

**King Fahd University Of Petroleum and Minerals**

*College of Computer Science and Engineering*

**ICS 202 – Project Report**

Student Details:

|  |  |  |
| --- | --- | --- |
| **Student Name** | **Identification Number** | **Section Number** |
| Ibrahim Ali BinAlshikh | 201139750 | 51 |

|  |  |
| --- | --- |
| **Instructor** | Ahmad Irfan |

Table of Contents

[1 Problem Statement 4](#_Toc387380074)

[2 Used Data Structures 4](#_Toc387380075)

[2.1 Linked List 4](#_Toc387380076)

[2.2 Sortable Linked List 4](#_Toc387380077)

[2.3 Queue 4](#_Toc387380078)

[3 Class Diagrams 4](#_Toc387380079)

[3.1 Package ics102.project.edges 4](#_Toc387380080)

[3.2 Package ics102.project.graph 5](#_Toc387380081)

[3.3 Package icc202.project.exceptions 5](#_Toc387380082)

[3.4 Package ics202.project.util 6](#_Toc387380083)

[3.5 Package ics202.project.vertices 7](#_Toc387380084)

[4 Java Classes 7](#_Toc387380085)

[4.1 Edges: 7](#_Toc387380086)

[4.1.1 AbstractEdge.java 7](#_Toc387380087)

[4.1.2 UndirectedEdge.java 7](#_Toc387380088)

[4.1.3 DirectedEdge.java 7](#_Toc387380089)

[4.1.4 UndirectedWightedEdge.java 7](#_Toc387380090)

[4.1.5 DirectedWeightedEdge.java 7](#_Toc387380091)

[4.2 Vertices: 7](#_Toc387380092)

[4.2.1 AbstractVertex.java (useless) 7](#_Toc387380093)

[4.2.2 Vertex.java 7](#_Toc387380094)

[4.2.3 TraversalVertex.java 8](#_Toc387380095)

[4.3 Graphs: 8](#_Toc387380096)

[4.3.1 AbstractGraph.java 8](#_Toc387380097)

[4.3.2 UndirectedGraph.java 8](#_Toc387380098)

[4.3.3 DirectedGraph.java 8](#_Toc387380099)

[4.3.4 UndirectedWeightedGraph.java 8](#_Toc387380100)

[4.3.5 DirectedWeightedGraph.java 8](#_Toc387380101)

[4.4 Utilities: 9](#_Toc387380102)

[4.4.1 LinkedNode.java 9](#_Toc387380103)

[4.4.2 DoublyLinkedNode.java 9](#_Toc387380104)

[4.4.3 LinkedList.java 9](#_Toc387380105)

[4.4.4 SortableLinkedList.java 9](#_Toc387380106)

[4.4.5 Stack.java 9](#_Toc387380107)

[4.4.6 Queue.java 9](#_Toc387380108)

[4.5 Traversal: 9](#_Toc387380109)

[4.5.1 BreadthFirst.java 9](#_Toc387380110)

[4.5.2 DepthFirst.java 9](#_Toc387380111)

[4.5.3 TopologicalOrder.java 9](#_Toc387380112)

[4.5.4 ShortestPath.java 9](#_Toc387380113)

[4.5.5 MinimumSpanningTree.java 9](#_Toc387380114)

[5 Testing Stages 10](#_Toc387380115)

[5.1 Edges Testing 10](#_Toc387380116)

[5.2 Vertex Testing 11](#_Toc387380117)

[5.3 Graphs Testing 12](#_Toc387380118)

[5.3.1 Undirected Graph 12](#_Toc387380119)

[5.3.2 Undirected Weighted Graph 13](#_Toc387380120)

[5.3.3 Directed Graph 14](#_Toc387380121)

[6 Conclusion 14](#_Toc387380122)

# Problem Statement

The problem is to design and implement my own graph data structure using java programing language. The library should support the following types of graphs: Directed, Undirected, Directed Weighted and Undirected weighted graph. Also the library should implements the following traversal algorithms: Depth First, Breadth First, Topological sort (For Directed graph only), Dijkstra and prim. The graphs should support the basic graph operations such as adding and removing vertices, adding and removing edges.

