Group BM Project Three Proposal

GROUP: BM	TA NAME: Sam
FLIPGRID: https://flip.com/cb37f526	TA EMAIL: sdchurch@wisc.edu

Name	Email	Team	P1 Role	P2 Role	P3 Role
Jeonghyeon Park	jpark634@wisc.edu	Red	BD	DW	FD
Jinwoong Shin	shin225@wisc.edu	Red	FD	AE	BD
MAX Martin	mrmartin6@wisc.edu	Red	FD	BD	AE
Sumanth Karnati	karnati2@wisc.edu	Red	BD	FD	DW
Keene Huynh	kwhuynh@wisc.edu	Blue	BD	DW	FD
Austin Kuo	akuo6@wisc.edu	Blue	FD	AE	BD
Mari Garey	mgarey@wisc.edu	Blue	FD	BD	AE
Jonah Bostrom	bostrom2@wisc.edu	Blue	BD	FD	DW

Project Title: Maze Mapper

Brief Project Description:

“Maze Mapper" is a software application that allows users to solve and visualize mazes stored in the DOT format. This application can be used by anyone who needs to solve a maze and understand its underlying structure.

The program makes use of graph algorithms and data structures Dijkstra's algorithm, to find the shortest path through any maze. The DOT format is a plain text graph description language that is used to represent graphs.

When a user inputs a maze in DOT format, the program first reads and parses the file to create a graph representation of the maze. Then, the program applies the selected algorithm to find the shortest path from the start to the end of the maze. Finally, the program generates a visual representation of the maze, highlighting the shortest path found.

The visual representation of the maze can be helpful for users to understand the structure of the maze and how the algorithm found the shortest path. The program can also be used to compare the performance of different algorithms on the same maze or on different mazes.

Overall, "Maze Mapper" is a useful tool for anyone who needs to solve and understand the structure of a Dijkstra, from students to professionals in various fields.

Representative Tasks Performed Using this Application:

1. Create custom maps that are preloaded for the user (DW)
2. Solve maps for the user using dijkstra's algorithm (BD)
3. Visualize graphs (demo by FD)
4. The application proves that it finds the shortest path through the maze (demo by AE)

Data Wrangler (DW) Role: **Sumanth Karnati, Jonah Bostrom**

The primary responsibility of the Data Wrangler in this project is to handle the loading and parsing of data from the DOT graph description language formatted file. As a DW, we read the file and extract the data fields which can be used by AE and BD.

