;
                public PlotData()
                {
                        m_treePart = new ArrayList <SectionData>();
                }
}
```

6. SectionData.java

```
package com.PHYLIP.Util;

import java.awt.geom.CubicCurve2D;
import java.awt.geom.Line2D;
import java.util.ArrayList;

public class SectionData {
         public Double strokewidth;
         public ArrayList <Line2D.Double> lines;
         public ArrayList <CubicCurve2D.Double> curves;
         public ArrayList <LabelData> texts;
         public SectionData()
         {
                 strokewidth = -1.0;
                 texts = new ArrayList<LabelData>();
                 lines = new ArrayList<Line2D.Double>();
                 curves = new ArrayList<CubicCurve2D.Double>();
         }
}
```

7. TestFileNames.java

```
package com.PHYLIP.Util;

import java.io.File;
import java.io.IOException;

import javax.swing.JOptionPane;

public class TestFileNames {
         public boolean DuplicateFileNames(
                         String file1, String file1name, String file2, String file2name)
         {
                 File testfile1 = new File(file1);
                 File testfile2 = new File(file2);

                 if (testfile1.exists() && testfile2.exists()){

                         // check if file1 and file2 are the same
```

```
                        String file1path = "";
                        try {
                                file1path = testfile1.getCanonicalPath();
                        } catch (IOException e) {
                                // should never happen
                                e.printStackTrace();
                        }

                        String file2path = "";
                        try {
                                file2path = testfile2.getCanonicalPath();
                        } catch (IOException e) {
                                // should never happen
                                e.printStackTrace();
                        }

                        if (file1path.equals(file2path))
                        {
                                String msg = file1name;
                                msg += " and ";
                                msg += file2name;
                                msg += " files are both named \"";
                                msg += file1;
                                msg += "\" which will not work.";
                                JOptionPane.showMessageDialog(null, msg,
                                                "Error", JOptionPane.ERROR_MESSAGE);
                                return false;
                        }
                }
                return true;
        }

        public boolean FileAvailable(String file, String filename)
        {
                File infile = new File(file);
                if (!infile.exists()){
                        String msg = filename;
                        msg += " *File: ";
                        msg += file;
                        msg += " does not exist.";
                        JOptionPane.showMessageDialog(null, msg,
                                        "Error", JOptionPane.ERROR_MESSAGE);
                        return false;
                }
                return true;
        }

        public String FileAlreadyExists(String file, String filename)
        {
                Object[] options = {"Quit", "Append", "Replace"};

                File outfile = new File(file);
                if (outfile.exists()){
                        String msg = filename;
                        msg += " File: ";
                        msg += file;
                        msg += " exists. Overwrite?";
                        int retval = JOptionPane.showOptionDialog(null,
                                        msg, "Warning", JOptionPane.
                                        YES_NO_CANCEL_OPTION, JOptionPane.
                                        WARNING_MESSAGE, null,options,options[0]);
                        if (retval == JOptionPane.CANCEL_OPTION){
                                return "w";
                        } else{
                                if (retval == JOptionPane.NO_OPTION){
                                        return "a";
                                } else{
                                        return "q";
                                }
                        }
                }
                return "w";
        }
}
```

# 5. View

1. PhenTor.java

```
package com.View;

import java.awt.BorderLayout;
import java.awt.Color;
import java.awt.Container;
import java.awt.Cursor;
import java.awt.Dimension;
import java.awt.EventQueue;
import java.awt.Font;
import java.awt.GridLayout;
import java.awt.Image;
import java.awt.Toolkit;
import java.awt.event.ActionEvent;
import java.awt.event.ActionListener;
import java.awt.event.WindowEvent;
import java.awt.event.WindowAdapter;
import java.awt.image.BufferedImage;
import java.io.BufferedReader;
```

```
import java.io.BufferedWriter;
import java.io.File;
import java.io.FileWriter;
import java.io.IOException;
import java.io.InputStreamReader;
import java.net.InetSocketAddress;
import java.net.Socket;
import java.net.URL;
import java.util.ArrayList;

import javax.imageio.ImageIO;
import javax.swing.BorderFactory;
import javax.swing.ButtonGroup;
import javax.swing.DefaultListModel;
import javax.swing.ImageIcon;
import javax.swing.JButton;
import javax.swing.JComboBox;
import javax.swing.JFileChooser;
import javax.swing.JFrame;
import javax.swing.JLabel;
import javax.swing.JList;
import javax.swing.JMenu;
import javax.swing.JMenuBar;
import javax.swing.JMenuItem;
import javax.swing.JOptionPane;
import javax.swing.JPanel;
import javax.swing.JProgressBar;
import javax.swing.JRadioButton;
import javax.swing.JScrollPane;
import javax.swing.JTable;
import javax.swing.JTextArea;
import javax.swing.JTextField;
import javax.swing.ListSelectionModel;
import javax.swing.ScrollPaneConstants;
import javax.swing.SpringLayout;
import javax.swing.SwingWorker;
import javax.swing.UIManager;
import javax.swing.UnsupportedLookAndFeelException;
import javax.swing.filechooser.FileNameExtensionFilter;

import com.Controller.GUIHandler;
import com.PHYLIP.DrawGram.DrawgramInterface;
import com.PHYLIP.Util.DrawPreview;

@SuppressWarnings({ "serial" })
public class PhenTor
        extends JFrame implements ActionListener{

        private static final Font FONT
                = new Font("Helvetica",Font.PLAIN ,10);

        public final String IMG_FOLDER
                = "/images/open_file.png";
        public final String IMG_PLAY
                = "/images/train.jpg";
        public final String IMG_WELCOME
                = "/images/welcome.jpg";
        public final String IMG_ADD
                = "/images/add-icon.png";
        public final String IMG_GRID
                = "/images/grid.png";
        public final String IMG_LOADING
                = "/images/loading.gif";
        public final String IMG_ADD2
                = "/images/addsp.png";
        public final String IMG_REMOVE
                = "/images/remsp.png";
        public final String IMG_PHENOM
                = "files/phenom.png";

        private final ImageIcon ICON_OPEN
                = new ImageIcon(
                                getClass().
                                getResource(IMG_FOLDER));
        private final ImageIcon ICON_PLAY
                = new ImageIcon(
                                getClass().
                                getResource(IMG_PLAY));
        private final ImageIcon ICON_WELCOME
                = new ImageIcon(
                                getClass().
                                getResource(IMG_WELCOME));
        private final ImageIcon ICON_ADD
                = new ImageIcon(
                                getClass().
                                getResource(IMG_ADD));
        private final ImageIcon ICON_LOADING
                = new ImageIcon(
                                getClass().
                                getResource(IMG_LOADING));
        private final ImageIcon ICON_ADD2
                = new ImageIcon(
                                getClass().
                                getResource(IMG_ADD2));
        private final ImageIcon ICON_REMOVE
                = new ImageIcon(
                                getClass().
                                getResource(IMG_REMOVE));
```

```
private final int HEIGHT = 550;
private final int WIDTH = 1000;

private final int SCREEN_HEIGHT
        = (int) Toolkit.
        getDefaultToolkit().
        getScreenSize().getHeight() - 30;
private final int SCREEN_WIDTH
        = (int) Toolkit.
        getDefaultToolkit().
        getScreenSize().getWidth();


private final int build = 1;
private final int fs = 2;
private final int fp = 3;
private final int gs = 4;

String[] data;
GUIHandler guih;
String guihout;
DrawPreview dp;

ArrayList<String> indexSpe;
int choice_;
boolean isOpen = false;

int pwayI;
String spe;

public PhenTor() throws IOException{
        try {
    UIManager.setLookAndFeel(
                UIManager.
                getSystemLookAndFeelClassName());
} catch (ClassNotFoundException
                | InstantiationException
                | IllegalAccessException
                | UnsupportedLookAndFeelException ex) {
}

        init();

        frame = this;
        c = frame.getContentPane();

        frame.setSize(WIDTH, HEIGHT);
        frame.setDefaultCloseOperation(
                        JFrame.DO_NOTHING_ON_CLOSE);
    frame.addWindowListener(new WindowAdapter() {
        @Override
        public void windowClosing
                (WindowEvent event) {
            deleteFiles();
        }
    });
        frame.setLocationRelativeTo(null);
        frame.setResizable(false);
        frame.setTitle("Methabolic Pathway");

        cSL = new SpringLayout();
        frame.setLayout(cSL);

        createMenuBar();
        setInputPanel();

        setOutputPanel();
        setProgressBar();

        welcomePanel();


        frame.setTitle("PhenTor");
        URL url = getClass()
                        .getResource("/image/icon.png");
        frame.setIconImage(ImageIO.read(url));
}

public void init(){
        (new GUIHandler()).getOptions();
}

public void setOptions
        (DefaultListModel<String> model){
        for(String specie : GUIHandler.options.get(0))
                model.addElement(specie);
}

public void deleteFiles(){
        File file = new File("outfile");
        if(file.exists())
                file.delete();

        file = new File("plotfile.ps");
        if(file.exists())
                file.delete();

        file = new File("intree");
        if(file.exists())
```

```
                file.delete();

        file = new File("font3");
        if(file.exists())
                file.delete();

        file = new File("phenom.png");
        if(file.exists())
                file.delete();

        file = new File("treedata");
        if(file.exists())
                file.delete();

        file = new File("treedata2");
        if(file.exists())
                file.delete();

        file = new File("drawgram.dll");
        if(file.exists())
                file.delete();

