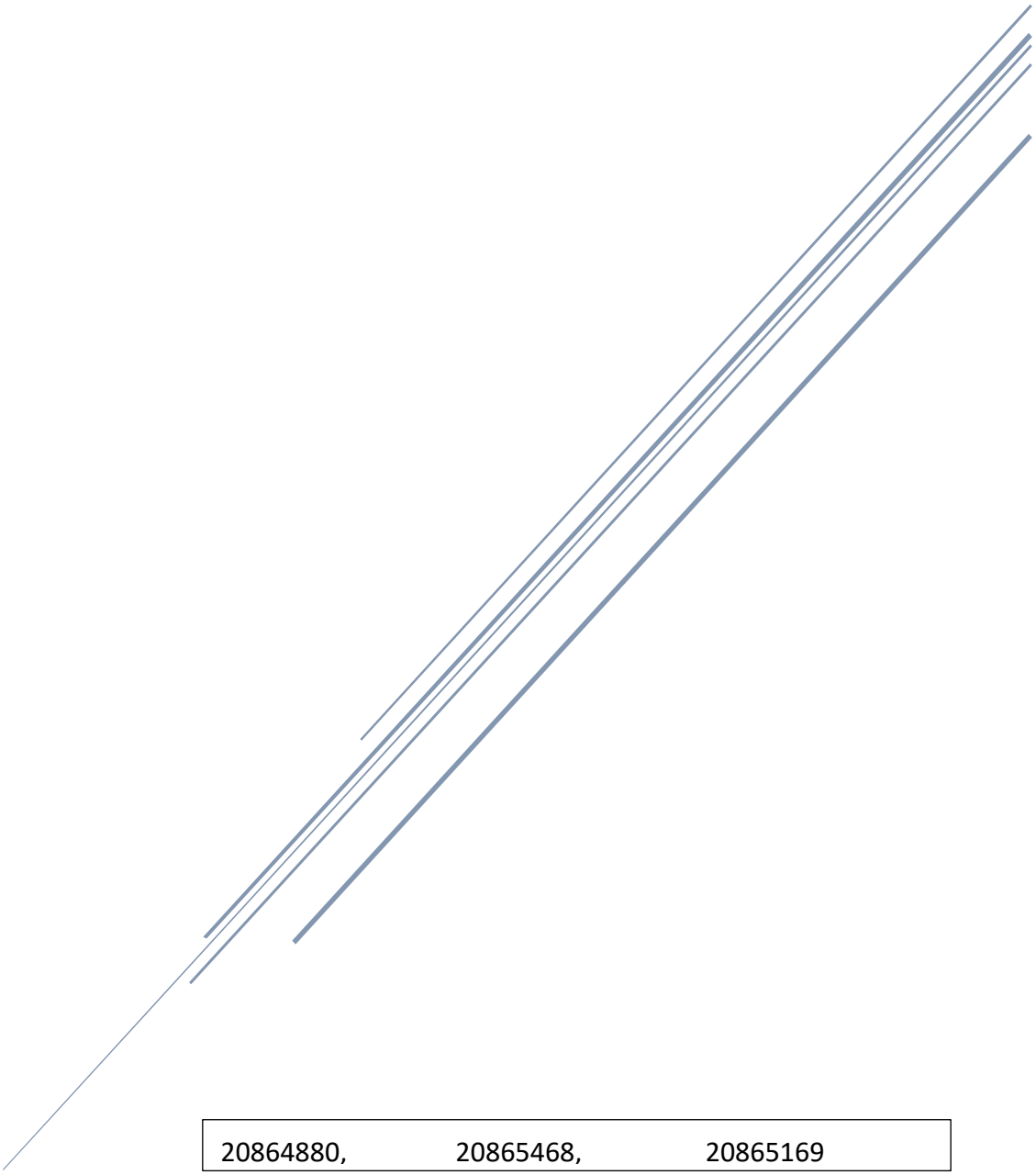
PROJECT GROUP 47

GitHub : https://github.com/bosco713/Comp3111F23G47

YouTube Video : https://youtu.be/qBGfB44JWlw



20864880,	20865468,	20865169

Yuen Man Him, Tsang Hing Ki, Man Lai Chuen

TABLE OF CONTENTS

Meeting Minutes

COMP3111: Software Engineering

Minutes of the 1st Project Meeting

Maze Game

Date: 13/10

Time: 22:00-0000

Place: Online (Discord)

Attending: Yuen Man Him (Bosco), Tsang Hing Ki (Jimmy), Man Lai Chuen (Wesley)

Recorder: Yuen Man Him (Bosco)

1. **Report on progress during the past week**

Name	Tasks worked on in the past week	Total hours
Bosco Yuen	Draw a draft class diagram.	2
Jimmy Tsang	Draw a draft class diagram.	2
Wesley Man	Draw a draft class diagram.	2

2. **Discussion of impediment and resolution**

   We discussed the correctness of the class diagram as we need to submit it this week. Also, we discuss how we are writing the use case specification.

3. **Goals for the coming week**

Name	Tasks that will be worked on in the coming week
Bosco yuen	Finish the use case specification of function A, start reading the template and algorithm website provided.
Jimmy Tsang	Finish the use case specification of function B, start how to design an algorithm to find the shortest path.
Wesley Man	Finish the use case specification of function C, start reading the template snack game.

4. **Meeting adjournment and next meeting**

   After we finish our function's use case specification.

   Around 15-10-2023 night.

COMP3111: Software Engineering

Minutes of the 2nd Project Meeting

Maze Game

Date: 15 – 10 – 2023

Time: 2100-2300

Place: Online (Discord)

Attending: Yuen Man Him (Bosco), Tsang Hing Ki (Jimmy), Man Lai Chuen (Wesley)

Recorder: Yuen Man Him (Bosco)

1. **Report on progress during the past week**

Name	Tasks worked on in the past week	Total hours
Bosco Yuen	Finish the use case specification of function A.	3
Jimmy Tsang	Finish the use case specification of function B.	3
Wesley Man	Finish the use case specification of function C.	3

2. **Discussion of impediment and resolution**

   We discussed the correctness of our use case specification as we needed to hand it in this week. We remove redundant options like "give user choosing different size of maze map generating" and give opinions to one another to give a better specification.

3. **Goals for the coming week**

Name	Tasks that will be worked on in the coming week
Bosco yuen	Try to start writing algorithm using the template.
Jimmy Tsang	Try to start developing the shortest path algorithm.
Wesley Man	Try to start with the code with snake game as reference.

4. **Meeting adjournment and next meeting**

   Wait till we have a code draft on our respective function. We all need to do revision on midterm so the next meeting can be arranged 3 – 4 weeks later. Around 17 – 11 – 2023.

OMP3111: Software Engineering

Minutes of the 3rd Project Meeting

Maze Game

Date: 15 – 11 – 2023

Time: 1030 – 1200

Place: COMP lab

Attending: Yuen Man Him (Bosco), Man Lai Chuen (Wesley)

Recorder: Yuen Man Him (Bosco)

1. **Report on progress during the past week**

Name	Tasks worked on in the past week	Total hours
Bosco Yuen	Finish a draft on generate maze from starch, fixing bug.	13
Jimmy Tsang	NA	NA
Wesley Man	Have a draft of function C from snack game, struggling how to merge it into the system.	10

2. **Discussion of impediment and resolution**

   There are still minor mistakes in our code. So, we decided we should continue building our own function first. We both have a brief understanding of what function we need to build a GUI. We thought of using common functions in building the GUI.

3. **Goals for the coming week**

Name	Tasks that will be worked on in the coming week
Bosco yuen	Continue building function A.
Jimmy Tsang	**NA**
Wesley Man	Try to fix bugs first.

4. **Meeting adjournment and next meeting**

   We have decided to have a quick meeting this weekend to update one another on the progress.

COMP3111: Software Engineering

Minutes of the 4th Project Meeting

Maze Game

Date: 18 – 11 – 2023

Time: 2200 – 0000

Place: Online (Discord)

Attending: Yuen Man Him (Bosco), Tsang Hing Ki (Jimmy), Man Lai Chuen (Wesley)

Recorder: Yuen Man Him (Bosco)

1. **Report on progress during the past week**

Name	Tasks worked on in the past week	Total hours
Bosco Yuen	Build a few classes from starch to generate a maze with more than one path and build a few test cases.	20
Jimmy Tsang	Build the ShortestPath class that can return all the nodes in the shortest path.	20
Wesley Man	Finished the classes: JerryController, TomController, KeyboardListener, PlayGUI and Tuple.	23

2. **Discussion of impediment and resolution**

We have two workable functions that can already corrupt each other: Function A and Function B. The maze map generated by A can use B to get the shortest path from the entry to the exit point. Also, we have a draft of function C that needs more help in merging it into the whole system.

3. **Goals for the coming week**

Name	Tasks that will be worked on in the coming week
Bosco yuen	Start working on more test case, finishing the function A, and explore on the GUI.
Jimmy Tsang	Finalize the ShortestPath class and start working on the unit tests.
Wesley Man	Finish all test Cases for the classes.

4. **Meeting adjournment and next meeting**

We could finish many things using the weekend, so we decided to meet again the next day, 19 – 11 – 2023.

COMP3111: Software Engineering

Minutes of the 5th Project Meeting

Maze Game

Date: 19 – 11 – 2023

Time: 20:00-2300

Place: Online (Discord)

Attending: Yuen Man Him (Bosco), Tsang Hing Ki (Jimmy), Man Lai Chuen (Wesley)

Recorder: Yuen Man Him (Bosco)

1. **Report on progress during the past week**

Name	Tasks worked on in the past week	Total hours
Bosco Yuen	Finish a simple GUI displaying the main menu and function A result.	7
Jimmy Tsang	Finish the unit tests on the ShortestPath class.	3
Wesley Man	Combine maze map in function A and shortest path of function B into function C.	3

2. **Discussion of impediment and resolution**

   Check the correctness of the classes and their functions of all Function A, B and C. We discussed the implementation of shared classes or functions. We gave advice to one another to try to make our code more precise.

3. **Goals for the coming week**

Name	Tasks that will be worked on in the coming week
Bosco yuen	Modify the GUI to make it more flexible, start the documentation.
Jimmy Tsang	Working on the GUI class to showcase the shortest path.
Wesley Man	Finish all test Cases for the classes.

4. **Meeting adjournment and next meeting**

   We need to finish nearly all parts of our system and discuss the test case on the GUI. We arrange our meeting on 25 – 11 – 2023.

COMP3111: Software Engineering

Minutes of the 1st Project Meeting

Maze Game

Date: 22 – 11 – 2023

Time: 1030 – 1200

Place: COMP lab

Attending: Yuen Man Him (Bosco), Man Lai Chuen (Wesley)

Recorder: Yuen Man Him (Bosco)

1. **Report on progress during the past week**

Name	Tasks worked on in the past week	Total hours
Bosco Yuen	Finish the first version of function A GUI, planned to implement one more button to increase the flexibility.	10
Jimmy Tsang	NA	NA
Wesley Man	Finish the GUI for function C and combine it into the main menu. The game works.	8

2. **Discussion of impediment and resolution**

We share our progress, Bosco thought there were some non-functional issues on function C, like “the pre-set moving direction of Jerry”. Asked Wesley to change it.

3. **Goals for the coming week**

Name	Tasks that will be worked on in the coming week
Bosco yuen	Work on the “Generate another map” button.
Jimmy Tsang	**NA**
Wesley Man	Make the game more logical.

4. **Meeting adjournment and next meeting**

This is just a meeting at a random time before the 3111 lectures. We kept our next meeting on 25 – 11 – 2023.

COMP3111: Software Engineering

Minutes of the 7th Project Meeting

Maze Game

Date: 25 – 11 – 2023

Time: 15:00-00:00

Place: Online (Discord)

Attending: Yuen Man Him (Bosco), Tsang Hing Ki (Jimmy), Man Lai Chuen (Wesley)

Recorder: Yuen Man Him (Bosco)

1. **Report on progress during the past week**

Name	Tasks worked on in the past week	Total hours
Bosco Yuen	Finish all the function A and GUI class.	15
Jimmy Tsang	Finish the GUI and the video to demonstrate our system.	12
Wesley Man	Finish all test Cases for the classes.	21

2. **Discussion of impediment and resolution**

   We discussed how we are trying to modify the unit test to achieve high coverage but a smaller number of test cases. At the same time, Jimmy focused on making the video and we were surprised by the result.

3. **Goals for the coming week**

Name	Tasks that will be worked on in the coming week
Bosco yuen	Need to finish the test case for the GUI.
Jimmy Tsang	Start the documentation.
Wesley Man	Try to increase the coverage.

4. **Meeting adjournment and next meeting**

   We arrange our meeting for 26 – 11 – 2023, as we need to submit it by that time.

COMP3111: Software Engineering

Minutes of the 8th Project Meeting

Maze Game

Date: 26 – 11 – 2023

Time: 1600 – 2200

Place: Online (Discord)

Attending: Yuen Man Him (Bosco), Tsang Hing Ki (Jimmy), Man Lai Chuen (Wesley)

Recorder: Yuen Man Him (Bosco)

1. **Report on progress during the past week**

Name	Tasks worked on in the past week	Total hours
Bosco Yuen	Finish all the test cases and report.	20
Jimmy Tsang	Finish all the test cases and report.	17
Wesley Man	Finish all the test cases and report.	26

2. **Discussion of impediment and resolution**

   We discussed how we are doing the GUI testing and the documentation.

3. **Goals for the coming week**

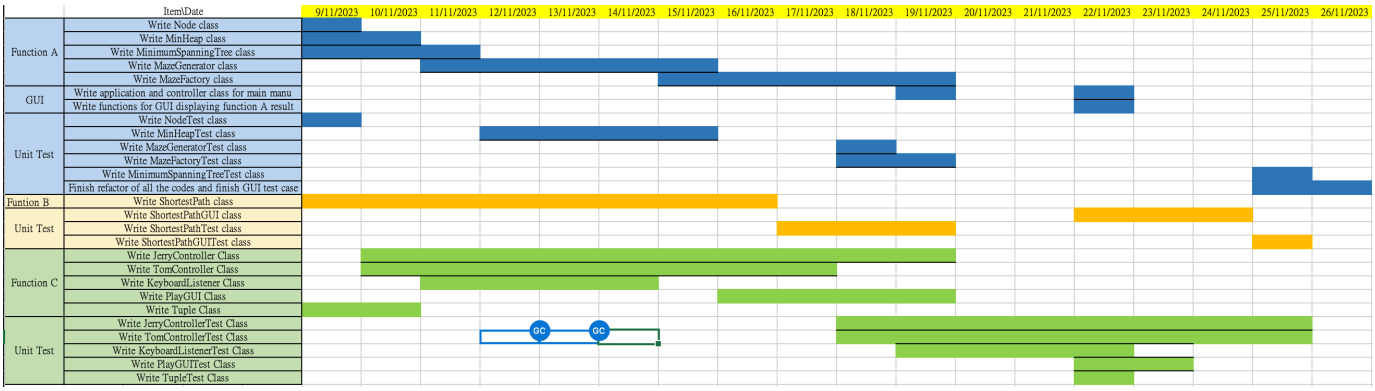
Name	Tasks that will be worked on in the coming week
Bosco yuen	NA
Jimmy Tsang	NA
Wesley Man	NA

4. **Meeting adjournment and next meeting**

   No more meeting after submission

Gantt Chart &
Burndown Chart

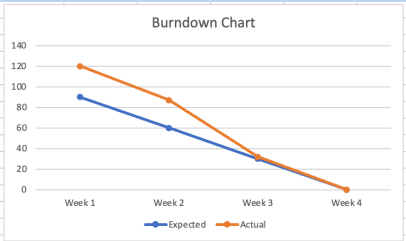
Gantt Chart:



Burndown Chart:

	Sprint Tasks (Sprint Backlog)	Week 1 30-10-2023 - 5-11-2023	Week 2 6-11-2023 - 12-11-2023	Week 3 13-11-2023 - 19-11-2023	Week 4 20-11-2023 - 26-11-2023	
Function A	Write the essential class to contruct the Maze Generator: Node, MinHeap, MinimumSpanningTree		10			
	Build the class to construct of maze map: MazeGenerator, MazeFactory		3	8		
GUI	Write GUI for main menu, and its controller			5		
	Write A GUI to show Function A result and the "Generate new map" button			10	5	
Function A_Unit test	Write test case for Node and MinHeap class		1	4		
	Write test case for MinimumSpanningTree, MazeGerenator, and MazeFactory class			2		
	Finish all the test case for MainMenu and MazeMapGUI				12	
Function B	Write class for generating the Shortest Path		8	2		
	Write GUI for demonstrating the Shortest Path in the maze				3	
Function B_Unit test	Write test case for the ShortestPath class			2		
	Write test case for the ShortestPathGUI class				1	
Function C	Write essential Classes for Jerry: Tuple, KeyboardListener, JerryController		8	9		
	Write the remaing Tom related classes: Tom Controller		2	2		
	Generate the GUI to play the game		1	1		
Function C_Unit test	Write test case for JerryController and Tom Controller			5	4	
	Write test case for Tuple and KeyboardListener			3	2	
	Write test case for PlayGUI			2	5	
	Total hours	0	33	55	32	120

	Expected	Actual
Week 1	90	120
Week 2	60	87
Week 3	30	32
Week 4	0	0

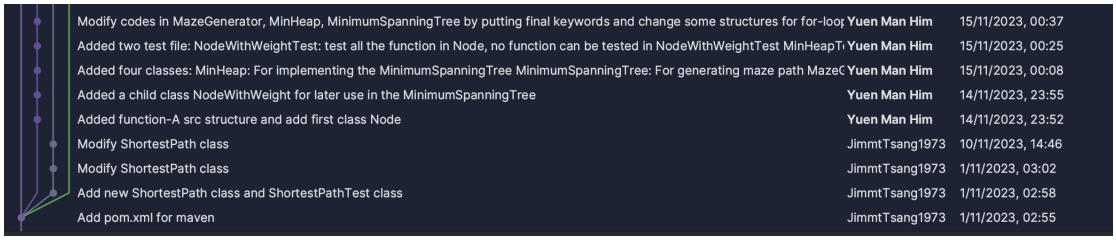


Representative Git Commit Log

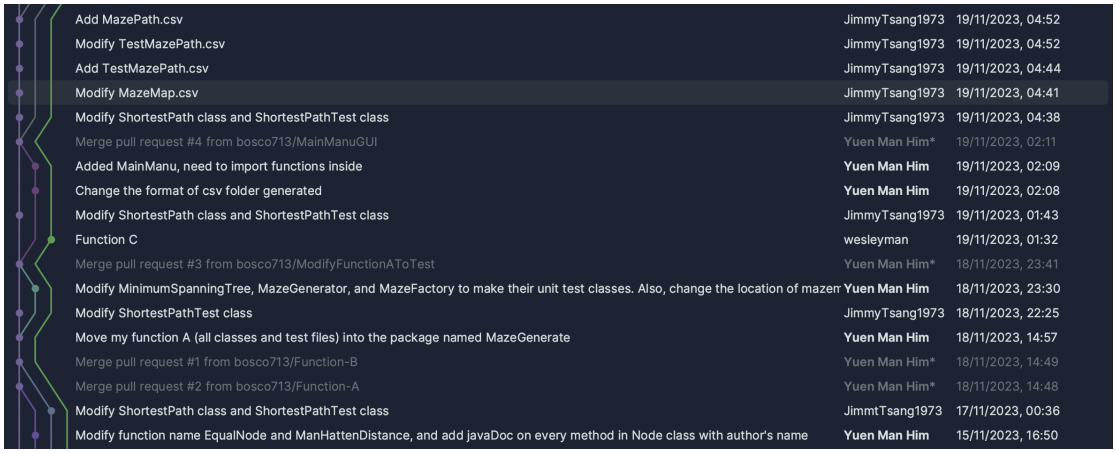
Representative Git Commit Log:

During Data modeling:

Start implementing our function:

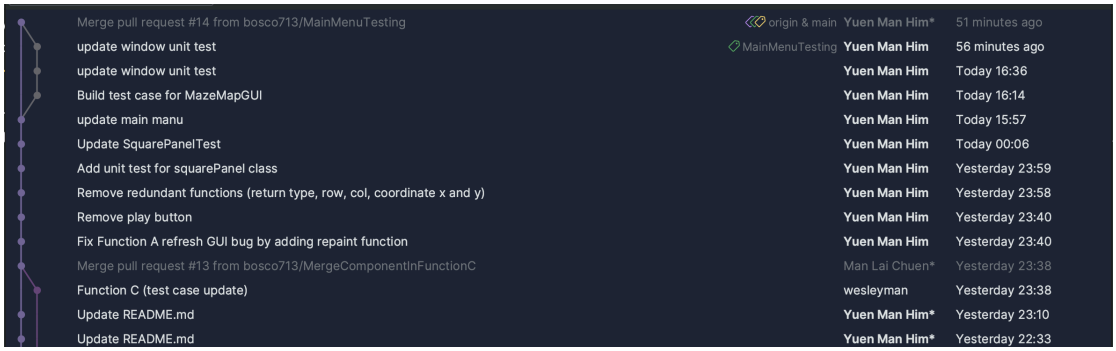


In the middle stage of implementation:



At the final stage of implementation:

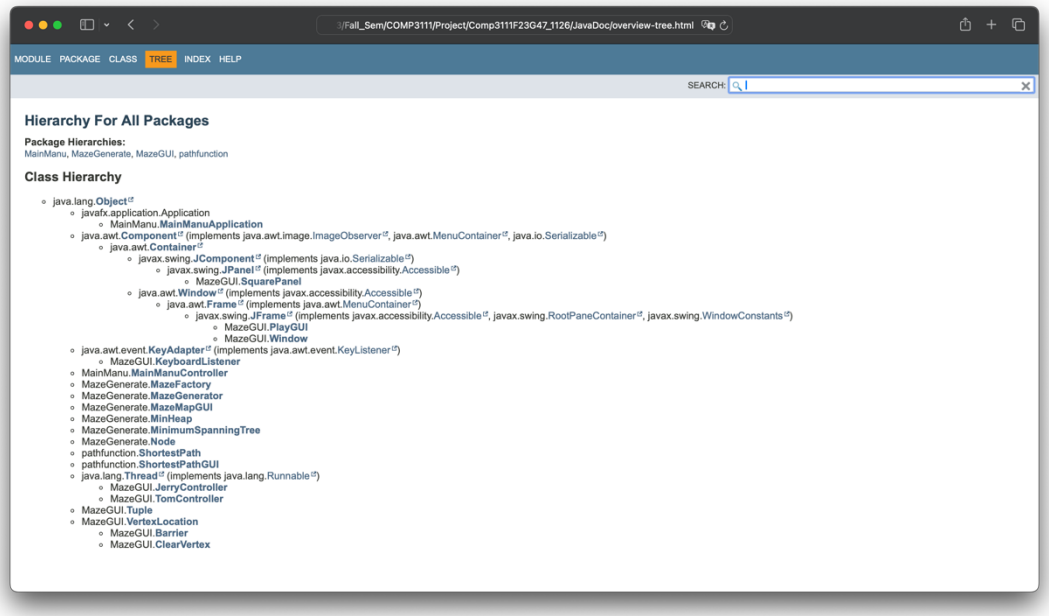
At the end:



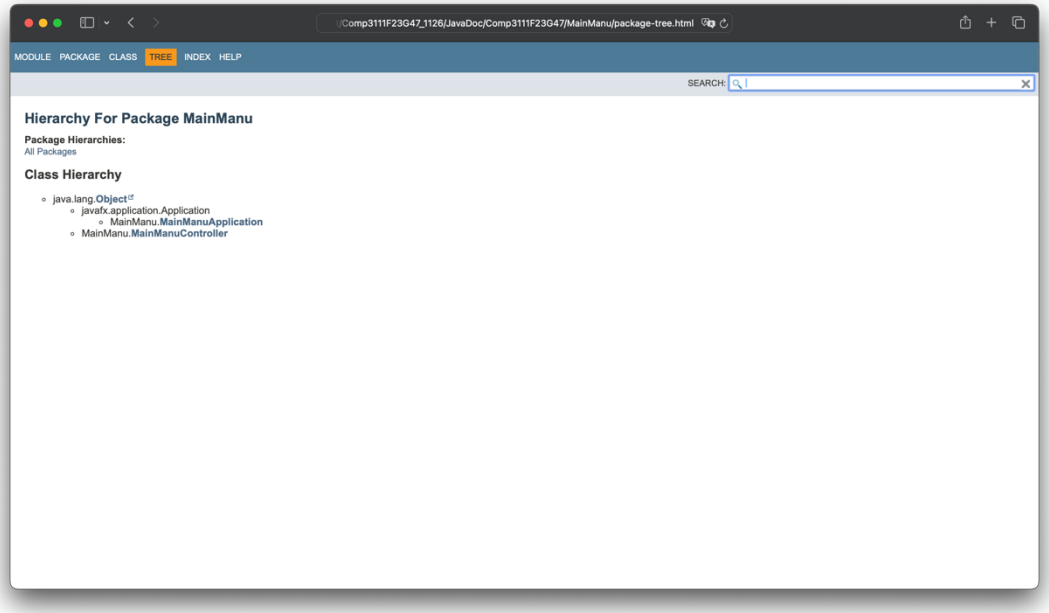
Documentation on the implemented tasks using Javadoc

