COMP3111 Software Engineering
Fall 2023 Group 30 TnJ1
Documentation

Name [Student ID]:

Wong Hiu Tung [20859316]
Ma Sum Yi [20861280]
Meng Hin Lok [20860339]

Link of the GitHub Repository:

https://github.com/nychopkie/Comp3111F23G30

Link of the Demo Video:

https://www.youtube.com/watch?v=zwIuZ3GZ0Yo

Content page

Task 211: Meeting Minutes---------- 2

Task 212a: Gantt Chart---------- 8

Task 212b: Burndown Chart---------- 9

Task 213: Git Commit Log---------- 11

Task 214: Documentation of the code (JavaDoc)----------12

Task 215: Screenshots of the application software----------13

Class A:MazeMap---------- 15
Class B: Shortestpath----------16
Class C:MazeGame----------17

Task 232a: Unit Testing----------18

Interface----------18
Class A:MazeMap---------- 19
Class B: Shortestpath---------- 20
Class C:MazeGame----------21

Task 232b: Coverage---------- 22

Interface----------22
Class A:MazeMap---------- 25
Class B: Shortestpath----------29
Class C:MazeGame----------31
Test-per-function Ratio---------- 34

Appendix---------- 35

JavaDoc for Launcher---------- 35
JavaDoc for Interface---------- 37
JavaDoc for MazeMap----------44
JavaDoc for Vertex---------- 50
JavaDoc for ShortestPath---------- 58
JavaDoc for MazeGame---------- 64

Task 211: Meeting Minutes

COMP3111 Team30
Meeting #1 Meeting Minutes

Date: 10/10/2023 (Tuesday)

Time: 15:05 – 17:18 (2 hr 13 min)

Location: Zoom (15:05 – 16:05), Discord (16:05 – 17:18)

Present: Christine Ma, Annie Wong, Gordon Meng

Recorder: Annie Wong

Agenda:

1. **Start meeting** **[15:05]**

2. **Reviewed Project specifications and changes** **[15:05 – 15:10]**
    a. On the part that changes: Jerry is controlled by user, and Tom is controlled by computer
    b. Definition of some of the tasks required, task 120 details

3. **Discussed logistics for project** **[15:10 – 15:15]**
    a. Created shared google drive for documents not sharable in Github e.g. UML diagram
    b. Discussed on whether to use draw.io or figma for drawing UML and user case diagrams
        i. Main concern is whether if draw.io can support collaboration between users
        ii. Tested draw.io and it can support real-time collaboration, therefore chose draw.io for future diagram drawing tasks
    c. Created Discord Channel for future meetings and easier communication
        i. Originally planned to use zoom for meeting, but Zoom would only allow 40 minutes, therefore decided to use discord for meetings

4. **Drawing UML – first draft** **[15:15 – 15:50]**
    a. Created the 12 classes required in the provided document
    b. Reviewed requirements for the association and generalization between classes
    c. Left the cardinality for revision later

5. **Discussed improvements for first draft** **[15:50 – 16:15]**
    a. Changed the generalization rule for tom location and jerry location to be a part of clear vertex

b. Discussed where moving class and crystal location could possibly be associated with

6. Raised questions – first draft [16:15 – 16:20]

a. Listed some questions to ask the TA in Wednesday lab session
   i. Clarification on the mechanism of the crystal
   ii. Clarification on the swapping role

7. Break [16:20 – 16:25]

a. Changed meeting platform from zoom to discord

8. Drawing UML – second draft [16:25 – 16:50]

a. Added generalization coverage
b. Added max min multiplicity to the association
c. Connected crystal location and moving object class to the other classes
d. Changed generalization of vertex location and clear vertex class

9. Checked with grading scheme of task 120 [16:50 – 16:54]

10. Raised question – second draft [16:54 – 17:00]

a. Questioned on the definition of the moving class and possible path class, whether if the possible path class is referring to the path currently the object is moving at, or is it just referring to the possible paths that it could be in
b. Raised question on the grading details of the project task 120
c. Asked if XOR can be used in generalization cases where the coverage is overlapping
d. Questioned on the state of the entry and exit, and whether if the crystal can spawn in those 2 locations

11. Started on user case diagram [17:00 – 17:15]

a. Not covered yet in lecture, therefore plotted the provided outline to the drawing software
b. Planned date for next meeting to finish the user case diagram: 11/10/2023
   i. Aim to at least finish half of user case diagram

12. Wrapping up [17:15 – 17:18]

a. Checked if there is any other questions we want to ask the TA next lab session (11/10/2023)

13. End meeting [17:18]

COMP3111 Team30
Meeting #2 Meeting Minutes

Date: 08/11/2023 (Wednesday)

Time: 16:00 – 17:00 (1 hr)

Location: Room 2407

Present: Christine Ma, Annie Wong, Gordon Meng

Recorder: Annie Wong

Agenda:

1. **Start meeting** **[16:00]**

2. **Report on progress** **[16:00 – 16:10]**

Name	Tasks worked on	Total hours
Christine Ma	Studied task specification	4
Annie Wong	Drafted a interface design	2
Gordon Meng	Studied Java	4

3. **Discussed logistics for project** **[16:10 – 16:30]**
   - 3.1. Discussed the expected time-line for the project
   - 3.2. Estimated the time required for each component
   - 3.3. Proposed deadlines for each person's tasks

Name	Task and completeness	Date
Annie Wong	Plot base of interface	15/11
	Finish edit maze map function	18/11
	Finish unit testing	23/11
Christine Ma	Finish shortest path function	17/11
	Finish unit testing	23/11
Gordon Meng	Finish basic function of playing maze game	15/11
	Finish integrating code of all other function needed	20/11
	Finish unit testing	23/11
Project Documentation		24/11
Finalize code		25/11

4. **Discussion on documentation** **[16:30 – 16:50]**
   - 4.1. Planned to discuss this in future meetings
   - 4.2. Re-read the requirements on the documents

5. Wrapping up [16:50 – 17:00]

5.1. Proposed next meeting date: 21/11/2023

6. End meeting [17:00]

COMP3111 Team30
Meeting #3 Meeting Minutes

Date: 21/11/2023 (Tuesday)

Time: 20:50 – 22:10 (1 hour 20 mins)

Location: Discord Voice Call

Present: Christine Ma, Annie Wong, Gordon Meng

Recorder: Annie Wong

Agenda:

1. **Start meeting** **[20:50]**

2. **Report on progress** **[20:50 – 21:10]**

Name	Tasks worked on	Total hours
Christine Ma	Finished Task B: shortest path	10
Annie Wong	Finished Interface Finished Task A: edit maze map	13
Gordon Meng	Finished Task C: play game	12

3. **Discussed logistics for project documentation** **[21:10 – 21:43]**
   - 3.1. Discussed the tasks required for documentation
   - 3.2. Estimated the time required for each component
   - 3.3. Proposed deadlines for each person's tasks

Name	Task and completeness	Date
Annie Wong	Finish writing the meeting minutes	21/11
	Finish documenting Interface, MazeMap, Vertex	22/11
	Finish unit testing	24/11
Christine Ma	Finalize the burndown chart	17/11
	Finish documenting ShortestPath	23/11
	Finish unit testing	24/11
Gordon Meng	Finalize the Gantt chart	23/11
	Finish documenting MazeGame	23/11
	Finish unit testing	23/11
Project Documentation		24/11
Project Presentation Video		23/11
Finalize code		25/11

4. Discussion on documentation [21:43 – 21:58]

4.1. Reviewed requirements
4.2. Would be handled by Annie
4.3. Reminded that the JavaDoc need to be finished and documented

5. Discussion on Video [21:58 – 22:10]

5.1. Discussed time for making video together: 23/11/2023 20:30
5.2. Planned out flow of the video
5.3. Record screen then, then add voice over afterwards
5.4. Planned editor for video: Annie

6. Wrapping up [22:10 – 22:20]

6.1. Reminded next meeting date: 23/11/2023 for video
6.2. Reminded to leave 26/11 for emergency correction and buffer

7. End meeting [22:20]

Task 212a: Gantt Chart

Task 212b: Burndown Chart

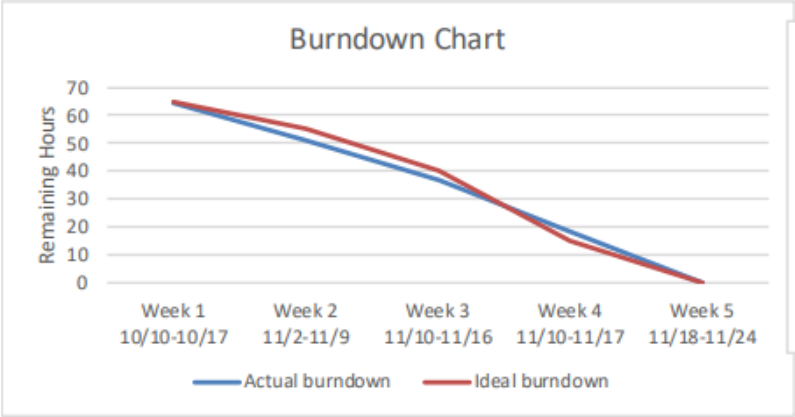
Group 30

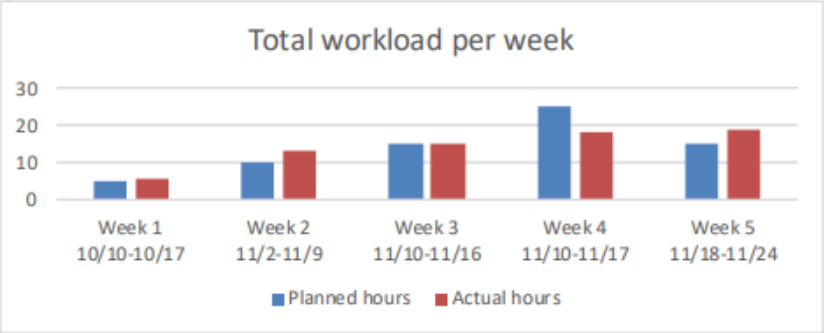
Sprint Backlog:Actual time

Sprint Tasks (Sprint Backlog)	Week 1 10/10-10/17	Week 2 11/2-11/9	Week 3 11/10-11/16	Week 4 11/10-11/17	Week 5 11/18-11/24
Job111 Sign up Group Formation	0.05				
Job112 Team Repo Setup(GitHub)	0.05				
Job 211 Meeting Minutes i.e meeting and write meeting minute	2.6	1.4			2
Job121 Class Diagram	2				
Job 122 Use Case Specification	0.5				
Job 212 Gantt Chart & Burndown Chart (i.e. effort including drawing and estimating the time required for each task)	0.2	0.4			1
Job 231 Functional Programming Quality. (i.e. Class A,B,C, check code style and format)		10	15	14	7
Job 232 Unit Testing & Coverage Report		1		4	4
Job 214 Documentation with JavaDoc		0.3			1.5
Job 222 YouTube Video Demo					3
Planned hours	**5**	**10**	**15**	**25**	**15**
Actual hours	**5.4**	**13.1**	**15**	**18**	**18.5**
Actual burndown	**64.6**	**51.5**	**36.5**	**18.5**	**0**
Ideal burndown	**65**	**55**	**40**	**15**	**0**

Group 30

Sprint backlog:Estimated time





Note that the burndown chart has some discrepancies with the Gantt chart, but the reason for this is due to the burndown chart focusing mainly on the weeks the work is most focused at. Since we focused most of the work at certain bursts, therefore, the date listed on the burndown chart is different to the ones on the Gantt chart, which is used to track work across the whole project.

Task 213: Git Commit Log

Since the commit history is too long to be included in the document, therefore, the respective logs for each branch are linked below, with explanation as to the contribution of each branch and its purpose.