        System.exit(1);
}

private void createMenuBar() {
        menuBar = new JMenuBar();
        menuBar.setPreferredSize(
                        new Dimension(frame.getWidth(), 30));
        frame.setJMenuBar(menuBar);

        file = new JMenu("File");
        menuBar.add(file);

        eTxt = new JMenuItem("Save Current Output");
        eTxt.addActionListener(this);
        eTxt.setEnabled(false);
        file.add(eTxt);

        exit = new JMenuItem("Exit");
        exit.addActionListener(new ActionListener() {
                @Override
                public void actionPerformed(ActionEvent e) {
                        System.exit(1);
                }
        });
        file.add(exit);

        online = new JMenu("Online Features");
        menuBar.add(online);

        view = new JMenuItem(
                        "View/Download Metabolic Pathway "+
                        "(Glycolysis, Krebb)");
        view.addActionListener(new ActionListener() {
                @Override
                public void actionPerformed(ActionEvent e) {
                        setUpView();
                }
        });
        online.add(view);
}

public void setInputPanel() {
        inputPanel = new JPanel();

        inputPanel.setFont(FONT);
        inputPanel.setBorder(
                        BorderFactory.
                        createTitledBorder("PhenTor Functions"));
        inputPanel.setBackground(Color.WHITE);

        inputSL = new SpringLayout();
        inputPanel.setLayout(inputSL);

        inputJsp = new JScrollPane(inputPanel);
        inputJsp.setVerticalScrollBarPolicy(
                        ScrollPaneConstants.VERTICAL_SCROLLBAR_NEVER);
        cSL.putConstraint(SpringLayout.NORTH,
                        inputJsp, 0, SpringLayout.NORTH, c);
        cSL.putConstraint(SpringLayout.WEST,
                        inputJsp, 0, SpringLayout.WEST, c);
        cSL.putConstraint(SpringLayout.EAST,
                        inputJsp, 270, SpringLayout.WEST, c);
        cSL.putConstraint(SpringLayout.SOUTH,
                        inputJsp, -30, SpringLayout.SOUTH, c);
        frame.add(inputJsp);

        createInsertPanel();
}

public void createInsertPanel(){
        JLabel orgL = new JLabel("Organisms : ");
        inputSL.putConstraint(SpringLayout.NORTH,
                        orgL, 8, SpringLayout.NORTH, inputPanel);
        inputSL.putConstraint(SpringLayout.WEST,
                        orgL, 8, SpringLayout.WEST, inputPanel);
        inputPanel.add(orgL);
```

```
orgSel = new JTextField(13);
orgSel.setEditable(false);
inputSL.putConstraint(SpringLayout.NORTH,
                orgSel, 7, SpringLayout.NORTH, inputPanel);
inputSL.putConstraint(SpringLayout.WEST,
                orgSel, 3, SpringLayout.EAST, orgL);
inputPanel.add(orgSel);

addB = new JButton(ICON_ADD);
addB.setToolTipText("Add Species To Test");
addB.setCursor(new Cursor(Cursor.HAND_CURSOR));
addB.setBackground(null);
addB.setBorder(null);
addB.setBorderPainted(false);
addB.addActionListener(this);
inputSL.putConstraint(SpringLayout.NORTH,
                addB, 6, SpringLayout.NORTH, inputPanel);
inputSL.putConstraint(SpringLayout.WEST,
                addB, 1, SpringLayout.EAST, orgSel);
inputPanel.add(addB);

JLabel indexL = new JLabel("Index : ");
inputSL.putConstraint(SpringLayout.NORTH,
                indexL, 10, SpringLayout.SOUTH, orgL);
inputSL.putConstraint(SpringLayout.WEST,
                indexL, 8, SpringLayout.WEST, inputPanel);
inputPanel.add(indexL);

String[] indices= {"Jaccard Index", "Hamming Distance"};
index = new JComboBox<String>(indices);
inputSL.putConstraint(SpringLayout.NORTH,
                index, 10, SpringLayout.SOUTH, orgL);
inputSL.putConstraint(SpringLayout.SOUTH,
                index, 30, SpringLayout.SOUTH, orgL);
inputSL.putConstraint(SpringLayout.WEST,
                index,  10, SpringLayout.EAST, indexL);
inputSL.putConstraint(SpringLayout.EAST,
                index, -10, SpringLayout.EAST, inputPanel);
inputPanel.add(index);

JLabel taskL = new JLabel("Choose Task: ");
inputSL.putConstraint(SpringLayout.NORTH,
                taskL, 10, SpringLayout.SOUTH, index);
inputSL.putConstraint(SpringLayout.WEST,
                taskL, 8, SpringLayout.WEST, inputPanel);
inputPanel.add(taskL);

buildR = new JRadioButton("Build Phenogram", true);
buildR.setBackground(Color.WHITE);
buildR.addActionListener(this);
inputSL.putConstraint(SpringLayout.NORTH,
                buildR, 3, SpringLayout.SOUTH, taskL);
inputSL.putConstraint(SpringLayout.WEST,
                buildR, 15, SpringLayout.WEST, inputPanel);
inputPanel.add(buildR);

gsR = new JRadioButton("Similarity of Species");
gsR.setBackground(Color.WHITE);
gsR.addActionListener(this);
inputSL.putConstraint(SpringLayout.NORTH,
                gsR, 1, SpringLayout.SOUTH, buildR);
inputSL.putConstraint(SpringLayout.WEST,
                gsR, 15, SpringLayout.WEST, inputPanel);
inputPanel.add(gsR);

fsR = new JRadioButton(
                "Find Reactions between Species");
fsR.setBackground(Color.WHITE);
fsR.addActionListener(this);
inputSL.putConstraint(SpringLayout.NORTH,
                fsR, 1, SpringLayout.SOUTH, gsR);
inputSL.putConstraint(SpringLayout.WEST,
                fsR, 15, SpringLayout.WEST, inputPanel);
inputPanel.add(fsR);

fpR = new JRadioButton(
                "Find Pathways using Reactions");
fpR.setBackground(Color.WHITE);
fpR.addActionListener(this);
inputSL.putConstraint(SpringLayout.NORTH,
                fpR, 1, SpringLayout.SOUTH, fsR);
inputSL.putConstraint(SpringLayout.WEST,
                fpR, 15, SpringLayout.WEST, inputPanel);
inputPanel.add(fpR);

task = new ButtonGroup();
task.add(buildR);
task.add(fsR);
task.add(fpR);
task.add(gsR);

taskPanel = new JPanel();
taskPanel.setFont(FONT);
taskPanel.setBackground(Color.WHITE);
taskSL = new SpringLayout();
taskPanel.setLayout(taskSL);
inputSL.putConstraint(SpringLayout.NORTH,
                taskPanel, 10, SpringLayout.SOUTH, fpR);
inputSL.putConstraint(SpringLayout.WEST,
```

```
                        taskPanel, 0, SpringLayout.WEST, inputPanel);
        inputSL.putConstraint(SpringLayout.EAST,
                        taskPanel, 260, SpringLayout.WEST, inputPanel);
        inputSL.putConstraint(SpringLayout.SOUTH,
                        taskPanel, 0, SpringLayout.SOUTH, inputPanel);
        inputPanel.add(taskPanel);

        setTaskPanel(buildR);
}

public void setTaskPanel(JRadioButton jb){
        taskPanel.removeAll();
        inputPanel.remove(taskPanel);

        taskPanel = new JPanel();
        taskPanel.setFont(FONT);
        taskPanel.setBackground(Color.WHITE);
        taskSL = new SpringLayout();
        taskPanel.setLayout(taskSL);
        inputSL.putConstraint(SpringLayout.NORTH,
                        taskPanel, 10, SpringLayout.SOUTH, fpR);
        inputSL.putConstraint(SpringLayout.WEST,
                        taskPanel, 0, SpringLayout.WEST, inputPanel);
        inputSL.putConstraint(SpringLayout.EAST,
                        taskPanel, 260, SpringLayout.WEST, inputPanel);
        inputSL.putConstraint(SpringLayout.SOUTH,
                        taskPanel, 0, SpringLayout.SOUTH, inputPanel);
        inputPanel.add(taskPanel);

        runB = new JButton(ICON_PLAY);
        runB.setToolTipText("Run Test");
        runB.setCursor(new Cursor(Cursor.HAND_CURSOR));
        runB.setBackground(null);
        runB.setBorder(null);
        runB.setBorderPainted(false);
        runB.addActionListener(this);
        taskSL.putConstraint(SpringLayout.WEST,
                        runB, 115, SpringLayout.WEST, taskPanel);

        if(jb == buildR || jb == gsR){
                taskSL.putConstraint(SpringLayout.NORTH,
                                runB, 0, SpringLayout.NORTH, taskPanel);
                taskPanel.add(runB);
        }else if(jb == fsR){
                JLabel treshL = new JLabel("Treshhold: ");
                taskSL.putConstraint(SpringLayout.NORTH,
                                treshL, 0, SpringLayout.NORTH, taskPanel);
                taskSL.putConstraint(SpringLayout.WEST,
                                treshL, 25, SpringLayout.WEST, taskPanel);
                taskPanel.add(treshL);

                treshT = new JTextField(3);
                taskSL.putConstraint(SpringLayout.NORTH,
                                treshT, 0, SpringLayout.NORTH, taskPanel);
                taskSL.putConstraint(SpringLayout.WEST,
                                treshT, 3, SpringLayout.EAST, treshL);
                taskSL.putConstraint(SpringLayout.EAST,
                                treshT, 30, SpringLayout.EAST, treshL);
                taskPanel.add(treshT);