JavaDoc:

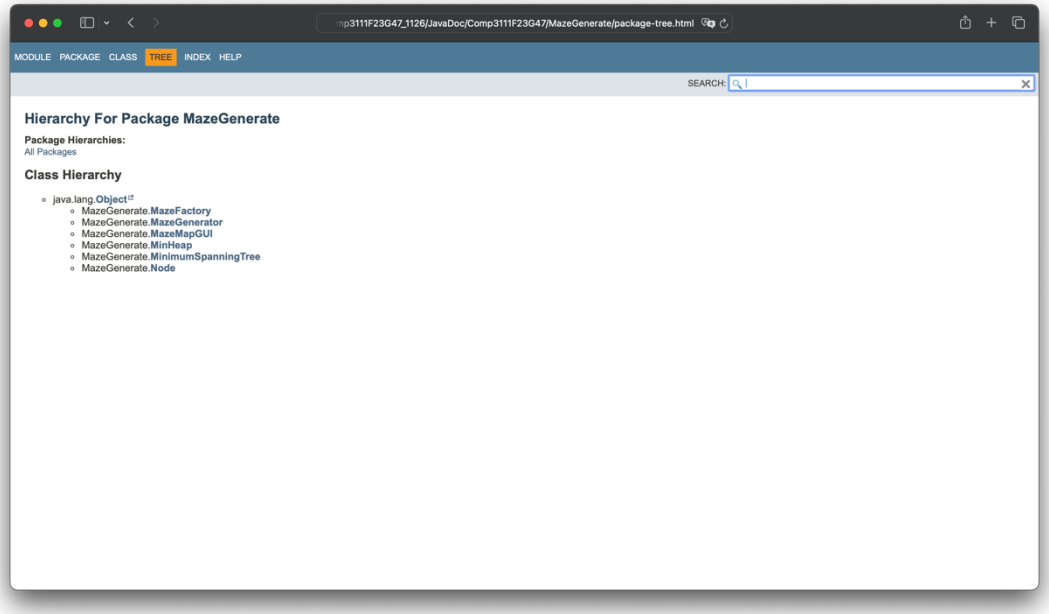
The whole project hierarchy:



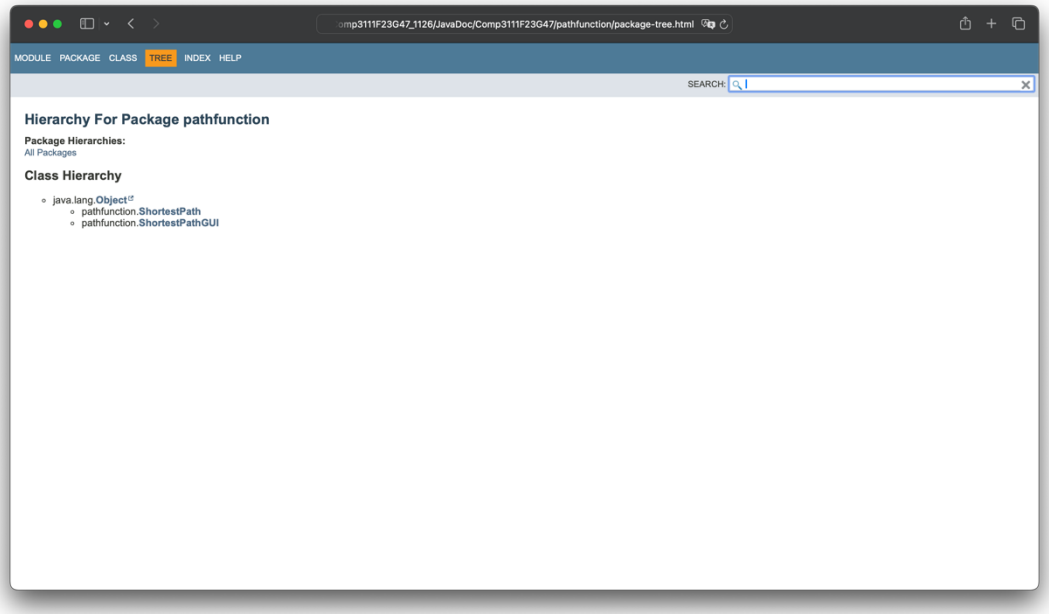
Main Menu:



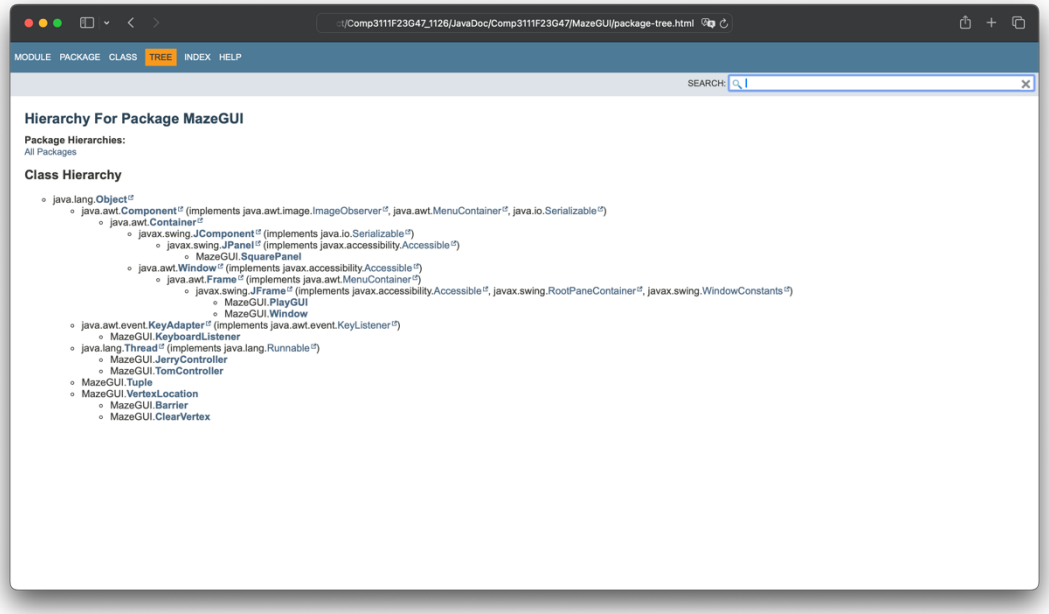
Maze Generate (Function A):



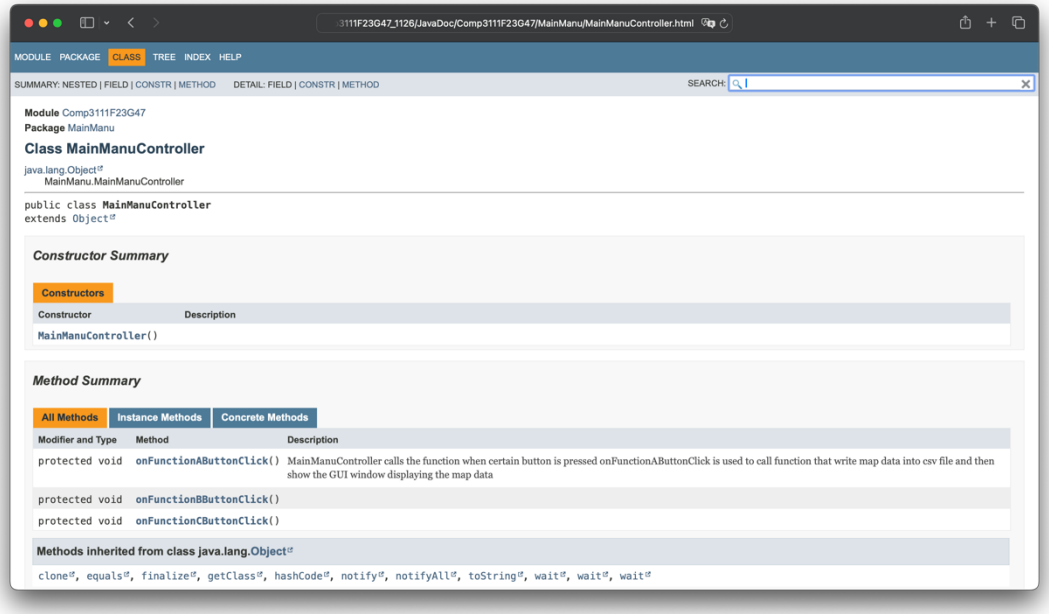
Shortest Path (Function B):



Tom catches Jerry (Function C) together with GUI:



Main Menu Controller (control which functions to test):



3111F23G47_1126/JavaDoc/Comp3111F23G47/MainManu/MainManuController.html

MODULE PACKAGE CLASS TREE INDEX HELP

SUMMARY: NESTED | FIELD | CONSTR | METHOD DETAIL: FIELD | CONSTR | METHOD SEARCH:

Module Comp3111F23G47
Package MainManu

Class MainManuController

java.lang.Object
MainManu.MainManuController

```
public class MainManuController
extends Object
```

Constructor Summary

Constructors

Constructor	Description
MainManuController()	

Method Summary

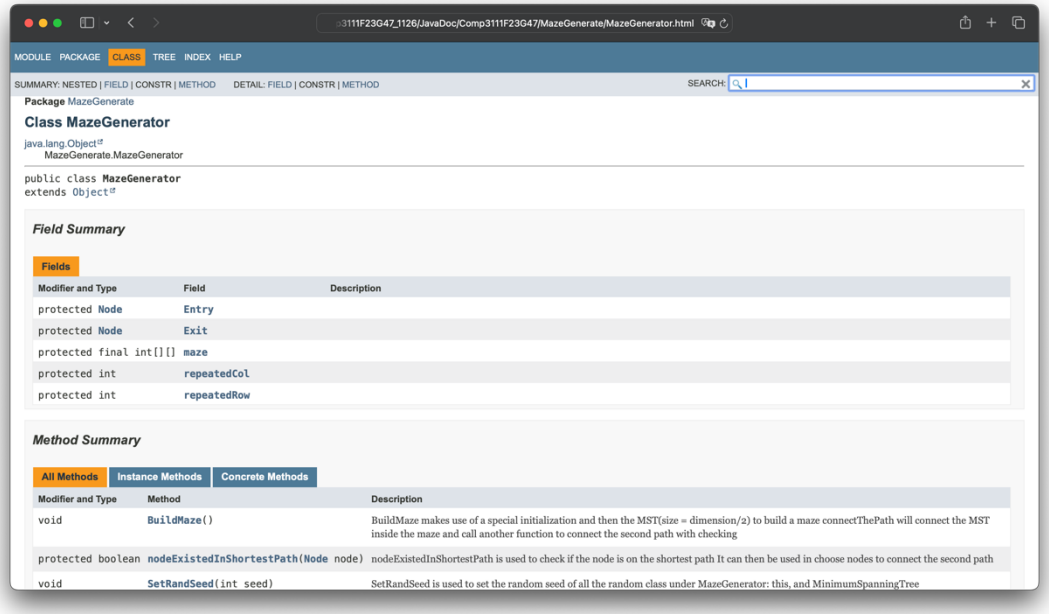
All Methods | Instance Methods | Concrete Methods

Modifier and Type	Method	Description
protected void	onFunctionAButtonClick()	MainManuController calls the function when certain button is pressed onFunctionAButtonClick is used to call function that write map data into csv file and then show the GUI window displaying the map data
protected void	onFunctionBButtonClick()	
protected void	onFunctionCButtonClick()	

Methods inherited from class java.lang.Object

clone, equals, finalize, getClass, hashCode, notify, notifyAll, toString, wait, wait, wait

Essential classes for maze generation (function A):



3111F23G47_1126/JavaDoc/Comp3111F23G47/MazeGenerate/MazeGenerator.html

MODULE PACKAGE CLASS TREE INDEX HELP

SUMMARY: NESTED | FIELD | CONSTR | METHOD DETAIL: FIELD | CONSTR | METHOD SEARCH:

Package MazeGenerate

Class MazeGenerator

java.lang.Object
MazeGenerate.MazeGenerator

```
public class MazeGenerator
extends Object
```

Field Summary

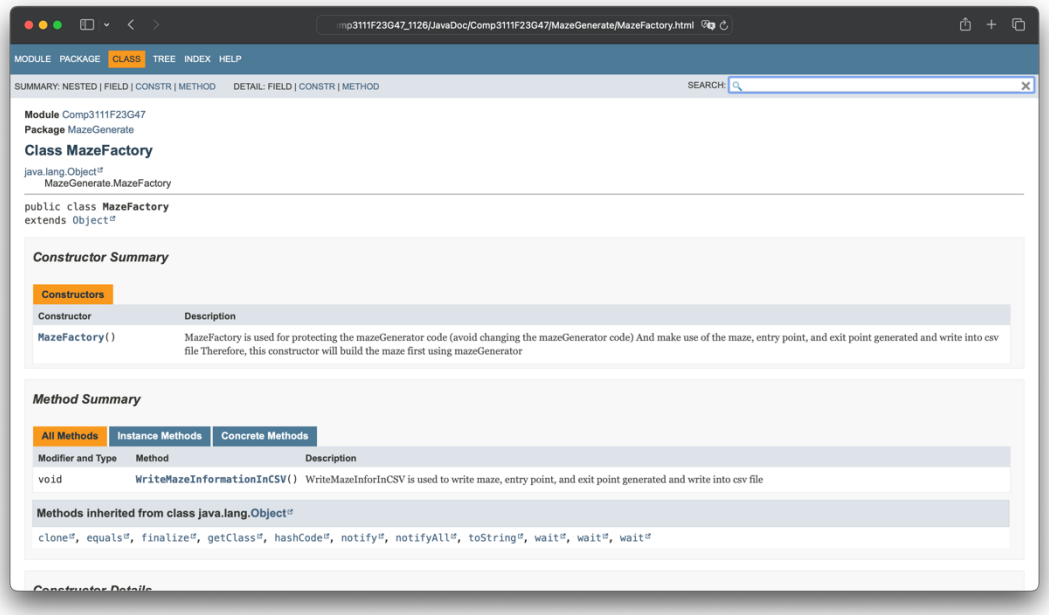
Fields

Modifier and Type	Field	Description
protected Node	Entry	
protected Node	Exit	
protected final int[][]	maze	
protected int	repeatedCol	
protected int	repeatedRow	

Method Summary

All Methods | Instance Methods | Concrete Methods

Modifier and Type	Method	Description
void	BuildMaze()	BuildMaze makes use of a special initialization and then the MST(size = dimension/2) to build a maze connectThePath will connect the MST inside the maze and call another function to connect the second path with checking
protected boolean	nodeExistedInShortestPath(Node node)	nodeExistedInShortestPath is used to check if the node is on the shortest path It can then be used in choose nodes to connect the second path
void	SetRandSeed(int seed)	SetRandSeed is used to set the random seed of all the random class under MazeGenerator: this, and MinimumSpanningTree



mp3111F23G47_1126/JavaDoc/Comp3111F23G47/MazeGenerate/MazeFactory.html

MODULE PACKAGE CLASS TREE INDEX HELP

SUMMARY: NESTED | FIELD | CONSTR | METHOD DETAIL: FIELD | CONSTR | METHOD SEARCH:

Module Comp3111F23G47
Package MazeGenerate

Class MazeFactory

java.lang.Object
MazeGenerate.MazeFactory

```
public class MazeFactory
extends Object
```

Constructor Summary

Constructors

Constructor	Description
MazeFactory()	MazeFactory is used for protecting the mazeGenerator code (avoid changing the mazeGenerator code) And make use of the maze, entry point, and exit point generated and write into csv file Therefore, this constructor will build the maze first using mazeGenerator

Method Summary

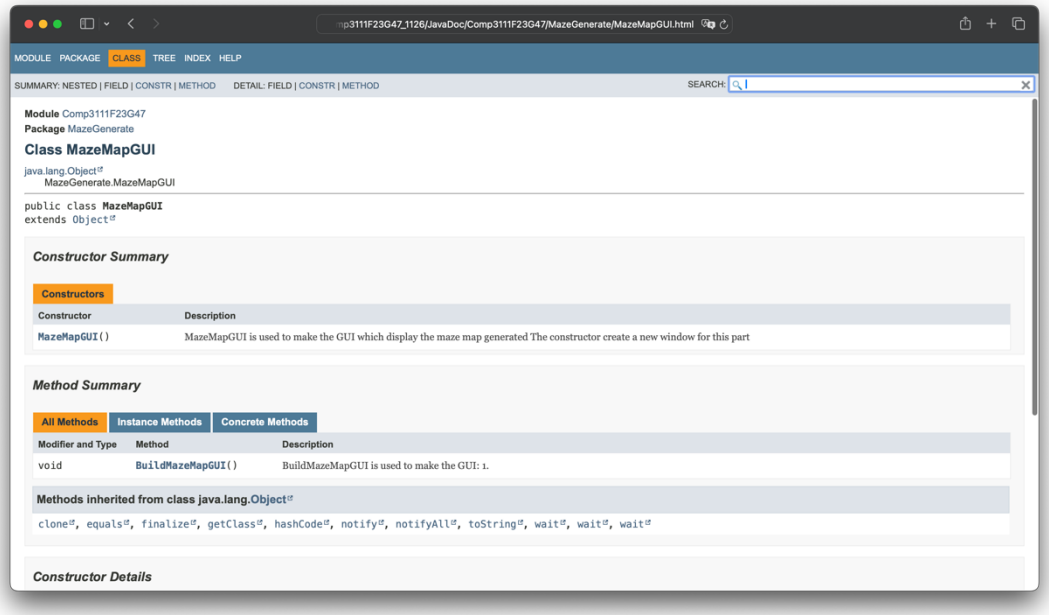
All Methods | Instance Methods | Concrete Methods

Modifier and Type	Method	Description
void	WriteMazeInformationInCSV()	WriteMazeInforInCSV is used to write maze, entry point, and exit point generated and write into csv file

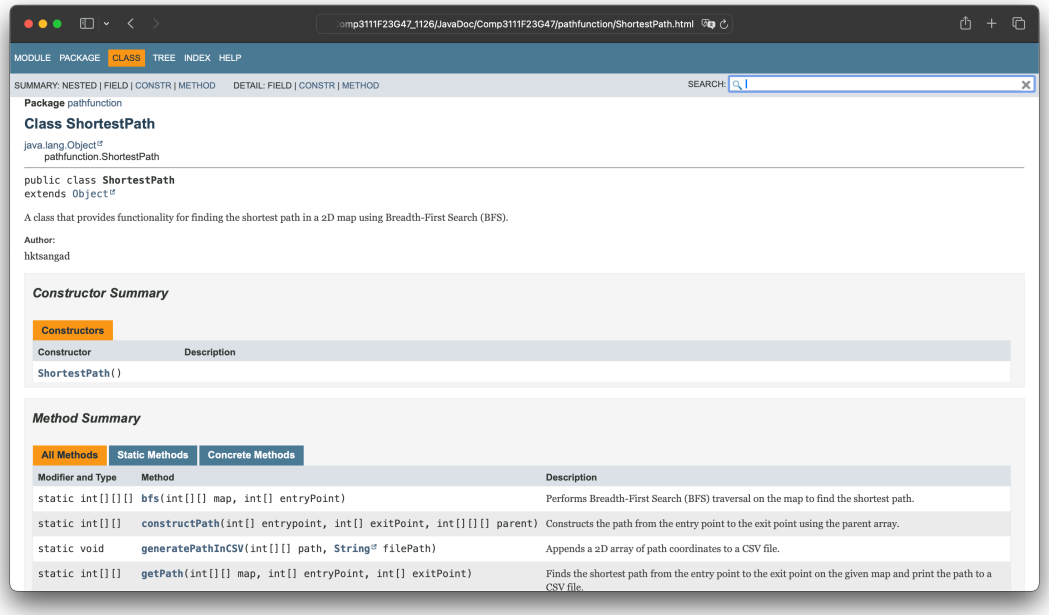
Methods inherited from class java.lang.Object

clone, equals, finalize, getClass, hashCode, notify, notifyAll, toString, wait, wait, wait

Constructor Details


mp3111F23G47_1126/JavaDoc/Comp3111F23G47/MazeGenerate/MazeMapGUI.html
MODULE PACKAGE CLASS TREE INDEX HELP
SUMMARY: NESTED | FIELD | CONSTR | METHOD DETAIL: FIELD | CONSTR | METHOD
SEARCH:
Module Comp3111F23G47
Package MazeGenerate
Class MazeMapGUI
java.lang.Object
MazeGenerate.MazeMapGUI
public class MazeMapGUI
extends Object
Constructor Summary
Constructors
Constructor
Description
MazeMapGUI()
MazeMapGUI is used to make the GUI which display the maze map generated The constructor create a new window for this part
Method Summary
All Methods
Instance Methods
Concrete Methods
Modifier and Type
Method
Description
void
BuildMazeMapGUI()
BuildMazeMapGUI is used to make the GUI: 1.
Methods inherited from class java.lang.Object
clone, equals, finalize, getClass, hashCode, notify, notifyAll, toString, wait, wait, wait
Constructor Details

Essential classes for finding shortest path (function B):



MODULE PACKAGE CLASS TREE INDEX HELP

SUMMARY: NESTED | FIELD | CONSTR | METHOD DETAIL: FIELD | CONSTR | METHOD SEARCH:

Package pathfunction

Class ShortestPath

java.lang.Object
pathfunction.ShortestPath

```
public class ShortestPath
extends Object
```

A class that provides functionality for finding the shortest path in a 2D map using Breadth-First Search (BFS).

Author:
hktsangad

Constructor Summary

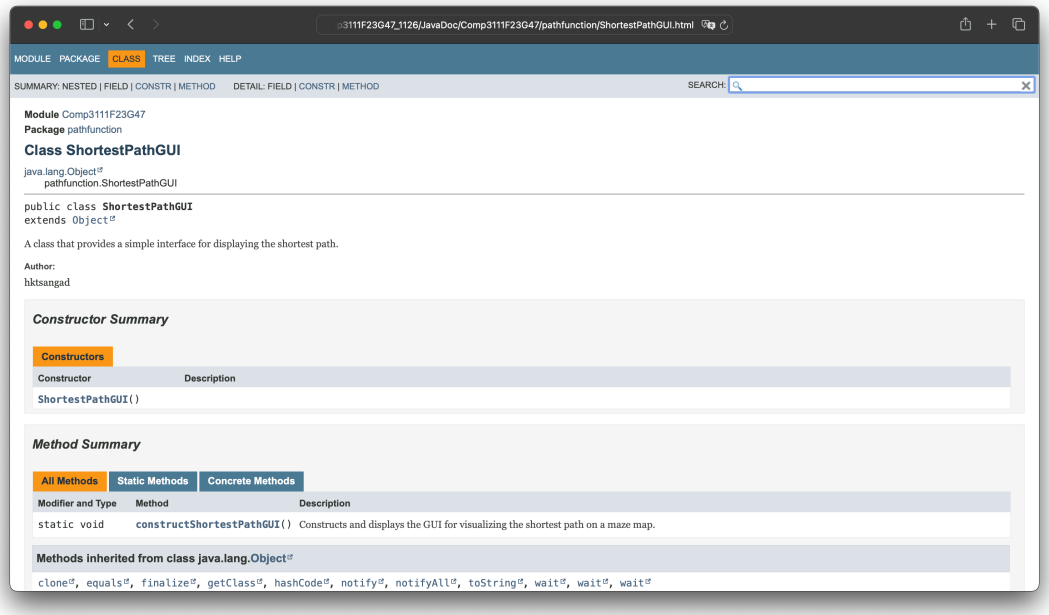
Constructors

Constructor	Description
`ShortestPath()`	

Method Summary

All Methods | **Static Methods** | **Concrete Methods**

Modifier and Type	Method	Description
`static int[][][]`	`bfs(int[][] map, int[] entryPoint)`	Performs Breadth-First Search (BFS) traversal on the map to find the shortest path.
`static int[][]`	`constructPath(int[] entrypoint, int[] exitPoint, int[][][] parent)`	Constructs the path from the entry point to the exit point using the parent array.
`static void`	`generatePathInCSV(int[][] path, String filePath)`	Appends a 2D array of path coordinates to a CSV file.
`static int[][]`	`getPath(int[][] map, int[] entryPoint, int[] exitPoint)`	Finds the shortest path from the entry point to the exit point on the given map and print the path to a CSV file.