# Used Data Structures

## Linked List

This class is the heart of graph library, used to create a list of edges inside the class Vertex. Also used to create a list of Vertices inside the super graph class. The implementation of adding and removing edges, adding and removing vertices manly depends on this class.

## Sortable Linked List

The main purpose of this class is to perform Depth First traversal.

## Queue

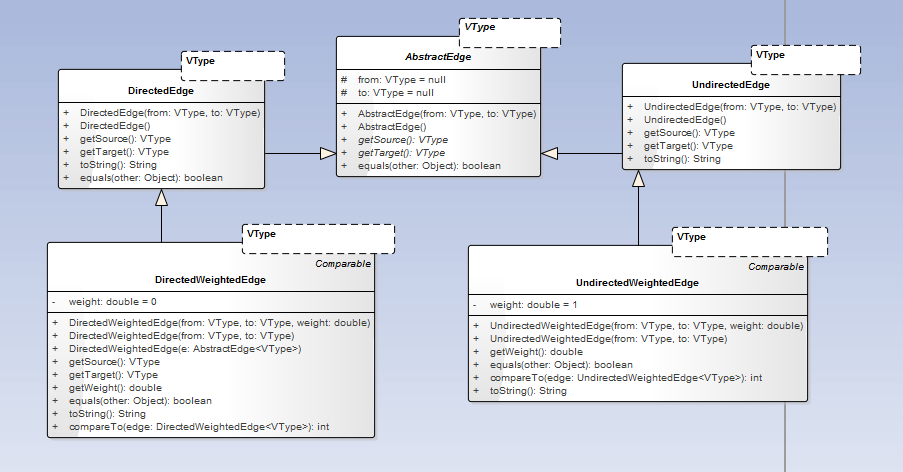
The main purpose of this class is to perform Breadth First traversal. But also it’s been used in all traversal classes in order to implement specific methods.

## Stack

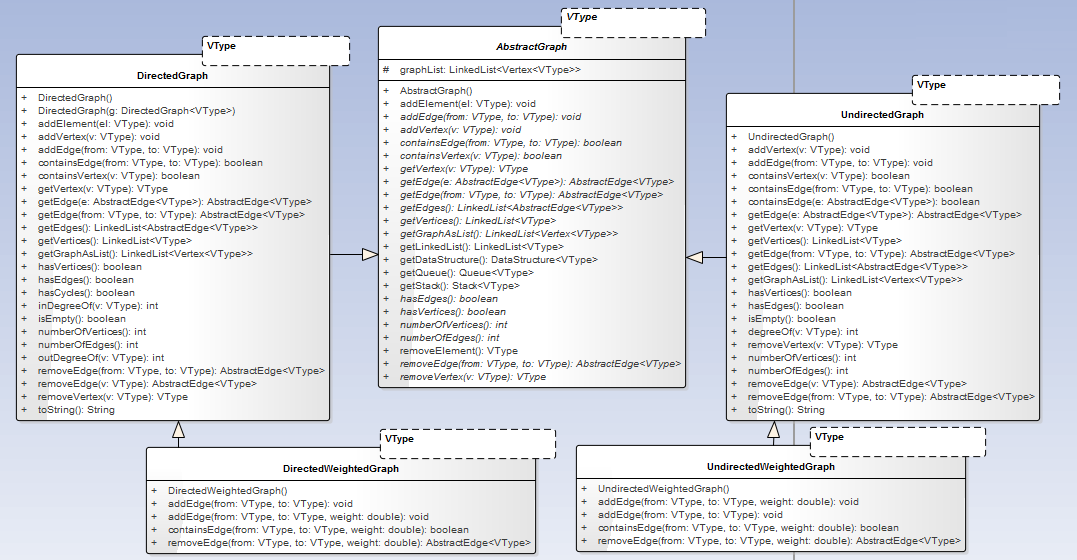
Used for traversal.