                JLabel percentL = new JLabel("%");
                taskSL.putConstraint(SpringLayout.NORTH,
                                percentL, 0, SpringLayout.NORTH, taskPanel);
                taskSL.putConstraint(SpringLayout.WEST,
                                percentL, 2, SpringLayout.EAST, treshT);
                taskPanel.add(percentL);

                taskSL.putConstraint(SpringLayout.NORTH,
                                runB, 30, SpringLayout.NORTH, treshL);
                taskPanel.add(runB);
        }else if(jb == fpR){
                JLabel subL = new JLabel("Reaction List Path: ");
                taskSL.putConstraint(SpringLayout.NORTH,
                                subL, 0, SpringLayout.NORTH, taskPanel);
                taskSL.putConstraint(SpringLayout.WEST,
                                subL, 25, SpringLayout.WEST, taskPanel);
                taskPanel.add(subL);

                subPath = new JTextField(10);
                taskSL.putConstraint(SpringLayout.NORTH,
                                subPath, 0, SpringLayout.NORTH, taskPanel);
                taskSL.putConstraint(SpringLayout.WEST,
                                subPath, 2, SpringLayout.EAST, subL);
                taskPanel.add(subPath);

                openFile = new JButton(ICON_OPEN);
                openFile.setCursor(new Cursor(Cursor.HAND_CURSOR));
                openFile.setToolTipText("Open XML File");
                openFile.addActionListener(this);
                taskSL.putConstraint(SpringLayout.NORTH,
                                openFile, -5, SpringLayout.NORTH, taskPanel);
                taskSL.putConstraint(SpringLayout.WEST,
                                openFile, 2, SpringLayout.EAST, subPath);
                taskSL.putConstraint(SpringLayout.EAST,
                                openFile, -2, SpringLayout.EAST, taskPanel);
                taskSL.putConstraint(SpringLayout.SOUTH,
                                openFile, 30, SpringLayout.NORTH, taskPanel);
                taskPanel.add(openFile);
```

```
                taskSL.putConstraint(SpringLayout.NORTH,
                                runB, 30, SpringLayout.NORTH, subL);
                taskPanel.add(runB);
        }
        revalidate();
}

public void setOutputPanel() {
        outputPanel = new JPanel();
        outputPanel.setBackground(Color.WHITE);

        outputJsp = new JScrollPane(outputPanel);
        outputJsp.setVerticalScrollBarPolicy(
                        ScrollPaneConstants.VERTICAL_SCROLLBAR_AS_NEEDED);
        cSL.putConstraint(SpringLayout.NORTH,
                        outputJsp, 0, SpringLayout.NORTH, c);
        cSL.putConstraint(SpringLayout.WEST,
                        outputJsp, 0, SpringLayout.EAST, inputJsp);
        cSL.putConstraint(SpringLayout.EAST,
                        outputJsp, 0, SpringLayout.EAST, c);
        cSL.putConstraint(SpringLayout.SOUTH,
                        outputJsp, -30, SpringLayout.SOUTH, c);
frame.add(outputJsp);

outputSL = new SpringLayout();
        outputPanel.setLayout(outputSL);
}

public void welcomePanel(){

        outputPanel.removeAll();

        JLabel welcome = new JLabel(ICON_WELCOME);
        outputSL.putConstraint(SpringLayout.NORTH,
                        welcome, 0, SpringLayout.NORTH, outputPanel);
        outputSL.putConstraint(SpringLayout.WEST,
                        welcome, 0, SpringLayout.WEST, outputPanel);
        outputSL.putConstraint(SpringLayout.EAST,
                        welcome, 0, SpringLayout.EAST, outputPanel);
        outputSL.putConstraint(SpringLayout.SOUTH,
                        welcome, 0, SpringLayout.SOUTH, outputPanel);
outputPanel.add(welcome);
}

public void loadingPanel(){

        outputPanel.removeAll();

        JLabel load = new JLabel(ICON_LOADING);
        outputSL.putConstraint(SpringLayout.NORTH,
                        load, 0, SpringLayout.NORTH, outputPanel);
        outputSL.putConstraint(SpringLayout.WEST,
                        load, 0, SpringLayout.WEST, outputPanel);
        outputSL.putConstraint(SpringLayout.EAST,
                        load, 0, SpringLayout.EAST, outputPanel);
        outputSL.putConstraint(SpringLayout.SOUTH,
                        load, 0, SpringLayout.SOUTH, outputPanel);
outputPanel.add(load);
}


public void viewPanel(){
        outputPanel.removeAll();

        viewPanel = new JPanel();
        viewPanel.setBorder(
                        BorderFactory.createTitledBorder(
                                        "View Metabolic Pathways from KEGG"));
        viewSL = new SpringLayout();
        viewPanel.setLayout(viewSL);

        outputSL.putConstraint(SpringLayout.NORTH,
                        viewPanel, 0, SpringLayout.NORTH, outputPanel);
        outputSL.putConstraint(SpringLayout.SOUTH,
                        viewPanel, 0, SpringLayout.SOUTH, outputPanel);
        outputSL.putConstraint(SpringLayout.WEST,
                        viewPanel,  0, SpringLayout.WEST, outputPanel);
        outputSL.putConstraint(SpringLayout.EAST,
                        viewPanel, 0, SpringLayout.EAST, outputPanel);
        outputPanel.add(viewPanel);

        revalidate();
}

public void setProgressBar(){
        progressBar = new JProgressBar();
        progressBar.setVisible(true);
        progressBar.setStringPainted(true);

        progressSL = new SpringLayout();
        progressPanel = new JPanel(progressSL);
        cSL.putConstraint(SpringLayout.NORTH,
                        progressPanel, 0, SpringLayout.SOUTH, inputPanel);
        cSL.putConstraint(SpringLayout.WEST,
                        progressPanel, 0, SpringLayout.WEST, c);
        cSL.putConstraint(SpringLayout.EAST,
                        progressPanel, 0, SpringLayout.EAST, c);
        cSL.putConstraint(SpringLayout.SOUTH,
```

```
                        progressPanel, 0, SpringLayout.SOUTH, c);
        frame.add(progressPanel);

        progressSL.putConstraint(SpringLayout.WEST,
                        progressBar, 0, SpringLayout.WEST, progressPanel);
        progressSL.putConstraint(SpringLayout.EAST,
                        progressBar, 0, SpringLayout.EAST, progressPanel);
        progressSL.putConstraint(SpringLayout.SOUTH,
                        progressBar, 0, SpringLayout.SOUTH, progressPanel);
        progressPanel.add(progressBar);
}

public void setTable(){
        String[] colNames = {"#", "Code", "Value"};
        Object[][] object
                = new Object[data.length][colNames.length];

        for(int i = 0 ; i < data.length ; i++){
                object[i][0] = new Integer(i + 1);
                object[i][1] = new String(data[i]);
                object[i][2] = new String(
                                GUIHandler.options.get(0).get(
                                                GUIHandler.options.
                                                get(1).indexOf(data[i])));
        }

        table = new JTable(object, colNames);
        table.setPreferredScrollableViewportSize(
                        new Dimension(300, 300));
        table.setFillsViewportHeight(true);
        table.setEnabled(false);
        table.setAutoResizeMode(
                        JTable.AUTO_RESIZE_ALL_COLUMNS);
        table.getColumnModel().
                getColumn(0).setPreferredWidth(20);
        table.getColumnModel().
                getColumn(1).setPreferredWidth(50);
        table.getColumnModel().
                getColumn(2).setPreferredWidth(400);

        tableJSP = new JScrollPane(table);
        tableJSP.setHorizontalScrollBarPolicy(
                        ScrollPaneConstants.
                        HORIZONTAL_SCROLLBAR_AS_NEEDED);
        tableJSP.setVerticalScrollBarPolicy(
                        ScrollPaneConstants.
                        VERTICAL_SCROLLBAR_AS_NEEDED);
}

public void inputData(){
        addSpeSL = new SpringLayout();
        addSpePanel = new JPanel(addSpeSL);

        addSpe = new JFrame();
        addSpe.setSize(800, 500);
        addSpe.setTitle("Add Species");
        addSpe.add(addSpePanel);
        addSpe.setVisible(true);
        addSpe.setResizable(false);
        addSpe.setLocationRelativeTo(null);
        addSpe.setDefaultCloseOperation(DISPOSE_ON_CLOSE);

        listPanel = new JPanel();
        listPanel.setBorder(
                        BorderFactory.createTitledBorder("Input Specie"));
        listSL = new SpringLayout();
        listPanel.setLayout(listSL);

        addSpeSL.putConstraint(SpringLayout.NORTH,
                        listPanel, 0, SpringLayout.NORTH, addSpePanel);
        addSpeSL.putConstraint(SpringLayout.SOUTH,
                        listPanel, 0, SpringLayout.SOUTH, addSpePanel);
        addSpeSL.putConstraint(SpringLayout.WEST,
                        listPanel, 0, SpringLayout.WEST, addSpePanel);
        addSpeSL.putConstraint(SpringLayout.EAST,
                        listPanel, 0, SpringLayout.EAST, addSpePanel);
        addSpePanel.add(listPanel);

        speNameL = new JLabel("Organism Name or KEGG Code: ");
        listSL.putConstraint(SpringLayout.NORTH,
                        speNameL, 10, SpringLayout.NORTH, listPanel);
        listSL.putConstraint(SpringLayout.WEST,
                        speNameL, 1, SpringLayout.WEST, listPanel);
        listPanel.add(speNameL);

        speName = new JTextField(15);
        listSL.putConstraint(SpringLayout.NORTH,
                        speName, 10, SpringLayout.NORTH, listPanel);
        listSL.putConstraint(SpringLayout.WEST,
                        speName, 2, SpringLayout.EAST, speNameL);
        listSL.putConstraint(SpringLayout.EAST,
                        speName, 350, SpringLayout.WEST, listPanel);
        listPanel.add(speName);

        addS2 = new JButton(ICON_ADD2);
        addS2.setCursor(new Cursor(Cursor.HAND_CURSOR));
        addS2.setToolTipText("Insert");
        addS2.addActionListener(this);
        listSL.putConstraint(SpringLayout.NORTH,
```