MODULE PACKAGE CLASS TREE INDEX HELP

SUMMARY: NESTED | FIELD | CONSTR | METHOD DETAIL: FIELD | CONSTR | METHOD SEARCH:

Module Comp3111F23G47
Package pathfunction

Class ShortestPathGUI

java.lang.Object
pathfunction.ShortestPathGUI

```
public class ShortestPathGUI
extends Object
```

A class that provides a simple interface for displaying the shortest path.

Author:
hktsangad

Constructor Summary

Constructors

Constructor	Description
`ShortestPathGUI()`	

Method Summary

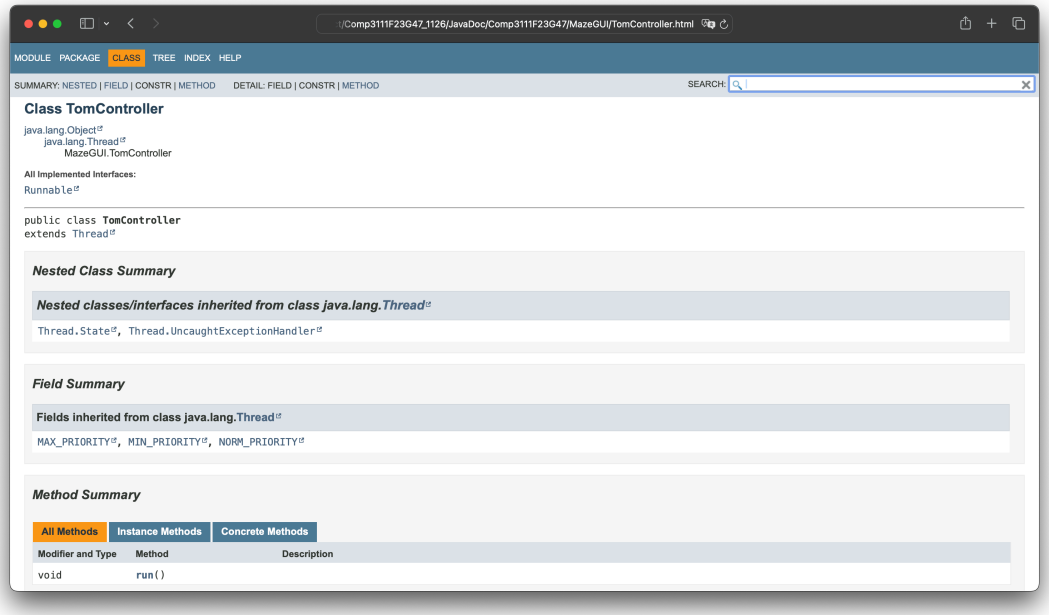
All Methods | **Static Methods** | **Concrete Methods**

Modifier and Type	Method	Description
`static void`	`constructShortestPathGUI()`	Constructs and displays the GUI for visualizing the shortest path on a maze map.

Methods inherited from class java.lang.Object

`clone`, `equals`, `finalize`, `getClass`, `hashCode`, `notify`, `notifyAll`, `toString`, `wait`, `wait`, `wait`

Essential classes for the game (function C):



MODULE PACKAGE CLASS TREE INDEX HELP

SUMMARY: NESTED | FIELD | CONSTR | METHOD DETAIL: FIELD | CONSTR | METHOD SEARCH:

Class TomController

java.lang.Object
java.lang.Thread
MazeGUI.TomController

All Implemented Interfaces:

Runnable

```
public class TomController
extends Thread
```

Nested Class Summary

Nested classes/interfaces inherited from class java.lang.Thread

Thread.State, Thread.UncaughtExceptionHandler

Field Summary

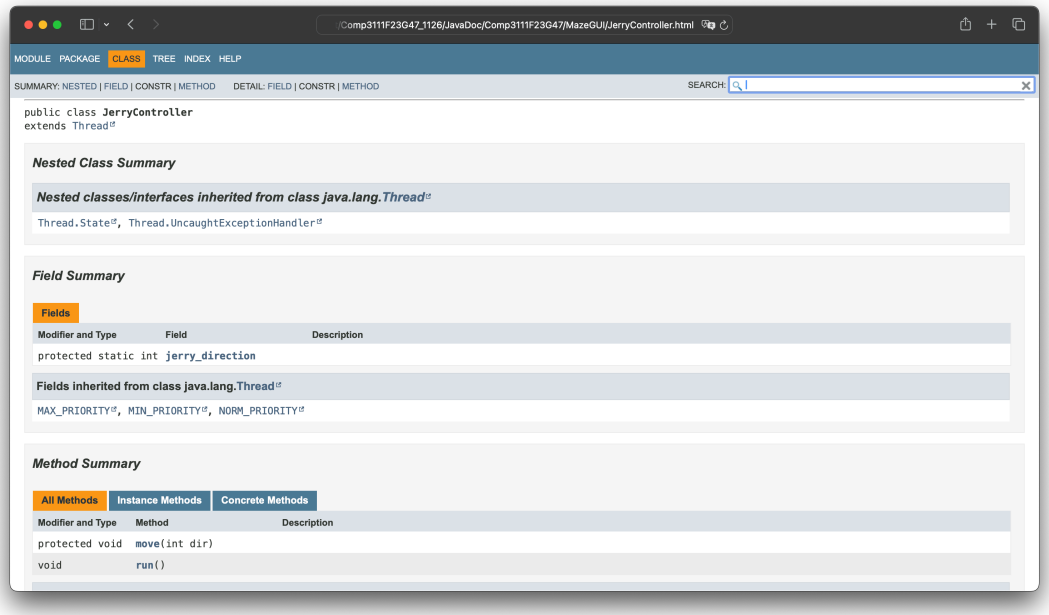
Fields inherited from class java.lang.Thread

MAX_PRIORITY, MIN_PRIORITY, NORM_PRIORITY

Method Summary

All Methods | Instance Methods | Concrete Methods

Modifier and Type	Method	Description
void	run()	



MODULE PACKAGE CLASS TREE INDEX HELP

SUMMARY: NESTED | FIELD | CONSTR | METHOD DETAIL: FIELD | CONSTR | METHOD SEARCH:

```
public class JerryController
extends Thread
```

Nested Class Summary

Nested classes/interfaces inherited from class java.lang.Thread

Thread.State, Thread.UncaughtExceptionHandler

Field Summary

Fields

Modifier and Type	Field	Description
protected static int	jerry_direction	

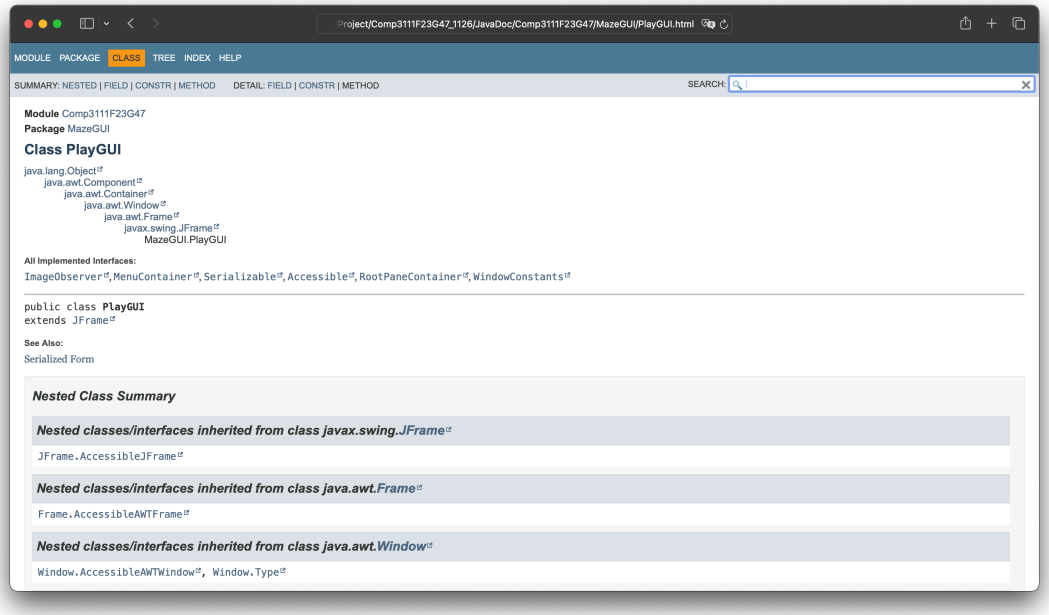
Fields inherited from class java.lang.Thread

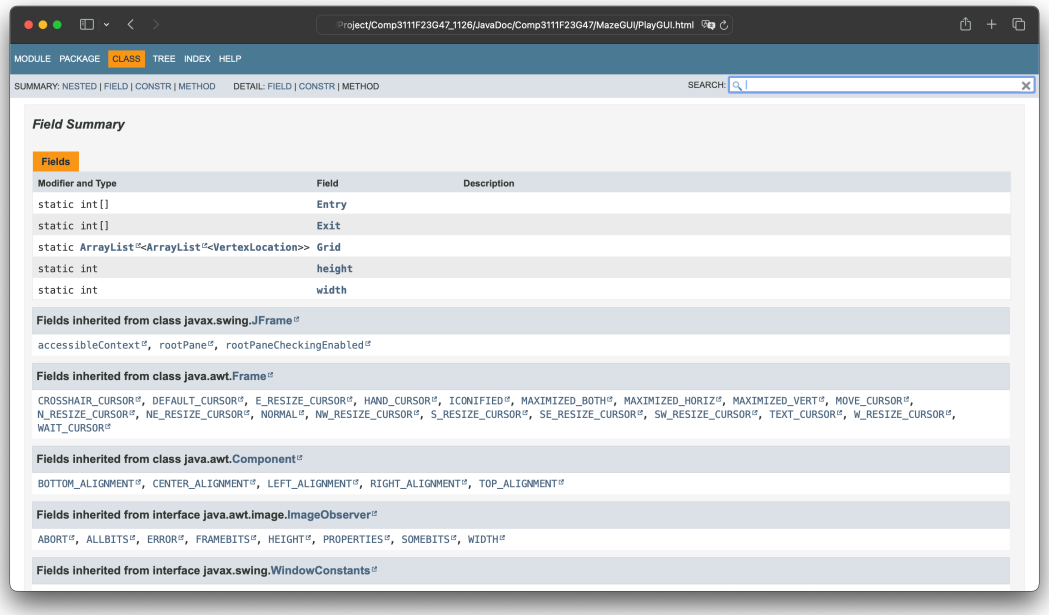
MAX_PRIORITY, MIN_PRIORITY, NORM_PRIORITY

Method Summary

All Methods | Instance Methods | Concrete Methods

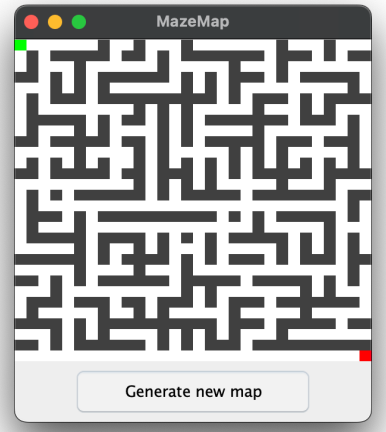
Modifier and Type	Method	Description
protected void	move(int dir)	
void	run()	

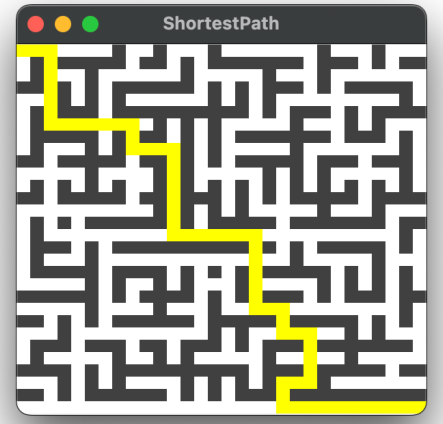

Project/Comp3111F23G47_1126/JavaDoc/Comp3111F23G47/MazeGUI/PlayGUI.html
MODULE PACKAGE CLASS TREE INDEX HELP
SUMMARY: NESTED | FIELD | CONSTR | METHOD DETAIL: FIELD | CONSTR | METHOD SEARCH:
Module Comp3111F23G47
Package MazeGUI
Class PlayGUI
java.lang.Object
java.awt.Component
java.awt.Container
java.awt.Window
java.awt.Frame
javax.swing.JFrame
MazeGUI.PlayGUI
All Implemented Interfaces:
ImageObserver, MenuContainer, Serializable, Accessible, RootPaneContainer, WindowConstants
public class PlayGUI
extends JFrame
See Also:
Serialized Form
Nested Class Summary
Nested classes/interfaces inherited from class javax.swing.JFrame
JFrame.AccessibleJFrame
Nested classes/interfaces inherited from class java.awt.Frame
Frame.AccessibleAWTFrame
Nested classes/interfaces inherited from class java.awt.Window
Window.AccessibleAWTWindow, Window.Type


Project/Comp3111F23G47_1126/JavaDoc/Comp3111F23G47/MazeGUI/PlayGUI.html
MODULE PACKAGE CLASS TREE INDEX HELP
SUMMARY: NESTED | FIELD | CONSTR | METHOD DETAIL: FIELD | CONSTR | METHOD SEARCH:
Field Summary
Fields
Modifier and Type Field Description
static int[] Entry
static int[] Exit
static ArrayList<ArrayList<VertexLocation>> Grid
static int height
static int width
Fields inherited from class javax.swing.JFrame
accessibleContext, rootPane, rootPaneCheckingEnabled
Fields inherited from class java.awt.Frame
CROSSHAIR_CURSOR, DEFAULT_CURSOR, E_RESIZE_CURSOR, HAND_CURSOR, ICONIFIED, MAXIMIZED_BOTH, MAXIMIZED_HORIZ, MAXIMIZED_VERT, MOVE_CURSOR, N_RESIZE_CURSOR, NE_RESIZE_CURSOR, NORMAL, NW_RESIZE_CURSOR, S_RESIZE_CURSOR, SE_RESIZE_CURSOR, SW_RESIZE_CURSOR, TEXT_CURSOR, W_RESIZE_CURSOR, WAIT_CURSOR
Fields inherited from class java.awt.Component
BOTTOM_ALIGNMENT, CENTER_ALIGNMENT, LEFT_ALIGNMENT, RIGHT_ALIGNMENT, TOP_ALIGNMENT
Fields inherited from interface java.awt.image.ImageObserver
ABORT, ALLBITS, ERROR, FRAMEBITS, HEIGHT, PROPERTIES, SOMEBITS, WIDTH
Fields inherited from interface javax.swing.WindowConstants

Screenshots of the Application Software

Maze map GUI and shortest path GUI





It’s respective csv folder

MazeMap.csv

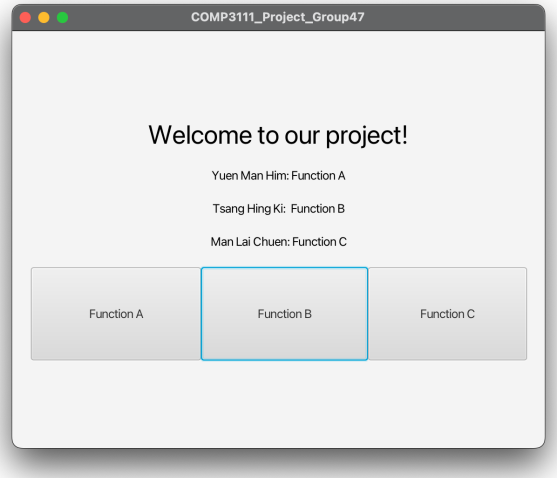
```
0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 1, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 1, 0, 0, 0
0, 1, 0, 1, 1, 1, 1, 1, 0, 1, 1, 0, 1, 0, 1, 1, 1, 1, 1, 1, 1, 0, 1, 1, 1, 0, 1, 0, 1, 1
0, 1, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 1, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0
1, 1, 0, 1, 0, 1, 0, 1, 1, 1, 1, 1, 1, 0, 1, 0, 1, 0, 1, 1, 1, 0, 1, 0, 1, 1, 1, 1, 1, 0
0, 0, 0, 1, 0, 1, 0, 1, 0, 0, 0, 0, 0, 0, 1, 0, 1, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0
0, 1, 0, 1, 1, 1, 0, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 0, 1, 0, 1, 1, 1, 0, 1, 0, 1, 1, 1, 1
0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 1, 0, 1, 0, 0, 0, 0, 0, 1, 0, 0, 0, 1, 0, 1, 0, 0, 0
0, 1, 1, 1, 1, 1, 0, 1, 0, 1, 1, 0, 1, 0, 1, 0, 1, 1, 1, 1, 1, 1, 1, 0, 1, 1, 1, 0, 1, 0
0, 1, 0, 0, 0, 1, 0, 1, 0, 0, 0, 0, 1, 0, 1, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 1, 0
1, 1, 0, 1, 1, 1, 1, 1, 0, 1, 1, 0, 1, 0, 1, 0, 1, 1, 1, 1, 1, 0, 1, 1, 1, 0, 1, 1, 1, 0
0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 1, 0, 1, 0, 0, 0, 0, 0, 1, 0, 1, 0, 1, 0, 0, 0, 0, 0, 1, 0
0, 1, 0, 1, 0, 1, 0, 1, 0, 0, 1, 0, 1, 0, 1, 0, 1, 0, 1, 0, 1, 0, 1, 0, 1, 0, 1, 0, 1, 0
1, 1, 0, 1, 1, 1, 1, 1, 0, 1, 1, 0, 1, 1, 1, 0, 1, 0, 1, 0, 1, 1, 1, 1, 1, 0, 1, 1, 1, 1
0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 1, 0, 1, 0, 1, 0, 0, 0, 0, 0, 1, 0, 0, 0, 1, 0, 0, 0
0, 1, 0, 1, 0, 1, 1, 1, 1, 1, 1, 1, 0, 1, 0, 1, 1, 1, 1, 1, 0, 1, 1, 1, 1, 1, 1, 0, 1, 1
0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0
0, 1, 1, 1, 0, 1, 0, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 0, 1, 0, 1, 0, 1, 1, 1, 1, 1, 0, 1, 0
0, 1, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 1, 0, 0, 0
0, 1, 1, 1, 1, 1, 0, 1, 1, 1, 1, 0, 1, 0, 1, 0, 1, 0, 1, 1, 1, 1, 1, 0, 1, 0, 1, 0, 1, 0
0, 0, 0, 0, 0, 1, 0, 1, 0, 0, 1, 0, 1, 0, 0, 0, 1, 0, 0, 0, 0, 0, 1, 0, 1, 0, 1, 0, 1, 0
1, 1, 1, 1, 1, 1, 0, 1, 0, 1, 1, 1, 1, 0, 1, 0, 1, 0, 1, 1, 1, 0, 1, 0, 1, 1, 1, 0, 1, 1
0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 1, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0
1, 1, 0, 1, 0, 1, 0, 1, 1, 1, 1, 1, 1, 0, 1, 1, 1, 1, 1, 0, 1, 1, 1, 1, 1, 0, 1, 0, 1, 0
0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 1, 0, 0, 0, 0, 0, 0, 0, 1, 0, 1, 0, 0, 0
0, 1, 1, 1, 0, 1, 1, 1, 0, 1, 1, 0, 1, 1, 1, 0, 1, 0, 1, 1, 1, 0, 1, 1, 1, 1, 1, 1, 1, 0
0, 1, 0, 0, 0, 0, 0, 1, 0, 0, 1, 0, 1, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0
1, 1, 1, 1, 0, 1, 0, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 0, 1, 1, 1, 0, 1, 1, 1, 1, 1, 0, 1, 1
0, 0, 0, 1, 0, 1, 0, 1, 0, 0, 0, 0, 1, 0, 1, 0, 1, 0, 1, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0
1, 1, 0, 1, 0, 1, 1, 1, 1, 1, 1, 1, 0, 1, 0, 1, 0, 1, 1, 1, 0, 1, 1, 1, 1, 1, 1, 1, 1, 1
0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0
```

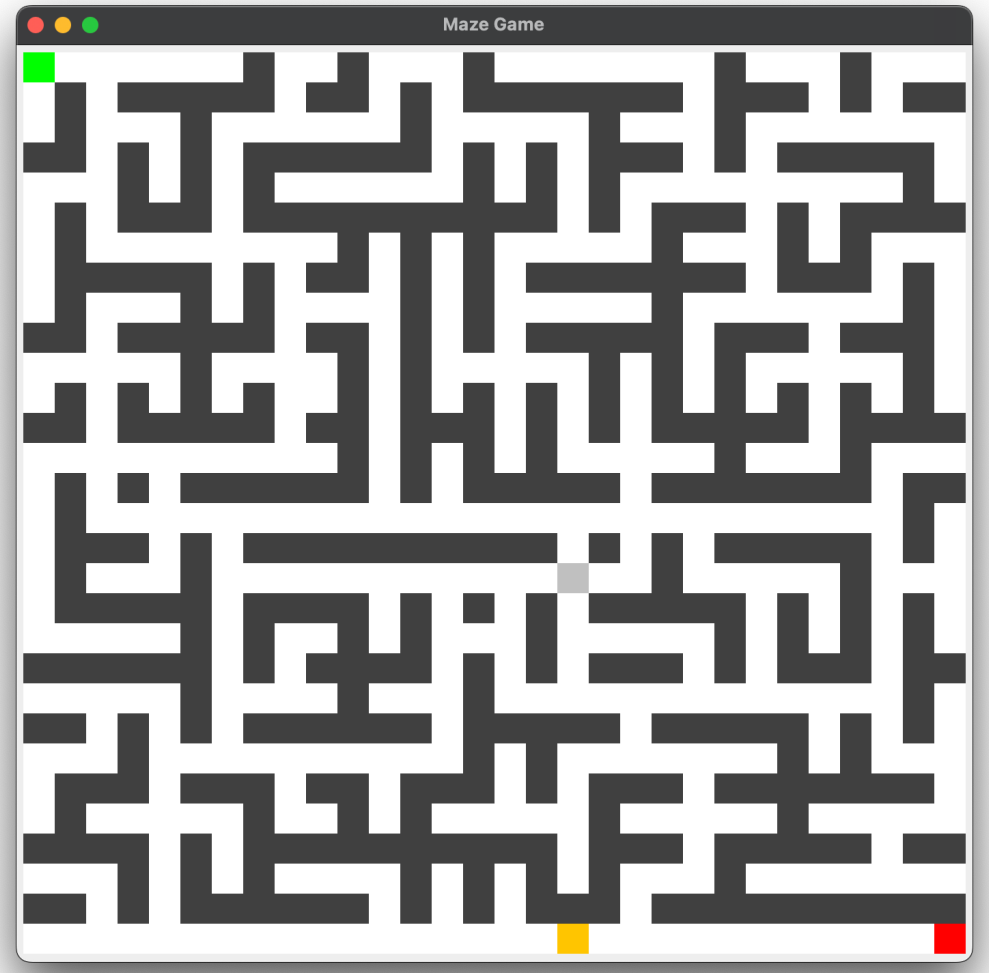
MazeMap.csv MazePath.csv

```
SP, 1, 1, 29, 29;
SP, 1, 2, 29, 28;
SP, 1, 3, 29, 27;
SP, 1, 4, 29, 26;
SP, 1, 5, 29, 25;
SP, 1, 6, 29, 24;
SP, 1, 7, 29, 23;
SP, 1, 8, 29, 22;
SP, 1, 9, 29, 21;
SP, 1, 10, 29, 20;
SP, 1, 11, 29, 19;
SP, 1, 12, 28, 19;
SP, 1, 13, 27, 19;
SP, 1, 14, 27, 20;
SP, 1, 15, 27, 21;
SP, 1, 16, 26, 21;
SP, 1, 17, 25, 21;
SP, 1, 18, 24, 21;
SP, 1, 19, 23, 21;
SP, 1, 20, 23, 20;
SP, 1, 21, 23, 19;
SP, 1, 22, 22, 19;
SP, 1, 23, 21, 19;
SP, 1, 24, 21, 18;
SP, 1, 25, 21, 17;
SP, 1, 26, 20, 17;
SP, 1, 27, 19, 17;
SP, 1, 28, 18, 17;
SP, 1, 29, 17, 17;
SP, 1, 30, 16, 17;
SP, 1, 31, 15, 17;
SP, 1, 32, 15, 16;
SP, 1, 33, 15, 15;
```

```
SP, 1, 33, 15, 15;
SP, 1, 34, 15, 14;
SP, 1, 35, 15, 13;
SP, 1, 36, 15, 12;
SP, 1, 37, 15, 11;
SP, 1, 38, 14, 11;
SP, 1, 39, 13, 11;
SP, 1, 40, 12, 11;
SP, 1, 41, 11, 11;
SP, 1, 42, 10, 11;
SP, 1, 43, 9, 11;
SP, 1, 44, 8, 11;
SP, 1, 45, 8, 10;
SP, 1, 46, 8, 9;
SP, 1, 47, 8, 8;
SP, 1, 48, 7, 8;
SP, 1, 49, 6, 8;
SP, 1, 50, 6, 7;
SP, 1, 51, 6, 6;
SP, 1, 52, 6, 5;
SP, 1, 53, 6, 4;
SP, 1, 54, 6, 3;
SP, 1, 55, 6, 2;
SP, 1, 56, 5, 2;
SP, 1, 57, 4, 2;
SP, 1, 58, 3, 2;
SP, 1, 59, 2, 2;
SP, 1, 60, 1, 2;
SP, 1, 61, 0, 2;
SP, 1, 62, 0, 1;
SP, 1, 63, 0, 0;
```