# Class Diagrams

## Package ics102.project.edges

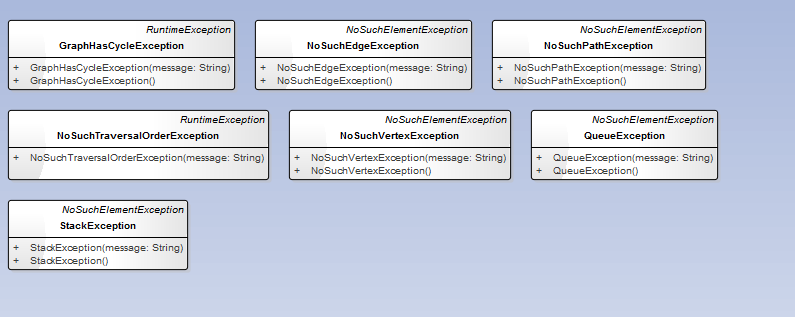
This package contains 5 classes: Abstract Edge, Directed Edge, Undirected Edge, Undirected Weighted Edge and a Directed Weighted Edge.

## Package ics102.project.graph

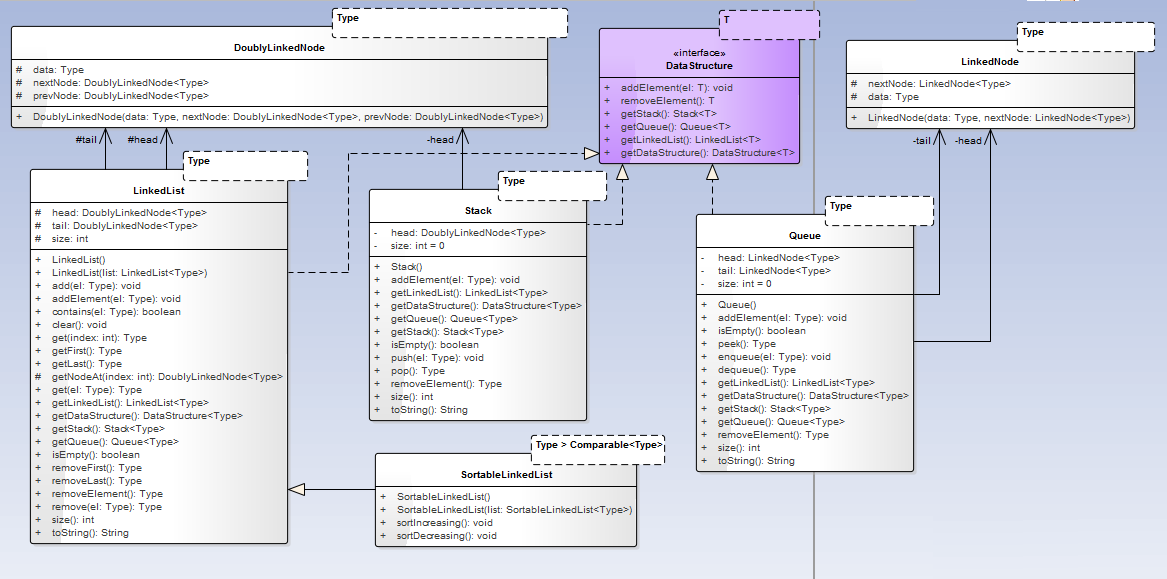


## Package icc202.project.exceptions

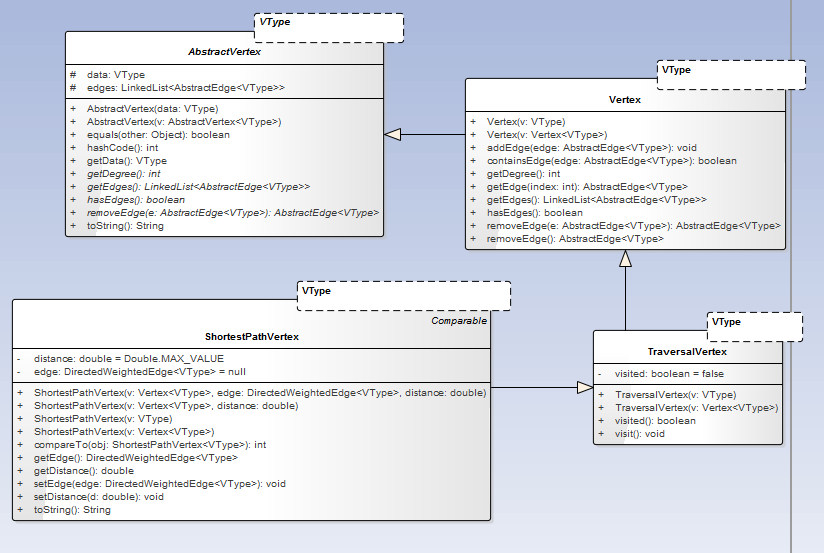
A package that contains all of the exceptions for the graph library.



## Package ics202.project.util

Utilities that was used to build the graph library (data structures).

## Package ics202.project.vertices



# Java Classes

## Edges:

Edges used to link between two vertices. Since we have different types of graphs, we need a different type for each one. The common thing between all of the edges is that they point to a source vertex and a target vertex.

### AbstractEdge.java

This class is the base class for all edges. It contains two instance variables that are generic. One is the source vertex and the other is the target vertex. Also this class contains two abstract methods. One is get Source , and the other is get Target . Also the class has equals method.

### UndirectedEdge.java

This class inherits all the information on the abstract edge class. In addition to that, it has to String method.

### DirectedEdge.java

This class inherits all the information on the abstract edge class. In addition to that, it has to String method. The difference between this one and the undirected edge is in the equal method only.

### UndirectedWightedEdge.java

This class inherits all the information on the undirected edge. In addition to that, it has an extra variable called weight. The default value for that variable is zero.

### DirectedWeightedEdge.java

This class inherits all the information on the undirected edge. In addition to that, it has an extra variable called weight. The default value for that variable is zero.