Commits on the master Branch:

https://github.com/nychopkie/Comp3111F23G30/commits/master

This Branch is the final product of the whole project. It is merged with dev-build only if dev-build is stable

Commits on the dev-build branch:

https://github.com/nychopkie/Comp3111F23G30/commits/dev-build

This branch mainly is used to handle and resolve conflicts of merging function A, B and C, and acts as a backup to master to prevent any detrimental damages such as accidental deletion.

Commits on the branch for Function A:

https://github.com/nychopkie/Comp3111F23G30/commits/func-A

This branch is the bench where function A is developed. After it is stable it would be merged to the dev-build branch.

Commits on the branch for Function B:

https://github.com/nychopkie/Comp3111F23G30/commits/b

This branch is the bench where function B is developed. After it is stable it would be merged to the dev-build branch.

Commits on the branch for Function C:

https://github.com/nychopkie/Comp3111F23G30/commits/func-C

This branch is the bench where function C is developed. After it is stable it would be merged to the dev-build branch.

Task 214: Documentation of the code (JavaDoc)

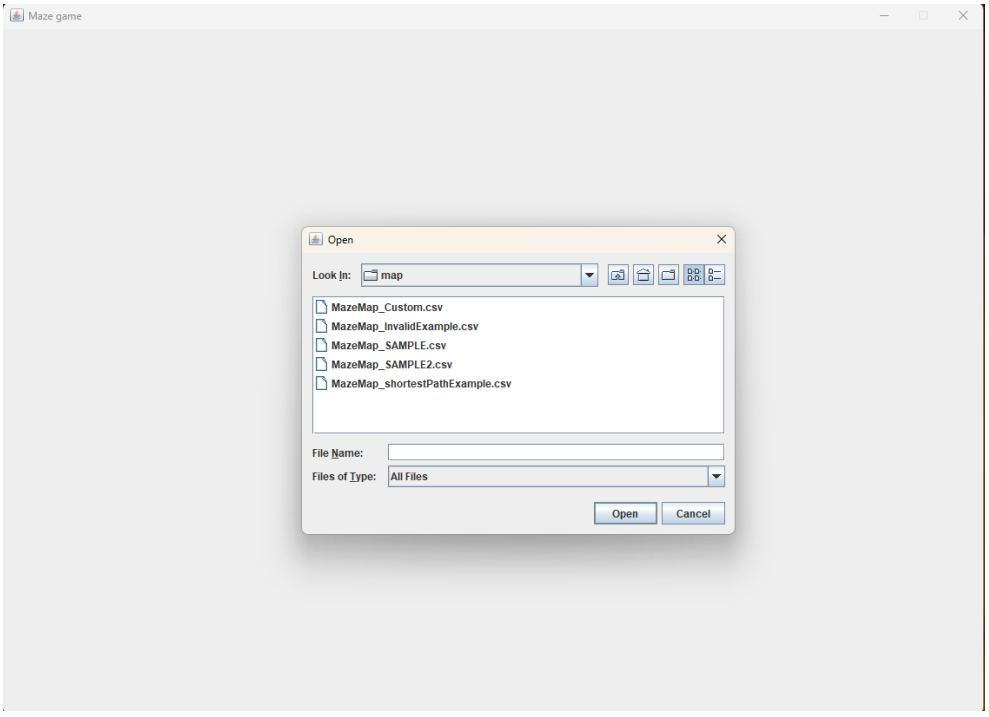
As it is exported as a html document, the JavaDoc is included in the repository.
Link: https://github.com/nychopkie/Comp3111F23G30/blob/master/javadoc/

In case the JavaDoc cannot be viewed in a html viewer, screenshots are included in the appendix.

Task 215: Screenshots of the application software

Starting Screen

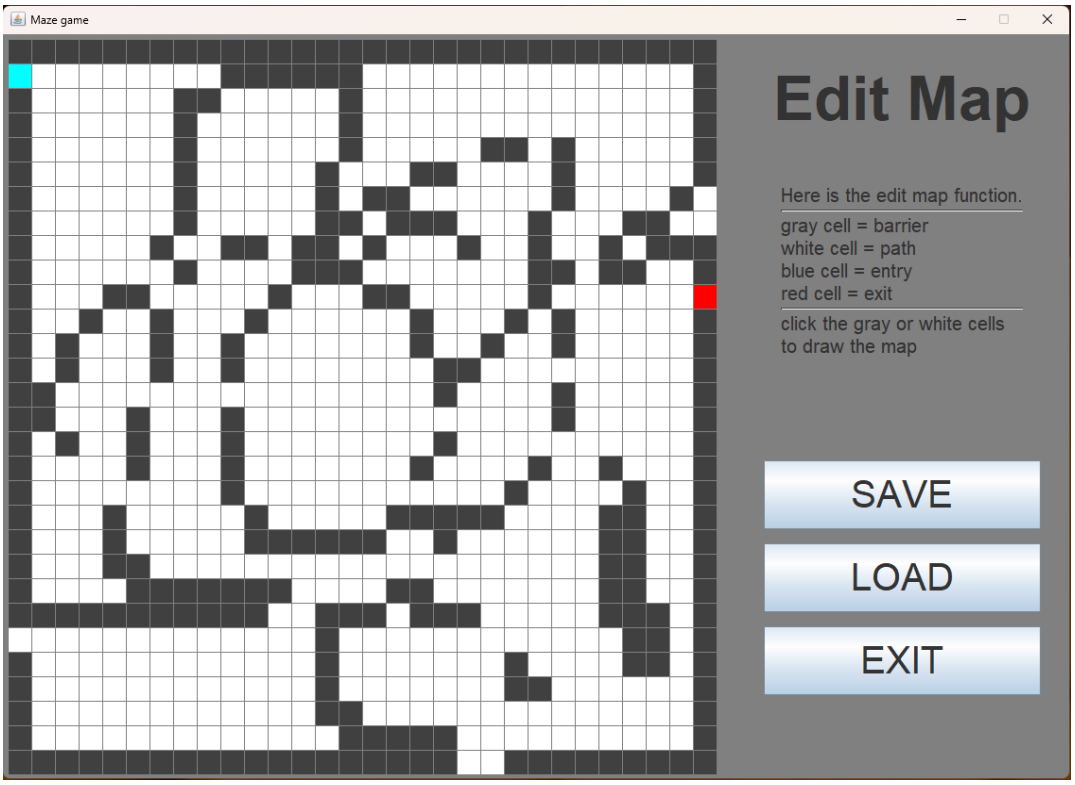
choosing the map



Playing the game: Game start

The function to display shortest path from entry to exit

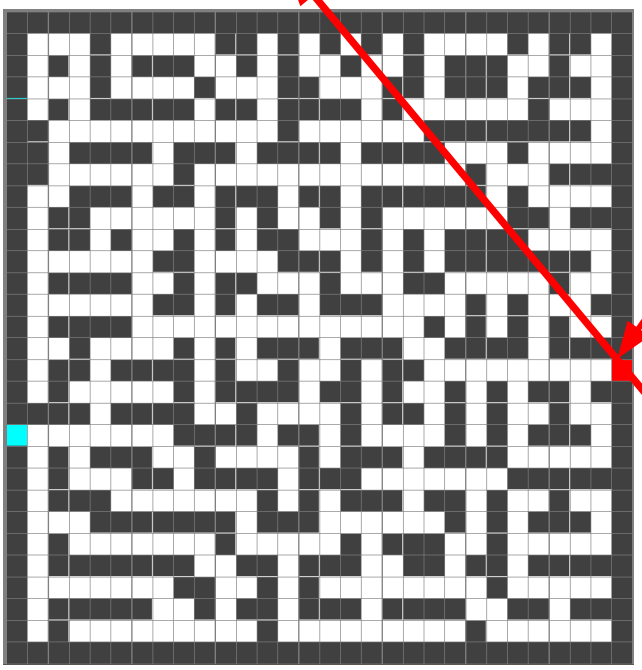
Function to edit your own maze map, load, and save it



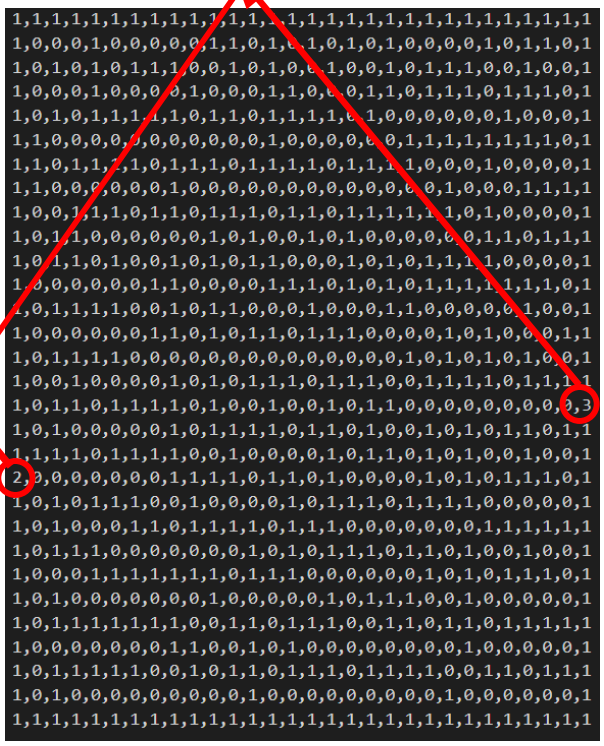
Class A:MazeMap

Entry point located at Vertex[21,1]

Exit point located at Vertex[18,30]

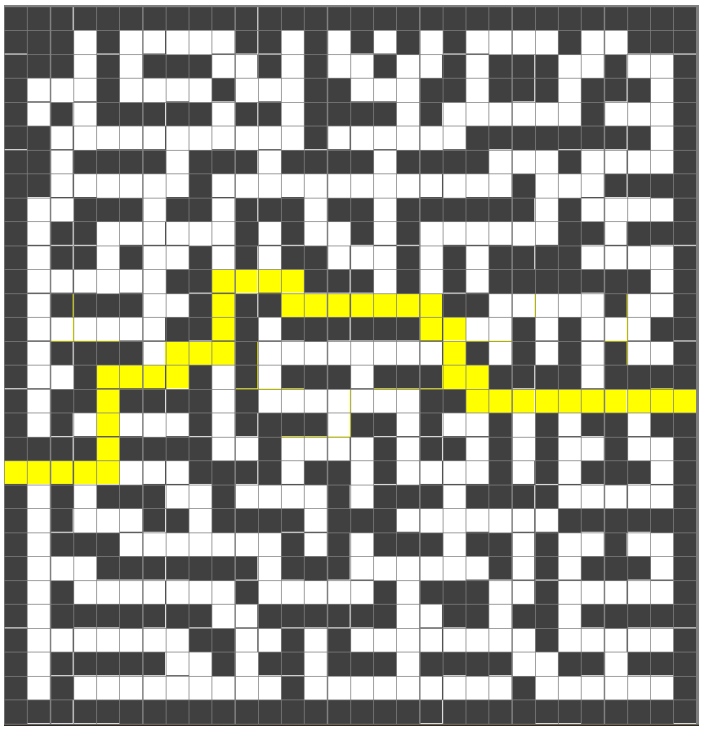


The maze game is operated in a square-shaped grid with 30x30 Vertex squares. Each vertex is 25 x 25 pixel.