```
                addS2, 10, SpringLayout.NORTH, listPanel);
        listSL.putConstraint(SpringLayout.SOUTH,
                addS2, 30, SpringLayout.NORTH, listPanel);
        listSL.putConstraint(SpringLayout.WEST,
                addS2, 1, SpringLayout.EAST, speName);
        listSL.putConstraint(SpringLayout.EAST,
                addS2, 39, SpringLayout.EAST, speName);
        listPanel.add(addS2);

        speList =  new DefaultListModel<String>();
        setOptions(speList);
        species = new JList<String>(speList);
        species.setSelectionMode(
                ListSelectionModel.MULTIPLE_INTERVAL_SELECTION);
        species.setVisibleRowCount(10);

        speJSP = new JScrollPane(species);
        speJSP.setVerticalScrollBarPolicy(
                ScrollPaneConstants.VERTICAL_SCROLLBAR_AS_NEEDED);
        listSL.putConstraint(SpringLayout.NORTH,
                speJSP, 10, SpringLayout.SOUTH, speNameL);
        listSL.putConstraint(SpringLayout.SOUTH,
                speJSP, 0, SpringLayout.SOUTH, listPanel);
        listSL.putConstraint(SpringLayout.WEST,
                speJSP, 1, SpringLayout.WEST, listPanel);
        listSL.putConstraint(SpringLayout.EAST,
                speJSP, 400, SpringLayout.WEST, listPanel);
        listPanel.add(speJSP);

        addS = new JButton(ICON_ADD2);
        addS.setCursor(new Cursor(Cursor.HAND_CURSOR));
        addS.setToolTipText("Insert");
        addS.addActionListener(this);
        listSL.putConstraint(SpringLayout.NORTH,
                addS, 100, SpringLayout.NORTH, speJSP);
        listSL.putConstraint(SpringLayout.SOUTH,
                addS, 120, SpringLayout.NORTH, speJSP);
        listSL.putConstraint(SpringLayout.WEST,
                addS, 1, SpringLayout.EAST, speJSP);
        listSL.putConstraint(SpringLayout.EAST,
                addS, 39, SpringLayout.EAST, speJSP);
        listPanel.add(addS);

        selectedL = new JLabel("Selected Specie");
        listSL.putConstraint(SpringLayout.NORTH,
                selectedL, 0, SpringLayout.NORTH, listPanel);
        listSL.putConstraint(SpringLayout.WEST,
                selectedL, 1, SpringLayout.EAST, addS);
        listPanel.add(selectedL);

        indexSpe = new ArrayList<String>();
        selList =  new DefaultListModel<String>();
        selected = new JList<String>(selList);
        selected.setSelectionMode(
                ListSelectionModel.MULTIPLE_INTERVAL_SELECTION);
        selected.setVisibleRowCount(10);
        listSL.putConstraint(SpringLayout.NORTH,
                selected, 1, SpringLayout.SOUTH, selectedL);
        listSL.putConstraint(SpringLayout.WEST,
                selected, 1, SpringLayout.EAST, addS);
        listSL.putConstraint(SpringLayout.EAST,
                selected, -1, SpringLayout.EAST, listPanel);
        listSL.putConstraint(SpringLayout.SOUTH,
                selected, -30, SpringLayout.SOUTH, listPanel);
        listPanel.add(selected);

        removeB = new JButton(ICON_REMOVE);
        removeB.setCursor(new Cursor(Cursor.HAND_CURSOR));
        removeB.setToolTipText("Remove");
        removeB.addActionListener(this);
        listSL.putConstraint(SpringLayout.NORTH,
                removeB, 2, SpringLayout.SOUTH, addS);
        listSL.putConstraint(SpringLayout.SOUTH,
                removeB, 22, SpringLayout.SOUTH, addS);
        listSL.putConstraint(SpringLayout.WEST,
                removeB, 1, SpringLayout.EAST, speJSP);
        listSL.putConstraint(SpringLayout.EAST,
                removeB, 39, SpringLayout.EAST, speJSP);
        listPanel.add(removeB);

        closeB = new JButton("CLOSE");
        closeB.setCursor(new Cursor(Cursor.HAND_CURSOR));
        closeB.setToolTipText("Hide Frame");
        closeB.addActionListener(this);
        listSL.putConstraint(SpringLayout.NORTH,
                closeB, 8, SpringLayout.SOUTH, selected);
        listSL.putConstraint(SpringLayout.EAST,
                closeB, -20, SpringLayout.EAST, listPanel);
        listPanel.add(closeB);
}

public void addTableInBuild(){
        outputSL.putConstraint(SpringLayout.NORTH,
                tableJSP, 0, SpringLayout.NORTH, outputPanel);
        outputSL.putConstraint(SpringLayout.WEST,
                tableJSP, 0, SpringLayout.WEST, outputPanel);
        outputSL.putConstraint(SpringLayout.EAST,
                tableJSP, -10, SpringLayout.WEST, drawPanel);
        outputSL.putConstraint(SpringLayout.SOUTH,
```

```
                        tableJSP, 0, SpringLayout.NORTH, ojsp);
        outputPanel.add(tableJSP);
}

public void addTable(){
        outputSL.putConstraint(SpringLayout.NORTH,
                        tableJSP, 10, SpringLayout.NORTH, outputPanel);
        outputSL.putConstraint(SpringLayout.WEST,
                        tableJSP, 10, SpringLayout.WEST, outputPanel);
        outputSL.putConstraint(SpringLayout.SOUTH,
                        tableJSP, -10, SpringLayout.SOUTH, outputPanel);
        outputPanel.add(tableJSP);
}

public void setGroups(){
        output = new JTextArea();
        output.setText(guihout);
        output.setEditable(false);
        ojsp = new JScrollPane(output);
        ojsp.setHorizontalScrollBarPolicy(
                        ScrollPaneConstants.HORIZONTAL_SCROLLBAR_AS_NEEDED);
        outputSL.putConstraint(SpringLayout.SOUTH,
                        ojsp, 0, SpringLayout.SOUTH, outputPanel);
        outputSL.putConstraint(SpringLayout.WEST,
                        ojsp, 0, SpringLayout.WEST, outputPanel);
        outputSL.putConstraint(SpringLayout.EAST,
                        ojsp, 0, SpringLayout.WEST, drawPanel);
        outputPanel.add(ojsp);
}

public void gsrOutput(){
        String[] colNames = {"Pair", "% Similarity"};
        int num = 0;
        for(int i = data.length - 1; i > 0 ; i--)
                num += i;

        int indX = 0 , indY = 0;
        Object[][] object = new Object[num][2];
        double[] sim = new double[num];
        for(int i = 0 ; i < num ; i++){
                        if(indX == indY){
                                indY++;
                        }

                        object[i][0]
                                        = new String("(" +
                                        data[indX] + ", " + data[indY] + ")");
                        object[i][1]
                                        = new String(
                                        guih.gsrSimMatrix[indX][indY] + "%");
                        sim[i] = guih.gsrSimMatrix[indX][indY];
                        indY++;
                        if(indY == data.length){
                                indX++;
                                indY = indX;
                        }
        }

        for(int i = 0 ; i < num - 1; i++){
                for(int j = i + 1 ; j < num ; j++){
                        if(sim[i] < sim[j]){
                                double temp = sim[i];
                                sim[i] = sim[j];
                                sim[j] = temp;

                                Object[] tempO = object[i];
                                object[i] = object[j];
                                object[j] = tempO;
                        }
                }
        }

        gsrTable = new JTable(object, colNames);
        gsrTable.setPreferredScrollableViewportSize(
                        new Dimension(300, 300));
        gsrTable.setFillsViewportHeight(true);
        gsrTable.setEnabled(false);
        gsrTable.setAutoResizeMode(
                        JTable.AUTO_RESIZE_ALL_COLUMNS);