Main menu & Function C





Report on the unit testing for the implemented tasks & Report on the coverage test

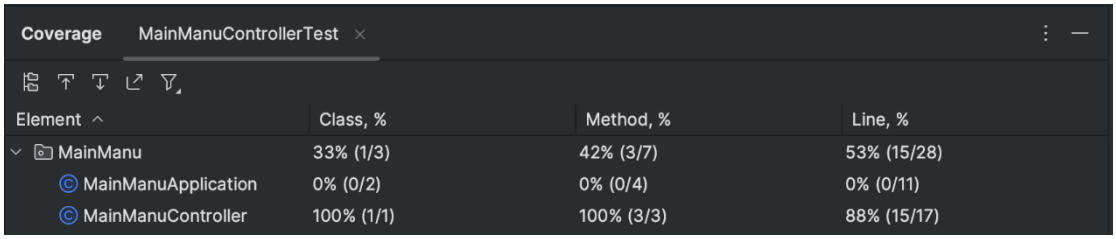
Main Menu

			All implemented functions		
Test case	Target function		onFunctionA ButtonClick()	onFunctionB ButtonClick()	onFunctionC ButtonClick()
TestOnFunction AButtonClick()	onFunctionAB uttonClick()		All covered	na	na
TestOnFunction BButtonClick()	onFunctionBB uttonClick()		na	All covered	na
TestOnFunction CButtonClick()	onFunctionCB uttonClick()		na	na	All covered

Coverage report & coverage

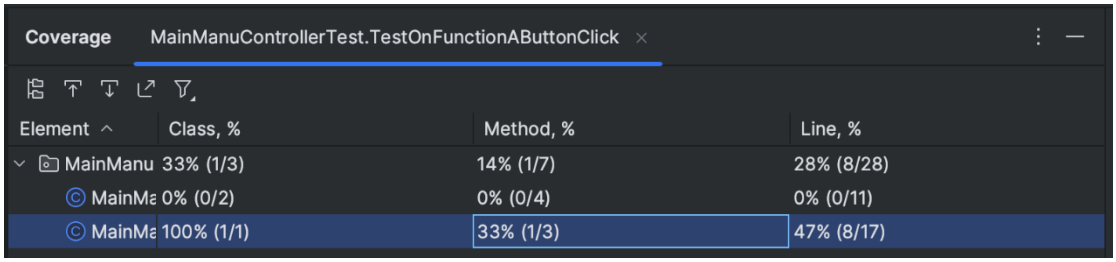
MainManuControllerTest:

Overall:



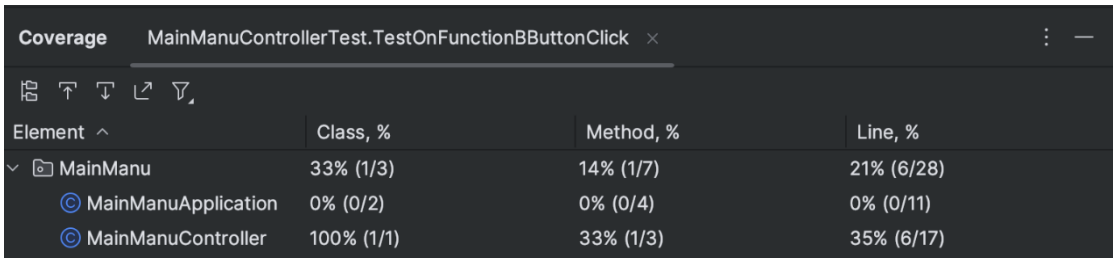
TestOnFunctionAButtonClick: All covered

(Assume no map is inside the map folder when the program first runs)

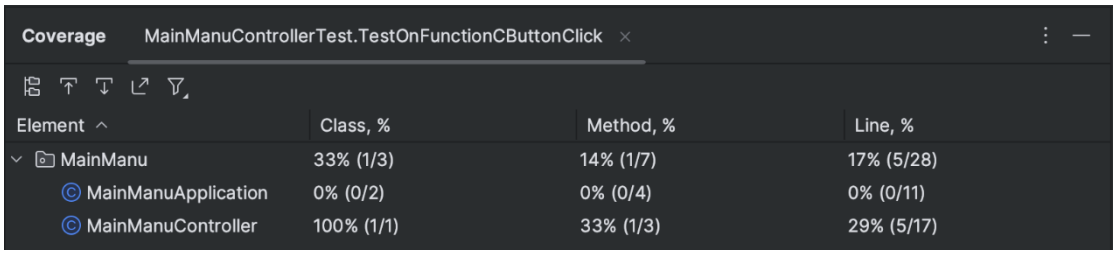


TestOnFunctionBButtonClick: All covered

(Assume no map is inside the map folder when the program first runs)



TestOnFunctionCButtonClick: All covered



For MainManuApplication, we cannot make significant test case testing the window, we have it's coverage but it cannot be done automatically. So we don't decide to include here.

Function A – Maze map

In Node class,

		All implemented functions	
Test cases	Target function	EqualNode()	ManhanttanDistance()
TestEqualNode()	EqualNode()	All covered	na
TestManhanttanDistance()	ManhanttanDistance()	na	All covered

In MinHeap class,

		All implemented functions	
Test cases	Target function	Insert()	ExtractMin()
TestInsert()	Insert()	All covered	na
TestExtractMin()	ExtractMin()	na	All covered

		All implemented functions	
Test cases	Target function	DecreaseKey()	notEmpty()
TestDecreaseKey()	DecreaseKey()	All covered	na
TestNotEmpty()	notEmpty()	na	All covered

		All implemented functions	
Test cases	Target function	swap()	na
TestSwap()	swap()	All covered	na
		na	na

In MinimumSpanningTree class,

		All implemented functions	
Test cases	Target function	SetRandSeed()	GetParentArray()
TestSetRandSeed()	SetRandSeed()	All covered	na
TestGetParentArray()	GetParentArray()	na	All covered

		All implemented functions	
Test cases	Target function	PrimsAlgorithm()	na
TestPrimsAlgorithm()	PrimsAlgorithm()	All covered	na
		na	na

In MazeGenerate class,

		All implemented functions	
Test cases	Target function	SetRandomSeed()	BuildMaze()
TestSetRandomSeed()	SetRandomSeed()	75-77, 83-85, 87-88	na
TestBuildMaze()	BuildMaze()	na	203, 209, 222-227, 230, 239, 244-250, 263-268, 271, 279-284

		All implemented functions
Test cases	Target function	nodeExistedInShortestPath()
TestNodeExistedInShortestPath()	nodeExistedInShortestPath()	All covered

		All implemented functions	
Test cases	Target function	vertices2MazeRow()	vertices2MazeCol()
TestVertices2MazeRow()	vertices2MazeRow()	All covered	na
TestVertices2MazeCol()	vertices2MazeCol()	na	All covered

In MazeFactory class,

		All implemented functions
Test cases	Target function	WriteMazeInformationInCSV()
TestWriteMazeInformationInCSV()	WriteMazeInformationInCSV()	42-43, 63-64

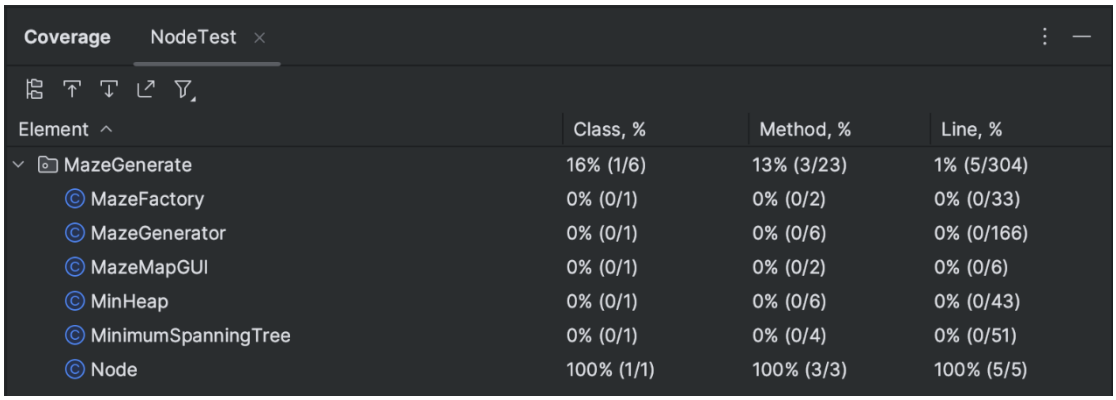
In MazeMapGUI class,

		All implemented functions
Test cases	Target function	BuildMazeMapGUI()
TestBuildMazeMapGUI()	BuildMazeMapGUI()	All covered

Coverage report & any uncovered lines:

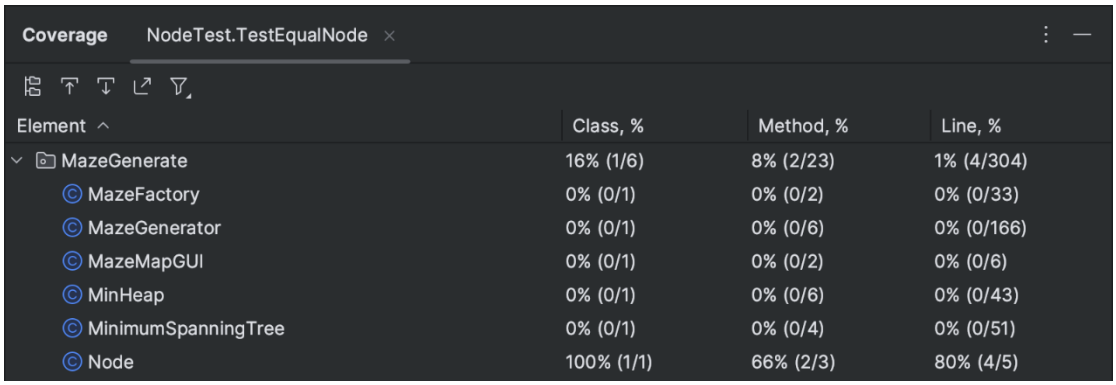
NodeTest class:

Overall:



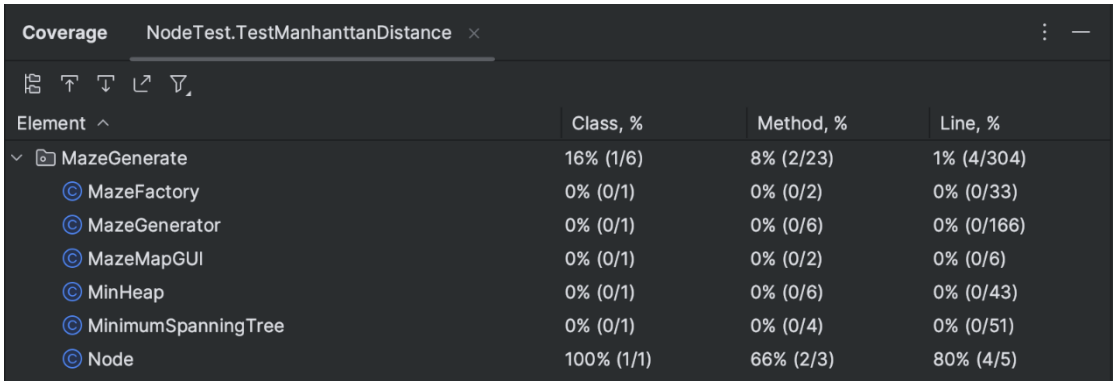
Element	Class, %	Method, %	Line, %
MazeGenerate	16% (1/6)	13% (3/23)	1% (5/304)
MazeFactory	0% (0/1)	0% (0/2)	0% (0/33)
MazeGenerator	0% (0/1)	0% (0/6)	0% (0/166)
MazeMapGUI	0% (0/1)	0% (0/2)	0% (0/6)
MinHeap	0% (0/1)	0% (0/6)	0% (0/43)
MinimumSpanningTree	0% (0/1)	0% (0/4)	0% (0/51)
Node	100% (1/1)	100% (3/3)	100% (5/5)

TestEqualNode: All covered



Element	Class, %	Method, %	Line, %
MazeGenerate	16% (1/6)	8% (2/23)	1% (4/304)
MazeFactory	0% (0/1)	0% (0/2)	0% (0/33)
MazeGenerator	0% (0/1)	0% (0/6)	0% (0/166)
MazeMapGUI	0% (0/1)	0% (0/2)	0% (0/6)
MinHeap	0% (0/1)	0% (0/6)	0% (0/43)
MinimumSpanningTree	0% (0/1)	0% (0/4)	0% (0/51)
Node	100% (1/1)	66% (2/3)	80% (4/5)

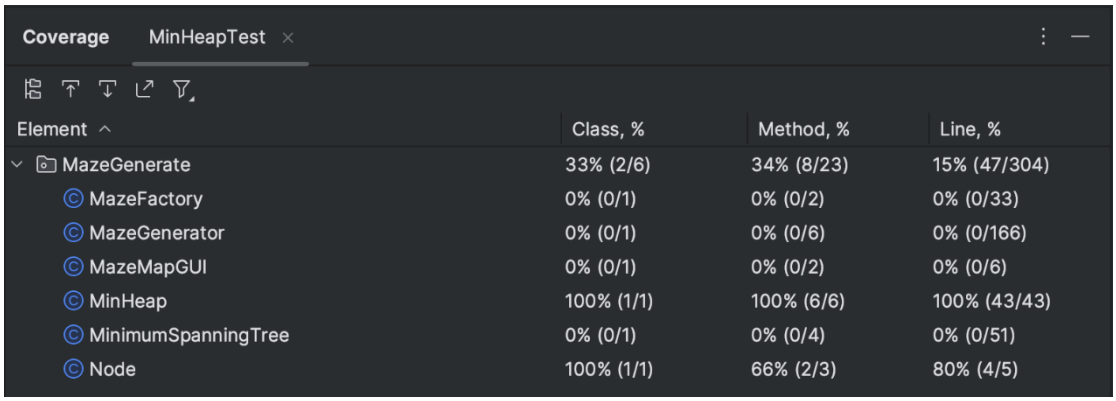
TestManhanttanDistance: All covered



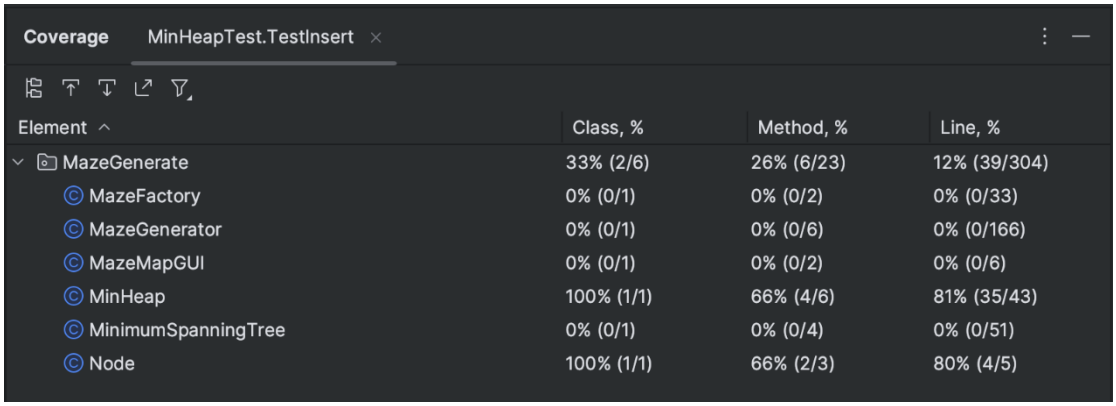
Element	Class, %	Method, %	Line, %
MazeGenerate	16% (1/6)	8% (2/23)	1% (4/304)
MazeFactory	0% (0/1)	0% (0/2)	0% (0/33)
MazeGenerator	0% (0/1)	0% (0/6)	0% (0/166)
MazeMapGUI	0% (0/1)	0% (0/2)	0% (0/6)
MinHeap	0% (0/1)	0% (0/6)	0% (0/43)
MinimumSpanningTree	0% (0/1)	0% (0/4)	0% (0/51)
Node	100% (1/1)	66% (2/3)	80% (4/5)

MinHeapTest:

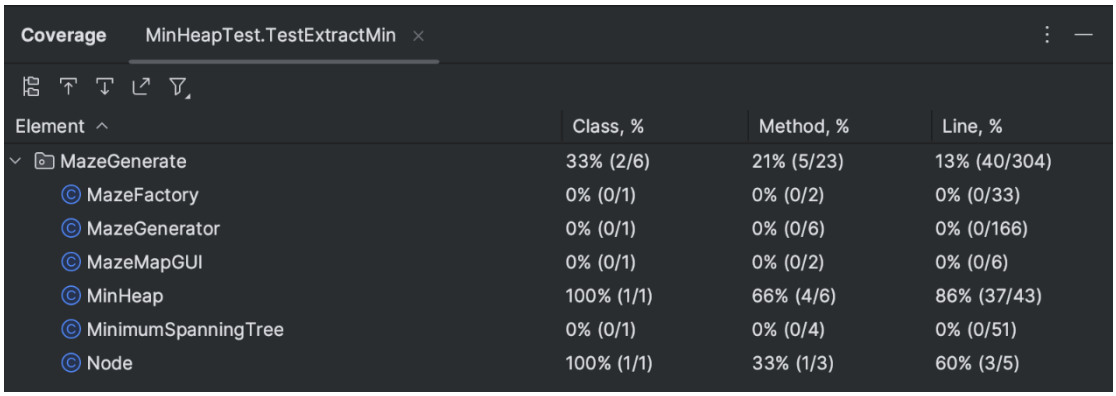
Overall:



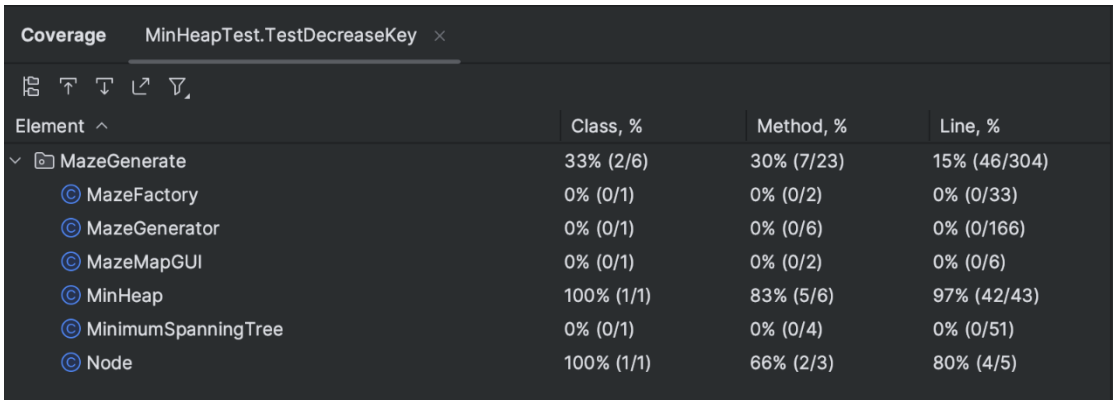
TestInsert: All covered



TestExtractMin: All covered



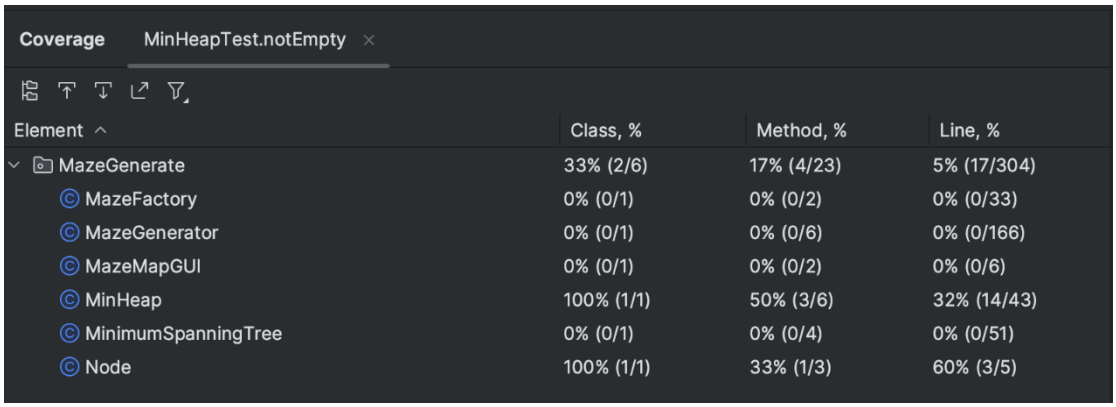
TestDecreaseKey: All covered



Coverage | MinHeapTest.TestDecreaseKey

Element	Class, %	Method, %	Line, %
MazeGenerate	33% (2/6)	30% (7/23)	15% (46/304)
MazeFactory	0% (0/1)	0% (0/2)	0% (0/33)
MazeGenerator	0% (0/1)	0% (0/6)	0% (0/166)
MazeMapGUI	0% (0/1)	0% (0/2)	0% (0/6)
MinHeap	100% (1/1)	83% (5/6)	97% (42/43)
MinimumSpanningTree	0% (0/1)	0% (0/4)	0% (0/51)
Node	100% (1/1)	66% (2/3)	80% (4/5)

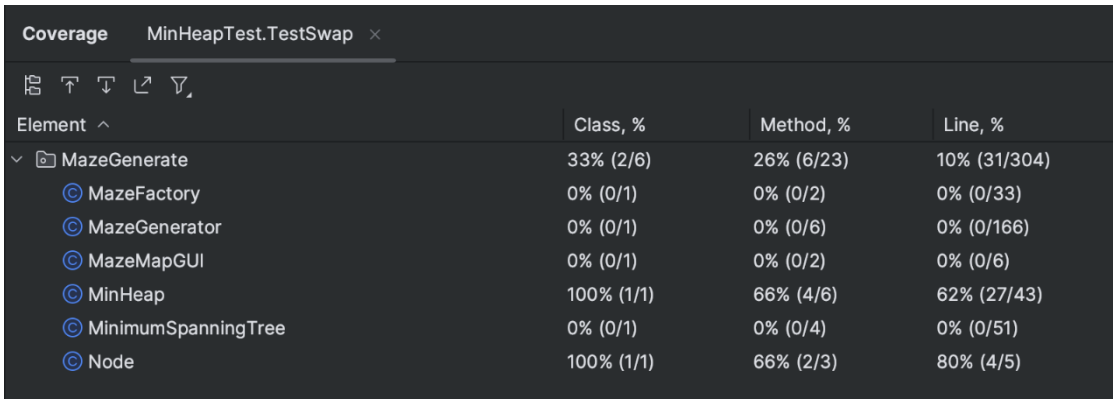
TestNotEmpty: All covered



Coverage | MinHeapTest.notEmpty

Element	Class, %	Method, %	Line, %
MazeGenerate	33% (2/6)	17% (4/23)	5% (17/304)
MazeFactory	0% (0/1)	0% (0/2)	0% (0/33)
MazeGenerator	0% (0/1)	0% (0/6)	0% (0/166)
MazeMapGUI	0% (0/1)	0% (0/2)	0% (0/6)
MinHeap	100% (1/1)	50% (3/6)	32% (14/43)
MinimumSpanningTree	0% (0/1)	0% (0/4)	0% (0/51)
Node	100% (1/1)	33% (1/3)	60% (3/5)