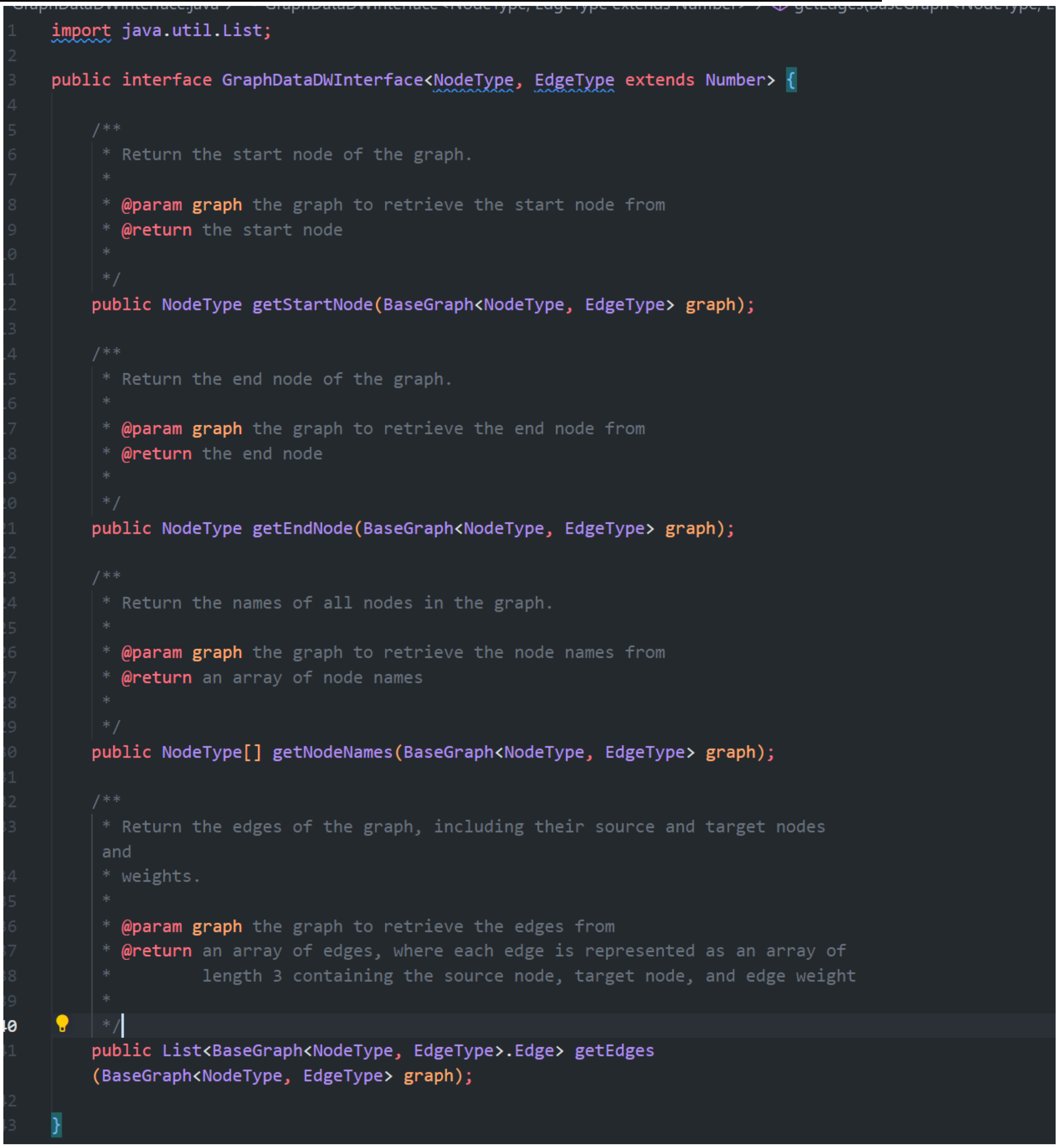
Data Description:

The data will be stored in a DOT format, with nodes and edges (represented by ->). The edges are representative of distance, and the nodes are distinct locations that one can travel. The source of the data will be 3 manually created files created by the DW. These files will be the mazes that are then used by other roles.

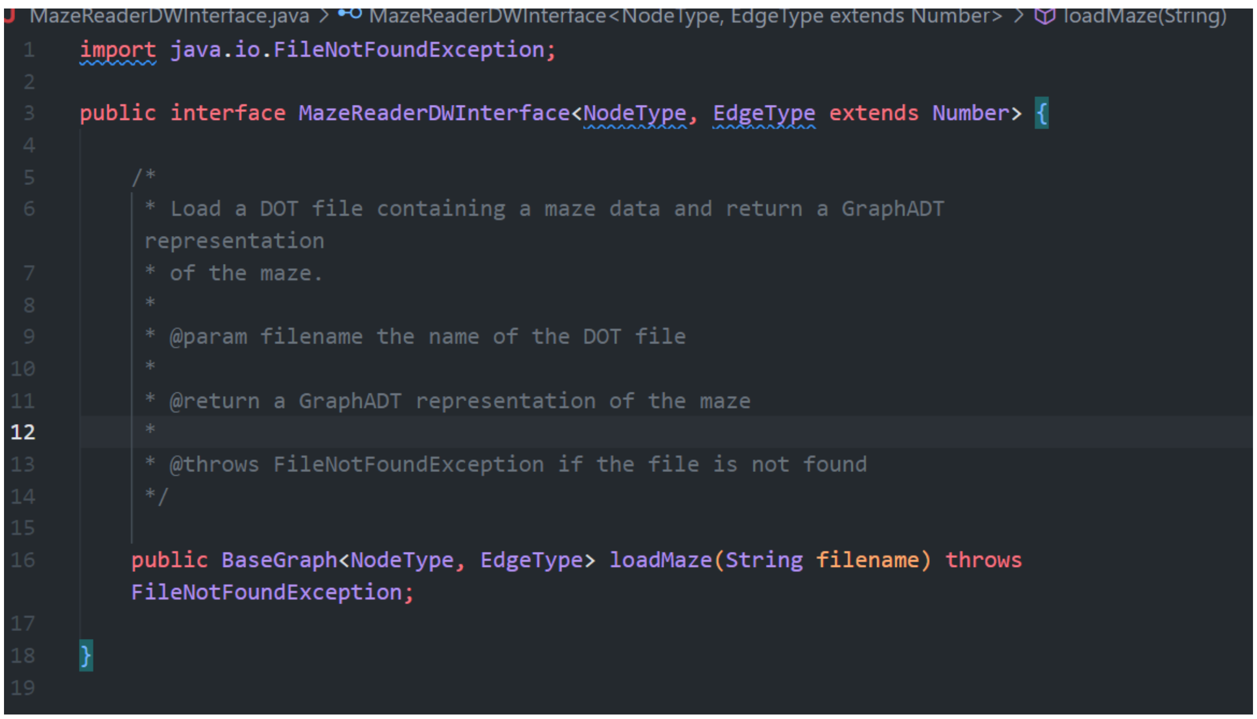
Development Responsibilities:

Class	**Description**	**Interfaces**
MazeReader	Reads in a file DOT file and presents the value	MazeReaderInterface
GraphData	responsible for storing extracted data fields from a DOT file. It contains two arrays, one to store node names and another to store	GraphDataInterface

	adjacency lists for each node.	



```java
import java.util.List;

public interface GraphDataDWInterface<NodeType, EdgeType extends Number> {

    /**
     * Return the start node of the graph.
     *
     * @param graph the graph to retrieve the start node from
     * @return the start node
     *
     */
    public NodeType getStartNode(BaseGraph<NodeType, EdgeType> graph);

    /**
     * Return the end node of the graph.
     *
     * @param graph the graph to retrieve the end node from
     * @return the end node
     *
     */
    public NodeType getEndNode(BaseGraph<NodeType, EdgeType> graph);

    /**
     * Return the names of all nodes in the graph.
     *
     * @param graph the graph to retrieve the node names from
     * @return an array of node names
     *
     */
    public NodeType[] getNodeNames(BaseGraph<NodeType, EdgeType> graph);

    /**
     * Return the edges of the graph, including their source and target nodes
     and
     * weights.
     *
     * @param graph the graph to retrieve the edges from
     * @return an array of edges, where each edge is represented as an array of
     *         length 3 containing the source node, target node, and edge weight
     *
     */
    public List<BaseGraph<NodeType, EdgeType>.Edge> getEdges
    (BaseGraph<NodeType, EdgeType> graph);

}
```


MazeReaderDWInterface.java > MazeReaderDWInterface<NodeType, EdgeType extends Number> > loadMaze(String)

```java
import java.io.FileNotFoundException;

public interface MazeReaderDWInterface<NodeType, EdgeType extends Number> {

    /*
     * Load a DOT file containing a maze data and return a GraphADT
     representation
     * of the maze.
     *
     * @param filename the name of the DOT file
     *
     * @return a GraphADT representation of the maze
     *
     * @throws FileNotFoundException if the file is not found
     */

    public BaseGraph<NodeType, EdgeType> loadMaze(String filename) throws
    FileNotFoundException;

}
```

Presentation Responsibilities:

Data Wrangler will be responsible for showing a full run-through of the application showing the shortestPath, shortestCost, and display of maze 3.

Algorithm Engineer (AE) Role: **MAX Martin, Mari Garey**

The Algorithm Engineer's responsibilities for this project are to give the tools to create the Graph data structure that represents the maze from the .dot file, as well as create the shortest path algorithm to be used to display the solution to the maze using Dijkstra's algorithm.