The maze map is denoted by a 30x30 matrix list in csv separated each row by a new line

Class B: Shortestpath

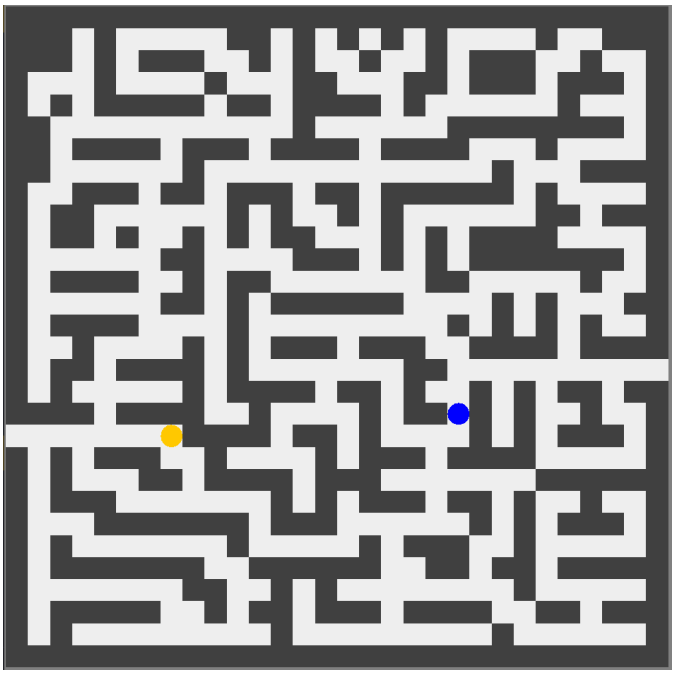


Screen layout of finding the shortest path in user generated map. The left-most corner starts at (0,0), and follows the format (x,y), where x is the horizontal position, y is the vertical position

PathType	PathNo	Index	Row_X	Col_Y
SP	1	1	19	0
SP	1	2	19	1
SP	1	3	19	2
SP	1	4	19	3
SP	1	5	19	4
SP	1	6	18	4
SP	1	7	17	4
SP	1	8	16	4
SP	1	9	15	4
SP	1	10	15	5
SP	1	11	15	6
SP	1	12	15	7
SP	1	13	14	7
SP	1	14	14	8
SP	1	15	14	9
SP	1	16	13	9
SP	1	17	12	9
SP	1	18	11	9
SP	1	19	11	10
SP	1	20	11	11
SP	1	21	11	12
SP	1	22	12	12
SP	1	23	12	13
SP	1	24	12	14
SP	1	25	12	15
SP	1	26	12	16
SP	1	27	12	17
SP	1	28	12	18
SP	1	29	13	18
SP	1	30	13	19
SP	1	31	14	19
SP	1	32	15	19
SP	1	33	15	20
SP	1	34	16	20
SP	1	35	16	21
SP	1	36	16	22
SP	1	37	16	23
SP	1	38	16	24
SP	1	39	16	25
SP	1	40	16	26
SP	1	41	16	27
SP	1	42	16	28
SP	1	43	16	29

Data example of the shortest path csv generated

Class C:MazeGame



Map screen layout of the Tom and Jerry Maze Game

```
1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1
1,0,0,0,1,0,0,0,0,0,1,1,0,1,0,1,0,1,0,1,0,0,0,0,1,0,1,1,0,1
1,0,1,0,1,0,1,1,1,0,0,1,0,1,0,0,1,0,0,1,0,1,1,1,0,0,1,0,0,1
1,0,0,0,1,0,0,0,0,1,0,0,0,1,1,0,0,0,1,1,0,1,1,1,0,1,1,1,0,1
1,0,1,0,1,1,1,1,1,0,1,1,0,1,1,1,1,0,1,0,0,0,0,0,0,1,0,0,0,1
1,1,0,0,0,0,0,0,0,0,0,0,0,1,0,0,0,0,0,0,1,1,1,1,1,1,1,0,1
1,1,0,1,1,1,1,0,1,1,1,0,1,1,1,1,0,1,1,1,1,0,0,0,1,0,0,0,1
1,1,0,0,0,0,0,0,1,0,0,0,0,0,0,0,0,0,0,0,0,1,0,0,0,1,1,1,1
1,0,0,1,1,1,0,1,1,0,1,1,0,1,1,0,1,1,1,1,1,1,0,1,0,0,0,0,1
1,0,1,1,0,0,0,0,0,0,1,0,1,0,0,1,0,1,0,0,0,0,0,0,1,1,0,1,1,1
1,0,1,1,0,1,0,0,1,0,1,0,1,1,0,0,0,1,0,1,0,1,1,1,1,0,0,0,0,1
1,0,0,0,0,0,0,1,1,0,0,0,0,1,1,1,0,1,0,1,0,1,1,1,1,1,1,0,1
1,0,1,1,1,1,0,0,1,0,1,1,0,0,0,1,0,0,0,1,1,0,0,0,0,0,1,0,1
1,0,0,0,0,0,0,1,1,0,1,0,1,1,0,1,1,1,0,0,0,1,0,1,0,0,0,1,1
1,0,1,1,1,1,0,0,0,0,0,0,0,0,0,0,0,0,1,0,1,0,1,0,1,0,0,1
1,0,0,1,0,0,0,0,1,0,1,0,1,1,0,1,1,1,0,0,1,1,1,1,0,1,1,1,1
1,0,1,0,1,1,0,0,1,0,1,1,0,1,1,1,0,1,1,1,0,0,0,0,0,1,1,0,3
1,0,1,0,0,0,0,0,1,0,1,1,1,1,0,1,1,0,1,0,0,1,0,1,0,1,1,0,1,1
1,1,1,1,0,1,1,1,1,0,0,1,0,0,0,0,1,0,1,1,0,1,0,1,0,0,1,0,0,1
2,0,0,0,0,0,0,0,1,1,1,1,0,1,1,0,1,0,0,0,0,1,0,1,0,1,1,1,0,1
1,0,1,0,1,1,1,0,0,1,0,0,0,0,1,0,1,1,1,0,1,1,1,1,0,0,0,0,0,1
1,0,1,0,0,0,1,1,0,1,1,1,1,0,1,1,1,0,0,0,0,0,0,0,1,1,1,1,1,1
1,0,1,1,1,0,0,0,0,0,0,1,0,1,0,1,1,0,1,1,0,1,0,0,1,0,0,1
1,0,0,0,1,1,1,1,1,1,1,0,1,1,0,0,0,0,0,0,1,0,1,0,1,1,0,1
1,0,1,0,0,0,0,0,0,0,1,0,0,0,0,0,1,0,1,1,1,0,0,1,0,0,0,0,0,1
1,0,1,1,1,1,1,1,1,0,0,1,1,0,1,1,0,0,1,1,0,1,1,0,1,1,1,1,1
1,0,0,0,0,0,0,0,1,1,0,0,1,0,1,0,0,0,0,0,0,0,0,1,0,0,0,0,0,1
1,0,1,1,1,1,1,0,0,1,0,1,1,0,1,1,1,0,1,1,1,0,0,1,1,0,1,1,1
1,0,1,0,0,0,0,0,0,0,0,1,0,0,0,0,0,0,0,0,0,1,0,0,0,0,0,0,1
1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1
```

The csv of the Maze Map that the game is running on

Task 232a: Unit Testing

Interface

1. Test Cases

The following lists the test cases conducted and the type of test it is checking
:

- **test_display()** : Tests if the display can display the correct state, since the function is controlled by the program, therefore the domain of possible inputs is only the valid cases. Checks all states of the Interface state 0, 1, 2, 5.
- **test_handleNav()** : Checks the handler for button press. Since it would only pass valid values within the set {0,1,2,5}, therefore only checks valid is sufficient.
- **test_showMainMenu()**: checks if the main menu can be displayed correctly.
- **test_showGameWindow()** : checks if the game screen can be displayed correctly.
- **test_showMapEdit()** : checks if the edit map screen can be displayed correctly.
- **test_showTestB()** : checks if the shortest path screen can be displayed correctly.
- **test_handleSPLoad()** : checks if the handler for loading map to shortest map screen can handle valid, invalid and null inputs.
- **test_constructor()** : checks if the object Interface can be instantiated correctly

Class A:MazeMap

For MazeMap.java

1. Test Cases

The following lists the test cases conducted and the type of test it is checking
:

- **test_getEntry()** : Checks if the entry can be returned.
- **test_getExit()** : Checks if the exit can be returned .
- **test_getSIZE()** : Checks if the size of the map can be returned.
- **test_refreshColour()** : Tests if when called, the map can revert back to the color of each vertex.
- **test_load_MazeMap()** : Checks if the map can be loaded. Passes valid, invalid and null path to the test.
- **test_save_MazeMap()** : Checks if the map can be saved. All types of map should be able to save since this is just an editing stage.
- **test_getMazeData()** : Checks if the maze map that is a 2D array of Vertex objects can be returned.
- **test_constructor()** : Checks if mazemap can be created.

For Vertex.java

1. Test Cases

The following lists the test cases conducted and the type of test it is checking
:

- **test_constructor()** : Checks if Vertex can be created.
- **test_getVertex_type()** : Checks if the vertex type can be returned.
- **test_gety()** : Checks if the y-coordinates can be returned.
- **test_getx()** : Checks if the x-coordinates can be returned.
- **test_handleMouseClick()** : checks if the action of the mouse click on the screen can be handled, and changes the state of the editing.
- **test_handleMousePressed()** : checks if the action of the mouse press on the screen can be handled, and changes the state of the editing
- **test_handleMouseReleased()** : checks if the action of the mouse release on the screen can be handled, and changes the state of the editing
- **test_handleMouseEntered()** : checks if the action of the mouse entering a vertex while boeing pressed or released on the screen can be handled, and changes the state of the editing

Class B: Shortestpath

1. Test Cases

The following lists the test cases conducted and the type of test it is checking

- **test_addNeighbors_boundary()** : Tests the boundary cases of the add Neighbors to see if they can add the neighboring vertices correctly
- **test_addNeighbors_corner()** : Tests the corner cases of the add Neighbors to see if they can add the neighboring vertices correctly
- **test_addNeighbors_normal()**: Tests the normal cases of the add Neighbors to see if they can add the neighboring vertices correctly
- **test_getNeighbor()** : checks if the neighboring vertices can be returned.
- **test_shortestPath()** : checks if it can find and return the shortest path correctly

Class C:MazeGame

1. Test Cases

The following lists the test cases conducted and the type of test it is checking
:

- **testMazeGameConstructor()** : Tests if the game can be created
- **testLoadMaze()** : Checks the game can load the maze map received, since the function calls the load_maze function of the MazeMap class, all paths passed must be valid.
- **testGetJerryPositionAsVertex()**: test if Jerry's position can be returned
- **testGetPanel()** : checks if the game panel can be returned.
- **testDraw()** : checks if the map can be drawn correctly.
- **testGetx()** : checks if the x-coordinates can be returned.
- **testGety()** : checks if the y-coordinates can be returned.
- **testGetColor()** : checks if the color of the requested GameEntity can be returned.
- **testFollowPathStepByStep()** : Test if the Game Entity can follow the path step by step
- **testGetCurrentPosition()** : checks if the current position of the GameEntity can be returned.
- **testTomMove()** : checks if Tom moves when this is invoked, and whether the movement is valid.
- **testJerryMove()** : checks if Jerry moves when this is invoked, and whether the movement is valid.
- **testJerryConstructor()** : Tests if Jerry can be created.
- **testPaintComponentColors()** : tests if the map can be painted correctly.
- **testGetPreferedSize()** : checks if the preferred size can be returned.
- **testGetMazeMap()** : checks if the MazeMap can be returned.

Task 232b: Coverage

Interface

JUnit line coverage report for interfaceTest.java covering Interface.java

Coverage | java in Comp3111F23G30

Element	Class, %	Method, %	Line, %
all	100% (29/29)	83% (79/95)	91% (686/751)
Interface	100% (16/16)	73% (28/38)	91% (393/428)
Interface	100% (15/15)	72% (27/37)	91% (391/426)
Launcher	100% (1/1)	100% (1/1)	100% (2/2)
MazeGame	100% (8/8)	96% (26/27)	85% (91/106)
MazeGame	100% (8/8)	96% (26/27)	85% (91/106)
MazeMap	100% (5/5)	83% (25/30)	93% (202/217)
MazeMap	100% (3/3)	100% (13/13)	93% (93/100)
Shortestpath	100% (1/1)	100% (3/3)	97% (79/81)
Vertex	100% (1/1)	64% (9/14)	83% (30/36)

1. Coverage Summary

The JUnit testing for Interface.java has achieved a line coverage of 91% (393 out of 428 lines). This indicates a high degree of test coverage, demonstrating that nearly all of the codebase is exercised during testing. However, specific sections remain untested, as detailed below.