        JScrollPane gsrTableJsp
                        = new JScrollPane(gsrTable);
        gsrTableJsp.setHorizontalScrollBarPolicy(
                        ScrollPaneConstants.HORIZONTAL_SCROLLBAR_AS_NEEDED);
        gsrTableJsp.setVerticalScrollBarPolicy(
                        ScrollPaneConstants.VERTICAL_SCROLLBAR_AS_NEEDED);
        outputSL.putConstraint(SpringLayout.NORTH,
                        gsrTableJsp, 10, SpringLayout.NORTH, outputPanel);
        outputSL.putConstraint(SpringLayout.WEST,
                        gsrTableJsp, 10, SpringLayout.EAST, tableJSP);
        outputSL.putConstraint(SpringLayout.EAST,
                        gsrTableJsp, -30, SpringLayout.EAST, outputPanel);
        outputSL.putConstraint(SpringLayout.SOUTH,
                        gsrTableJsp, 0, SpringLayout.SOUTH, outputPanel);
        outputPanel.add(gsrTableJsp);
}

public void fprOutput(){
        output = new JTextArea();
```

```
        String s = "Reactions Exists in species:\n";
        int count = 1;
        for(int i = 0 ; i < guih.speSub.length ; i++){
                if(guih.speSub[i] == 1){
                        s += "("+count+") " + data[i] + "\n";
                        count++;
                }
        }
        output.setEditable(false);
        output.setText(s);
        JScrollPane outputAJSP = new JScrollPane(output);
        outputAJSP.setVerticalScrollBarPolicy(
                        ScrollPaneConstants.VERTICAL_SCROLLBAR_AS_NEEDED);
        outputSL.putConstraint(SpringLayout.NORTH,
                        outputAJSP, 10, SpringLayout.NORTH, outputPanel);
        outputSL.putConstraint(SpringLayout.WEST,
                        outputAJSP, 10, SpringLayout.EAST, tableJSP);
        outputSL.putConstraint(SpringLayout.EAST,
                        outputAJSP, -20, SpringLayout.EAST, outputPanel);
        outputSL.putConstraint(SpringLayout.SOUTH,
                        outputAJSP, 10, SpringLayout.SOUTH, outputPanel);
        outputPanel.add(outputAJSP);
}

public void fsrOutput(){
        JLabel fsL = new JLabel("Reactions with " +
                        (guih.treshold*100) + "% treshold");
        outputSL.putConstraint(SpringLayout.NORTH,
                        fsL, 5, SpringLayout.NORTH, outputPanel);
        outputSL.putConstraint(SpringLayout.WEST,
                        fsL, 5, SpringLayout.WEST, outputPanel);
        outputPanel.add(fsL);

        fsrPanel = new JPanel(new GridLayout(0, 1));
        JPanel northOnlyPanel = new JPanel();
        northOnlyPanel.setLayout(new BorderLayout());
        northOnlyPanel.add(fsrPanel, BorderLayout.NORTH);
        JScrollPane fsrJsp = new JScrollPane();
        fsrJsp.setViewportView(northOnlyPanel);
        fsrJsp.setVerticalScrollBarPolicy(
                        ScrollPaneConstants.VERTICAL_SCROLLBAR_ALWAYS);
        fsrJsp.setHorizontalScrollBarPolicy(
                        ScrollPaneConstants.HORIZONTAL_SCROLLBAR_NEVER);
        outputSL.putConstraint(SpringLayout.NORTH,
                        fsrJsp, 5, SpringLayout.SOUTH, fsL);
        outputSL.putConstraint(SpringLayout.WEST,
                        fsrJsp, 5, SpringLayout.WEST, outputPanel);
        outputSL.putConstraint(SpringLayout.EAST,
                        fsrJsp, -10, SpringLayout.EAST, outputPanel);
        outputSL.putConstraint(SpringLayout.SOUTH,
                        fsrJsp, -10, SpringLayout.SOUTH, outputPanel);
        outputPanel.add(fsrJsp);


        int dfsSize = guih.dfs.size();
        JTextArea[] subgraphs = new JTextArea[dfsSize];
        JScrollPane[] subJsp = new JScrollPane[dfsSize];
        JLabel[] subgL = new JLabel[dfsSize];
        JPanel[] reactions = new JPanel[dfsSize];

        System.out.println("size: " + guih.eList.size());
        for(int i = 0 ; i < dfsSize ; i++){
                reactions[i] = new JPanel(new GridLayout(0, 1));
                subgraphs[i] = new JTextArea();
                String text = "";
                int size = 0;
                for(int j = 0 ; j < guih.dfs.get(i).size() ; j++){
                        int start
                                = guih.indices.get(guih.dfs.get(i).get(j))[0];
                        int last
                                = guih.indices.get(guih.dfs.get(i).get(j))[1];
                        for(int k = start ; k <= last ; k++){
                                int eIndex = guih.edgeIndex[k];
                                String s = guih.eList.get(eIndex).reaction;
                                if(k != start){
                                        text += ", ";
                                        if(size > 85){
                                                size = 0;
                                                text += "\n";
                                        }
                                }
                                size += s.length();
                                text += s;

                        }
                }
                subgraphs[i].setEditable(false);
                subgraphs[i].setColumns(20);
                subgraphs[i].setText(text);
                subgL[i] = new JLabel("Reaction List #" + (i+1));
                reactions[i].add(subgL[i]);

                subJsp[i] = new JScrollPane(subgraphs[i]);
                subJsp[i].setVerticalScrollBarPolicy(
                                ScrollPaneConstants.VERTICAL_SCROLLBAR_AS_NEEDED);
                subJsp[i].setHorizontalScrollBarPolicy(
                                ScrollPaneConstants.HORIZONTAL_SCROLLBAR_AS_NEEDED);
                subJsp[i].setLocation(0, 0);
                reactions[i].add(subJsp[i]);
```

```
                fsrPanel.add(reactions[i]);
        }
}

public void setUpView(){
        if(!checkIntConnection()){
                JOptionPane.showMessageDialog(null,
                                "Can't Connect to www.kegg.jp. "+
                                "Please Check your internet connection.");
                return;
        }

        onlineF = new JFrame("PhenTor: Online");
        onlineF.setResizable(false);
        onlineF.setSize(SCREEN_WIDTH, SCREEN_HEIGHT);

        oSL = new SpringLayout();
        onlineF.setLayout(oSL);
        Container oc = onlineF.getContentPane();
        listPSL = new SpringLayout();
        listP = new JPanel(listPSL);
        listP.setBorder(
                        BorderFactory.createTitledBorder("List"));
        oSL.putConstraint(SpringLayout.NORTH,
                        listP, 0, SpringLayout.NORTH, oc);
        oSL.putConstraint(SpringLayout.WEST,
                        listP, 0, SpringLayout.WEST, oc);
        oSL.putConstraint(SpringLayout.EAST,
                        listP, 350, SpringLayout.WEST, oc);
        oSL.putConstraint(SpringLayout.SOUTH,
                        listP, 0, SpringLayout.SOUTH, oc);
        onlineF.add(listP);

        JLabel selPway = new JLabel("Select Pathway: ");
        listPSL.putConstraint(SpringLayout.NORTH,
                        selPway, 7, SpringLayout.NORTH, listP);
        listPSL.putConstraint(SpringLayout.WEST,
                        selPway,  10, SpringLayout.WEST, listP);
        listP.add(selPway);

        String[] pway= {"Gylcolysis", "Citrate Cycle (Kreb)"};
        pathway = new JComboBox<String>(pway);
        listPSL.putConstraint(SpringLayout.NORTH,
                        pathway, 5, SpringLayout.SOUTH, selPway);
        listPSL.putConstraint(SpringLayout.WEST,
                        pathway,  10, SpringLayout.WEST, listP);
        listP.add(pathway);

        run = new JButton("View Pathway");
        run.setToolTipText(
                        "View Pathway of selected specie");
        run.setCursor(new Cursor(Cursor.HAND_CURSOR));
        run.addActionListener(new ActionListener() {
                @Override
                public void actionPerformed(ActionEvent e) {
                        onlineFunctions();
                        workTask wt = new workTask(run);
                        wt.execute();
                }
        });

        listPSL.putConstraint(SpringLayout.NORTH,
                        run, 5, SpringLayout.SOUTH, selPway);
        listPSL.putConstraint(SpringLayout.EAST,
                        run,  -10, SpringLayout.EAST, listP);
        listP.add(run);

        dload = new JButton("Download XML file");
        dload.setToolTipText(
                        "Download the XML file of the "+
                        "pathway of the selected specie");
        dload.setCursor(new Cursor(Cursor.HAND_CURSOR));
        dload.addActionListener(new ActionListener() {
                @Override
                public void actionPerformed(ActionEvent e) {
                        onlineFunctions();
                        workTask wt = new workTask(dload);
                        wt.execute();
                }
        });
        listPSL.putConstraint(SpringLayout.SOUTH,
                        dload, -5, SpringLayout.NORTH, run);
        listPSL.putConstraint(SpringLayout.EAST,
                        dload, -10, SpringLayout.EAST, listP);
        listP.add(dload);

        JLabel speNameVL = new JLabel(
                        "Organism Name or KEGG Code: ");
        listPSL.putConstraint(SpringLayout.NORTH,
                        speNameVL, 10, SpringLayout.SOUTH, pathway);
        listPSL.putConstraint(SpringLayout.WEST,
                        speNameVL, 1, SpringLayout.WEST, listP);
        listP.add(speNameVL);

        speNameV = new JTextField(20);
        listPSL.putConstraint(SpringLayout.NORTH,
                        speNameV, 10, SpringLayout.SOUTH, pathway);
        listPSL.putConstraint(SpringLayout.WEST,
```