TestSwap: All covered



Coverage | MinHeapTest.TestSwap

Element	Class, %	Method, %	Line, %
MazeGenerate	33% (2/6)	26% (6/23)	10% (31/304)
MazeFactory	0% (0/1)	0% (0/2)	0% (0/33)
MazeGenerator	0% (0/1)	0% (0/6)	0% (0/166)
MazeMapGUI	0% (0/1)	0% (0/2)	0% (0/6)
MinHeap	100% (1/1)	66% (4/6)	62% (27/43)
MinimumSpanningTree	0% (0/1)	0% (0/4)	0% (0/51)
Node	100% (1/1)	66% (2/3)	80% (4/5)

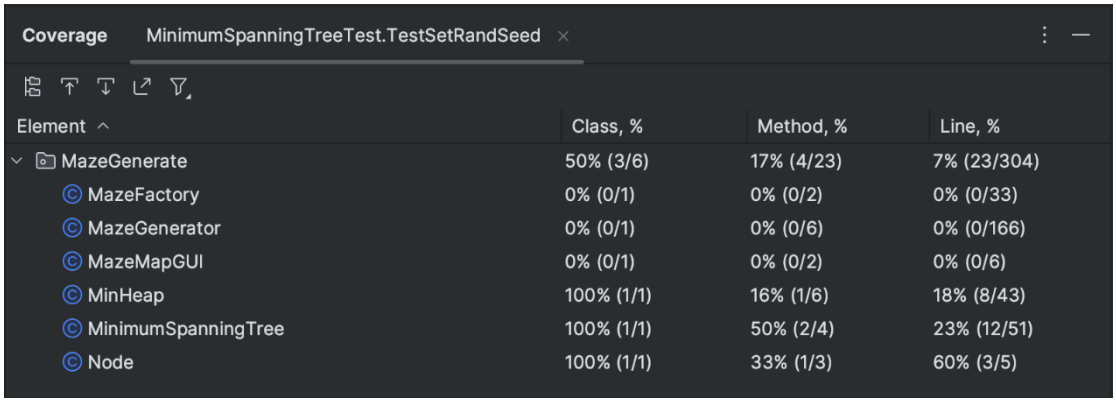
MinimumSpanningTreeTest class:

Overall:

Coverage | MinimumSpanningTreeTest

Element	Class, %	Method, %	Line, %
MazeGenerate	50% (3/6)	52% (12/23)	31% (96/304)
MazeFactory	0% (0/1)	0% (0/2)	0% (0/33)
MazeGenerator	0% (0/1)	0% (0/6)	0% (0/166)
MazeMapGUI	0% (0/1)	0% (0/2)	0% (0/6)
MinHeap	100% (1/1)	100% (6/6)	95% (41/43)
MinimumSpanningTree	100% (1/1)	100% (4/4)	100% (51/51)
Node	100% (1/1)	66% (2/3)	80% (4/5)

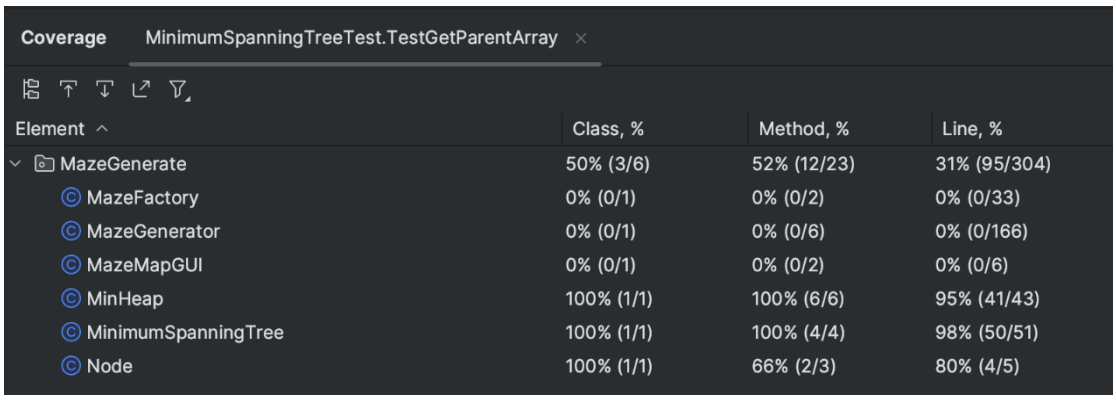
TestSetRandSeed: All covered



Coverage | MinimumSpanningTreeTest.TestSetRandSeed

Element	Class, %	Method, %	Line, %
MazeGenerate	50% (3/6)	17% (4/23)	7% (23/304)
MazeFactory	0% (0/1)	0% (0/2)	0% (0/33)
MazeGenerator	0% (0/1)	0% (0/6)	0% (0/166)
MazeMapGUI	0% (0/1)	0% (0/2)	0% (0/6)
MinHeap	100% (1/1)	16% (1/6)	18% (8/43)
MinimumSpanningTree	100% (1/1)	50% (2/4)	23% (12/51)
Node	100% (1/1)	33% (1/3)	60% (3/5)

TestGetParentArray: All covered



Coverage | MinimumSpanningTreeTest.TestGetParentArray

Element	Class, %	Method, %	Line, %
MazeGenerate	50% (3/6)	52% (12/23)	31% (95/304)
MazeFactory	0% (0/1)	0% (0/2)	0% (0/33)
MazeGenerator	0% (0/1)	0% (0/6)	0% (0/166)
MazeMapGUI	0% (0/1)	0% (0/2)	0% (0/6)
MinHeap	100% (1/1)	100% (6/6)	95% (41/43)
MinimumSpanningTree	100% (1/1)	100% (4/4)	98% (50/51)
Node	100% (1/1)	66% (2/3)	80% (4/5)

TestPrimsAlgorithm: All covered

Coverage | MinimumSpanningTreeTest.TestPrimsAlgorithm

Element	Class, %	Method, %	Line, %
MazeGenerate	50% (3/6)	47% (11/23)	31% (95/304)
MazeFactory	0% (0/1)	0% (0/2)	0% (0/33)
MazeGenerator	0% (0/1)	0% (0/6)	0% (0/166)
MazeMapGUI	0% (0/1)	0% (0/2)	0% (0/6)
MinHeap	100% (1/1)	100% (6/6)	95% (41/43)
MinimumSpanningTree	100% (1/1)	75% (3/4)	98% (50/51)
Node	100% (1/1)	66% (2/3)	80% (4/5)

MazeGeneratorTest class:

Overall:

Coverage | MazeGeneratorTest

Element	Class, %	Method, %	Line, %
MazeGenerate	66% (4/6)	82% (19/23)	77% (235/304)
MazeFactory	0% (0/1)	0% (0/2)	0% (0/33)
MazeGenerator	100% (1/1)	100% (6/6)	81% (136/166)
MazeMapGUI	0% (0/1)	0% (0/2)	0% (0/6)
MinHeap	100% (1/1)	100% (6/6)	100% (43/43)
MinimumSpanningTree	100% (1/1)	100% (4/4)	100% (51/51)
Node	100% (1/1)	100% (3/3)	100% (5/5)

TestSetRandomSeed: 75-77, 83-85, 87-88

Coverage | MazeGeneratorTest.TestSetRandomSeed

Element	Class, %	Method, %	Line, %
MazeGenerate	66% (4/6)	30% (7/23)	16% (49/304)
MazeFactory	0% (0/1)	0% (0/2)	0% (0/33)
MazeGenerator	100% (1/1)	33% (2/6)	15% (25/166)
MazeMapGUI	0% (0/1)	0% (0/2)	0% (0/6)
MinHeap	100% (1/1)	16% (1/6)	18% (8/43)
MinimumSpanningTree	100% (1/1)	50% (2/4)	23% (12/51)
Node	100% (1/1)	66% (2/3)	80% (4/5)

```java
public void SetRandSeed(int seed) { // for unit-testing: need to set up a rand seed and then regener
    rand.setSeed(seed);
    repeatedRow = 2 * (rand.nextInt( bound: (dimension - 2) / 2) + 1);
    repeatedCol = 2 * (rand.nextInt( bound: (dimension - 2) / 2) + 1);
    // selectEntryWithExitPoint()
    // selectEntryWithExitPoint is used to random choose the Entry and Exit Nodes
    // START selectEntryWithExitPoint()
    int randNum = rand.nextInt( bound: 4);
    switch (randNum) {
        case (0) -> {
            Entry = new Node( x: 0,  y: 0);
            Exit = new Node( x: 29,  y: 29);
        }
        case (1) -> {
            Entry = new Node( x: 0,  y: 29);
            Exit = new Node( x: 29,  y: 0);
        }
        case (2) -> {
            Entry = new Node( x: 29,  y: 0);
            Exit = new Node( x: 0,  y: 29);
        }
        case (3) -> {
            Entry = new Node( x: 29,  y: 29);
            Exit = new Node( x: 0,  y: 0);
        }
    }
    // END selectEntryWithExitPoint()
    MST = new MinimumSpanningTree( dim: dimension / 2, randSize, Entry);
    MST.SetRandSeed(seed);
}
```

I test one random seed to generate one set of entry and exit point

TestBuildMaze: 203, 209, 222-227, 230, 239, 244-250, 263-268, 271, 279-284

Coverage | MazeGeneratorTest.TestBuildMaze

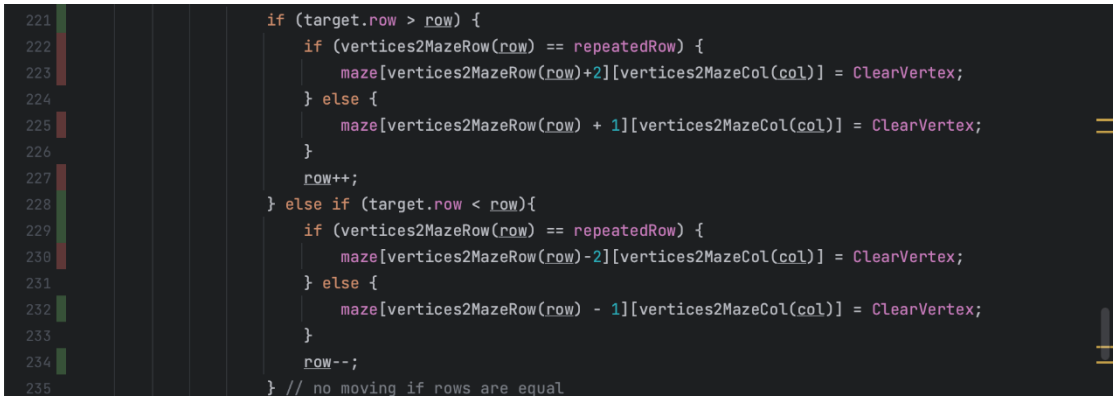
Element	Class, %	Method, %	Line, %
MazeGenerate	66% (4/6)	82% (19/23)	74% (225/304)
MazeFactory	0% (0/1)	0% (0/2)	0% (0/33)
MazeGenerator	100% (1/1)	100% (6/6)	75% (126/166)
MazeMapGUI	0% (0/1)	0% (0/2)	0% (0/6)
MinHeap	100% (1/1)	100% (6/6)	100% (43/43)
MinimumSpanningTree	100% (1/1)	100% (4/4)	100% (51/51)
Node	100% (1/1)	100% (3/3)	100% (5/5)

```java
        // and pick 3 Nodes, 2 from the shortest path and 1 outside
        // connect these 3 Nodes to form the second path
        // START generateSecondPath()
        int num1 = rand.nextInt( bound: ShortestPath.length-2)+1;
        int num2 = rand.nextInt( bound: ShortestPath.length-2)+1;
        while (num2 == num1) {
            num2 = rand.nextInt( bound: ShortestPath.length-2)+1;
        }
        Node otherNode = new Node(rand.nextInt( bound: dimension/2), rand.nextInt( bound: dimension/2));
        while (nodeExistedInShortestPath(otherNode)
                || otherNode.ManhanttanDistance(ShortestPath[num1]) <= 5
                || otherNode.ManhanttanDistance(ShortestPath[num2]) <= 5) {
            otherNode = new Node(rand.nextInt( bound: dimension/2), rand.nextInt( bound: dimension/2));
        }
```

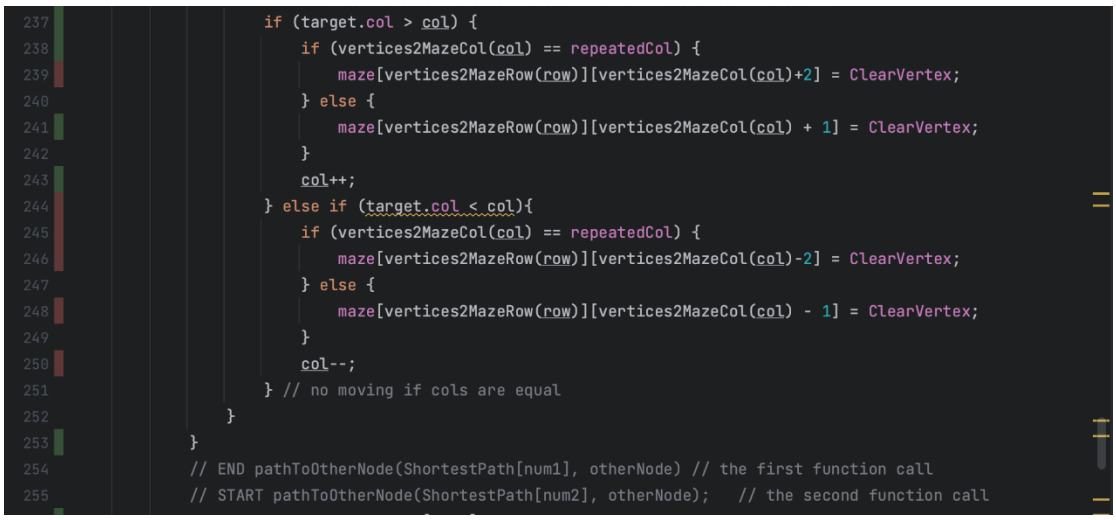
This ensures the num2 will not be the same number with num1

And the otherNode will

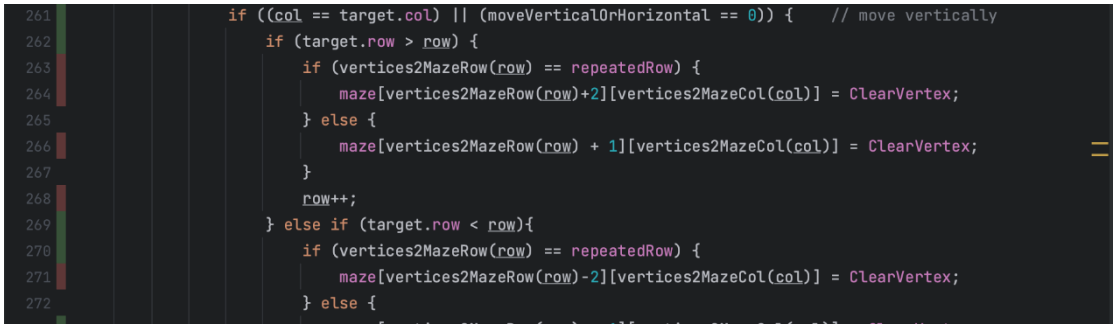
1. Not on the original only path
2. This node will be far enough that the second generated is far enough to be a good game



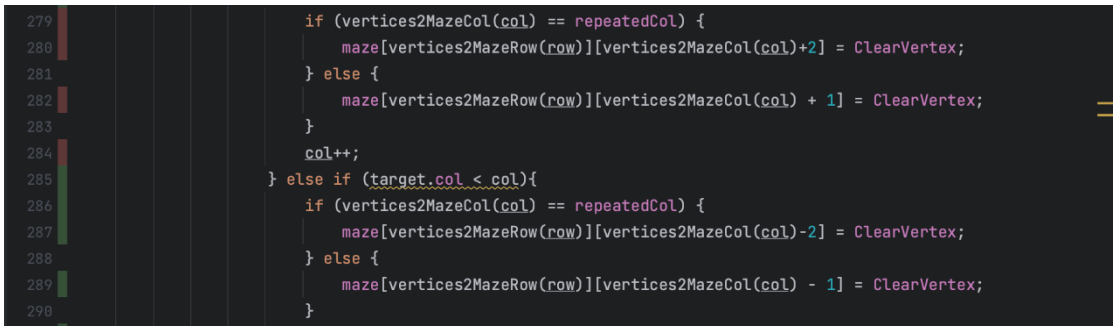
```
                if (target.row > row) {
                    if (vertices2MazeRow(row) == repeatedRow) {
                        maze[vertices2MazeRow(row)+2][vertices2MazeCol(col)] = ClearVertex;
                    } else {
                        maze[vertices2MazeRow(row) + 1][vertices2MazeCol(col)] = ClearVertex;
                    }
                    row++;
                } else if (target.row < row){
                    if (vertices2MazeRow(row) == repeatedRow) {
                        maze[vertices2MazeRow(row)-2][vertices2MazeCol(col)] = ClearVertex;
                    } else {
                        maze[vertices2MazeRow(row) - 1][vertices2MazeCol(col)] = ClearVertex;
                    }
                    row--;
                } // no moving if rows are equal
```



```
                if (target.col > col) {
                    if (vertices2MazeCol(col) == repeatedCol) {
                        maze[vertices2MazeRow(row)][vertices2MazeCol(col)+2] = ClearVertex;
                    } else {
                        maze[vertices2MazeRow(row)][vertices2MazeCol(col) + 1] = ClearVertex;
                    }
                    col++;
                } else if (target.col < col){
                    if (vertices2MazeCol(col) == repeatedCol) {
                        maze[vertices2MazeRow(row)][vertices2MazeCol(col)-2] = ClearVertex;
                    } else {
                        maze[vertices2MazeRow(row)][vertices2MazeCol(col) - 1] = ClearVertex;
                    }
                    col--;
                } // no moving if cols are equal
            }
        }
        // END pathToOtherNode(ShortestPath[num1], otherNode) // the first function call
        // START pathToOtherNode(ShortestPath[num2], otherNode);   // the second function call
```



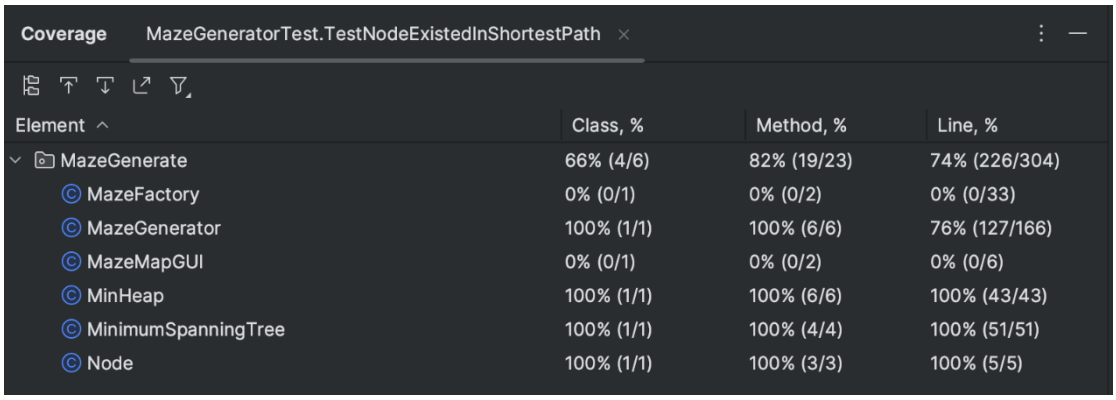
```
            if ((col == target.col) || (moveVerticalOrHorizontal == 0)) {    // move vertically
                if (target.row > row) {
                    if (vertices2MazeRow(row) == repeatedRow) {
                        maze[vertices2MazeRow(row)+2][vertices2MazeCol(col)] = ClearVertex;
                    } else {
                        maze[vertices2MazeRow(row) + 1][vertices2MazeCol(col)] = ClearVertex;
                    }
                    row++;
                } else if (target.row < row){
                    if (vertices2MazeRow(row) == repeatedRow) {
                        maze[vertices2MazeRow(row)-2][vertices2MazeCol(col)] = ClearVertex;
                    } else {
```



```
                    if (vertices2MazeCol(col) == repeatedCol) {
                        maze[vertices2MazeRow(row)][vertices2MazeCol(col)+2] = ClearVertex;
                    } else {
                        maze[vertices2MazeRow(row)][vertices2MazeCol(col) + 1] = ClearVertex;
                    }
                    col++;
                } else if (target.col < col){
                    if (vertices2MazeCol(col) == repeatedCol) {
                        maze[vertices2MazeRow(row)][vertices2MazeCol(col)-2] = ClearVertex;
                    } else {
                        maze[vertices2MazeRow(row)][vertices2MazeCol(col) - 1] = ClearVertex;
                    }
```

These don’t include as it is already pass the target’s row is already smaller than row, no more walking to the other side. Col will be the same.