2. Uncovered Lines

The analysis reveals 9 specific lines that are not covered by any test case. These are as follows:

- **line 150, 160, 170, 341-343, 353-355, 495, 698, 708** : The absence of coverage in these lines indicates that the test cases do not cover the branch of ActionEvents.
- **line 223, 247, 272, 554, 578,603** : The absence of coverage in these lines suggests that the test cases do not cover branches that have Interface.testMode set as 0.
- **line 238-240, 263-265, 568-570, 593-595** : The absence of coverage in line 169-170 and line 197-198 indicates that the test case does not cover the branch if an interruption exception behavior exists.
- **line 471, 481-484** : The absence of coverage in these lines suggests that the test cases do not cover branches that handle loading and saving of the maps.

3. Analysis of Uncovered Lines

- **line 150, 160, 170, 341-343, 353-355, 495, 698, 708** : all of these lines associate a function call after a button is pressed. Although the functions enclosed in the ActionEvents are uncovered, as the test cases cannot reach these branches, these functions, namely handleNav() is tested separately, and is covered. Therefore, while these lines cannot be tested, their functionalities are tested in other test cases.
- **line 223, 247, 272, 554, 578,603** : To facilitate with the test cases, since the function is designed to use a file chooser when choosing the file to load, therefore there is a testMode checking before entering the branches. As it is running tests, testMode 0 is for user manual input, which is disabled for manual testing for unit tests. Therefore it is unreachable for the tests and is uncovered.
- **line 238-240, 263-265, 568-570, 593-595** : The dialog box is set to be closed after a certain time passed, unless the thread handling the sleep function breaks, which is highly unlikely in user cases, since most likely they would click the ok button on the warning message. Therefore, we cannot test the case where the thread breaks and the line is uncovered.
- **line 471, 481-484** : These lines all call functions that are from a different class, and are handled by the respective class. Therefore although uncovered, they are tested and in those tests they are mostly covered.

All the functions below are in the Java Class: Interface

Test case	Target function	Implemented Function
test_display()	display()	display() all covered
test_handleNav()	handleNav()	handleNav() all covered
test_showMainMenu()	showMainMenu()	showMainMenu() did not cover 150,160,170
test_showGameWindow()	showGameWindow()	showGameWindow() did not cover 223, 238-240, 247, 263-265, 272, 341-343, 353-355
test_showMapEdit()	showMapEdit()	showMapEdit() did not cover 471, 481-484, 495
test_showTestB()	showTestB()	showTestB() did not cover 698, 708
test_handleSPLoad()	handleSPLoad()	handleSPLoad() did not cover 554, 568-570, 578, 593-595, 603
test_constructor()	Interface()	Interface() all covered

Class A:MazeMap

JUnit line coverage report for MazeMapTest.java covering MazeMap.java

Coverage | java in Comp3111F23G30

Element	Class, %	Method, %	Line, %
all	100% (29/29)	83% (79/95)	91% (686/751)
Interface	100% (16/16)	73% (28/38)	91% (393/428)
Interface	100% (15/15)	72% (27/37)	91% (391/426)
Launcher	100% (1/1)	100% (1/1)	100% (2/2)
MazeGame	100% (8/8)	96% (26/27)	85% (91/106)
MazeGame	100% (8/8)	96% (26/27)	85% (91/106)
MazeMap	100% (5/5)	83% (25/30)	93% (202/217)
MazeMap	100% (3/3)	100% (13/13)	93% (93/100)
Shortestpath	100% (1/1)	100% (3/3)	97% (79/81)
Vertex	100% (1/1)	64% (9/14)	83% (30/36)

1. Coverage Summary

The JUnit testing for MazeMap.java has achieved a line coverage of 93% (93 out of 100 lines). This indicates a high degree of test coverage, demonstrating that nearly all of the codebase is exercised during testing. However, specific sections remain untested, as detailed below.

2. Uncovered Lines

The analysis reveals 9 specific lines that are not covered by any test case. These are as follows:

- **line 159-161, 178-180** : The absence of coverage in line 159-161 and line 178-180 indicates that the test case does not cover the branch if it exhibits interruption exception behavior.
- **line 209, 246-247** : The absence of coverage in line 209 and line 246-247 indicates that the test case does not cover the branch if a file exception behavior exists.

3. Analysis of Uncovered Lines

- **line 159-161, 178-180** : The dialog box is set to be closed after a certain time passed, unless the thread handling the sleep function breaks, which is highly unlikely in user cases, since most likely they would click the ok button on the warning message. Therefore, we cannot test the case where the thread breaks and the line is uncovered.

- **line 209, 246-247** : Before the chosen file is loaded to the FileBuffer to scan through the csv file, branches above had already validated the file of its format and existence, therefore there is no chance that the file handled in the branch to be invalid and catch exceptions. Therefore we cannot test these lines and it is uncovered.

All the functions below are in the Java Class: MazeMap

Test case	Target function	Implemented Function
test_getEntry()	getEntry()	getEntry() all covered
test_getExit()	getExit()	getExit() all covered
test_getSIZE()	getSIZE()	getSIZE() all covered
test_refreshColour()	refreshColour()	refreshColour() all covered
test_load_MazeMap()	load_MazeMap()	load_MazeMap() did not cover 159-161, 178-180, 209
test_save_MazeMap()	save_MazeMap()	save_MazeMap() did not cover 246-247
test_getMazeData()	getMazeData()	getMazeData() all covered
test_constructor()	MazeMap()	MazeMap() all covered

JUnit line coverage report for VertexTest.java covering Vertex.java

Coverage | java in Comp3111F23G30

Element	Class, %	Method, %	Line, %
all	100% (29/29)	83% (79/95)	91% (686/751)
Interface	100% (16/16)	73% (28/38)	91% (393/428)
Interface	100% (15/15)	72% (27/37)	91% (391/426)
Launcher	100% (1/1)	100% (1/1)	100% (2/2)
MazeGame	100% (8/8)	96% (26/27)	85% (91/106)
MazeGame	100% (8/8)	96% (26/27)	85% (91/106)
MazeMap	100% (5/5)	83% (25/30)	93% (202/217)
MazeMap	100% (3/3)	100% (13/13)	93% (93/100)
Shortestpath	100% (1/1)	100% (3/3)	97% (79/81)
Vertex	100% (1/1)	64% (9/14)	83% (30/36)

1. Coverage Summary

The JUnit testing for Vertex.java has achieved a line coverage of 83% (30 out of 36 lines), with specific sections remaining untested listed below.

2. Uncovered Lines

The analysis reveals 6 specific lines that are not covered by any test case. These are as follows:

- **line 109, 140-141, 157, 174, 190:** the absence of coverage in the listed lines indicates that the test case does not cover the branch of mouse event.

3. Analysis of Uncovered Lines

- **line 109, 190**: due to the fact that if not implemented, the class cannot run properly, therefore, the void functions simply returns with no code executed, therefore it cannot be tested, nor would it need to be tested since it does nothing.
- **line 140-141, 157, 174**: these functions are all mouse action handlers, it responds to the mouse actions. However, although the mouse inputs are not tested, as the mouse actions pass the action to handler functions to perform work, the handler function for these functions are tested instead. Therefore they are uncovered.

All the functions below are in the Java Class: Vertex

Test case	Target function	Implemented Function
test_constructor()	Vertex()	Vertex() all covered
test_getVertex_type()	getVertex_type()	getVertex_type() all covered
test_gety()	gety()	gety() all covered
test_getx()	getx()	getx() all covered
	mouseClicked()	109: not covered since does nothing
test_handleMouseClick()	handleMouseClick()	handleMouseClick() all covered
	mousePressed()	140-141: mouse event
test_handleMousePressed()	handleMousePressed()	handleMousePressed() all covered
	mouseReleased()	157: mouse event
test_handleMouseReleased()	handleMouseReleased()	handleMouseReleased() all covered
	mouseEntered()	174: mouse event
test_handleMouseEntered()	handleMouseEntered()	handleMouseEntered() all covered
	mouseReleased()	190: not covered since does nothing

Class B: Shortestpath

JUnit line coverage report for ShortestPathTest.java covering Shortestpath.java

Coverage | java in Comp3111F23G30

Element	Class, %	Method, %	Line, %
all	100% (29/29)	83% (79/95)	91% (686/751)
Interface	100% (16/16)	73% (28/38)	91% (393/428)
Interface	100% (15/15)	72% (27/37)	91% (391/426)
Launcher	100% (1/1)	100% (1/1)	100% (2/2)
MazeGame	100% (8/8)	96% (26/27)	85% (91/106)
MazeGame	100% (8/8)	96% (26/27)	85% (91/106)
MazeMap	100% (5/5)	83% (25/30)	93% (202/217)
MazeMap	100% (3/3)	100% (13/13)	93% (93/100)
Shortestpath	100% (1/1)	100% (3/3)	97% (79/81)
Vertex	100% (1/1)	64% (9/14)	83% (30/36)

1. Coverage Summary

The JUnit testing for Shortestpath.java has achieved a line coverage of 97% (79 out of 81 lines). This indicates a high degree of test coverage, demonstrating that nearly all of the codebase is exercised during testing. However, specific sections remain untested, as detailed below.

2. Uncovered Lines

The analysis reveals specific lines that are not covered by any test case. These are as follows:

- **line 107-108** : the absence of coverage in line 107-108 indicates that the test case does not cover the branch if a file exception behavior exists.
-

3. Analysis of Uncovered Lines

- **line 107-108** : noted that the path of the csv file is fixed, so it is impossible to have file exception behavior. However, if we delete the file exception branch, it will cause syntax errors. Therefore, we could not test line 107-108.

All the functions below are in the Java Class: Shortestpath

Test case	Target function	Implemented Function
test_addNeighbors_boundary()	addNeighbors()	addNeighbors() all covered
test_addNeighbors_corner()	addNeighbors()	addNeighbors() all covered
test_addNeighbors_normal()	addNeighbors()	addNeighbors() all covered
test_getNeighbor()	getNeighbors()	getNeighbors() all covered
test_shortestPath()	shortestpath()	shortestpath() 107-108 uncovered

Class C:MazeGame

JUnit line coverage report for MazeGameTest.java covering MazeGame.java

Coverage | java in Comp3111F23G30

Element	Class, %	Method, %	Line, %
all	100% (29/29)	83% (79/95)	91% (686/751)
Interface	100% (16/16)	73% (28/38)	91% (393/428)
Interface	100% (15/15)	72% (27/37)	91% (391/426)
Launcher	100% (1/1)	100% (1/1)	100% (2/2)
MazeGame	100% (8/8)	96% (26/27)	85% (91/106)
MazeGame	100% (8/8)	96% (26/27)	85% (91/106)
MazeMap	100% (5/5)	83% (25/30)	93% (202/217)
MazeMap	100% (3/3)	100% (13/13)	93% (93/100)
Shortestpath	100% (1/1)	100% (3/3)	97% (79/81)
Vertex	100% (1/1)	64% (9/14)	83% (30/36)

1. Coverage Summary

The JUnit testing for MazeGame.java has achieved a line coverage of 85%(91/106). This indicates a high degree of test coverage, demonstrating that a significant portion of the codebase is exercised during testing. However, specific sections remain untested, as detailed below.

2. Uncovered Lines

The analysis reveals specific lines that are not covered by any test case. These are as follows:

- **Lines 79, 81-82, 84-85, 87-88, 90, 94** : These lines pertain to a key listener implementation. The absence of coverage here suggests that the test cases do not simulate key presses, which are crucial for triggering this portion of the code.
- **Lines 404-406, 409-411 :** These lines are integral to the game's real-time logic, particularly involving the interaction between characters Jerry and Tom. This portion of code requires player input to move Jerry, which in turn influences Tom's behavior in the game.