```
                        speNameV, 3, SpringLayout.EAST, speNameVL);
        listP.add(speNameV);

        orgList =  new DefaultListModel<String>();
        setOptions(orgList);
        list = new JList<String>(orgList);
        list.setVisibleRowCount(10);
        list.setSelectionMode(
                        ListSelectionModel.SINGLE_INTERVAL_SELECTION);
        listJ = new JScrollPane(list);
        listJ.setVerticalScrollBarPolicy(
                        ScrollPaneConstants.VERTICAL_SCROLLBAR_AS_NEEDED);
        listPSL.putConstraint(SpringLayout.NORTH,
                        listJ, 10, SpringLayout.SOUTH, speNameVL);
        listPSL.putConstraint(SpringLayout.SOUTH,
                        listJ, -1, SpringLayout.SOUTH, listP);
        listPSL.putConstraint(SpringLayout.WEST,
                        listJ, 1, SpringLayout.WEST, listP);
        listPSL.putConstraint(SpringLayout.EAST,
                        listJ, -1, SpringLayout.EAST, listP);
        listP.add(listJ);

        selLabel = new JLabel("Selected Specie: ");
        selLabel.setVisible(false);
        oSL.putConstraint(SpringLayout.NORTH,
                        selLabel, 10, SpringLayout.NORTH, oc);
        oSL.putConstraint(SpringLayout.WEST,
                        selLabel, 3, SpringLayout.EAST, listP);
        onlineF.add(selLabel);

viewPSL = new SpringLayout();
viewP = new JPanel(viewPSL);
viewP.setBackground(Color.WHITE);
viewP.setBorder(
                BorderFactory.createLineBorder(Color.BLACK));
        oSL.putConstraint(SpringLayout.NORTH,
                        viewP, 3, SpringLayout.SOUTH, selLabel);
        oSL.putConstraint(SpringLayout.WEST,
                        viewP, 0, SpringLayout.EAST, listP);
        oSL.putConstraint(SpringLayout.EAST,
                        viewP, 0, SpringLayout.EAST, oc);
        oSL.putConstraint(SpringLayout.SOUTH,
                        viewP, 0, SpringLayout.SOUTH, oc);
onlineF.add(viewP);

        onlineF.setLocationRelativeTo(null);
        onlineF.setDefaultCloseOperation(DISPOSE_ON_CLOSE);
        onlineF.setVisible(true);


}

public void onlineFunctions(){
        if(list.getSelectedValue() == null
                        && speNameV.getText() == null)
                return;

        int choice2 = -1;

        if(speNameV.getText() != null
                        || speNameV.getText().length() == 0){
                choice2 = JOptionPane.showConfirmDialog(
                                null, "Use text as input?");
        }

        if(choice2 == 0){
                String str = speNameV.getText().trim();
                int n = GUIHandler.options.get(2).
                                indexOf(str.toLowerCase());
                if(n == -1)
                        n = GUIHandler.options.get(1).
                        indexOf(str.toLowerCase());

                if(n == -1){
                        JOptionPane.showMessageDialog(null,
                                        str + " not in the options");
                        return;
                }

                spe = GUIHandler.options.get(1).get(n);
                speNameV.setText("");
        } else
                spe = GUIHandler.options.get(1).get(
                                GUIHandler.options.get(0).
                                indexOf(list.getSelectedValue()));

        if(pathway.getSelectedIndex() == 0)
                pwayI = 10;
        else
                pwayI = 20;
}

public boolean checkIntConnection() {

        boolean status = false;
Socket sock = new Socket();
InetSocketAddress address
        = new InetSocketAddress("www.kegg.jp", 80);
```

```
        try {
                sock.connect(address, 3000);
                if(sock.isConnected()) {
                        status=true;
                }
        } catch(Exception e) {
                e.printStackTrace();
        }

                try {
                        sock.close();
                } catch (IOException e) {
                        e.printStackTrace();
                }
        return status;
}

        public static void main(String[] args)
                        throws IOException {
                final Splash splash = new Splash();
                EventQueue.invokeLater(new Runnable() {
                        @Override
                        public void run() {
                                try {
                                        UIManager.setLookAndFeel(
                                                UIManager.
                                                getSystemLookAndFeelClassName());
                                        PhenTor p = new PhenTor();
                                        splash.setVisible(false);
                                        p.frame.setVisible(true);
                                } catch (Exception e) {
                                        e.printStackTrace();
                                }
                        }
                });
        }

        @Override
        public void actionPerformed(ActionEvent e) {
                if(e.getSource() == openFile){
                        try {
                                UIManager.setLookAndFeel(
                                                UIManager.
                                                getSystemLookAndFeelClassName());
                        } catch (ClassNotFoundException
                                        | InstantiationException
                                        | IllegalAccessException
                                        | UnsupportedLookAndFeelException e1) {
                                JOptionPane.showMessageDialog(null,
                                                "Look and feel cannot be loaded.\n"+
                                                "Switching to default look and feel.",
                                                "Warning", JOptionPane.WARNING_MESSAGE);
                        }

                        JFileChooser fileChooser = new JFileChooser();
                        fileChooser.setFileSelectionMode(
                                        JFileChooser.FILES_ONLY);
                        fileChooser.setAcceptAllFileFilterUsed(false);
                        fileChooser.addChoosableFileFilter(
                                        new FileNameExtensionFilter("XML","xml"));

                        int rVal = fileChooser.showOpenDialog(null);
                        if (rVal == JFileChooser.APPROVE_OPTION) {
                                subPath.setText(fileChooser.
                                                getSelectedFile().toString());
                    }
                }else if(e.getSource() == exit){
                        deleteFiles();
                }else if(e.getSource() == addB){
                        if(isOpen)
                                addSpe.setVisible(true);
                        else{
                                isOpen = true;
                                inputData();
                        }
                }else if(e.getSource() == closeB){
                        addSpe.setVisible(false);
                }else if(e.getSource() == runB){
                        int ind = index.getSelectedIndex();
                        workTask wt = new workTask(ind);
                        wt.execute();
                }else if(e.getSource() == buildR || e.getSource() == fsR ||
                                e.getSource() == fpR || e.getSource() == gsR){
                        setTaskPanel((JRadioButton)e.getSource());
                }else if(e.getSource() == eTxt){
                        //export
                        try {
                                UIManager.setLookAndFeel(
                                                UIManager.getSystemLookAndFeelClassName());
                        } catch (ClassNotFoundException | InstantiationException
                                        | IllegalAccessException
                                        | UnsupportedLookAndFeelException e1) {
                                JOptionPane.showMessageDialog(
                                                null, "Look and feel cannot be loaded.\n"+
                                "Switching to default look and feel.", "Warning",
                                JOptionPane.WARNING_MESSAGE);
                        }

                        JFileChooser jl = new JFileChooser();
```

```
                jl.setFileSelectionMode(
                                JFileChooser.DIRECTORIES_ONLY);
                jl.setDialogTitle("Choose Destination Folder");
                int save = jl.showSaveDialog(null);
                if (JFileChooser.APPROVE_OPTION == save){
                        String filename = jl.getSelectedFile().
                                        getName();
                        String filepath = jl.getSelectedFile().
                                        getAbsolutePath();
                        guih.exportFiles(choice_, filepath, filename);
                        JOptionPane.showMessageDialog(null, "Finished Exporting Files");
                }else
                        JOptionPane.showMessageDialog(null,
                                        "Activity was canceled by the user!");
        }else if(e.getSource() == addS2){
                if(speName.getText() == null)
                        return;
                String str = speName.getText().trim();
                int n = GUIHandler.options.get(2).indexOf(str.toLowerCase());
                if(n == -1)
                        n = GUIHandler.options.get(1).indexOf(str.toLowerCase());

                if(n == -1){
                        JOptionPane.showMessageDialog(null, str + " not in the options");
                        return;
                }

                String str2 = GUIHandler.options.get(0).get(n);
                if(selList.contains(str2)){
                        JOptionPane.showMessageDialog(null,
                                        str2+ " already in the testing set");
                        return;
                }
                indexSpe.add(n + "");
                selList.addElement(str2);
                speList.removeElement(str2);

                speName.setText("");

                data = new String[selList.size()];
                String out = "";
                for(int i = 0 ; i < selList.size() ; i++){
                        data[i] = GUIHandler.options.get(1).get(
                                        GUIHandler.options.get(0).indexOf(selList.get(i)));

                        if(i != 0)
                                out += ", ";
                        out += data[i];
                }
                orgSel.setText(out);

        }else if(e.getSource() == addS){
                if(species.getSelectedValue() == null)
                        return;
                int[] a = species.getSelectedIndices();
                String str = "";
                String[] sels = new String[a.length];
                for(int i = 0 ; i < a.length ; i++){
                        str = speList.getElementAt(a[i]);
                        selList.addElement(str);
                        indexSpe.add(GUIHandler.options.get(0).indexOf(str) + "");
                        sels[i] = str;
                }

                for(int i = 0 ; i < sels.length ; i++){
                        speList.removeElement(sels[i]);
                }

                data = new String[selList.size()];
                String out = "";
                for(int i = 0 ; i < selList.size() ; i++){
                        data[i] = GUIHandler.options.get(1).get(
                                        GUIHandler.options.get(0).indexOf(selList.get(i)));