## Vertices:

The vertex is one unite that is used to build a graph. But sometimes we need to perform specific task on the graph that requires additional information to be present within the vertex, and that’s why we have different types of vertices.

### AbstractVertex.java (useless)

### Vertex.java

This class is the base vertex. It is used in building all graphs. Inside this class, there is a linked list that contains a linked list of abstract edges. In addition to that, there is a generic filed that will hold the stored information inside the vertex.

### TraversalVertex.java

This class is used in traversal algorithms. The class inherits all of the information inside class vertex. In addition to that, it has an extra instance variable “visit”. This variable is used to indicate that if the vertex is visited or not.

## Graphs:

### AbstractGraph.java

This class is the base graph for all graphs. It contains the main methods that can be found in almost any graph. This graph has a linked list of vertices. This list is the graph itself.

### UndirectedGraph.java

This class inherits everything from the abstract graph. When adding an edge to this graph, the type of edge will be undirected edge.

### DirectedGraph.java

This class inherits everything from the abstract graph. When adding an edge to this graph, the type of edge will be directed edge.

### UndirectedWeightedGraph.java

This class inherits everything from the undirected graph. When adding an edge to this graph, the type of edge will be directed weighted edge.

### DirectedWeightedGraph.java

This class inherits everything from the directed graph. When adding an edge to this graph, the type of edge will be directed weighted edge.

## Utilities:

### LinkedNode.java

The class is used to build a stack and a queue.

### DoublyLinkedNode.java

This class is used to build a linked list.