3. Analysis of Uncovered Lines

- **Lines 79, 81-82, 84-85, 87-88, 90, 94 (KeyListener)**: Key listeners require actual user interaction, typically key presses, to be activated. Simulating this interaction within JUnit test cases poses a significant challenge, as it requires external input which is outside the scope of standard unit testing practices.
- **Lines 404-406, 409-411 (Real-Time Game Logic):** The uncovered lines here are part of a real-time game logic mechanism where the character Jerry needs to be moved by player input, which subsequently affects the actions of the character Tom. Testing such dynamic, real-time interactions is complex within the confines of JUnit, as it typically requires simulating player inputs and responses in a game environment – a scenario that extends beyond standard unit testing.

All the functions below are in the Java Class: MazeGame

Test case	Target function	Implemented Function
testMazeGameConstructor	MazeGame()	MazeGame() uncovered: 79, 81-82, 84-85, 87-88, 90, 94
testLoadMaze	MazeGame.loadMaze()	loadMaze all covered
testGetJerryPositionAsVertex	MazeGame.getJerryPositionAsVertex()	getJerryPositionAsVertex all covered
testGetPanel	MazeGame.getPanel()	MazeGame.getPanel() all covered
testConstructor	GameEntity()	GameEntity() all covered
testDraw	GameEntitydraw()	GameEntitydraw() all covered
testGetx	GameEntity.getx()	getx all covered
testGety	GameEntity.gety()	gety all covered
testGetColor	GameEntity.getColor()	getColor all covered
testFollowPathStepByStep	Tom.followPathStepByStep()	followPathStepByStep all covered

testGetCurrentPosition	Tom.getCurrentPosition()	getCurrentPosition all covered
testTomMove	Tom.move()	move all covered
testJerryMove	Jerry.move()	move all covered
testJerryConstructor	Jerry()	Jerry() all covered
testPaintComponentColors	GamePanel.paintComponent()	paintComponent all covered
testGetPreferedSize	GamePanel.getPreferedSize()	getPreferedSize all covered
testGetMazeMap	getMazeMap()	getMazeMap all covered
	gameLoop()	cannot be covered since requires key inputs

Test-per-function Ratio

The Test-per-function ratio is a metric that quantifies the density of test cases relative to the number of functions in the code. It is calculated as follows:
total number of test case: 47
number of functions: 51
Test-per-function ratio= 47/(160-max(51,80)) = 0.5875

In this context, the Test-per-function ratio is computed as 47/80. This implies that, on average, every function in MazeGame.java is tested by approximately 0.6 test cases. While this ratio provides a general idea of testing density, it is important to note that the effectiveness of tests is not solely dependent on quantity but also on the quality and comprehensiveness of each test case.

Appendix

JavaDoc for Launcher

Package Interface

Class Launcher

java.lang.Object
Interface.Launcher

```
public class Launcher
extends Object
```

use this to init the game

Constructor Summary

Constructors

Constructor	Description
`Launcher()`	

Method Summary

All Methods Static Methods Concrete Methods

Modifier and Type	Method	Description
`static void`	`main(String [] args0)`	start the game here

Methods inherited from class java.lang.Object

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

Constructor Details

Launcher

```
public Launcher()
```

Method Details

main

```
public static void main(String [] args0)
```

start the game here

Parameters:

`args0` -

JavaDoc for Interface

Package Interface

Class Interface

java.lang.Object
  java.awt.Component
    java.awt.Container
      java.awt.Window
        java.awt.Frame
          javax.swing.JFrame
            Interface.Interface

All Implemented Interfaces:

`ImageObserver` , `MenuContainer` , `Serializable` , `Accessible` , `RootPaneContainer` , `WindowConstants`

```
public class Interface
extends JFrame
```

class that displays the game

ATTRIBUTES
1. MazeGame: the maze map
2. static state: the state that the screen should be displaying
3. static testMode: for turning on test mode for unit test

OPERATIONS
1. Interface()
2. display()
3. showMainMenu()
4. showGameWindow()
5. showMapEdit()
6. showTestB()
7. handleNav(int state)
8. handleSPLoad()

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`

Nested classes/interfaces inherited from class java.awt.Container

`Container.AccessibleAWTContainer`

Nested classes/interfaces inherited from class java.awt.Component

`Component.AccessibleAWTComponent` , `Component.BaselineResizeBehavior` , `Component.BltBufferStrategy` , `Component.FlipBufferStrategy`

Field Summary

Fields

Modifier and Type	Field	Description
`static int`	`state`	state of the interface to control what page now to show 0: starting menu 1: edit map 2: play game 5: test shortestPath
`static int`	`testMode`	for testing: 1 = have choice 2 = no choice 3 = invalid inputs (not csv) 4 = invalid input (unplayable) 0 = not test

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

DISPOSE_ON_CLOSE , DO_NOTHING_ON_CLOSE , EXIT_ON_CLOSE , HIDE_ON_CLOSE

Constructor Summary

Constructors

Constructor	Description
Interface()	constructor of Interface

Method Summary

All Methods Instance Methods Concrete Methods

Modifier and Type	Method	Description
void	display()	refresh and display the interface according to the state
void	handleNav(int state)	handler for mouse button to test
boolean	handleSPLoad()	help test the button click of the shortest path function
boolean	showGameWindow()	state 1: to show the game screen
void	showMainMenu()	state 0: to show the menu screen
void	showMapEdit()	state 2: to show edit screen
void	showTestB()	state 5: show the shortest path

Methods inherited from class javax.swing.JFrame

addImpl , createRootPane , frameInit , getAccessibleContext , getContentPane , getDefaultCloseOperation , getGlassPane , getGraphics , getJMenuBar , getLayeredPane , getRootPane , getTransferHandler , isDefaultLookAndFeelDecorated , isRootPaneCheckingEnabled , paramString , processWindowEvent , remove , repaint , setContentPane , setDefaultCloseOperation , setDefaultLookAndFeelDecorated , setGlassPane , setIconImage , setJMenuBar , setLayeredPane , setLayout , setRootPane , setRootPaneCheckingEnabled , setTransferHandler , update

Methods inherited from class java.awt.Frame

addNotify , getCursorType , getExtendedState , getFrames , getIconImage , getMaximizedBounds , getMenuBar , getState , getTitle , isResizable , isUndecorated , remove , removeNotify , setBackground , setCursor , setExtendedState , setMaximizedBounds , setMenuBar , setOpacity , setResizable , setShape , setState , setTitle , setUndecorated

Methods inherited from class java.awt.Window

addPropertyChangeListener , addPropertyChangeListener , addWindowFocusListener , addWindowListener , addWindowStateListener , applyResourceBundle , applyResourceBundle , createBufferStrategy , createBufferStrategy , dispose , getBackground , getBufferStrategy , getFocusableWindowState , getFocusCycleRootAncestor , getFocusOwner , getFocusTraversalKeys , getIconImages , getInputContext , getListeners , getLocale , getModalExclusionType , getMostRecentFocusOwner , getOpacity , getOwnedWindows , getOwner , getOwnerlessWindows , getShape , getToolkit , getType , getWarningString , getWindowFocusListeners , getWindowListeners , getWindows , getWindowStateListeners , hide , isActive , isAlwaysOnTop , isAlwaysOnTopSupported , isAutoRequestFocus , isFocusableWindow , isFocusCycleRoot , isFocused , isLocationByPlatform , isOpaque , isShowing , isValidateRoot , pack , paint , postEvent , processEvent , processWindowFocusEvent , processWindowStateEvent , removeWindowFocusListener , removeWindowListener , removeWindowStateListener , reshape , setAlwaysOnTop , setAutoRequestFocus , setBounds , setBounds , setCursor , setFocusableWindowState , setFocusCycleRoot , setIconImages , setLocation , setLocation , setLocationByPlatform , setLocationRelativeTo , setMinimumSize , setModalExclusionType , setSize , setSize , setType , setVisible , show , toBack , toFront

Methods inherited from class java.awt.Container

add , add , add , add , add , addContainerListener , applyComponentOrientation , areFocusTraversalKeysSet , countComponents , deliverEvent , doLayout , findComponentAt , findComponentAt , getAlignmentX , getAlignmentY , getComponent , getComponentAt , getComponentAt , getComponentCount , getComponents , getComponentZOrder , getContainerListeners , getFocusTraversalPolicy , getInsets , getLayout , getMaximumSize , getMinimumSize , getMousePosition , getPreferredSize , insets , invalidate , isAncestorOf , isFocusCycleRoot , isFocusTraversalPolicyProvider , isFocusTraversalPolicySet , layout , list , list , locate , minimumSize , paintComponents , preferredSize , print , printComponents , processContainerEvent , remove , removeAll , removeContainerListener , setComponentZOrder , setFocusTraversalKeys , setFocusTraversalPolicy , setFocusTraversalPolicyProvider , setFont , transferFocusDownCycle , validate , validateTree

Methods inherited from class java.awt.Component

action , add , addComponentListener , addFocusListener , addHierarchyBoundsListener , addHierarchyListener , addInputMethodListener , addKeyListener , addMouseListener , addMouseMotionListener , addMouseWheelListener , bounds , checkImage , checkImage , coalesceEvents , contains , contains , createImage , createImage , createVolatileImage , createVolatileImage , disable , disableEvents , dispatchEvent , enable , enable , enableEvents , enableInputMethods , firePropertyChange , firePropertyChange , firePropertyChange , firePropertyChange , firePropertyChange , firePropertyChange , firePropertyChange , firePropertyChange , firePropertyChange , getBaseline , getBaselineResizeBehavior , getBounds , getBounds , getColorModel , getComponentListeners , getComponentOrientation , getCursor , getDropTarget , getFocusListeners , getFocusTraversalKeysEnabled , getFont , getFontMetrics , getForeground , getGraphicsConfiguration , getHeight , getHierarchyBoundsListeners

, getHierarchyListeners , getIgnoreRepaint , getInputMethodListeners , getInputMethodRequests , getKeyListeners , getLocation , getLocation , getLocationOnScreen , getMouseListeners , getMouseMotionListeners , getMousePosition , getMouseWheelListeners , getName , getParent , getPropertyChangeListeners , getPropertyChangeListeners , getSize , getSize , getTreeLock , getWidth , getX , getY , gotFocus , handleEvent , hasFocus , imageUpdate , inside , isBackgroundSet , isCursorSet , isDisplayable , isDoubleBuffered , isEnabled , isFocusable , isFocusOwner , isFocusTraversable , isFontSet , isForegroundSet , isLightweight , isMaximumSizeSet , isMinimumSizeSet , isPreferredSizeSet , isValid , isVisible , keyDown , keyUp , list , list , list , location , lostFocus , mouseDown , mouseDrag , mouseEnter , mouseExit , mouseMove , mouseUp , move , nextFocus , paintAll , prepareImage , prepareImage , printAll , processComponentEvent , processFocusEvent , processHierarchyBoundsEvent , processHierarchyEvent , processInputMethodEvent , processKeyEvent , processMouseEvent , processMouseMotionEvent , processMouseWheelEvent , removeComponentListener , removeFocusListener , removeHierarchyBoundsListener , removeHierarchyListener , removeInputMethodListener , removeKeyListener , removeMouseListener , removeMouseMotionListener , removeMouseWheelListener , removePropertyChangeListener , removePropertyChangeListener , repaint , repaint , repaint , requestFocus , requestFocus , requestFocus , requestFocus , requestFocusInWindow , requestFocusInWindow , requestFocusInWindow , resize , resize , revalidate , setComponentOrientation , setDropTarget , setEnabled , setFocusable , setFocusTraversalKeysEnabled , setForeground , setIgnoreRepaint , setLocale , setMaximumSize , setMixingCutoutShape , setName , setPreferredSize , show , size , toString , transferFocus , transferFocusBackward , transferFocusUpCycle