                        if(i != 0)
                                out += ", ";
                        out += data[i];
                }
                orgSel.setText(out);
        }else if(e.getSource() == removeB){
                if(selected.getSelectedValue() == null)
                        return;

                int[] a = selected.getSelectedIndices();
                String str = "";
                String[] sels = new String[a.length];
                String[] inds = new String[a.length];
                for(int i = 0 ; i < a.length ; i++){
                        str = selList.getElementAt(a[i]);
                        speList.add(Integer.parseInt(indexSpe.get(a[i])), str);
                        sels[i] = str;
                        inds[i] = indexSpe.get(a[i]);
                }

                for(int i = 0 ; i < sels.length ; i++){
                        selList.removeElement(sels[i]);
                        indexSpe.remove(inds[i]);
                }
```

```
                        data = new String[selList.size()];
                        String out = "";
                        for(int i = 0 ; i < selList.size() ; i++){
                                data[i] = GUIHandler.options.get(1).get(
                                        GUIHandler.options.get(0).indexOf(selList.get(i)));
                                if(i != 0)
                                        out += ", ";
                                out += data[i];
                        }

                        orgSel.setText(out);
                }
        }

private class workTask extends SwingWorker<Void, Integer> {

                int choice;
                JButton button;
                boolean isPhentorF;
                public workTask(JButton button){
                        this.button = button;
                        isPhentorF = false;
                }

                public workTask(int choice) {
                        this.choice = choice;
                        isPhentorF = true;
                }

                @Override
                protected Void doInBackground() throws Exception {
                        progressBar.setVisible(true);
                        progressBar.setIndeterminate(true);
                        frame.revalidate();
                        frame.repaint();

                        try {
                                if(isPhentorF){

                                        if(data.length == 0){
                                                JOptionPane.showMessageDialog(
                                                                null, "No Species to test in the dataset");
                                                return null;
                                        }

                                        outputPanel.removeAll();
                                        loadingPanel();
                                        //show loading gif
                                        revalidate();

                                        if(buildR.isSelected()){
                                                if(data.length < 3){
                                                        JOptionPane.showMessageDialog(null,
                                                        "Test Data for building phenogram"+
                                                        "must be greater than or equal to 3");
                                                        outputPanel.removeAll();
                                                        welcomePanel();
                                                        revalidate();

                                                        progressBar.setString("");
                                                        progressBar.setVisible(false);
                                                        progressBar.setValue(0);

                                                        return null;
                                                }

                                                choice_ = build;
                                                double start2 = System.currentTimeMillis()/1000;
                                                progressBar.setString("Reading Input....");
                                                System.out.println("Clicked");
                                                progressBar.setString("Processing....");
                                                guih = new GUIHandler(choice, progressBar);
                                                String out = guih.SimulateSelected(data, 1);

                                                if(out == null){
                                                        outputPanel.removeAll();
                                                        welcomePanel();
                                                        revalidate();

                                                        progressBar.setString("");
                                                        progressBar.setVisible(false);
                                                        progressBar.setValue(0);

                                                        return null;
                                                }

                                                System.out.println(out);
                                                guihout = out;
                                                progressBar.setString("Setting Tree Data....");
                                                DrawgramInterface dg = new DrawgramInterface();
                                                dg.DrawgramRun("treedata");
                                                String title = "Preview: ";
                                                String curDir = System.getProperty("user.dir");
                                                curDir += "/JavaPreview.ps";

                                                progressBar.setString("Drawing Tree....");

                                                outputPanel.removeAll();
                                                drawSL = new SpringLayout();
```

```
                drawPanel = new JPanel(drawSL);
                drawPanel.setBackground(Color.WHITE);

                drawPanel.setBorder(
                                BorderFactory.createLineBorder(Color.BLACK));
                outputSL.putConstraint(SpringLayout.NORTH,
                                drawPanel, 0, SpringLayout.NORTH, outputPanel);
                outputSL.putConstraint(SpringLayout.WEST,
                                drawPanel, -300, SpringLayout.EAST, outputPanel);
                outputSL.putConstraint(SpringLayout.EAST,
                                drawPanel, 0, SpringLayout.EAST, outputPanel);
                outputSL.putConstraint(SpringLayout.SOUTH,
                                drawPanel, 0, SpringLayout.SOUTH, outputPanel);
                outputPanel.add(drawPanel);

                new DrawPreview(title, curDir, "files", false);

                BufferedImage image = ImageIO.read(new File("phenom.png"));

                //URL url = getClass().getResource("/files/phenom.png");
                //BufferedImage image = ImageIO.read(url);
                ImageIcon imageIcon = new ImageIcon(image);
                Image dimg = imageIcon.getImage();
                Image img
                = dimg.getScaledInstance(299,455,Image.SCALE_SMOOTH);
                ImageIcon icon = new ImageIcon(img);

                JLabel imageL = new JLabel(icon);
                drawSL.putConstraint(SpringLayout.NORTH,
                                imageL, 0, SpringLayout.NORTH, drawPanel);
                drawSL.putConstraint(SpringLayout.WEST,
                                imageL, 0, SpringLayout.WEST, drawPanel);
                drawSL.putConstraint(SpringLayout.EAST,
                                imageL, 0, SpringLayout.EAST, drawPanel);
                drawSL.putConstraint(SpringLayout.SOUTH,
                                imageL, 0, SpringLayout.SOUTH, drawPanel);
                drawPanel.add(imageL);

                progressBar.setString("Setting table....");

                setGroups();
                setTable();
                addTableInBuild();
                eTxt.setEnabled(true);
                revalidate();
                double end2 = System.currentTimeMillis()/1000;

                File file = new File("outfile");
                if(file.exists())
                        file.delete();

                file = new File("intree");
                if(file.exists())
                        file.delete();

        }else if(fsR.isSelected()){
                choice_ = fs;
                String tresh = treshT.getText();
                int alpha = Integer.parseInt(tresh);
                double treshold = (double)alpha/100;
                guih = new GUIHandler(choice, progressBar, treshold);
                String out = guih.SimulateSelected(data, 2);

                if(out == null){
                        outputPanel.removeAll();
                        welcomePanel();
                        revalidate();

                        progressBar.setString("");
                        progressBar.setVisible(false);
                        progressBar.setValue(0);

                        return null;
                }

                System.out.println(out);

                outputPanel.removeAll();
                fsrOutput();
                eTxt.setEnabled(true);
                revalidate();
        }else if(fpR.isSelected()){
                choice_ = fp;
                String fileName = subPath.getText().trim();
                guih = new GUIHandler(choice, progressBar, fileName);
                String out = guih.SimulateSelected(data, 3);

                if(out == null){
                        outputPanel.removeAll();
                        welcomePanel();
                        revalidate();

                        progressBar.setString("");
                        progressBar.setVisible(false);
                        progressBar.setValue(0);

                        return null;
                }
```

```
                    System.out.println(out);
                    outputPanel.removeAll();
                    setTable();
                    addTable();
                    fprOutput();
                    eTxt.setEnabled(true);
                    revalidate();
            }else if(gsR.isSelected()){
                    choice_ = gs;
                    guih = new GUIHandler(choice, progressBar);
                    String out = guih.SimulateSelected(data, 4);

                    if(out == null){
                            outputPanel.removeAll();
                            welcomePanel();
                            revalidate();

                            progressBar.setString("");
                            progressBar.setVisible(false);
                            progressBar.setValue(0);

                            return null;
                    }

                    System.out.println(out);
                    outputPanel.removeAll();
                    setTable();
                    addTable();
                    gsrOutput();
                    eTxt.setEnabled(true);
                    revalidate();
            }
    } else {
            if(button == run){

                    viewP.removeAll();

                    JLabel load = new JLabel(ICON_LOADING);
                    viewPSL.putConstraint(SpringLayout.NORTH,
                                    load, 0, SpringLayout.NORTH, viewP);
                    viewPSL.putConstraint(SpringLayout.WEST,
                                    load, 0, SpringLayout.WEST, viewP);
                    viewPSL.putConstraint(SpringLayout.EAST,
                                    load, 0, SpringLayout.EAST, viewP);
                    viewPSL.putConstraint(SpringLayout.SOUTH,
                                    load, 0, SpringLayout.SOUTH, viewP);
                    viewP.add(load);

                    String path = "http://rest.kegg.jp/get/"+
                    spe + "000" + pwayI + "/image";

    URL url = new URL(path);
    BufferedImage image = ImageIO.read(url);

    Image dimg =
                    image.getScaledInstance(
                                    (int) viewP.getSize().getWidth(),
                                    (int) viewP.getSize().getHeight(),
                                    Image.SCALE_SMOOTH);
    ImageIcon icon = new ImageIcon(dimg);

    String label
    = "Selected Specie: " + spe + "000" + pwayI;
    selLabel.setVisible(true);
    selLabel.setText(label);

    viewP.removeAll();

                    JLabel imageL = new JLabel(icon);
                    imageL.setToolTipText(spe + "000" + pwayI);
                    viewP.add(imageL);
                    onlineF.revalidate();
            }else if(button == dload){
                    String path = "http://rest.kegg.jp/get/"+
            spe + "000" + pwayI + "/kgml";

                    URL kgml = new URL(path);
                BufferedReader in =
                            new BufferedReader(
                                    new InputStreamReader(
                                                    kgml.openStream()));

                String inputLine;
                String content = "";
                while ((inputLine = in.readLine()) != null){
                    content += inputLine + "\n";
                }
                in.close();

                    JFileChooser jl = new JFileChooser();
                    jl.setFileSelectionMode(JFileChooser.DIRECTORIES_ONLY);
                    jl.setDialogTitle("Choose Destination Folder");
                    int save = jl.showSaveDialog(null);
                    if (JFileChooser.APPROVE_OPTION == save){
                            String filename = spe + "000" + pwayI + ".xml";
                            String filepath = jl.getSelectedFile().getAbsolutePath();