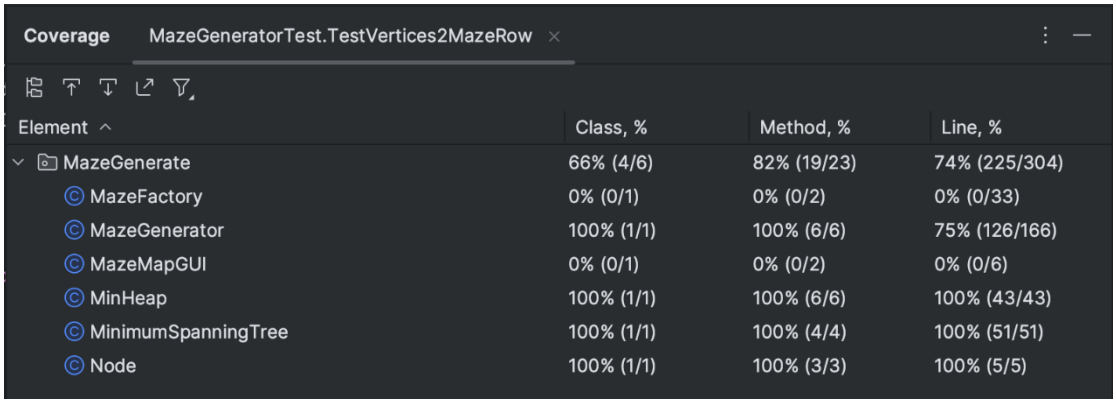
TestNodeExistedInShortestPath: All covered



Coverage | MazeGeneratorTest.TestNodeExistedInShortestPath

Element	Class, %	Method, %	Line, %
MazeGenerate	66% (4/6)	82% (19/23)	74% (226/304)
MazeFactory	0% (0/1)	0% (0/2)	0% (0/33)
MazeGenerator	100% (1/1)	100% (6/6)	76% (127/166)
MazeMapGUI	0% (0/1)	0% (0/2)	0% (0/6)
MinHeap	100% (1/1)	100% (6/6)	100% (43/43)
MinimumSpanningTree	100% (1/1)	100% (4/4)	100% (51/51)
Node	100% (1/1)	100% (3/3)	100% (5/5)

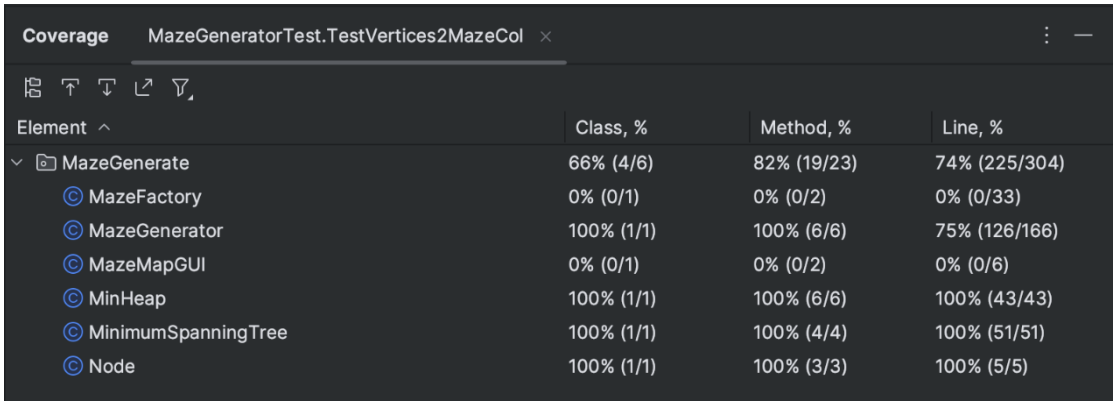
TestVertices2MazeRow: All covered



Coverage | MazeGeneratorTest.TestVertices2MazeRow

Element	Class, %	Method, %	Line, %
MazeGenerate	66% (4/6)	82% (19/23)	74% (225/304)
MazeFactory	0% (0/1)	0% (0/2)	0% (0/33)
MazeGenerator	100% (1/1)	100% (6/6)	75% (126/166)
MazeMapGUI	0% (0/1)	0% (0/2)	0% (0/6)
MinHeap	100% (1/1)	100% (6/6)	100% (43/43)
MinimumSpanningTree	100% (1/1)	100% (4/4)	100% (51/51)
Node	100% (1/1)	100% (3/3)	100% (5/5)

TestVertices2MazeCol: All covered



Coverage | MazeGeneratorTest.TestVertices2MazeCol

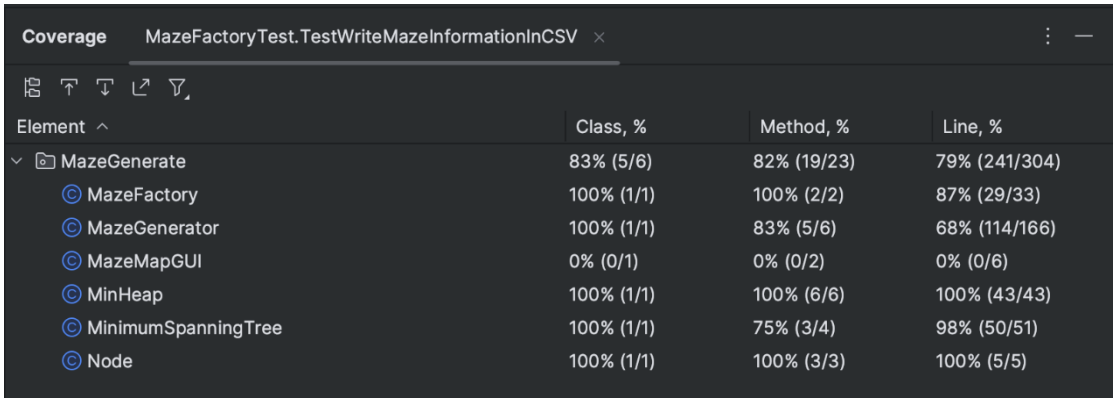
Element	Class, %	Method, %	Line, %
MazeGenerate	66% (4/6)	82% (19/23)	74% (225/304)
MazeFactory	0% (0/1)	0% (0/2)	0% (0/33)
MazeGenerator	100% (1/1)	100% (6/6)	75% (126/166)
MazeMapGUI	0% (0/1)	0% (0/2)	0% (0/6)
MinHeap	100% (1/1)	100% (6/6)	100% (43/43)
MinimumSpanningTree	100% (1/1)	100% (4/4)	100% (51/51)
Node	100% (1/1)	100% (3/3)	100% (5/5)

MazeFactoryTest class:

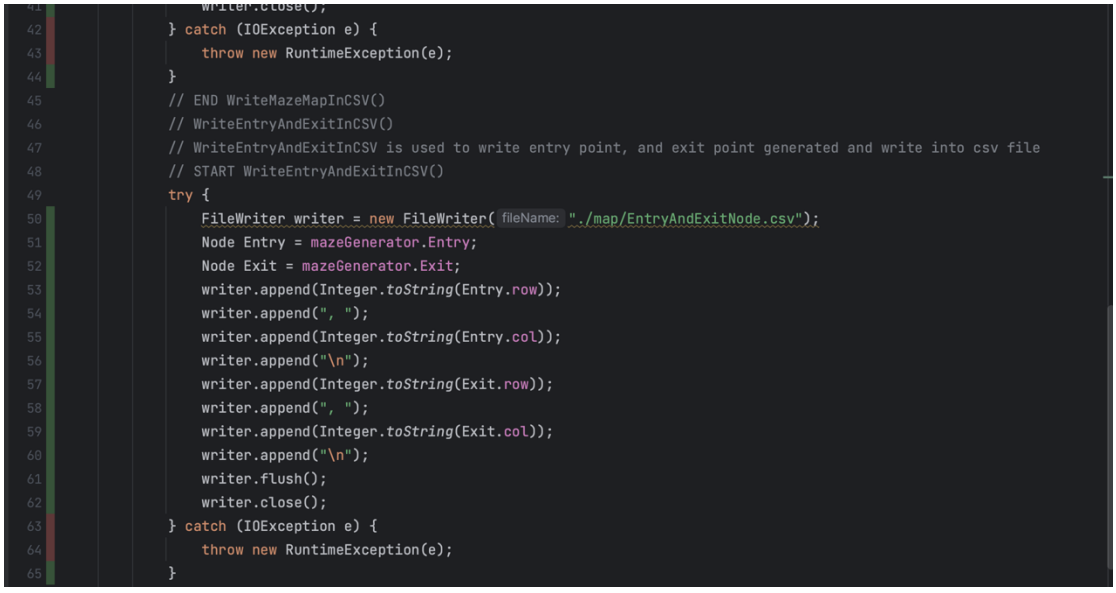
Overall:

Element	Class, %	Method, %	Line, %
MazeGenerate	83% (5/6)	82% (19/23)	80% (244/304)
MazeFactory	100% (1/1)	100% (2/2)	87% (29/33)
MazeGenerator	100% (1/1)	83% (5/6)	70% (117/166)
MazeMapGUI	0% (0/1)	0% (0/2)	0% (0/6)
MinHeap	100% (1/1)	100% (6/6)	100% (43/43)
MinimumSpanningTree	100% (1/1)	75% (3/4)	98% (50/51)
Node	100% (1/1)	100% (3/3)	100% (5/5)

TestWriteMazeInformationInCSV: 42-43, 63-64



Element	Class, %	Method, %	Line, %
MazeGenerate	83% (5/6)	82% (19/23)	79% (241/304)
MazeFactory	100% (1/1)	100% (2/2)	87% (29/33)
MazeGenerator	100% (1/1)	83% (5/6)	68% (114/166)
MazeMapGUI	0% (0/1)	0% (0/2)	0% (0/6)
MinHeap	100% (1/1)	100% (6/6)	100% (43/43)
MinimumSpanningTree	100% (1/1)	75% (3/4)	98% (50/51)
Node	100% (1/1)	100% (3/3)	100% (5/5)



```
        writer.close();
    } catch (IOException e) {
        throw new RuntimeException(e);
    }
    // END WriteMazeMapInCSV()
    // WriteEntryAndExitInCSV()
    // WriteEntryAndExitInCSV is used to write entry point, and exit point generated and write into csv file
    // START WriteEntryAndExitInCSV()
    try {
        FileWriter writer = new FileWriter( fileName: "./map/EntryAndExitNode.csv");
        Node Entry = mazeGenerator.Entry;
        Node Exit = mazeGenerator.Exit;
        writer.append(Integer.toString(Entry.row));
        writer.append(", ");
        writer.append(Integer.toString(Entry.col));
        writer.append("\n");
        writer.append(Integer.toString(Exit.row));
        writer.append(", ");
        writer.append(Integer.toString(Exit.col));
        writer.append("\n");
        writer.flush();
        writer.close();
    } catch (IOException e) {
        throw new RuntimeException(e);
    }
```

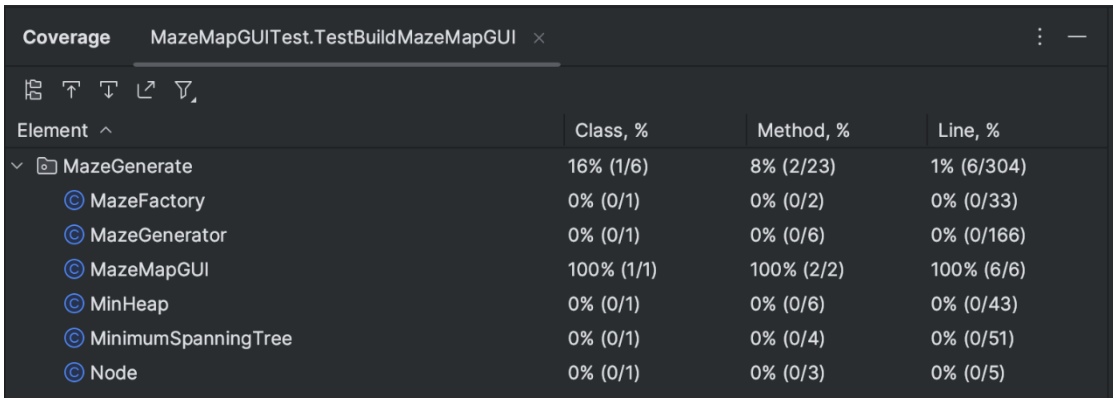
Catch region is not tested as no wrong input is tested.

MazeMapGUITest class:

Overall:

Element	Class, %	Method, %	Line, %
MazeGenerate	16% (1/6)	8% (2/23)	1% (6/304)
MazeFactory	0% (0/1)	0% (0/2)	0% (0/33)
MazeGenerator	0% (0/1)	0% (0/6)	0% (0/166)
MazeMapGUI	100% (1/1)	100% (2/2)	100% (6/6)
MinHeap	0% (0/1)	0% (0/6)	0% (0/43)
MinimumSpanningTree	0% (0/1)	0% (0/4)	0% (0/51)
Node	0% (0/1)	0% (0/3)	0% (0/5)

TestBuildMazeMapGUI: All covered



Element	Class, %	Method, %	Line, %
MazeGenerate	16% (1/6)	8% (2/23)	1% (6/304)
MazeFactory	0% (0/1)	0% (0/2)	0% (0/33)
MazeGenerator	0% (0/1)	0% (0/6)	0% (0/166)
MazeMapGUI	100% (1/1)	100% (2/2)	100% (6/6)
MinHeap	0% (0/1)	0% (0/6)	0% (0/43)
MinimumSpanningTree	0% (0/1)	0% (0/4)	0% (0/51)
Node	0% (0/1)	0% (0/3)	0% (0/5)

Function B – Shortest path

In ShortestPath class,

			All implemented function						
Test cases	Target function	initializeParentArray()	toArray()	isValidPosition()	isUnvisitedCell()	bfs()	constructPath()	getPath()	generatePathInCSV()
initializeParentArrayTest()	initializeParentArray()	All covered	na	na	na	na	na	na	na
toArrayTest()	toArray()	na	All covered	na	na	na	na	na	na
isValidPositionTest()	isValidPosition()	na	na	All covered	na	na	na	na	na
isUnvisitedCellTest()	isUnvisitedCell()	na	na	na	All covered	na	na	na	na
bfsTest()	bfs()	na	na	na	na	All covered	na	na	na
constructPathTest()	constructPath()	na	na	na	na	na	All covered	na	na
getPathTest()	getPath()	na	na	na	na	na	na	All covered	na
generatePathInCSVTest()	generatePathInCSV()	na	na	na	na	na	na	na	171

In ShortestPathGUI class,

	All implemented function	
Test cases	Target function	constructShortestPathGUI()
constructShortestPathGUITest()	constructShortestPathGUI()	All covered

ShortestPath class:

Overall:

Element	Class, %	Method, %	Line, %
pathfunction	50% (1/2)	90% (9/10)	91% (65/71)
ShortestPath	100% (1/1)	100% (9/9)	98% (65/66)
ShortestPathGUI	0% (0/1)	0% (0/1)	0% (0/5)

initializeParentArrayTest(): All covered

Coverage ShortestPathTest.initializeParentArrayTest

Element	Class, %	Method, %	Line, %
pathfunction	50% (1/2)	20% (2/10)	12% (9/71)
ShortestPath	100% (1/1)	22% (2/9)	13% (9/66)
ShortestPathGUI	0% (0/1)	0% (0/1)	0% (0/5)

toArrayTest(): All covered

Coverage ShortestPathTest.toArrayTest

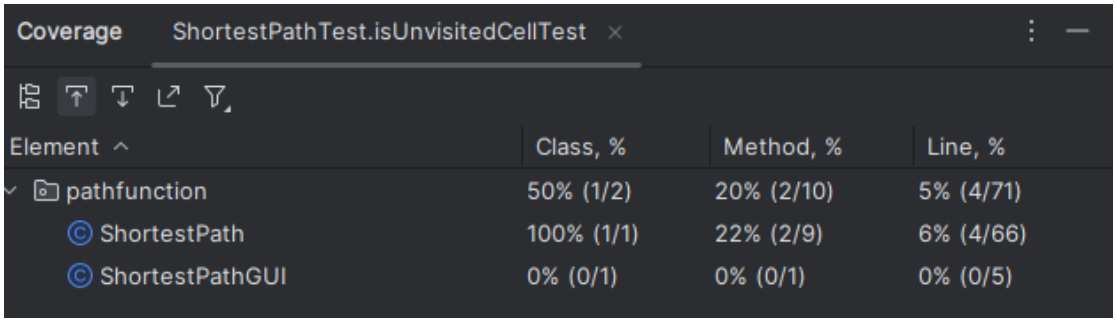
Element	Class, %	Method, %	Line, %
pathfunction	50% (1/2)	20% (2/10)	8% (6/71)
ShortestPath	100% (1/1)	22% (2/9)	9% (6/66)
ShortestPathGUI	0% (0/1)	0% (0/1)	0% (0/5)

isValidPositionTest(): All covered

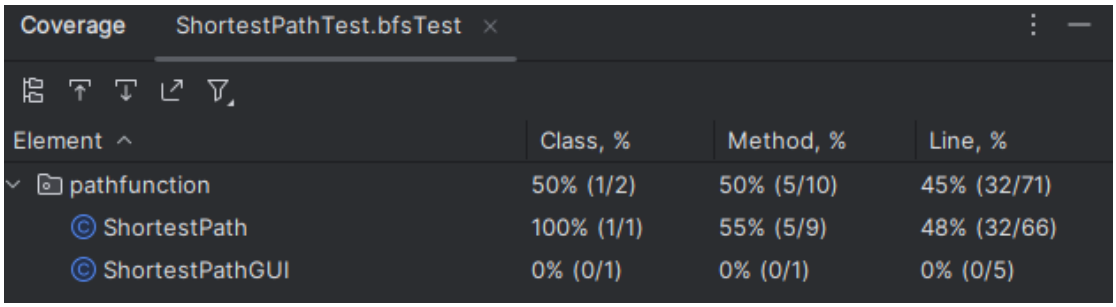
Coverage ShortestPathTest.isValidPositionTest

Element	Class, %	Method, %	Line, %
pathfunction	50% (1/2)	20% (2/10)	4% (3/71)
ShortestPath	100% (1/1)	22% (2/9)	4% (3/66)
ShortestPathGUI	0% (0/1)	0% (0/1)	0% (0/5)

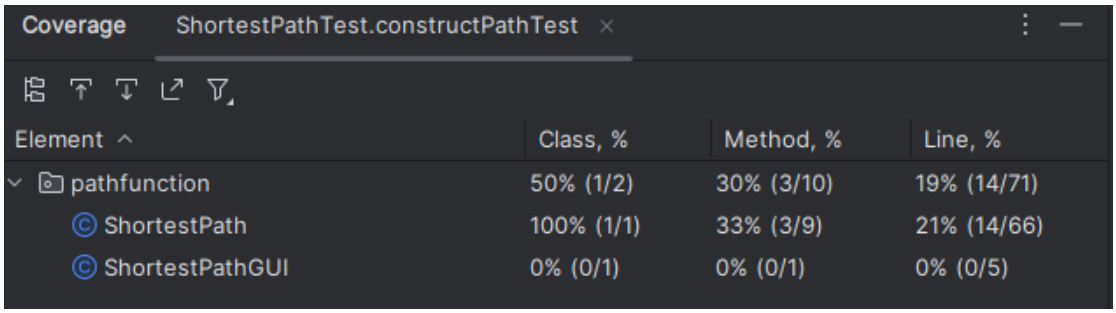
isUnvisitedCellTest(): All covered



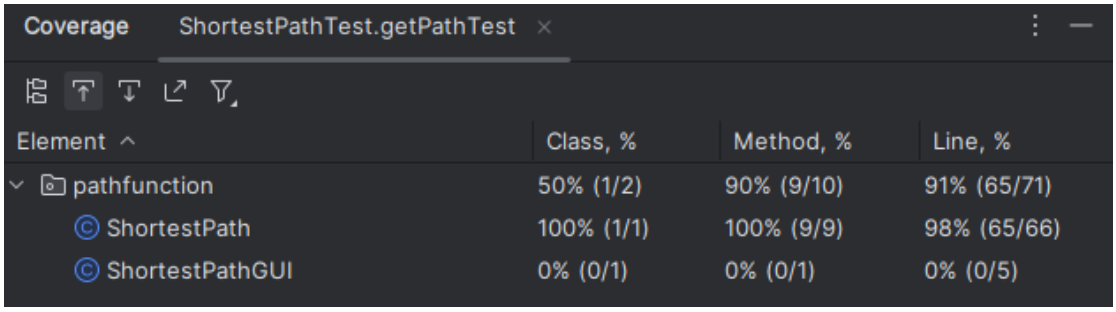
bfsTest(): All covered



constructPathTest(): All covered



getPathTest(): All covered



generatePathInCSVTest(): 171

Coverage | ShortestPathTest.generatePathInCSVTest

Element	Class, %	Method, %	Line, %
pathfunction	50% (1/2)	20% (2/10)	26% (19/71)
ShortestPath	100% (1/1)	22% (2/9)	28% (19/66)
ShortestPathGUI	0% (0/1)	0% (0/1)	0% (0/5)

```
150     public static void generatePathInCSV(int[][] path, String filePath) {
151         File file = new File(filePath);
152
153         try (FileWriter writer = new FileWriter(file)) {
154             for (int i = 0; i < path.length; i++) {
155                 writer.append("SP");
156                 writer.append(", ");
157                 writer.append(String.valueOf( i: 1));
158                 writer.append(", ");
159                 writer.append(String.valueOf( i: i + 1));
160                 writer.append(", ");
161                 writer.append(String.valueOf(path[i][0]));
162                 writer.append(", ");
163                 writer.append(String.valueOf(path[i][1]));
164                 writer.append(";");
165
166                 if (i < path.length - 1) {
167                     writer.append(System.lineSeparator());
168                 }
169             }
170
171         } catch (IOException e) {
172             throw new RuntimeException(e);
173         }
    }
```

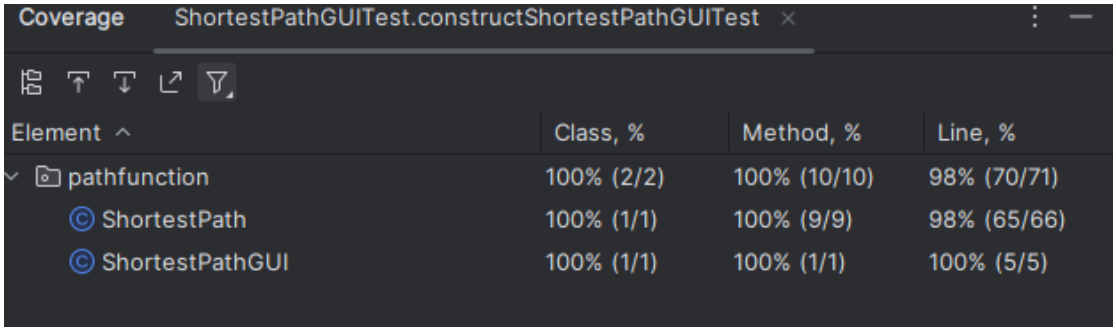
In ShortestPathGUI class:

Overall:

Coverage | ShortestPathGUITest

Element	Class, %	Method, %	Line, %
pathfunction	100% (2/2)	100% (10/10)	98% (70/71)
ShortestPath	100% (1/1)	100% (9/9)	98% (65/66)
ShortestPathGUI	100% (1/1)	100% (1/1)	100% (5/5)

constructShortestPathGUITest(): All covered



Function C – Gameplay

In JerryController Class,

		All implemented function					
Test Case	Target function	**JerryController.run()**	**JerryController.**CheckCollision()	**JerryController.StopTheGame()**	**JerryController.moveTest()**	**JerryController.DeleteOldPos()**	**JerryController.pauser()**
runTest()	**JerryController.run()**	All covered	na	na	na	na	na
CheckCollisionTest()	**JerryController.**CheckCollision()	na	All covered	na	na	na	na
StopTheGameTest()	**JerryController.StopTheGame()**	na	na	All covered	na	na	na
moveTest()	**JerryController.moveTest()**	na	na	na	All covered	na	na
DeleteOldPosTest()	**JerryController.DeleteOldPos()**	na	na	na	na	94	na

In TomController Class,

		All implemented function					
Test Case	Target function	**TomController.run()**	**TomController.**CheckCollision()	**TomController.StopTheGame()**	**TomController.moveTest()**	**TomController.DeleteOldPos()**	**TomController.pauser()**
runTest()	**TomController.run()**	All covered	na	na	na	na	na
CheckCollisionTest()	**TomController.**CheckCollision()	na	All covered	na	na	na	na
StopTheGameTes	**TomController.StopTheGam**	na	na	All covered	na	na	na

t()	**e()**						
moveTest()	**TomController.moveTest()**	na	na	na	All covered	na	na
DeleteOldPosTest()	**TomController.DeleteOldPos()**	na	na	na	na	76,78,80-81	na

In KeyboardListener Class,

		All implemented function
Test Case	Target function	KeyboardListener.KeyPressed()
KeyPressedTest()	KeyboardListener.KeyPressed()	All covered

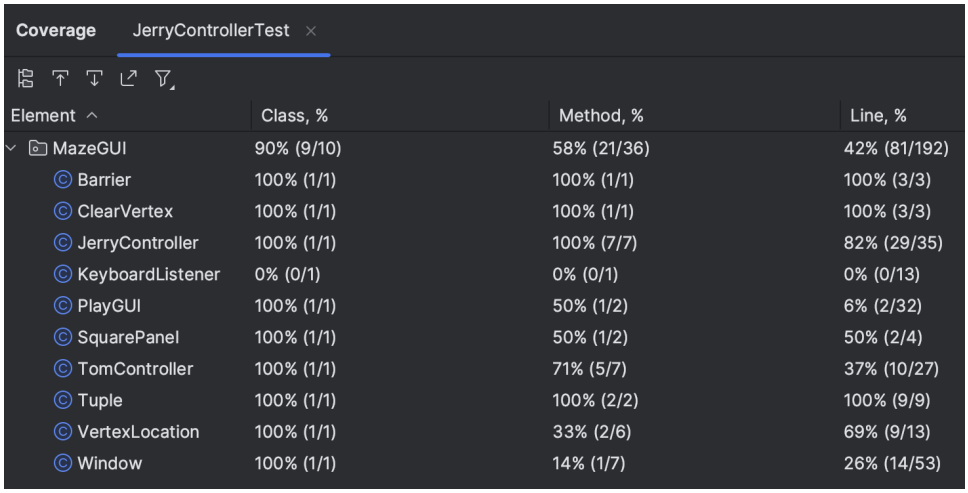
In Tuple Class,

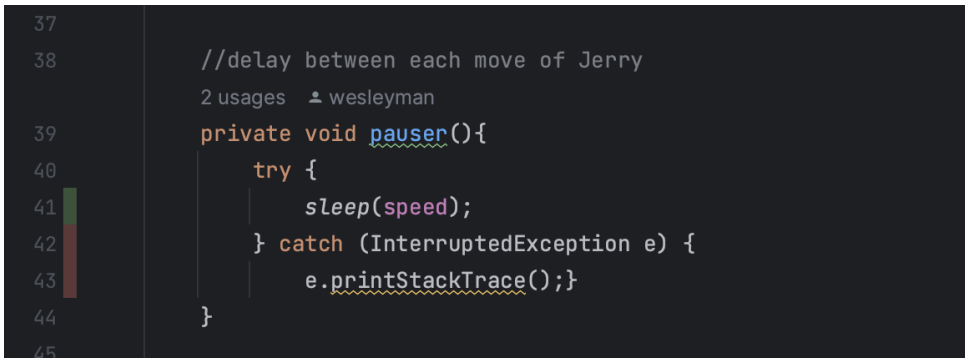
		All implemented function
Test Case	Target function	Tuple.ChangeData()
ChangeDataTest()	Tuple.ChangeData()	All covered

Coverage report & any uncovered lines:

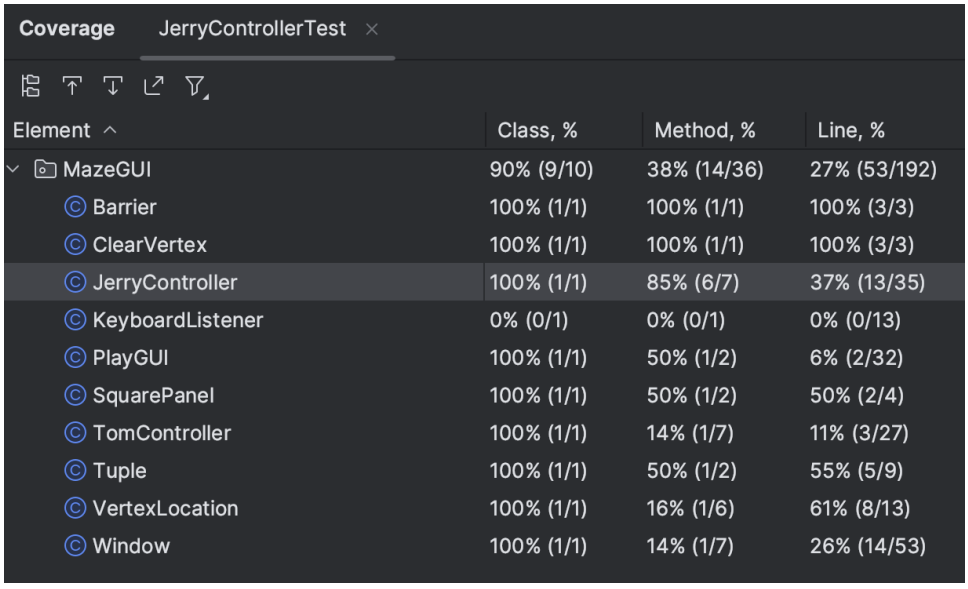
JerryController Class

Overall:

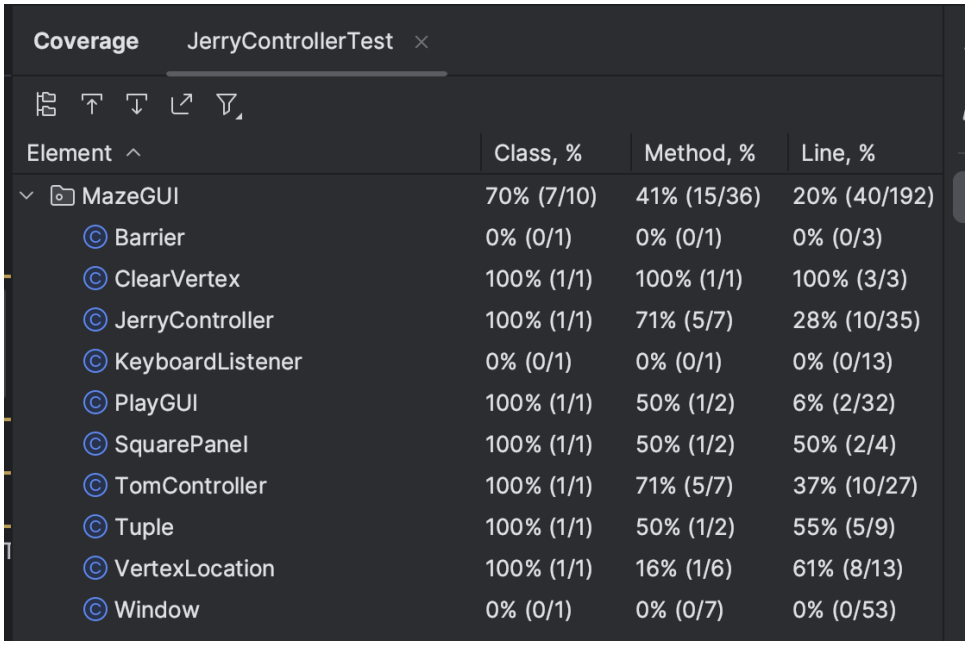




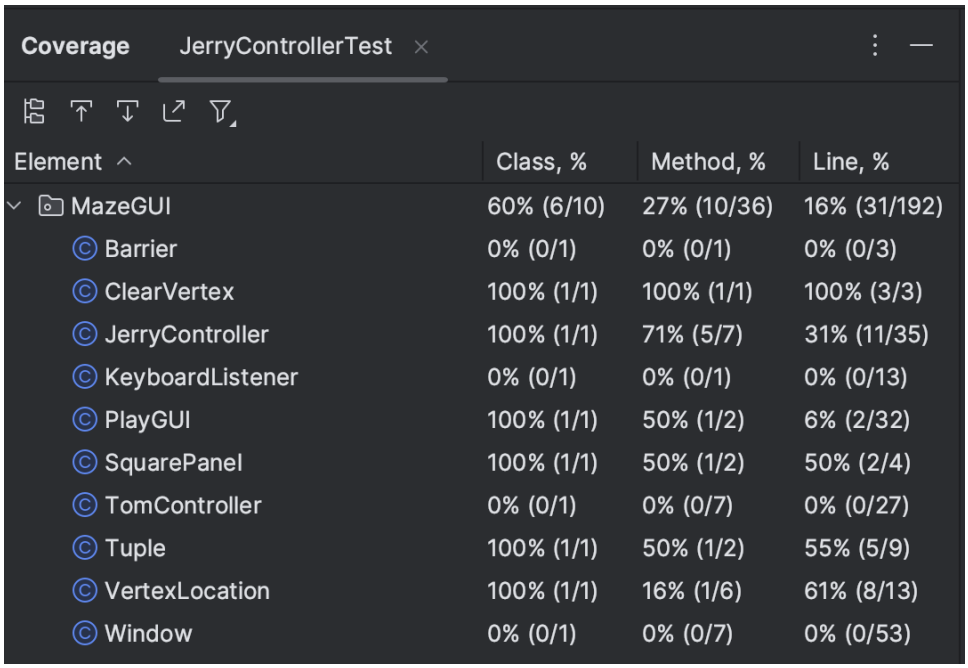
runTest(): All covered



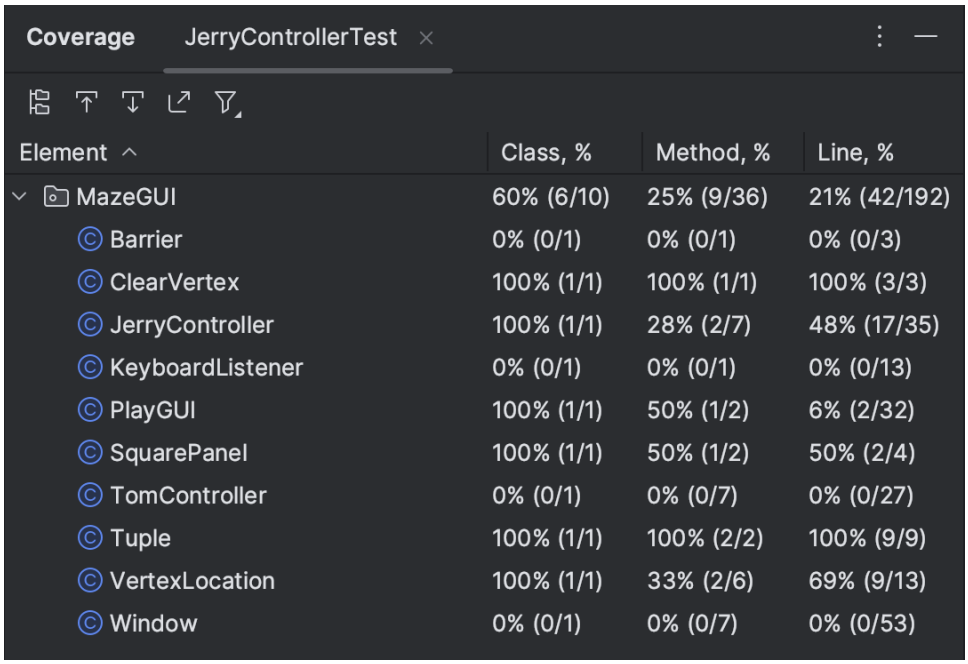
CheckCollisionTest(): All covered



StopTheGameTest(): All covered

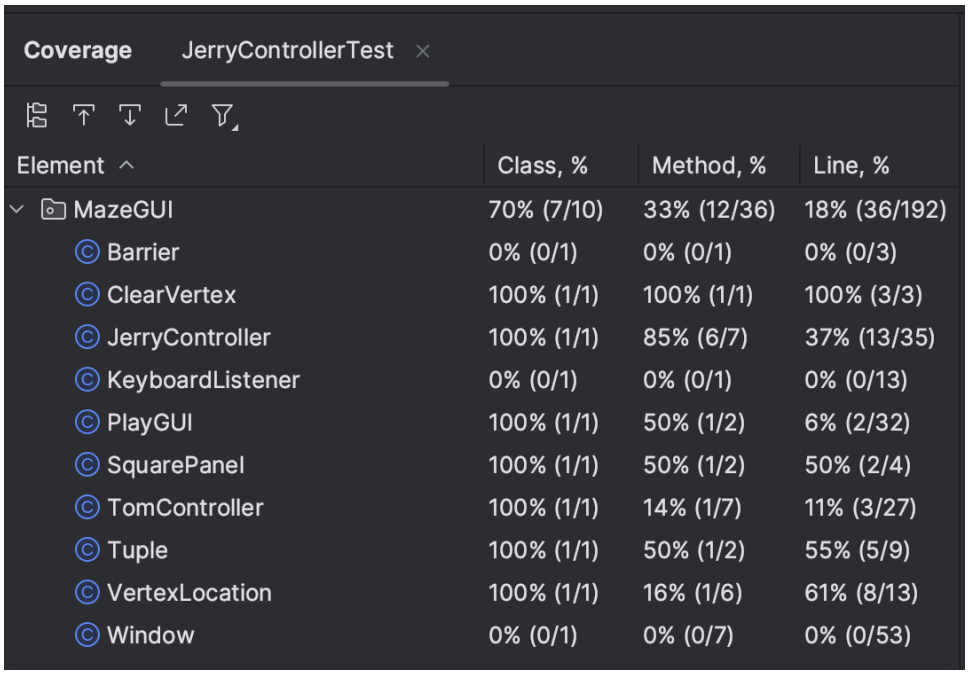


moveTest(): All covered



Element	Class, %	Method, %	Line, %
MazeGUI	60% (6/10)	25% (9/36)	21% (42/192)
Barrier	0% (0/1)	0% (0/1)	0% (0/3)
ClearVertex	100% (1/1)	100% (1/1)	100% (3/3)
JerryController	100% (1/1)	28% (2/7)	48% (17/35)
KeyboardListener	0% (0/1)	0% (0/1)	0% (0/13)
PlayGUI	100% (1/1)	50% (1/2)	6% (2/32)
SquarePanel	100% (1/1)	50% (1/2)	50% (2/4)
TomController	0% (0/1)	0% (0/7)	0% (0/27)
Tuple	100% (1/1)	100% (2/2)	100% (9/9)
VertexLocation	100% (1/1)	33% (2/6)	69% (9/13)
Window	0% (0/1)	0% (0/7)	0% (0/53)

DeleteOldPosTest(): Line 94



Element	Class, %	Method, %	Line, %
MazeGUI	70% (7/10)	33% (12/36)	18% (36/192)
Barrier	0% (0/1)	0% (0/1)	0% (0/3)
ClearVertex	100% (1/1)	100% (1/1)	100% (3/3)
JerryController	100% (1/1)	85% (6/7)	37% (13/35)
KeyboardListener	0% (0/1)	0% (0/1)	0% (0/13)
PlayGUI	100% (1/1)	50% (1/2)	6% (2/32)
SquarePanel	100% (1/1)	50% (1/2)	50% (2/4)
TomController	100% (1/1)	14% (1/7)	11% (3/27)
Tuple	100% (1/1)	50% (1/2)	55% (5/9)
VertexLocation	100% (1/1)	16% (1/6)	61% (8/13)
Window	0% (0/1)	0% (0/7)	0% (0/53)

```
92      //Refresh the squares that needs to be
        1 usage  wesleyman
93      private void DeleteOldPos(){
94          if(Jerry_Pos.Row != Jerry_Pos.Old_Row || Jerry_Pos.Col != Jerry_Pos.Old_Col) {
95              //add color to new position
96              PlayGUI.Grid.get(Jerry_Pos.Row).get(Jerry_Pos.Col).square.ChangeColor(Color.orange);
97              //delete color of old position
98              if(Jerry_Pos.Old_Row == PlayGUI.Entry[1] && Jerry_Pos.Old_Col == PlayGUI.Entry[0]){
99                  PlayGUI.Grid.get(Jerry_Pos.Old_Row).get(Jerry_Pos.Old_Col).square.ChangeColor(Color.red);
100             }
101             else{
102                 PlayGUI.Grid.get(Jerry_Pos.Old_Row).get(Jerry_Pos.Old_Col).square.ChangeColor(Color.white);
103             }
104         }
105     }
106 }
107
```

TomController Class

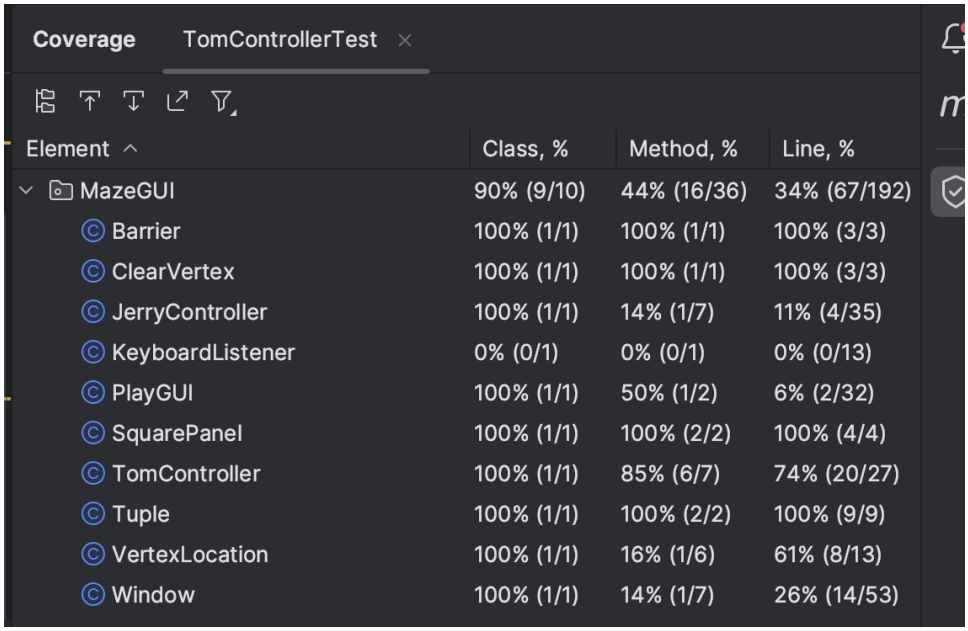
Overall:

Coverage | TomControllerTest

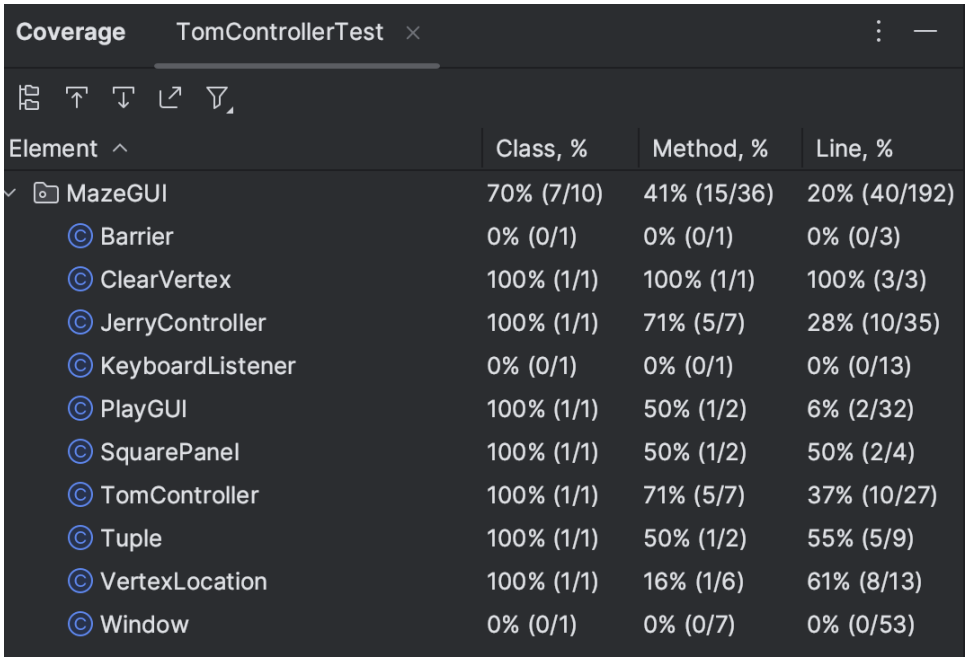
Element	Class, %	Method, %	Line, %
MazeGUI	90% (9/10)	58% (21/36)	40% (78/192)
Barrier	100% (1/1)	100% (1/1)	100% (3/3)
ClearVertex	100% (1/1)	100% (1/1)	100% (3/3)
JerryController	100% (1/1)	71% (5/7)	31% (11/35)
KeyboardListener	0% (0/1)	0% (0/1)	0% (0/13)
PlayGUI	100% (1/1)	50% (1/2)	6% (2/32)
SquarePanel	100% (1/1)	100% (2/2)	100% (4/4)
TomController	100% (1/1)	100% (7/7)	88% (24/27)
Tuple	100% (1/1)	100% (2/2)	100% (9/9)
VertexLocation	100% (1/1)	16% (1/6)	61% (8/13)
Window	100% (1/1)	14% (1/7)	26% (14/53)

```
40      //delay between each move of Tom
        2 usages  wesleyman
41      private void pauser(){
42          try {
43              sleep(speed);
44          } catch (InterruptedException e) {
45              e.printStackTrace();}
46      }
47
```

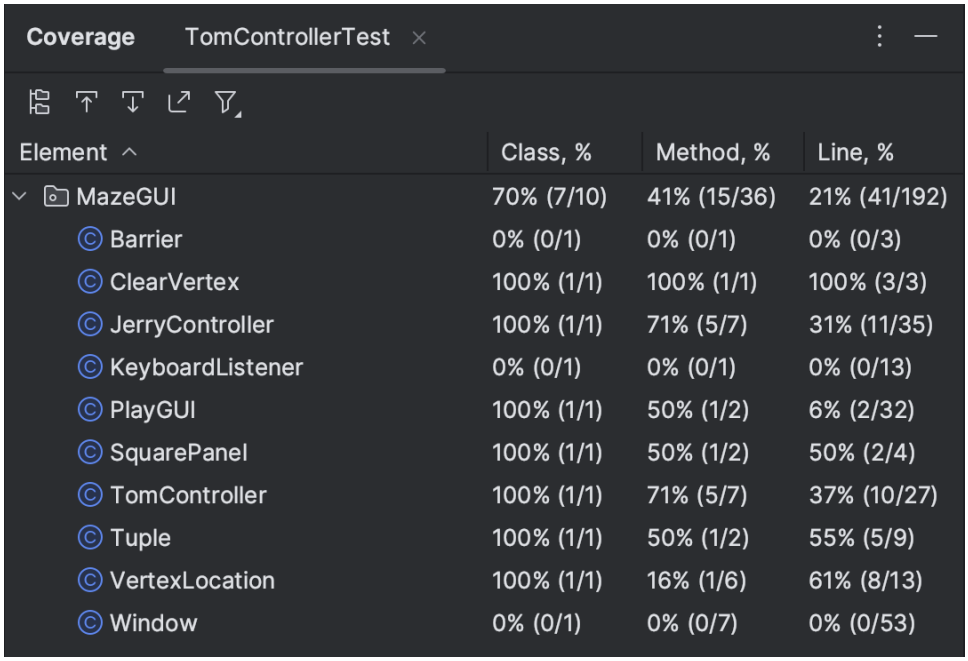
runTest(): All covered



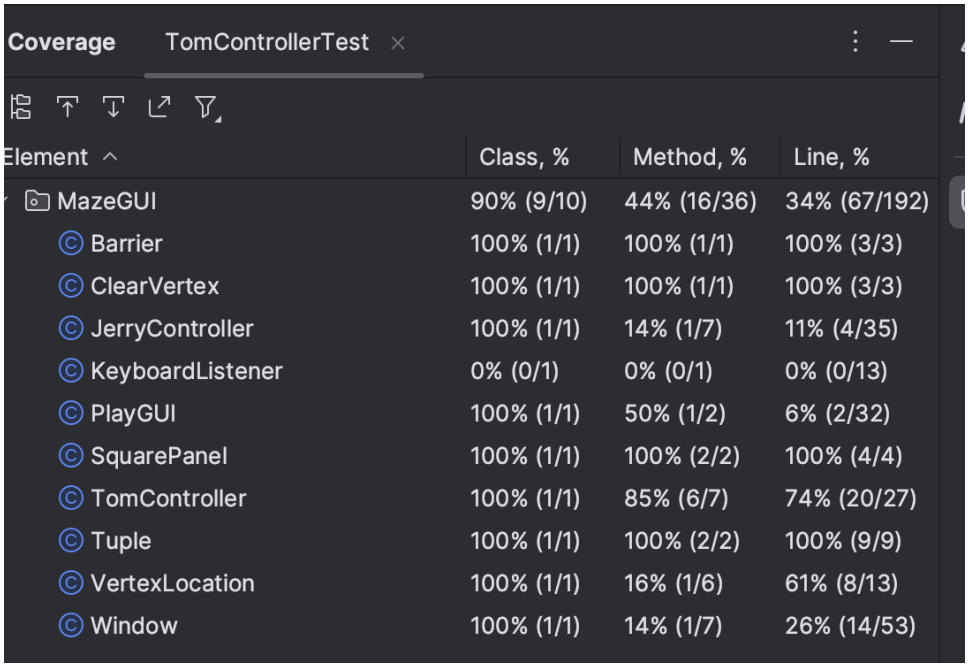
CheckCollisionTest(): All covered



StopTheGameTest(): All covered



moveTest(): All covered



DeleteOldPosTest(): Line 76,78,80-81

Coverage TomControllerTest

Element	Class, %	Method, %	Line, %
MazeGUI	90% (9/10)	44% (16/36)	34% (67/192)
Barrier	100% (1/1)	100% (1/1)	100% (3/3)
ClearVertex	100% (1/1)	100% (1/1)	100% (3/3)
JerryContr	100% (1/1)	14% (1/7)	11% (4/35)
KeyboardLi	0% (0/1)	0% (0/1)	0% (0/13)
PlayGUI	100% (1/1)	50% (1/2)	6% (2/32)
SquarePane	100% (1/1)	100% (2/2)	100% (4/4)
TomContro	100% (1/1)	85% (6/7)	74% (20/27)
Tuple	100% (1/1)	100% (2/2)	100% (9/9)
VertexLoca	100% (1/1)	16% (1/6)	61% (8/13)
Window	100% (1/1)	14% (1/7)	26% (14/53)

```
    //Refresh the squares that needs to be
    1 usage  wesleyman
    private void DeleteOldPos(){
        if(Tom_Pos.Row != Tom_Pos.Old_Row || Tom_Pos.Col != Tom_Pos.Old_Col) {
            //add color to new position
            PlayGUI.Grid.get(Tom_Pos.Row).get(Tom_Pos.Col).square.ChangeColor(Color.lightGray);
            //delete color of old position
            if(Tom_Pos.Old_Row == PlayGUI.Exit[1] && Tom_Pos.Old_Col == PlayGUI.Exit[0]){
                PlayGUI.Grid.get(Tom_Pos.Old_Row).get(Tom_Pos.Old_Col).square.ChangeColor(Color.green);
            }
            else{
                PlayGUI.Grid.get(Tom_Pos.Old_Row).get(Tom_Pos.Old_Col).square.ChangeColor(Color.white);
            }
        }
    }
}
```