### LinkedList.java

The whole graph library depends on that class. If there is a bug in adding or removing vertices and edges it will be here for sure.

### SortableLinkedList.java

This class is used as a priority queue for prim’s algorithm. The aim of the algorithm is to find the minimum spanning tree.

### Stack.java

This class is used in traversal.

### Queue.java

This class is used in traversal.

## Traversal:

### BreadthFirst.java

This class is used to perform breadth first traversal on all types of graphs.

### DepthFirst.java

This class is used to perform depth first traversal on all types of graphs.

### TopologicalOrder.java

This class used to perform topological sorting on directed graphs.

### ShortestPath.java

This class is used to find the shortest path to visit all graph vertices.

### MinimumSpanningTree.java

This class is used to find the minimum path tree for undirected weighted graphs.

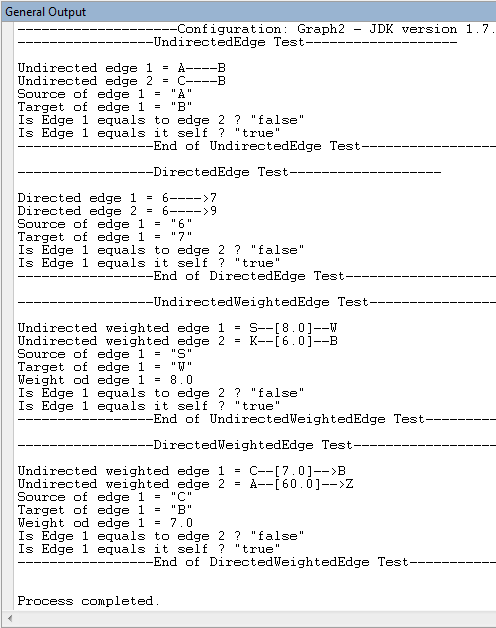
# Testing Stages

All tests classes can be found under the package “ics202.project.testes”.

Note: to run each test file, click on the file and from “Run” menu select “Run File”.

## Edges Testing

This test is done by creating two edges from each class and then using the methods inside each class.



## Vertex Testing

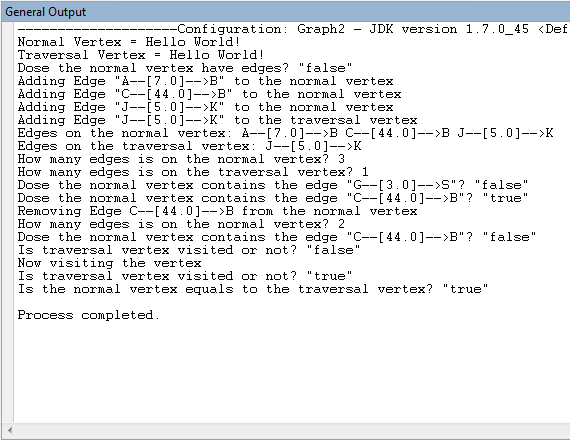
This test is done by creating two vertices, one is normal and the other is traversal vertex. On each class the following exception might be thrown:

1. “ NoSuchEdgeException ”

This exception is thrown when removing an edge from the vertex and the edge is not on the vertex. Also thrown if the vertex dose not contains any edges and the user try to remove an edge.