Methods inherited from class java.lang.Object

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

Methods inherited from interface java.awt.MenuContainer

getFont , postEvent

Field Details

state

```
public static int state
```

state of the interface to control what page now to show
0: starting menu
1: edit map
2: play game
5: test shortestPath

testMode

```
public static int testMode
```

for testing:
1 = have choice
2 = no choice
3 = invalid inputs (not csv)
4 = invalid input (unplayable)
0 = not test

Constructor Details

Interface

```
public Interface()
```

constructor of Interface

Method Details

display

```
public void display()
```

refresh and display the interface according to the state

handleNav

```
public void handleNav(int state)
```

handler for mouse button to test

Parameters:

`state` -

showMainMenu

```
public void showMainMenu()
```

state 0: to show the menu screen

showGameWindow

```
public boolean showGameWindow()
```

state 1: to show the game screen

showMapEdit

```
public void showMapEdit()
```

state 2: to show edit screen

handleSPLoad

```
public boolean handleSPLoad()
```

help test the button click of the shortest path function

Returns:

true if cannot load, false if can load or cancel operation

showTestB

```
public void showTestB()
```

state 5: show the shortest path

JavaDoc for MazeMap

Package MazeMap

Class MazeMap

java.lang.Object
- java.awt.Component
  - java.awt.Container
    - javax.swing.JComponent
      - javax.swing.JPanel
        - MazeMap.MazeMap

All Implemented Interfaces:

`ImageObserver` ,`MenuContainer` ,`Serializable` ,`Accessible`

```
public class MazeMap
extends JPanel
```

class that shows the map of the game

ATTRIBUTES
1. MazeMapData: the map data of all vertices
2. SIZE: the size of the map
3. PIXEL_SIZE: the pixel size of each vertex
4. GAP: the gap size between each vertex
5. entry: the entry's y-coordinate
6. exit: the exit's y-coordinate
7. savePath: the path where custom map would be saved

OPERATIONS
1. Default Constructor MazeMap()
2. getEntry()
3. getExit()
4. getSIZE()
5. setSavePath(String path)
6. refreshColour()
7. load_MazeMap(String filePath)
8. save_MazeMap(String filePath)
9. getMazeData()

See Also:

Serialized Form

Nested Class Summary

Nested classes/interfaces inherited from class javax.swing.JPanel

`JPanel.AccessibleJPanel`

Nested classes/interfaces inherited from class javax.swing.JComponent

`JComponent.AccessibleJComponent`

Nested classes/interfaces inherited from class java.awt.Container

`Container.AccessibleAWTContainer`

Nested classes/interfaces inherited from class java.awt.Component

`Component.AccessibleAWTComponent` , `Component.BaselineResizeBehavior` , `Component.BltBufferStrategy` , `Component.FlipBufferStrategy`

Field Summary

Fields

Modifier and Type	Field	Description
`static String`	`savePath`	the path that the user customized map stores to

Fields inherited from class javax.swing.JComponent

`listenerList` , `TOOL_TIP_TEXT_KEY` , `ui` , `UNDEFINED_CONDITION` , `WHEN_ANCESTOR_OF_FOCUSED_COMPONENT` , `WHEN_FOCUSED` , `WHEN_IN_FOCUSED_WINDOW`

Fields inherited from class java.awt.Component

`accessibleContext` , `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`

Constructor Summary

Constructors

Constructor	Description
`MazeMap()`	MazeMap Constructor This constructor initializes an empty MazeMap

Method Summary

All Methods Instance Methods Concrete Methods

Modifier and Type	Method	Description
`Vertex`	`getEntry()`	Returns the entry Vertex of the map

Vertex	getExit()	Returns the exit Vertex of the map
Vertex[][]	getMazedata()	Returns the map data as a 2D array of Vertex objects
int	getSIZE()	Returns the size of the map
boolean	load_MazeMap(String filePath)	Load the map data csv into MazeMap.
void	refreshColour()	helper function to refresh map colour
void	save_MazeMap()	Save the MazeMap into a csv file.

Methods inherited from class javax.swing.JPanel

getAccessibleContext , getUI , getUIClassID , paramString , setUI , updateUI

Methods inherited from class javax.swing.JComponent

addAncestorListener , addNotify , addVetoableChangeListener , computeVisibleRect , contains , createToolTip , disable , enable , firePropertyChange , firePropertyChange , firePropertyChange , fireVetoableChange , getActionForKeyStroke , getActionMap , getAlignmentX , getAlignmentY , getAncestorListeners , getAutoscrolls , getBaseline , getBaselineResizeBehavior , getBorder , getBounds , getClientProperty , getComponentGraphics , getComponentPopupMenu , getConditionForKeyStroke , getDebugGraphicsOptions , getDefaultLocale , getFontMetrics , getGraphics , getHeight , getInheritsPopupMenu , getInputMap , getInputMap , getInputVerifier , getInsets , getInsets , getListeners , getLocation , getMaximumSize , getMinimumSize , getNextFocusableComponent , getPopupLocation , getPreferredSize , getRegisteredKeyStrokes , getRootPane , getSize , getToolTipLocation , getToolTipText , getToolTipText , getTopLevelAncestor , getTransferHandler , getVerifyInputWhenFocusTarget , getVetoableChangeListeners , getVisibleRect , getWidth , getX , getY , grabFocus , hide , isDoubleBuffered , isLightweightComponent , isManagingFocus , isOpaque , isOptimizedDrawingEnabled , isPaintingForPrint , isPaintingOrigin , isPaintingTile , isRequestFocusEnabled , isValidateRoot , paint , paintBorder , paintChildren , paintComponent , paintImmediately , paintImmediately , print , printAll , printBorder , printChildren , printComponent , processComponentKeyEvent , processKeyBinding , processKeyEvent , processMouseEvent , processMouseMotionEvent , putClientProperty , registerKeyboardAction , registerKeyboardAction , removeAncestorListener , removeNotify , removeVetoableChangeListener , repaint , repaint , requestDefaultFocus , requestFocus , requestFocus , requestFocusInWindow , requestFocusInWindow , resetKeyboardActions , reshape , revalidate , scrollRectToVisible , setActionMap , setAlignmentX , setAlignmentY , setAutoscrolls , setBackground , setBorder , setComponentPopupMenu , setDebugGraphicsOptions , setDefaultLocale , setDoubleBuffered , setEnabled , setFocusTraversalKeys , setFont , setForeground , setInheritsPopupMenu , setInputMap , setInputVerifier , setMaximumSize , setMinimumSize , setNextFocusableComponent , setOpaque , setPreferredSize , setRequestFocusEnabled , setToolTipText , setTransferHandler , setUI , setVerifyInputWhenFocusTarget , setVisible , unregisterKeyboardAction , update

Methods inherited from class java.awt.Container

add , add , add , add , add , addContainerListener , addImpl , addPropertyChangeListener , addPropertyChangeListener , applyComponentOrientation , areFocusTraversalKeysSet , countComponents , deliverEvent , doLayout , findComponentAt , findComponentAt , getComponent , getComponentAt , getComponentAt , getComponentCount , getComponents , getComponentZOrder , getContainerListeners , getFocusTraversalKeys , getFocusTraversalPolicy , getLayout , getMousePosition , insets , invalidate , isAncestorOf , isFocusCycleRoot , isFocusCycleRoot , isFocusTraversalPolicyProvider , isFocusTraversalPolicySet , layout , list , list , locate , minimumSize , paintComponents , preferredSize , printComponents , processContainerEvent , processEvent , remove , remove , removeAll , removeContainerListener , setComponentZOrder , setFocusCycleRoot , setFocusTraversalPolicy , setFocusTraversalPolicyProvider , setLayout , transferFocusDownCycle , validate , validateTree

Methods inherited from class java.awt.Component

action , add , addComponentListener , addFocusListener , addHierarchyBoundsListener , addHierarchyListener , addInputMethodListener , addKeyListener , addMouseListener , addMouseMotionListener , addMouseWheelListener , bounds , checkImage , checkImage , coalesceEvents , contains , createImage , createImage , createVolatileImage , createVolatileImage , disableEvents , dispatchEvent , enable , enableEvents , enableInputMethods , firePropertyChange , firePropertyChange , firePropertyChange , firePropertyChange , firePropertyChange , firePropertyChange , getBackground , getBounds , getColorModel , getComponentListeners , getComponentOrientation , getCursor , getDropTarget , getFocusCycleRootAncestor , getFocusListeners , getFocusTraversalKeysEnabled , getFont , getForeground , getGraphicsConfiguration , getHierarchyBoundsListeners , getHierarchyListeners , getIgnoreRepaint , getInputContext , getInputMethodListeners , getInputMethodRequests , getKeyListeners , getLocale , getLocation , getLocationOnScreen , getMouseListeners , getMouseMotionListeners , getMousePosition , getMouseWheelListeners , getName , getParent , getPropertyChangeListeners , getPropertyChangeListeners , getSize , getToolkit , getTreeLock , gotFocus , handleEvent , hasFocus , imageUpdate , inside , isBackgroundSet , isCursorSet , isDisplayable , isEnabled , isFocusable , isFocusOwner , isFocusTraversable , isFontSet , isForegroundSet , isLightweight , isMaximumSizeSet , isMinimumSizeSet , isPreferredSizeSet , isShowing , isValid , isVisible , keyDown , keyUp , list , list , list , location , lostFocus , mouseDown , mouseDrag , mouseEnter , mouseExit , mouseMove , mouseUp , move , nextFocus , paintAll , postEvent , prepareImage , prepareImage , processComponentEvent , processFocusEvent , processHierarchyBoundsEvent , processHierarchyEvent , processInputMethodEvent , processMouseWheelEvent , remove , removeComponentListener , removeFocusListener , removeHierarchyBoundsListener , removeHierarchyListener , removeInputMethodListener , removeKeyListener , removeMouseListener , removeMouseMotionListener , removeMouseWheelListener , removePropertyChangeListener , removePropertyChangeListener , repaint , repaint , repaint , requestFocus , requestFocus , requestFocusInWindow , resize , resize , setBounds , setBounds , setComponentOrientation , setCursor , setDropTarget , setFocusable , setFocusTraversalKeysEnabled , setIgnoreRepaint , setLocale , setLocation , setLocation , setMixingCutoutShape , setName , setSize , setSize ,

`show` , `show` , `size` , `toString` , `transferFocus` , `transferFocusBackward` , `transferFocusUpCycle`

Methods inherited from class java.lang.Object

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

Field Details

savePath

```
public static String  savePath
```

the path that the user customized map stores to

Constructor Details

MazeMap

```
public MazeMap()
```

MazeMap Constructor
This constructor initializes an empty MazeMap

Method Details

getEntry

```
public Vertex getEntry()
```

Returns the entry Vertex of the map

Returns:
Vertex() entry

getExit

```
public Vertex getExit()
```

Returns the exit Vertex of the map

Returns:
Vertex() exit

getSIZE

```
public int getSIZE()
```

Returns the size of the map

Returns:
int size of map

refreshColour

```
public void refreshColour()
```

helper function to refresh map colour

load_MazeMap

```
public boolean load_MazeMap(String  filePath)
```

Load the map data csv into MazeMap.

This function will only write back if the passed file is a .csv file, or the path could be found.

Parameters:
`filePath` - the file path of the csv file

Returns:
true if the file is found, false if the file is not found or wrong

save_MazeMap

```
public void save_MazeMap()
```

Save the MazeMap into a csv file.