                            File xmlfile = new File(filepath + "/" + filename) ;
```

```
                                                        // if file doesnt exists, then create it
                                                        if (!xmlfile.exists()) {
                                                                xmlfile.createNewFile();
                                                        }
                                                        FileWriter xmlfw =
                                                                        new FileWriter(xmlfile.getAbsoluteFile())
                                                        BufferedWriter xmlbw = new BufferedWriter(xmlfw);
                                                        xmlbw.write(content);
                                                        xmlbw.close();

                                                        String addPath = "glyco";
                                                        if(pwayI == 20)
                                                                addPath = "citric";

                                                        String s = getClass().
                                                                        getResource("/files/"+
                                                        addPath + "/" + filename).toString();
                                                        s = s.substring(s.indexOf('C'), s.length());
                                                        File xmlfile2 = new File(s) ;
                                                        // if file doesnt exists, then create it
                                                        if (!xmlfile2.exists()) {
                                                                xmlfile2.createNewFile();
                                                        }
                                                        FileWriter xmlfw2 =
                                                                        new FileWriter(
                                                                                        xmlfile2.getAbsoluteFile(
                                                        BufferedWriter xmlbw2 =
                                                                        new BufferedWriter(xmlfw2);
                                                        xmlbw2.write(content);
                                                        xmlbw2.close();

                                                        JOptionPane.showMessageDialog(null,
                                                                        filename + " download complete. A copy wa
                                                                        "saved in the database");
                                                }else
                                                        JOptionPane.showMessageDialog(null,
                                                                        "Activity was canceled by the user!");
                                                }
                                        }
                                } catch (Exception e1) {
                                        JOptionPane.showMessageDialog(null, "Check Parameters.",
                                                        "Error", JOptionPane.ERROR_MESSAGE);
                                        e1.printStackTrace();

                                        outputPanel.removeAll();
                                        welcomePanel();
                                        revalidate();

                                }

                                progressBar.setString("");
                                progressBar.setVisible(false);
                                progressBar.setValue(0);
                                frame.revalidate();
                                frame.repaint();
                                return null;
                        }
                }

                private JFrame frame;
                private Container c;

                private JMenuBar menuBar;
                private JMenu file, online;
                private JMenuItem exit, eTxt, view;

                private SpringLayout cSL, inputSL, outputSL, viewSL, taskSL, addSpeSL,
                        drawSL, progressSL;

                private JPanel inputPanel, outputPanel, viewPanel, taskPanel, drawPanel,
                        addSpePanel, progressPanel, fsrPanel;
                private JTextField orgSel, treshT;
                private JScrollPane inputJsp, outputJsp, tableJSP;
                private JList<String> selected, species;
                private DefaultListModel<String> selList, speList;
                private JTextArea output;

                private JProgressBar progressBar;

                private JButton openFile, runB, addB, closeB;
                private JTextField subPath;
                private JComboBox<String> index;

                private JRadioButton buildR, fsR, fpR, gsR;
                private ButtonGroup task;

                private JTable table, gsrTable;

                private JFrame addSpe, onlineF;
                //frame elems
                private JPanel listPanel;
                private JScrollPane speJSP, ojsp;
                private SpringLayout listSL;
                private JButton addS, removeB, addS2;
                private JLabel selectedL, speNameL;
                private JTextField speName;

                //frame-view elem
                private SpringLayout oSL, listPSL, viewPSL;
```

```
        private JPanel listP, viewP;
        private JLabel selLabel;
        private JList<String> list;
        private DefaultListModel<String> orgList;
        private JButton run, dload;
        private JComboBox<String> pathway;
        private JScrollPane listJ;
        private JTextField speNameV;
}
```

2. Splash.java

```
package com.View;

import java.awt.Component;
import java.io.IOException;
import java.net.URL;

import javax.imageio.ImageIO;
import javax.swing.ImageIcon;
import javax.swing.JFrame;
import javax.swing.JLabel;
import javax.swing.JPanel;
import javax.swing.SwingConstants;

public class Splash extends JFrame {


        private static final long serialVersionUID =
                -2322807926670906301L;
        private JPanel contentPane;

        public Splash() throws IOException {
                setSize(475, 300);
                contentPane = new JPanel();
                contentPane.setBorder(null);
                setContentPane(contentPane);
                contentPane.setLayout(null);

                JLabel label = new JLabel(
                                "Developed by Lejun Christian Lor Osorio"+
                                ", 2015. All rights reserved.");
                label.setAlignmentX(
                                Component.CENTER_ALIGNMENT);
                label.setHorizontalTextPosition(
                                SwingConstants.CENTER);
                label.setHorizontalAlignment(
                                SwingConstants.CENTER);
                label.setBounds(0, 275, 500, 14);
                contentPane.add(label);

                JLabel lblNewLabel = new JLabel("");

                lblNewLabel.setIcon(
                                new ImageIcon(
                                                getClass().
                                                getResource(
                                                        "/images/splash.png"
                                                )));
                lblNewLabel.setBounds(0, 0, 475, 300);
                contentPane.add(lblNewLabel);

                URL url = getClass().
                                getResource("/image/icon.png");
                setIconImage(ImageIO.read(url));
                setUndecorated(true);
                setDefaultCloseOperation(
                                JFrame.EXIT_ON_CLOSE);
                setLocationRelativeTo(null);
                setVisible(true);
        }
}
```

# XII. Acknowledgement

I can say that this night, I've fulfilled one of my dreams. Ive dreamed of this day for so long, ever since I entered college. Know what my dream is? Right! To write an Acknowledgement. Thats it.

"Do not be anxious about anything, but in everything by prayer and supplication with thanksgiving let your requests be made known to God." (Phil. 4:6) (from Zee, pacopy :D ). First, I want to thank God, for answering my prayers about this special problem. Even thought, it feels like I'm in a rush, whenever I go to church to do my duties, I easily forgot the things that make my life complicated. After going to church, it feels like I can think of the thinks that seems complicated at the start.

To my parents, who always there to scold me on whatever I'm doing, thank you. Without you, I won't be here in this world. I will take this opportunity to say how much I love you. i love you Mommy and Daddy. Kahit lagi nyo kong pinapagalitan sa kahit anung bagay. To the other part of our family, thank you for you support (*"Meron ba?"*) :) .

To Sir Geoff Solano, my adviser in this study, thanks for entrusting me this topic. Thanks for the guide that you gave for this topic. Thank you for believing that I can finish this task. To my other professors, from 1st year - this year (" *Well, 5th year ko na*"), thank you, because you taught me all the things that make me a certified UP Student.

To the person who always makes me do something for the first time (not green, okay?), who is always there to support me (*andyan ka nga ba?*), who is always there to talk to me till 12 am, thank you. **KILALA MO KUNG SINO KA**. Im really happy that youve cried because of me (together with you baby sister). Thanks for the motivation. Thanks for making me laugh all the time, whenever we are together, or whenever we are chatting with each other. I will also take this opportunity to tell the world that I LOVE YOU. (**University Confession?!**). If you want to know her name, just text me. 09*********. Ilang permutations lang yan.

To my best friends, Allan and Allan, **BAKIT KAYO PAREHAS ALLAN?**, thanks for supporting me in the backgrounds. Even though all of us have our own lives now, you didnt even forget to keep touch with me. I know that Im not that a great friend, but still, the both of you still stick with me, well, of course! Both of you are not great also, joke. To Mark ALLAN and ALLAN Jay, both of you are my best friends and will always be.

To my blockmates who served as my laughing stock in the University, thank you. Oh wrong, who both served as my friends and my idols in UP, thank you. Thanks for making me copy your home works. Thanks for doing most of the things whenever there is a group activity. Well thanks for the memories, the outing, the shooting, and many more.

To my co-BOR, Editho Giray III, Joser Barron, Mark Borerros, and Rizza Gonzaga (plus Manuel Angeles), thanks for all the fun stuff in college. All of you made my college life fun and interesting. Especially to MarkBo, salamat sa pagiging laughing stock naming ha.

To my billiard mates, BOR, Sarah Tandoc, Sherwin Saringan, Alyra Escotido, Geraline de Guzman, and Celina Buena, billiards ulit!

To my closest friends from high school, the BU (Barkadang Unknown). Sana makasama na ko next time sa gala nyo.

Thanks to Narinig ko sa UP also known as Overheard at UP. This page makes me laugh all the time. Try this.

Lastly, this is for all the people who didn't believe in me. To think that I have too many haters. **WEW**. Well, I wont say that I didnt do anything bad, well, yes, I did something, but why over exaggerate it? And to those who backfires me, looks down on me, lying to me, and not telling me anything, well, thank you. All of you served as my motivators to do what I must do. *FYI*. Im not bitter or what, Im just saying thank you.

I always made some wrong decisions in life, but those decisions made me on who I am now. Of course, I regret most of those wrong decisions, but not all. Life is a never-ending journey. Ending this journey means entering a new one. Its a good thing that others are there to push me to reach this day. It is also a good thing that I didnt give up half way. All I can say is that, HERE I AM, right now, AT UP.

To close this acknowledgement, Aristotle says, "Dont believe anything written in the Internet, especially in WikiPedia." Thanks for this quote. This is one of the quotes that made me into a certified UP Student.