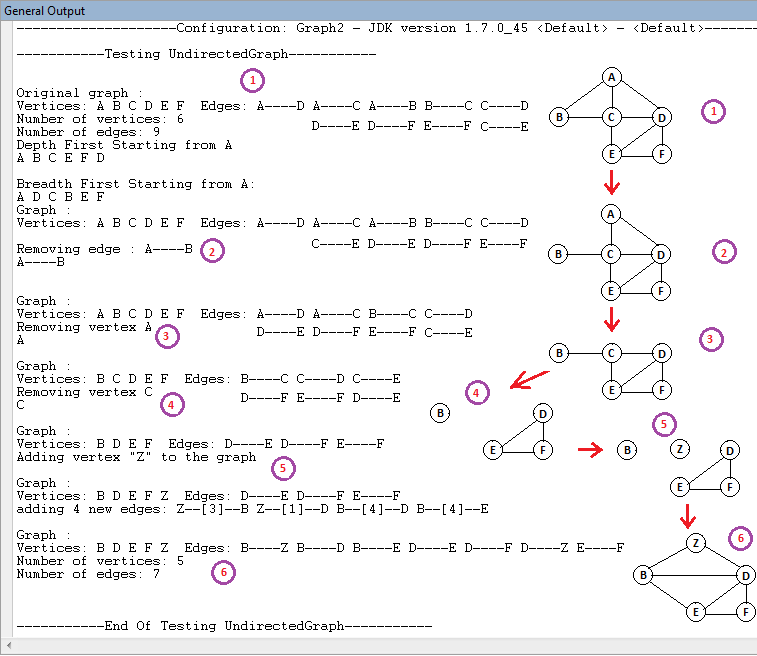
1. “IndexOutOfBoundsException”

This exception is if the user is trying to access a specific edge using the “get Edge (int index)” method.