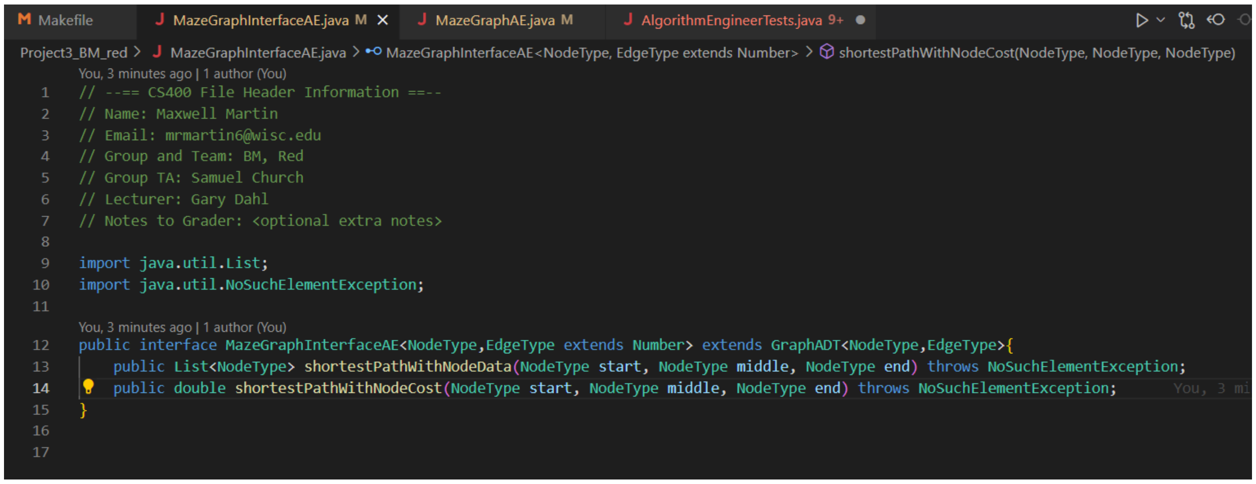
Capabilities Added to Required Data Structure:

The feature that we are adding to the GraphADT data structure is the ability to find the shortest path while including a specified node in the path. In the context of this project that would be you would have to pass through the "treasure" node, and then continue to the end of the path.

Development Responsibilities:

Class	**Description**	**Interfaces**
MazeGraph	This class extends the GraphADT interface,	GraphDataInterface

	which exposes all the methods that are required to build the graph. Additionally we add one method named shortestPathWithNode that will return the shortest path that must pass through a node	


Makefile
MazeGraphInterfaceAE.java M
MazeGraphAE.java M
AlgorithmEngineerTests.java 9+
Project3_BM_red > MazeGraphInterfaceAE.java > MazeGraphInterfaceAE<NodeType, EdgeType extends Number> > shortestPathWithNodeCost(NodeType, NodeType, NodeType)
You, 3 minutes ago | 1 author (You)

```java
// --== CS400 File Header Information ==--
// Name: Maxwell Martin
// Email: mrmartin6@wisc.edu
// Group and Team: BM, Red
// Group TA: Samuel Church
// Lecturer: Gary Dahl
// Notes to Grader: <optional extra notes>

import java.util.List;
import java.util.NoSuchElementException;

public interface MazeGraphInterfaceAE<NodeType,EdgeType extends Number> extends GraphADT<NodeType,EdgeType>{
    public List<NodeType> shortestPathWithNodeData(NodeType start, NodeType middle, NodeType end) throws NoSuchElementException;
    public double shortestPathWithNodeCost(NodeType start, NodeType middle, NodeType end) throws NoSuchElementException;
}
```

Presentation Responsibilities:

Algorithm Engineer will be responsible for showing a full run-through of the application showing the shortestPath, shortestCost, and display of maze 3.

Backend Developer (BD) Rol: **Jinwoong Shin, Austin Kuo**

The Backend Developer is responsible for designing, developing, and maintaining the logic and infrastructure of Maze Mapper App. The backend developer's code must make use of the algorithm engineer's code with the Dijkstra's shortest path algorithm in an application specific way.

Backend Functionality Description:

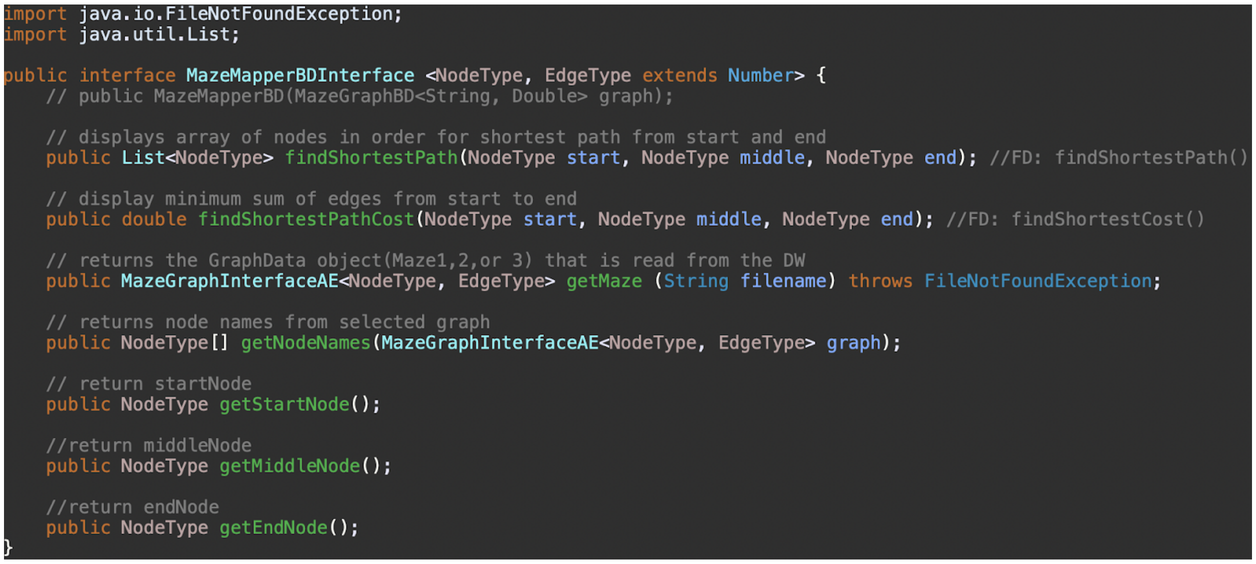
The BD will use the generic code from the AE to apply specifically to this app using String types for the node names and Double types for the edge values.

Development Responsibilities:

The Backend Developer is responsible for creating the code about Dijkstra's algorithm from Algorithm Engineer and providing the data for the Frontend Developer.

Responsible for creating the following .java files:

1. **MazeReaderBD.java** : (placeholder DW)
2. **GraphDataBD.java** : (placeholder DW)
3. **MazeGraphBD.java** : (placeholder AE)
4. **MazeMapperBD** : BD role



```java
import java.io.FileNotFoundException;
import java.util.List;

public interface MazeMapperBDInterface <NodeType, EdgeType extends Number> {
    // public MazeMapperBD(MazeGraphBD<String, Double> graph);

    // displays array of nodes in order for shortest path from start and end
    public List<NodeType> findShortestPath(NodeType start, NodeType middle, NodeType end); //FD: findShortestPath()

    // display minimum sum of edges from start to end
    public double findShortestPathCost(NodeType start, NodeType middle, NodeType end); //FD: findShortestCost()

    // returns the GraphData object(Maze1,2,or 3) that is read from the DW
    public MazeGraphInterfaceAE<NodeType, EdgeType> getMaze (String filename) throws FileNotFoundException;

    // returns node names from selected graph
    public NodeType[] getNodeNames(MazeGraphInterfaceAE<NodeType, EdgeType> graph);

    // return startNode
    public NodeType getStartNode();

    //return middleNode
    public NodeType getMiddleNode();

    //return endNode
    public NodeType getEndNode();
}
```

Presentation Responsibilities:

Backend Developer will be responsible for showing a full run-through of the application showing the shortestPath, shortestCost, and display of maze 1.

Frontend Developer (FD) Role: **Jeonghyeon Park, Keene Huynh**

Frontend developer will be responsible for creating a user interface with text-based-UI that visualizes the maze and each “room” in the maze for the user. It will allow the user to see the visualization of the maze, the shortest paths, and the shortest costs of it. Also, users can choose between a set amount of different mazes (different files) that were provided by the DW.