This function will save the file under Assets/map/MazeMap_Custom[i].csv where the i is the number of files that's custom

getMazedata

```
public Vertex[][] getMazedata()
```

Returns the map data as a 2D array of Vertex objects

Returns:
Vertex[SIZE][SIZE] the map data and the individual vertices

JavaDoc for Vertex

Package MazeMap

Class Vertex

java.lang.Object
  java.awt.Component
    java.awt.Container
      javax.swing.JComponent
        javax.swing.JPanel
          MazeMap.Vertex

All Implemented Interfaces:

`MouseListener` , `ImageObserver` , `MenuContainer` , `Serializable` , `EventListener` , `Accessible`

```
public class Vertex
extends JPanel
implements MouseListener
```

class that controls each grid cell of the map

ATTRIBUTES
1. x: the x-coordinate of the vertex
2. y: the y-coordinate of the vertex
3. draw: control to invoke drag drawing
4. vertex_type: the type of vertex it is
5. static canEdit: control whether the vertex can be edited
6. sizeOfSquare: the size of each square
7. CLEAR_VERTEX_COLOUR: white

OPERATIONS
1. Vertex(int sizeOfSquare, int x, int y, int vertex_type)
2. getVertex_type()
3. gety()
4. getx()
5. handleMouseClick()
6. handleMousePressed()
7. handleMouseReleased()
8. handleMouseEntered()

See Also:

Serialized Form

Nested Class Summary

Nested classes/interfaces inherited from class javax.swing.JPanel

`JPanel.AccessibleJPanel`

Nested classes/interfaces inherited from class javax.swing.JComponent

`JComponent.AccessibleJComponent`

Nested classes/interfaces inherited from class java.awt.Container

`Container.AccessibleAWTContainer`

Nested classes/interfaces inherited from class java.awt.Component

`Component.AccessibleAWTComponent` , `Component.BaselineResizeBehavior` , `Component.BltBufferStrategy` , `Component.FlipBufferStrategy`

Field Summary

Fields

Modifier and Type	Field	Description
`static boolean`	`canEdit`	checks if the vertex can change value
`static boolean`	`draw`	static control for whole map drag drawing

Fields inherited from class javax.swing.JComponent

`listenerList` , `TOOL_TIP_TEXT_KEY` , `ui` , `UNDEFINED_CONDITION` , `WHEN_ANCESTOR_OF_FOCUSED_COMPONENT` , `WHEN_FOCUSED` , `WHEN_IN_FOCUSED_WINDOW`

Fields inherited from class java.awt.Component

`accessibleContext` , `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`

Constructor Summary

Constructors

Constructor	Description
`Vertex(int sizeOfSquare, int x, int y, int vertex_type)`	the constructor for Vertex

Method Summary

All Methods Instance Methods Concrete Methods

Modifier and Type	Method	Description
int	getVertex_type()	returns the vertex type of the Vertex
int	getx()	returns the x-coordinates of the Vertex
int	gety()	returns the y-coordinates of the Vertex
void	handleMouseClick()	handles the action of mouse click
void	handleMouseEntered()	handles the action of mouse entered Vertex
void	handleMousePressed()	handles the action of mouse pressed
void	handleMouseReleased()	handles the action of mouse released
void	mouseClicked(MouseEvent e)	
void	mouseEntered(MouseEvent e)	action if mouse entered Vertex
void	mouseExited(MouseEvent e)	
void	mousePressed(MouseEvent e)	action if press mouse
void	mouseReleased(MouseEvent e)	action if released mouse

Methods inherited from class javax.swing.JPanel

getAccessibleContext , getUI , getUIClassID , paramString , setUI , updateUI

Methods inherited from class javax.swing.JComponent

addAncestorListener , addNotify , addVetoableChangeListener , computeVisibleRect , contains , createToolTip , disable , enable , firePropertyChange , firePropertyChange , firePropertyChange , fireVetoableChange , getActionForKeyStroke , getActionMap , getAlignmentX , getAlignmentY , getAncestorListeners , getAutoscrolls , getBaseline , getBaselineResizeBehavior , getBorder , getBounds , getClientProperty , getComponentGraphics , getComponentPopupMenu , getConditionForKeyStroke , getDebugGraphicsOptions , getDefaultLocale , getFontMetrics , getGraphics , getHeight , getInheritsPopupMenu , getInputMap , getInputMap , getInputVerifier , getInsets , getInsets , getListeners , getLocation , getMaximumSize , getMinimumSize , getNextFocusableComponent , getPopupLocation , getPreferredSize , getRegisteredKeyStrokes , getRootPane , getSize , getToolTipLocation , getToolTipText , getToolTipText , getTopLevelAncestor , getTransferHandler , getVerifyInputWhenFocusTarget , getVetoableChangeListeners , getVisibleRect , getWidth , getX , getY , grabFocus , hide , isDoubleBuffered , isLightweightComponent , isManagingFocus , isOpaque , isOptimizedDrawingEnabled , isPaintingForPrint , isPaintingOrigin , isPaintingTile , isRequestFocusEnabled , isValidateRoot , paint , paintBorder , paintChildren , paintComponent , paintImmediately , paintImmediately , print , printAll , printBorder , printChildren , printComponent , processComponentKeyEvent , processKeyBinding , processKeyEvent , processMouseEvent , processMouseMotionEvent , putClientProperty , registerKeyboardAction , registerKeyboardAction , removeAncestorListener , removeNotify , removeVetoableChangeListener , repaint , repaint ,

requestDefaultFocus , requestFocus , requestFocus , requestFocusInWindow , requestFocusInWindow , resetKeyboardActions , reshape , revalidate , scrollRectToVisible , setActionMap , setAlignmentX , setAlignmentY , setAutoscrolls , setBackground , setBorder , setComponentPopupMenu , setDebugGraphicsOptions , setDefaultLocale , setDoubleBuffered , setEnabled , setFocusTraversalKeys , setFont , setForeground , setInheritsPopupMenu , setInputMap , setInputVerifier , setMaximumSize , setMinimumSize , setNextFocusableComponent , setOpaque , setPreferredSize , setRequestFocusEnabled , setToolTipText , setTransferHandler , setUI , setVerifyInputWhenFocusTarget , setVisible , unregisterKeyboardAction , update

Methods inherited from class java.awt.Container

add , add , add , add , add , addContainerListener , addImpl , addPropertyChangeListener , addPropertyChangeListener , applyComponentOrientation , areFocusTraversalKeysSet , countComponents , deliverEvent , doLayout , findComponentAt , findComponentAt , getComponent , getComponentAt , getComponentAt , getComponentCount , getComponents , getComponentZOrder , getContainerListeners , getFocusTraversalKeys , getFocusTraversalPolicy , getLayout , getMousePosition , insets , invalidate , isAncestorOf , isFocusCycleRoot , isFocusCycleRoot , isFocusTraversalPolicyProvider , isFocusTraversalPolicySet , layout , list , list , locate , minimumSize , paintComponents , preferredSize , printComponents , processContainerEvent , processEvent , remove , remove , removeAll , removeContainerListener , setComponentZOrder , setFocusCycleRoot , setFocusTraversalPolicy , setFocusTraversalPolicyProvider , setLayout , transferFocusDownCycle , validate , validateTree

Methods inherited from class java.awt.Component

action , add , addComponentListener , addFocusListener , addHierarchyBoundsListener , addHierarchyListener , addInputMethodListener , addKeyListener , addMouseListener , addMouseMotionListener , addMouseWheelListener , bounds , checkImage , checkImage , coalesceEvents , contains , createImage , createImage , createVolatileImage , createVolatileImage , disableEvents , dispatchEvent , enable , enableEvents , enableInputMethods , firePropertyChange , firePropertyChange , firePropertyChange , firePropertyChange , firePropertyChange , firePropertyChange , getBackground , getBounds , getColorModel , getComponentListeners , getComponentOrientation , getCursor , getDropTarget , getFocusCycleRootAncestor , getFocusListeners , getFocusTraversalKeysEnabled , getFont , getForeground , getGraphicsConfiguration , getHierarchyBoundsListeners , getHierarchyListeners , getIgnoreRepaint , getInputContext , getInputMethodListeners , getInputMethodRequests , getKeyListeners , getLocale , getLocation , getLocationOnScreen , getMouseListeners , getMouseMotionListeners , getMousePosition , getMouseWheelListeners , getName , getParent , getPropertyChangeListeners , getPropertyChangeListeners , getSize , getToolkit , getTreeLock , gotFocus , handleEvent , hasFocus , imageUpdate , inside , isBackgroundSet , isCursorSet , isDisplayable , isEnabled , isFocusable , isFocusOwner , isFocusTraversable , isFontSet , isForegroundSet , isLightweight , isMaximumSizeSet , isMinimumSizeSet , isPreferredSizeSet , isShowing , isValid , isVisible , keyDown , keyUp , list , list , list , location , lostFocus , mouseDown , mouseDrag , mouseEnter , mouseExit , mouseMove , mouseUp , move , nextFocus , paintAll , postEvent , prepareImage , prepareImage , processComponentEvent , processFocusEvent , processHierarchyBoundsEvent ,

processHierarchyEvent , processInputMethodEvent , processMouseWheelEvent , remove , removeComponentListener , removeFocusListener , removeHierarchyBoundsListener , removeHierarchyListener , removeInputMethodListener , removeKeyListener , removeMouseListener , removeMouseMotionListener , removeMouseWheelListener , removePropertyChangeListener , removePropertyChangeListener , repaint , repaint , repaint , requestFocus , requestFocus , requestFocusInWindow , resize , resize , setBounds , setBounds , setComponentOrientation , setCursor , setDropTarget , setFocusable , setFocusTraversalKeysEnabled , setIgnoreRepaint , setLocale , setLocation , setLocation , setMixingCutoutShape , setName , setSize , setSize , show , show , size , toString , transferFocus , transferFocusBackward , transferFocusUpCycle

Methods inherited from class java.lang.Object

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

Field Details

draw

```
public static boolean draw
```

static control for whole map drag drawing

canEdit

```
public static boolean canEdit
```

checks if the vertex can change value

Constructor Details

Vertex

```
public Vertex(int sizeOfSquare,
              int x,
              int y,
              int vertex_type)
```

the constructor for Vertex

Parameters:

`sizeOfSquare` - pass in the size of each square

`x` - the x-coordinate of this Vertex

`y` - the y-coordinate of this Vertex

`vertex_type` - the type of this Vertex

Method Details

getVertex_type

```
public int getVertex_type()
```

returns the vertex type of the Vertex

Returns:

int vertex_type

gety

```
public int gety()
```

returns the y-coordinates of the Vertex

Returns:

int y

getx

```
public int getx()
```

returns the x-coordinates of the Vertex

Returns:

int x

mouseClicked

```
public void mouseClicked(MouseEvent  e)
```

Specified by:

`mouseClicked` in interface `MouseListener`

handleMouseClick

```
public void handleMouseClick()
```

handles the action of mouse click

if can edit then can change colour and type of the Vertex as long as it is not the entry or exit

mousePressed

```
public void mousePressed(MouseEvent e)
```

action if press mouse

Specified by:

`mousePressed` in interface `MouseListener`

handleMousePressed

```
public void handleMousePressed()
```

handles the action of mouse pressed

if mouse pressed then change the status of draw to true

mouseReleased

```
public void mouseReleased(MouseEvent e)
```

action if released mouse

Specified by:

`mouseReleased` in interface `MouseListener`

handleMouseReleased

```
public void handleMouseReleased()
```

handles the action of mouse released

if mouse released then change the status of draw to false

mouseEntered

```
public void mouseEntered(MouseEvent e)
```

action if mouse entered Vertex

Specified by:

`mouseEntered` in interface `MouseListener`

handleMouseEntered

```
public void handleMouseEntered()
```

handles the action of mouse entered Vertex

if mouse entered and is drawing time then draw!

mouseExited

```
public void mouseExited(MouseEvent  e)
```

Specified by:

`mouseExited` in interface `MouseListener`

JavaDoc for ShortestPath

Package MazeMap

Class Shortestpath

java.lang.Object
  java.awt.Component
    java.awt.Container
      javax.swing.JComponent
        javax.swing.JPanel
          MazeMap.Shortestpath

All Implemented Interfaces:

`ImageObserver` , `MenuContainer` , `Serializable` , `Accessible`

```
public class Shortestpath
extends JPanel
```

it contains a function to find the shortest path and other helper functions used by that function.

