Homework 1

Arik Rinberg 317156669

Barak Zan 305634487

Question 1

Subsection a

A graph needs the ability to have a node added and an edge added, therefore we chose to add the methods that capture this functionality.

We also need to be able to iterate over the nodes and the children of a given node in order to extract information from the graph.

We chose to not add the ability to iterate over the edges because although adding information it would not assist in this assignment.

Subsection b

We decided to implement the graph as an abstract class getting the node type T. We decided that each node would have a name, and we use this abstract method to get the original node stored in the graph.

We hold a HashMap from names to nodes, and a HashMap from nodes to a HashSet of nodes that are connected via an edge.

This makes all the needed operations very quick as HashMap's and HashSets are lookup and add $O(1)$ time complexity.

Subsection c

We chose to check all the functions that the graph should maintain, and checked every possible throw that can happen.

We tried adding nodes with the same name but different costs, we tried adding a null node. We tried adding edges to non-existing nodes, and tried adding an existing edge.

We made sure to try and check all the needed edge cases, to make sure we protect our class from the users (mostly from ourselves…).

Subsection d

Our namespace has a class called $Path$, so we cannot use import $java.nio.file.Path$ and use it, as we have a conflicting name. Therefore, we must use the full name.

Question 2

Subsection a

$DFS(A(2), D(1))$

Iteration #	Current $start$	$visited$
0	–	$\{\ \}$
1	$A(2)$	$\{A(2)\}$
2	$G(3)$	$\{A(2), G(3)\}$
3	$R(4)$	$\{A(2), G(3), R(4)\}$
4	$D(1)$	$\{A(2), G(3), R(4), D(1)\}$

This run returns true and $visited = \{A(2), G(3), R(4), D(1)\}$.

$DFS(B(3))$

Iteration #	Current $start$	$visited$
0	–	$\{\ \}$
1	$B(3)$	$\{B(3)\}$
2	$E(2)$	$\{B(3), E(2)\}$
3	$A(2)$	$\{B(3), E(2), A(2)\}$
4	$G(3)$	$\{B(3), E(2), A(2), G(3)\}$
5	$R(4)$	$\{B(3), E(2), A(2), G(3), R(4)\}$
6	$D(1)$	$\{B(3), E(2), A(2), G(3), R(4), D(1)\}$

This run returns false and $visited = \{B(3), E(2), A(2), G(3), R(4), D(1)\}$

Subsection c

there is a circle in the graph iff there is a backwards edge in it. Meaning that we got to a node with gray/black child.