Log of a Sample Execution of the App:

```
----------------------------------------------------------------------
---------
Welcome to the Maze Mapper.
----------------------------------------------------------------------
---------
Choose a command from the list below:
    [F]ind shortest path
    Find [S]hortest cost
    [C]hoose maze to display
    [Q]uit
Choose command: f
Shortest Path: a -> c
Choose a command from the list below:
    [F]ind shortest path
    Find [S]hortest cost
    [C]hoose maze to display
    [Q]uit
Choose command: s
Shortest Cost: 11.1
Choose a command from the list below:
    [F]ind shortest path
    Find [S]hortest cost
    [C]hoose maze to display
    [Q]uit
Choose command: c
Please input a number 1, 2, or 3 to choose one of the three mazes.
1
The rooms in the mazes are: Room A, Room B, Room C,
Choose a command from the list below:
    [F]ind shortest path
    Find [S]hortest cost
    [C]hoose maze to display
    [Q]uit
Choose command: q
----------------------------------------------------------------------
---------
Thank you for using the Maze Mapper!
----------------------------------------------------------------------
---------
```

Development Responsibilities:

The Java interface for the Frontend Developer's class will be:

```
public interface MazeMapperFDInterface {
    Public void runCommandLoop();
    Public char mainMenuPrompt();
    Public String mazeInput();
    public String findShortestPath();
    Public String findShortestCost();
    public void chooseMaze();
}
```

Presentation Responsibilities:

Frontend Developer will be responsible for showing a full run-through of the application showing the shortestPath, shortestCost, and display of maze 2.

Scope and Signatures:

Ideas for Scoping Up:

<briefly describe a couple of ideas for extra features that could be added to this project>

- Add specific nodes/sections within the maze that are needed to complete the project
  - Akin to a treasure/bonus room
- Alternatively add nodes/sections to avoid within the maze that cannot be accessed regardless of if it is the shortest path to the end
  - A la “room filled with lava or snakes”

Ideas for Scoping Down:

<briefly describe a couple of ideas for features that could be removed from this project>

-

Outside Libraries and Other Tools:

<briefly describe a couple of ideas for features that could be removed from this project>

- Java.utils

Team Signatures:

Name	Email	Team	Type Name As Signature
Jeonghyeon Park	jpark634@wisc.edu	Red	Jeonghyeon Park
Jinwoong Shin	shin225@wisc.edu	Red	Jinwoong Shin
MAX Martin	mrmartin6@wisc.edu	Red	Maxwell Martin
Sumanth Karnati	karnati2@wisc.edu	Red	Sumanth Karnati
Keene Huynh	kwhuynh@wisc.edu	Blue	Keene Huynh
Austin Kuo	akuo6@wisc.edu	Blue	Austin Kuo

Mari Garey	mgarey@wisc.edu	Blue	Mari Garey
Jonah Bostrom	bostrom2@wisc.edu	Blue	Jonah Bostrom

TA Feedback:

I like the project so far, but I think some slight tweaks could make it a bit more interesting. Currently, it seems like it is just a graph visualizer. My suggestion is that you let the user pick the start and end node (and maybe also the "treasure" node). This way they can see how the shortest path and costs change.

As far as presentation responsibilities, these should be more concrete and specific. For every role the presentation responsibility should demo the entire application (and a specific function of the application). An example might be, "load maze 2 and verify that the shortest path is correct." If you implement the suggestion I gave above it will be easier to come up with 4 unique tasks like this.

My main concern implementation-wise is that I don't see how the frontend and the backend communicate with each other. How does the frontend keep a reference to the backend (is this passed in like previous projects, or does the frontend create the backend?). Also, how does the frontend know what the graph is for visualization?

Team Response:

<After grading, if the TA Feedback above describes and required clarifications or changes to this proposal, please discuss as a group before acknowledging and addressing those concerns here.>

Proposal Amendments:

If your group needs to make any changes to what is described above after the proposal deadline, then 1) make sure everyone in your group agrees with those changes, 2) describe those changes in the first empty row below, and then 3) notify your group's TA about those changes and whey are being made. Your TA will then review your request and indicate whether they approve of such changes by adding their initials to the end of that amendment's row below.

Number	Description	TA Approval
1	Add a getMaze1-3() methods to the backend interface to allow the	

	frontend to access the graph utilized by the backend	
2	Changed most instances of ‘String’ in BD interface to NodeType	SC
3	Removed main method from MazeMapperFDInterface	SC
4	Incorporated	
5	Update MazeGraphInterface to extend DijkstraGraph, and add separate methods for both the data and cost	SC
6	Reduce getMaze1-3() methods to ‘getMaze()’ method to the backend interface to allow the frontend to access the graph utilized by the backend.	SC
7	Overhaul frontend implementation to be text-based instead of using JavaFX GUI	SC
8		