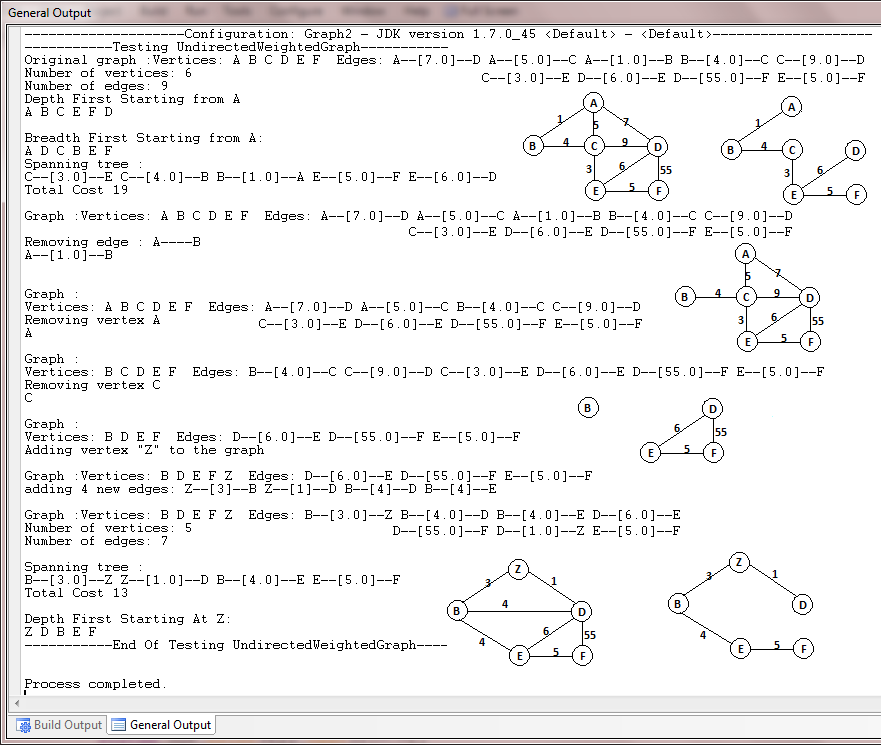


## Graphs Testing

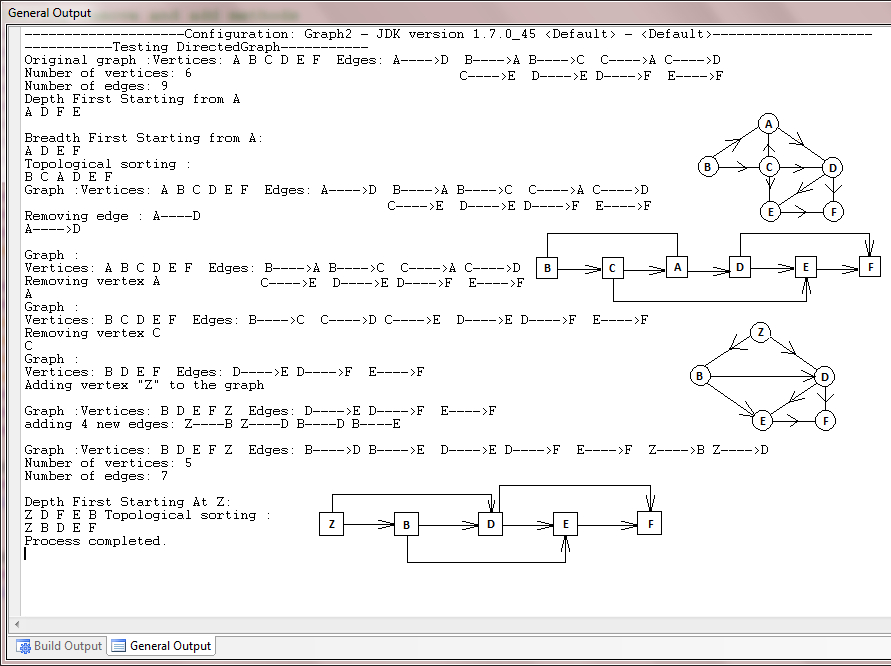
### Undirected Graph



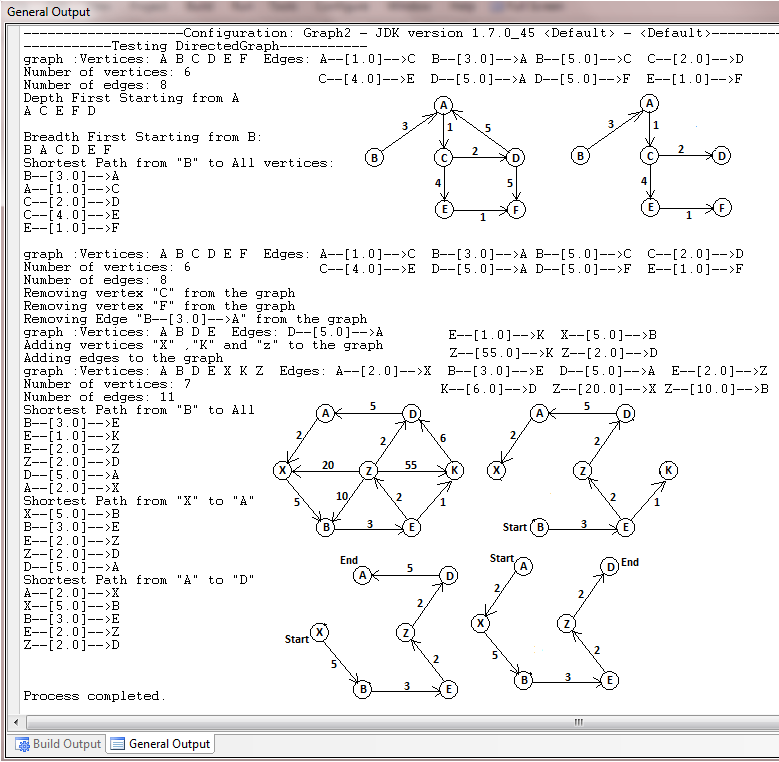
### Undirected Weighted Graph



### Directed Graph



### Directed Weighted Graph



# Conclusion

For the first time in my life I felt like I did something by working on that project. Also this project was my chance to apply everything I have learned thus far in software engineering. I started the project about one month ago. Every day I was working on it for like two hours. Only in the last week, I was working about 4 hours on it. The implementation part of the graph library was easy. The only hard part was implementing traversals and checking for exceptions that can happen.