KeyboardListener Class

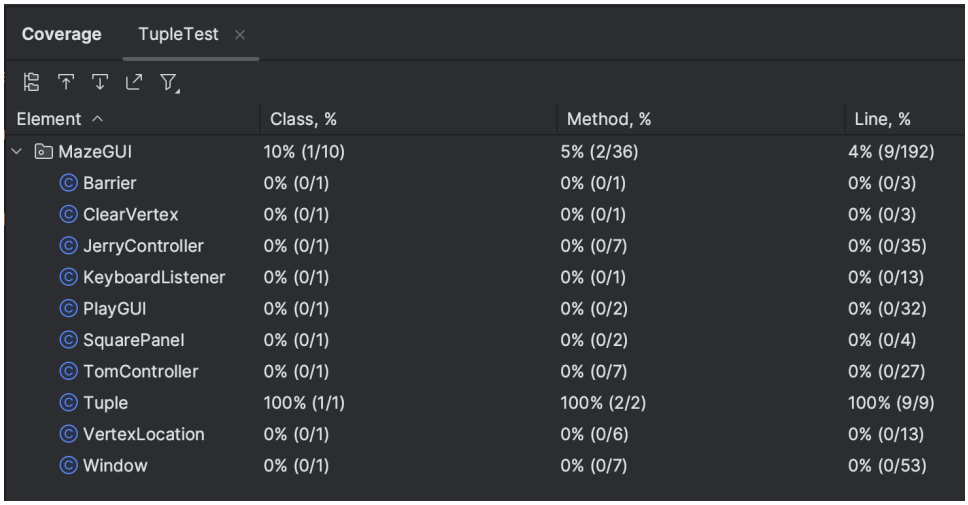
KeyPressedTest(): All covered

Coverage KeyboardListenerTest

Element	Class, %	Method, %	Line, %
MazeGUI	70% (7/10)	22% (8/36)	20% (39/193)
Barrier	0% (0/1)	0% (0/1)	0% (0/3)
ClearVertex	100% (1/1)	100% (1/1)	100% (3/3)
JerryController	100% (1/1)	14% (1/7)	11% (4/35)
KeyboardListener	100% (1/1)	100% (1/1)	100% (14/14)
PlayGUI	100% (1/1)	50% (1/2)	6% (2/32)
SquarePanel	100% (1/1)	50% (1/2)	50% (2/4)
TomController	0% (0/1)	0% (0/7)	0% (0/27)
Tuple	100% (1/1)	50% (1/2)	55% (5/9)
VertexLocation	100% (1/1)	33% (2/6)	69% (9/13)
Window	0% (0/1)	0% (0/7)	0% (0/53)

Tuple Class

ChangeDataTest(): **All covered**



Element	Class, %	Method, %	Line, %
MazeGUI	10% (1/10)	5% (2/36)	4% (9/192)
Barrier	0% (0/1)	0% (0/1)	0% (0/3)
ClearVertex	0% (0/1)	0% (0/1)	0% (0/3)
JerryController	0% (0/1)	0% (0/7)	0% (0/35)
KeyboardListener	0% (0/1)	0% (0/1)	0% (0/13)
PlayGUI	0% (0/1)	0% (0/2)	0% (0/32)
SquarePanel	0% (0/1)	0% (0/2)	0% (0/4)
TomController	0% (0/1)	0% (0/7)	0% (0/27)
Tuple	100% (1/1)	100% (2/2)	100% (9/9)
VertexLocation	0% (0/1)	0% (0/6)	0% (0/13)
Window	0% (0/1)	0% (0/7)	0% (0/53)

GUI testing (window, SquarePanel class)

In SquarePanelTest class:

		All implemented functions
Test case	Target function	ChangeColor()
changeColor()	ChangeColor()	All covered

In WindowTest class:

		All implemented function	
Test case	Target function	readCSV()	BuildGridWithMazeMap()
TestReadCSV()	readCSV()	67	na
TestBuildGridWithMazeMap()	BuildGridWithMazeMap()	na	All covered

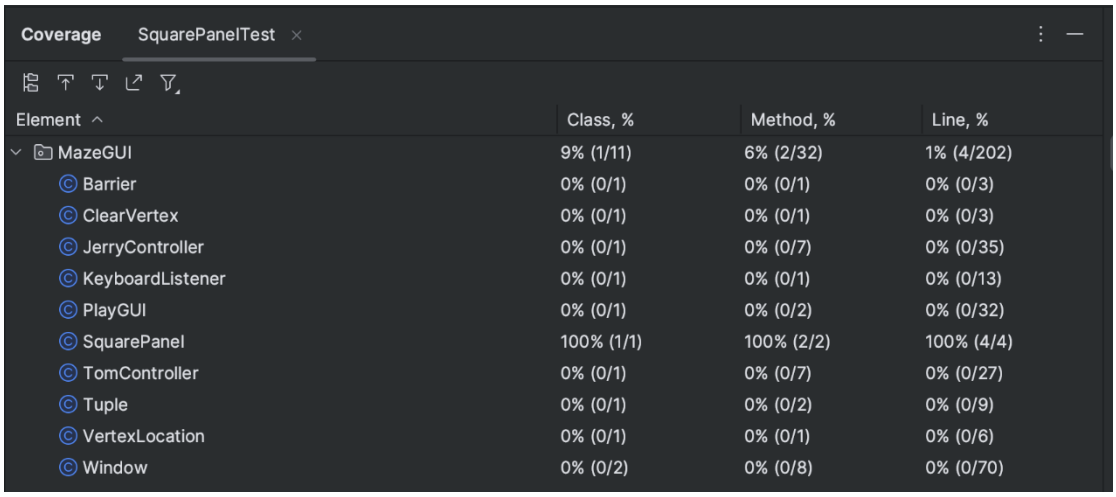
		All implemented function		
Test case	Target function	SetUpMazePanelUsingGrid()	ShowMaze()	showShortestPath()
TestBuildGridWithMazeMap()	SetUpMazePanelUsingGrid()	All covered	na	na
TestShowMaze()	ShowMaze()	na	125-131 (will be cover in button unit test)	na
TestShortestPath()	showShortestPath()	na	na	All covered

		All implmeneted function
Test case	Target function	Button inside ShowMaze()
TestFunctionAButton()	Button inside ShowMaze()	All covered

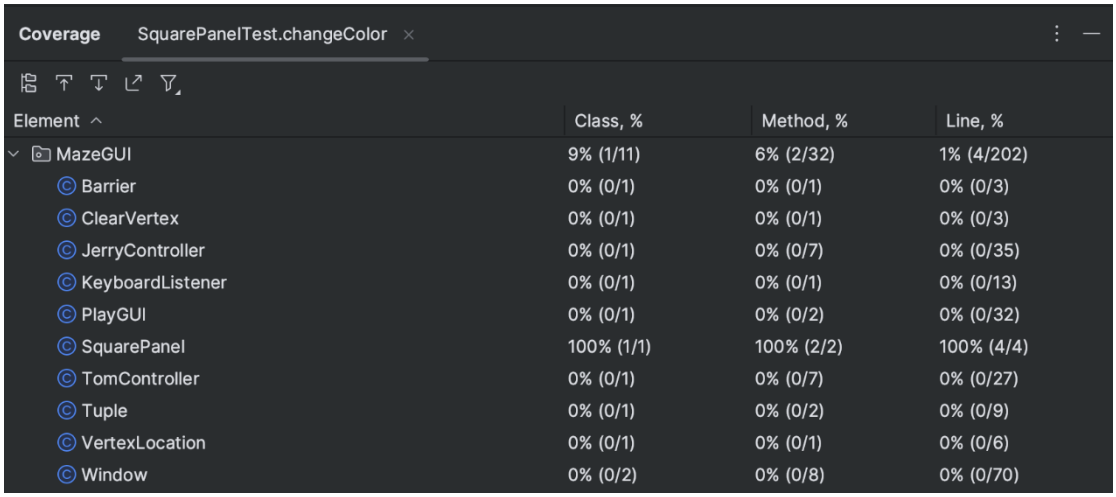
Coverage report & any uncovered lines:

SquarePanelTest class:

Overall:



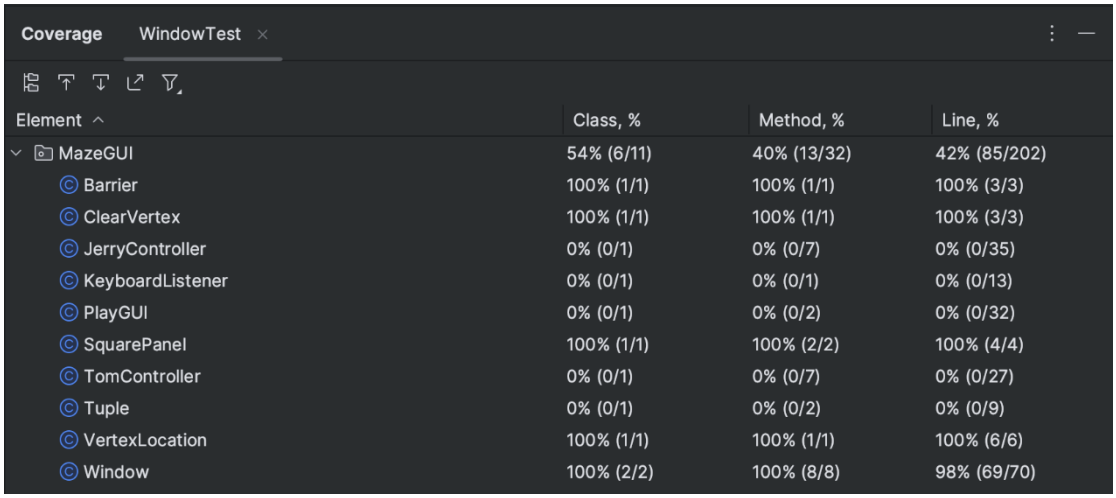
Element	Class, %	Method, %	Line, %
MazeGUI	9% (1/11)	6% (2/32)	1% (4/202)
Barrier	0% (0/1)	0% (0/1)	0% (0/3)
ClearVertex	0% (0/1)	0% (0/1)	0% (0/3)
JerryController	0% (0/1)	0% (0/7)	0% (0/35)
KeyboardListener	0% (0/1)	0% (0/1)	0% (0/13)
PlayGUI	0% (0/1)	0% (0/2)	0% (0/32)
SquarePanel	100% (1/1)	100% (2/2)	100% (4/4)
TomController	0% (0/1)	0% (0/7)	0% (0/27)
Tuple	0% (0/1)	0% (0/2)	0% (0/9)
VertexLocation	0% (0/1)	0% (0/1)	0% (0/6)
Window	0% (0/2)	0% (0/8)	0% (0/70)



Element	Class, %	Method, %	Line, %
MazeGUI	9% (1/11)	6% (2/32)	1% (4/202)
Barrier	0% (0/1)	0% (0/1)	0% (0/3)
ClearVertex	0% (0/1)	0% (0/1)	0% (0/3)
JerryController	0% (0/1)	0% (0/7)	0% (0/35)
KeyboardListener	0% (0/1)	0% (0/1)	0% (0/13)
PlayGUI	0% (0/1)	0% (0/2)	0% (0/32)
SquarePanel	100% (1/1)	100% (2/2)	100% (4/4)
TomController	0% (0/1)	0% (0/7)	0% (0/27)
Tuple	0% (0/1)	0% (0/2)	0% (0/9)
VertexLocation	0% (0/1)	0% (0/1)	0% (0/6)
Window	0% (0/2)	0% (0/8)	0% (0/70)

WindowTest class:

Overall:



Element	Class, %	Method, %	Line, %
MazeGUI	54% (6/11)	40% (13/32)	42% (85/202)
Barrier	100% (1/1)	100% (1/1)	100% (3/3)
ClearVertex	100% (1/1)	100% (1/1)	100% (3/3)
JerryController	0% (0/1)	0% (0/7)	0% (0/35)
KeyboardListener	0% (0/1)	0% (0/1)	0% (0/13)
PlayGUI	0% (0/1)	0% (0/2)	0% (0/32)
SquarePanel	100% (1/1)	100% (2/2)	100% (4/4)
TomController	0% (0/1)	0% (0/7)	0% (0/27)
Tuple	0% (0/1)	0% (0/2)	0% (0/9)
VertexLocation	100% (1/1)	100% (1/1)	100% (6/6)
Window	100% (2/2)	100% (8/8)	98% (69/70)

TestReadCSV: 67

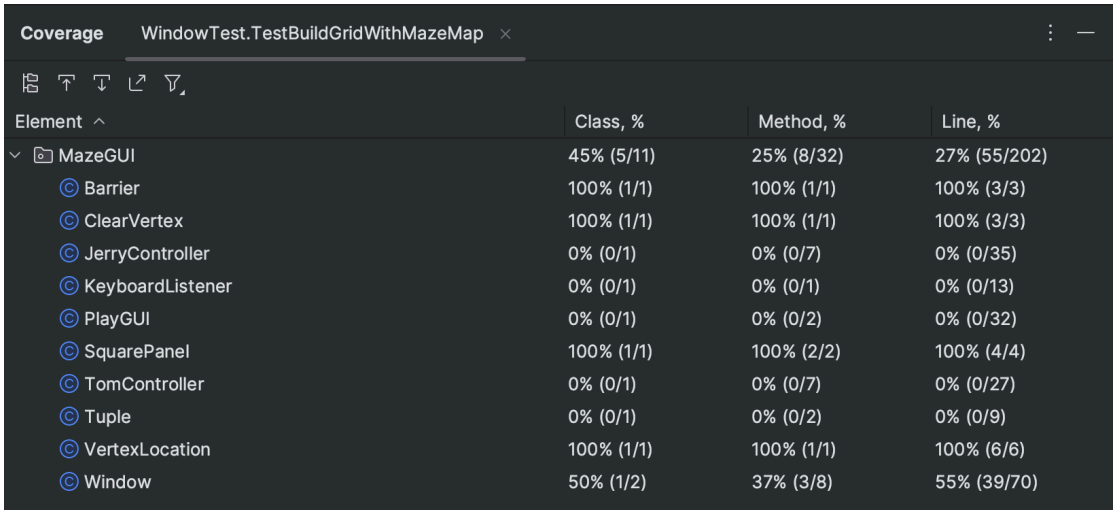
Coverage | WindowTest.TestReadCSV

Element	Class, %	Method, %	Line, %
MazeGUI	9% (1/11)	6% (2/32)	11% (23/202)
Barrier	0% (0/1)	0% (0/1)	0% (0/3)
ClearVertex	0% (0/1)	0% (0/1)	0% (0/3)
JerryController	0% (0/1)	0% (0/7)	0% (0/35)
KeyboardListener	0% (0/1)	0% (0/1)	0% (0/13)
PlayGUI	0% (0/1)	0% (0/2)	0% (0/32)
SquarePanel	0% (0/1)	0% (0/2)	0% (0/4)
TomController	0% (0/1)	0% (0/7)	0% (0/27)
Tuple	0% (0/1)	0% (0/2)	0% (0/9)
VertexLocation	0% (0/1)	0% (0/1)	0% (0/6)
Window	50% (1/2)	25% (2/8)	32% (23/70)

```
48      public static int[][] readCSV(String filePath) {
49          int[][] map = new int[0][];
50
51          try (BufferedReader br = new BufferedReader(new FileReader(filePath))) {
52              String line;
53              int rowCount = 0;
54
55              while ((line = br.readLine()) != null) {
56                  String[] values = line.split( regex: ",");
57                  int[] intValues = new int[values.length];
58
59                  for (int i = 0; i < values.length; i++) {
60                      intValues[i] = Integer.parseInt(values[i].trim());
61                  }
62
63                  map = Arrays.copyOf(map,  newLength: map.length + 1);
64                  map[rowCount++] = intValues;
65              }
66          } catch (IOException e) {
67              throw new RuntimeException(e);
68          }
69
70          return map;
71      }
```

Catch region not included as there is no invalid IO input.

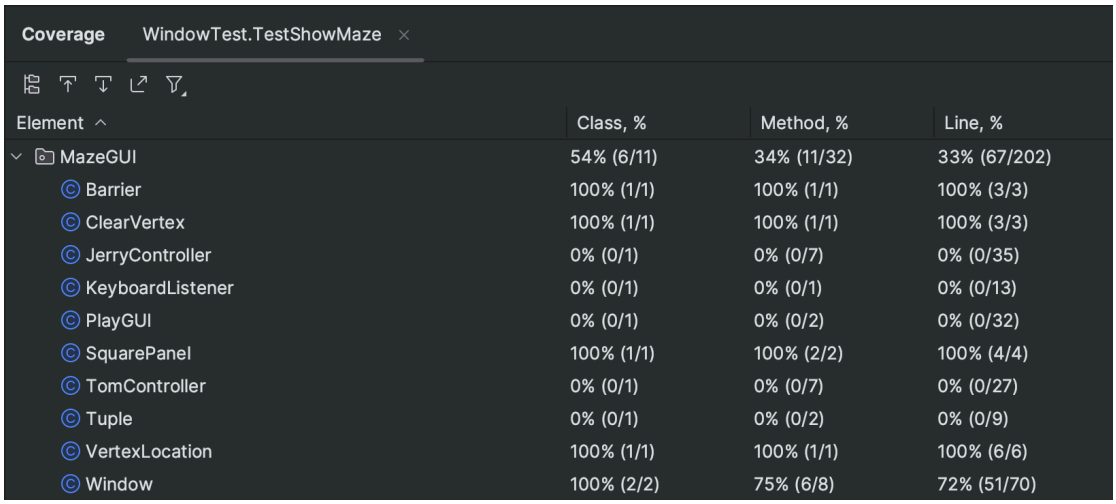
TestBuildGridWithMazeMap: All covered



Coverage | WindowTest.TestBuildGridWithMazeMap

Element	Class, %	Method, %	Line, %
MazeGUI	45% (5/11)	25% (8/32)	27% (55/202)
Barrier	100% (1/1)	100% (1/1)	100% (3/3)
ClearVertex	100% (1/1)	100% (1/1)	100% (3/3)
JerryController	0% (0/1)	0% (0/7)	0% (0/35)
KeyboardListener	0% (0/1)	0% (0/1)	0% (0/13)
PlayGUI	0% (0/1)	0% (0/2)	0% (0/32)
SquarePanel	100% (1/1)	100% (2/2)	100% (4/4)
TomController	0% (0/1)	0% (0/7)	0% (0/27)
Tuple	0% (0/1)	0% (0/2)	0% (0/9)
VertexLocation	100% (1/1)	100% (1/1)	100% (6/6)
Window	50% (1/2)	37% (3/8)	55% (39/70)

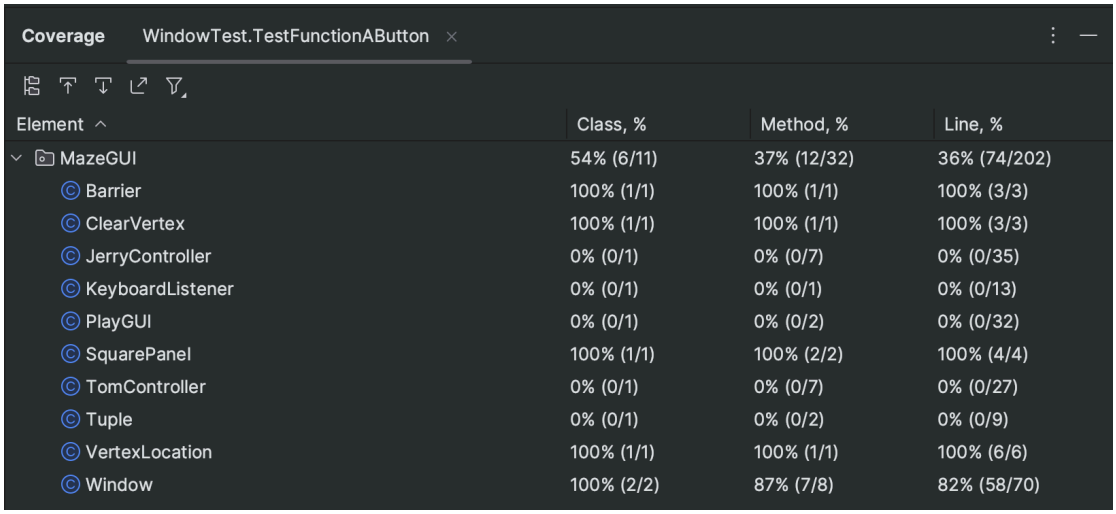
TestShowMaze: All covered



Coverage | WindowTest.TestShowMaze

Element	Class, %	Method, %	Line, %
MazeGUI	54% (6/11)	34% (11/32)	33% (67/202)
Barrier	100% (1/1)	100% (1/1)	100% (3/3)
ClearVertex	100% (1/1)	100% (1/1)	100% (3/3)
JerryController	0% (0/1)	0% (0/7)	0% (0/35)
KeyboardListener	0% (0/1)	0% (0/1)	0% (0/13)
PlayGUI	0% (0/1)	0% (0/2)	0% (0/32)
SquarePanel	100% (1/1)	100% (2/2)	100% (4/4)
TomController	0% (0/1)	0% (0/7)	0% (0/27)
Tuple	0% (0/1)	0% (0/2)	0% (0/9)
VertexLocation	100% (1/1)	100% (1/1)	100% (6/6)
Window	100% (2/2)	75% (6/8)	72% (51/70)

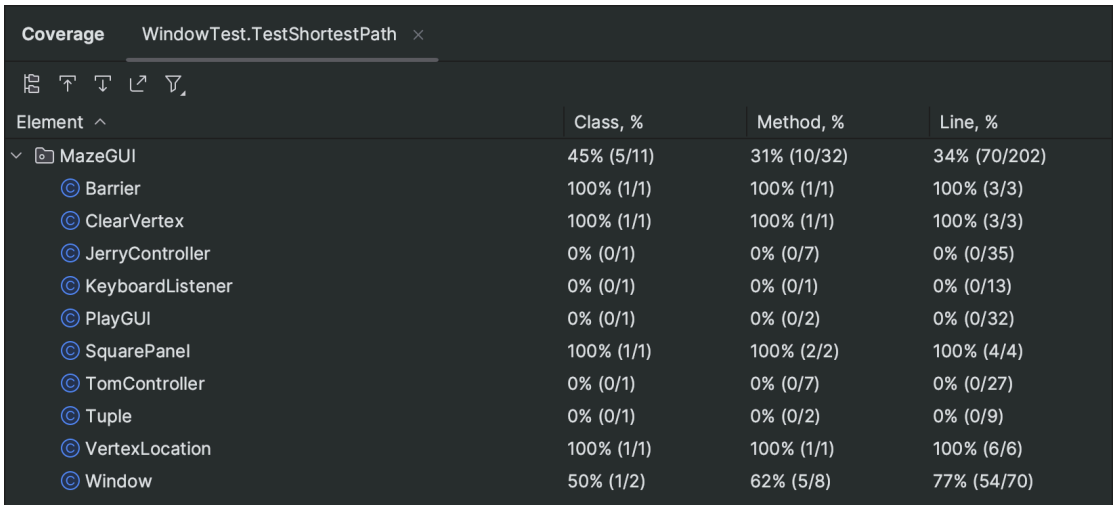
TestFunctionAButton: All covered



Coverage | WindowTest.TestFunctionAButton

Element	Class, %	Method, %	Line, %
MazeGUI	54% (6/11)	37% (12/32)	36% (74/202)
Barrier	100% (1/1)	100% (1/1)	100% (3/3)
ClearVertex	100% (1/1)	100% (1/1)	100% (3/3)
JerryController	0% (0/1)	0% (0/7)	0% (0/35)
KeyboardListener	0% (0/1)	0% (0/1)	0% (0/13)
PlayGUI	0% (0/1)	0% (0/2)	0% (0/32)
SquarePanel	100% (1/1)	100% (2/2)	100% (4/4)
TomController	0% (0/1)	0% (0/7)	0% (0/27)
Tuple	0% (0/1)	0% (0/2)	0% (0/9)
VertexLocation	100% (1/1)	100% (1/1)	100% (6/6)
Window	100% (2/2)	87% (7/8)	82% (58/70)

TestShortestPath: All covered



Element	Class, %	Method, %	Line, %
MazeGUI	45% (5/11)	31% (10/32)	34% (70/202)
Barrier	100% (1/1)	100% (1/1)	100% (3/3)
ClearVertex	100% (1/1)	100% (1/1)	100% (3/3)
JerryController	0% (0/1)	0% (0/7)	0% (0/35)
KeyboardListener	0% (0/1)	0% (0/1)	0% (0/13)
PlayGUI	0% (0/1)	0% (0/2)	0% (0/32)
SquarePanel	100% (1/1)	100% (2/2)	100% (4/4)
TomController	0% (0/1)	0% (0/7)	0% (0/27)
Tuple	0% (0/1)	0% (0/2)	0% (0/9)
VertexLocation	100% (1/1)	100% (1/1)	100% (6/6)
Window	50% (1/2)	62% (5/8)	77% (54/70)