Details could be found above each function.

OPERATIONS
1. shortestPath(MazeMap map,Vertex start,Vertex end, int mode)
2. addNeighbors(Vertex cell, List< Vertex > list, MazeMap map)

See Also:

Serialized Form

Nested Class Summary

Nested classes/interfaces inherited from class javax.swing.JPanel

`JPanel.AccessibleJPanel`

Nested classes/interfaces inherited from class javax.swing.JComponent

`JComponent.AccessibleJComponent`

Nested classes/interfaces inherited from class java.awt.Container

`Container.AccessibleAWTContainer`

Nested classes/interfaces inherited from class java.awt.Component

`Component.AccessibleAWTComponent` , `Component.BaselineResizeBehavior` , `Component.BltBufferStrategy` , `Component.FlipBufferStrategy`

Field Summary

Fields inherited from class javax.swing.JComponent

listenerList , TOOL_TIP_TEXT_KEY , ui , UNDEFINED_CONDITION , WHEN_ANCESTOR_OF_FOCUSED_COMPONENT , WHEN_FOCUSED , WHEN_IN_FOCUSED_WINDOW

Fields inherited from class java.awt.Component

accessibleContext , 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

Constructor Summary

Constructors

Constructor	Description
Shortestpath()	

Method Summary

All Methods Static Methods Concrete Methods

Modifier and Type	Method	Description
static void	addNeighbors(Vertex cell, List <Vertex> list, MazeMap map)	the function to add all valid neighbors of a target vertex into a list.
static Vertex	getNeighbor(Vertex cell, int distance, int[][] distances, MazeMap map)	the function to get the neighbor of the target cell, which is with a particular distance value.
static Vertex[]	shortestPath(MazeMap map, Vertex start, Vertex end, int mode)	the function to find the shortest path from the starting vertex to the ending vertex

Methods inherited from class javax.swing.JPanel

getAccessibleContext , getUI , getUIClassID , paramString , setUI , updateUI

Methods inherited from class javax.swing.JComponent

addAncestorListener , addNotify , addVetoableChangeListener , computeVisibleRect , contains , createToolTip , disable , enable , firePropertyChange , firePropertyChange , firePropertyChange , fireVetoableChange , getActionForKeyStroke , getActionMap , getAlignmentX , getAlignmentY , getAncestorListeners , getAutoscrolls , getBaseline , getBaselineResizeBehavior ,

getBorder , getBounds , getClientProperty , getComponentGraphics , getComponentPopupMenu , getConditionForKeyStroke , getDebugGraphicsOptions , getDefaultLocale , getFontMetrics , getGraphics , getHeight , getInheritsPopupMenu , getInputMap , getInputMap , getInputVerifier , getInsets , getInsets , getListeners , getLocation , getMaximumSize , getMinimumSize , getNextFocusableComponent , getPopupLocation , getPreferredSize , getRegisteredKeyStrokes , getRootPane , getSize , getToolTipLocation , getToolTipText , getToolTipText , getTopLevelAncestor , getTransferHandler , getVerifyInputWhenFocusTarget , getVetoableChangeListeners , getVisibleRect , getWidth , getX , getY , grabFocus , hide , isDoubleBuffered , isLightweightComponent , isManagingFocus , isOpaque , isOptimizedDrawingEnabled , isPaintingForPrint , isPaintingOrigin , isPaintingTile , isRequestFocusEnabled , isValidateRoot , paint , paintBorder , paintChildren , paintComponent , paintImmediately , paintImmediately , print , printAll , printBorder , printChildren , printComponent , processComponentKeyEvent , processKeyBinding , processKeyEvent , processMouseEvent , processMouseMotionEvent , putClientProperty , registerKeyboardAction , registerKeyboardAction , removeAncestorListener , removeNotify , removeVetoableChangeListener , repaint , repaint , requestDefaultFocus , requestFocus , requestFocus , requestFocusInWindow , requestFocusInWindow , resetKeyboardActions , reshape , revalidate , scrollRectToVisible , setActionMap , setAlignmentX , setAlignmentY , setAutoscrolls , setBackground , setBorder , setComponentPopupMenu , setDebugGraphicsOptions , setDefaultLocale , setDoubleBuffered , setEnabled , setFocusTraversalKeys , setFont , setForeground , setInheritsPopupMenu , setInputMap , setInputVerifier , setMaximumSize , setMinimumSize , setNextFocusableComponent , setOpaque , setPreferredSize , setRequestFocusEnabled , setToolTipText , setTransferHandler , setUI , setVerifyInputWhenFocusTarget , setVisible , unregisterKeyboardAction , update

Methods inherited from class java.awt.Container

add , add , add , add , add , addContainerListener , addImpl , addPropertyChangeListener , addPropertyChangeListener , applyComponentOrientation , areFocusTraversalKeysSet , countComponents , deliverEvent , doLayout , findComponentAt , findComponentAt , getComponent , getComponentAt , getComponentAt , getComponentCount , getComponents , getComponentZOrder , getContainerListeners , getFocusTraversalKeys , getFocusTraversalPolicy , getLayout , getMousePosition , insets , invalidate , isAncestorOf , isFocusCycleRoot , isFocusCycleRoot , isFocusTraversalPolicyProvider , isFocusTraversalPolicySet , layout , list , list , locate , minimumSize , paintComponents , preferredSize , printComponents , processContainerEvent , processEvent , remove , remove , removeAll , removeContainerListener , setComponentZOrder , setFocusCycleRoot , setFocusTraversalPolicy , setFocusTraversalPolicyProvider , setLayout , transferFocusDownCycle , validate , validateTree

Methods inherited from class java.awt.Component

action , add , addComponentListener , addFocusListener , addHierarchyBoundsListener , addHierarchyListener , addInputMethodListener , addKeyListener , addMouseListener , addMouseMotionListener , addMouseWheelListener , bounds , checkImage , checkImage , coalesceEvents , contains , createImage , createImage , createVolatileImage , createVolatileImage , disableEvents , dispatchEvent , enable , enableEvents , enableInputMethods , firePropertyChange ,

firePropertyChange , firePropertyChange , firePropertyChange , firePropertyChange , firePropertyChange , getBackground , getBounds , getColorModel , getComponentListeners , getComponentOrientation , getCursor , getDropTarget , getFocusCycleRootAncestor , getFocusListeners , getFocusTraversalKeysEnabled , getFont , getForeground , getGraphicsConfiguration , getHierarchyBoundsListeners , getHierarchyListeners , getIgnoreRepaint , getInputContext , getInputMethodListeners , getInputMethodRequests , getKeyListeners , getLocale , getLocation , getLocationOnScreen , getMouseListeners , getMouseMotionListeners , getMousePosition , getMouseWheelListeners , getName , getParent , getPropertyChangeListeners , getPropertyChangeListeners , getSize , getToolkit , getTreeLock , gotFocus , handleEvent , hasFocus , imageUpdate , inside , isBackgroundSet , isCursorSet , isDisplayable , isEnabled , isFocusable , isFocusOwner , isFocusTraversable , isFontSet , isForegroundSet , isLightweight , isMaximumSizeSet , isMinimumSizeSet , isPreferredSizeSet , isShowing , isValid , isVisible , keyDown , keyUp , list , list , list , location , lostFocus , mouseDown , mouseDrag , mouseEnter , mouseExit , mouseMove , mouseUp , move , nextFocus , paintAll , postEvent , prepareImage , prepareImage , processComponentEvent , processFocusEvent , processHierarchyBoundsEvent , processHierarchyEvent , processInputMethodEvent , processMouseWheelEvent , remove , removeComponentListener , removeFocusListener , removeHierarchyBoundsListener , removeHierarchyListener , removeInputMethodListener , removeKeyListener , removeMouseListener , removeMouseMotionListener , removeMouseWheelListener , removePropertyChangeListener , removePropertyChangeListener , repaint , repaint , repaint , requestFocus , requestFocus , requestFocusInWindow , resize , resize , setBounds , setBounds , setComponentOrientation , setCursor , setDropTarget , setFocusable , setFocusTraversalKeysEnabled , setIgnoreRepaint , setLocale , setLocation , setLocation , setMixingCutoutShape , setName , setSize , setSize , show , show , size , toString , transferFocus , transferFocusBackward , transferFocusUpCycle

Methods inherited from class java.lang.Object

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

Constructor Details

Shortestpath

```
public Shortestpath()
```

Method Details

shortestPath

```
public static Vertex[] shortestPath(MazeMap map,
                                    Vertex start,
                                    Vertex end,
                                    int mode)
```

the function to find the shortest path from the starting vertex to the ending vertex

Parameters:

`map` - the Maze map that the user decided to play. Only Maze map with at least 1 possible path should be passed in this parameter.

`start` - the starting point of the shortest path

`mode` - Either 0 or 1. Mode 0 represents the Tom and Jerry mode, where the system calls the shortestpath function and only want to receive the shortest path as return. Mode 1 represents the normal mode where the user directly calls the shortestpath function and want to receive a csv file containing the shortest path and seeing graphic representation of the shortest path.

Returns:

a Vertex array containing the shortest path between the strating point and the ending point. Null if map do not have a possible path from the starting vertex to the ending vertex.

addNeighbors

```
public static void addNeighbors(Vertex cell,
                                List <Vertex> list,
                                MazeMap map)
```

the function to add all valid neighbors of a target vertex into a list.

Parameters:

`cell` - the target vertex that we want to find its neighbors.

`list` - the list that storing the neighbors of target vertex after the function.

`map` - the Maze map that the user decided to play.

getNeighbor

```
public static Vertex getNeighbor(Vertex cell,
                                 int distance,
                                 int[][] distances,
                                 MazeMap map)
```

the function to get the neighbor of the target cell, which is with a particular distance value.

Parameters:

`cell` - the target vertex that we want to find its neighbor.

`distance` - distance from the starting point to that neighbor cell.

`distances` - a Vertex 2D array containing the distances from the starting point to all vertex.

`map` - the Maze map that the user decided to play.

Returns:

A vertex that is the neighbor of cell and has the specified distance. Null if such vertex does not exist.

JavaDoc for MazeGame

Package MazeGame

Class MazeGame

java.lang.Object
MazeGame.MazeGame

```
public class MazeGame
extends Object
```

Main class for the Tom and Jerry maze game.

ATTRIBUTES
- mazeMap: MazeMap object representing the game's maze layout.
- size: Integer representing the fixed size of the maze.
- tom: Tom instance representing the character Tom in the game.
- jerry: Jerry instance representing the character Jerry in the game.
- panel: JPanel for rendering game components.
- timer: Timer for managing the game loop and Jerry's movement.
- DELAY: Integer for delay time in milliseconds for Jerry's movement.
- tomTimer: Timer for Tom's movement.
- TOM_DELAY: Integer for delay time in milliseconds for Tom's movement.
- sizeOfSquare: Integer for the size of each square in the maze.
- jerryPosition: Vertex representing the current position of Jerry in the maze.
- entryPoint: Vertex representing the entry point of the maze.
- exitPoint: Vertex representing the exit point of the maze.

OPERATIONS
- MazeGame(): Constructor to initialize the game window, load the maze, set up game entities, and start the game timers.
- loadMaze(String filePath): Loads the maze configuration from a specified file path.
- getJerryPositionAsVertex(): Returns Jerry's current position as a Vertex object.
- gameLoop(): Main game loop that handles game logic, updates positions, checks win/lose conditions, and refreshes the display.
- isGameWon(): Checks if Jerry has reached the exit point, indicating a win.
- isGameLost(): Checks if Tom has caught Jerry, indicating a loss.

Additional methods for testing:
- getMazeMap(): Returns the MazeMap object.

Nested Class Summary

Nested Classes

Modifier and Type	Class	Description
static enum	MazeGame.Direction	Enum representing possible movement directions for Jerry in the game. ATTRIBUTES - Enum values: UP, DOWN, LEFT, RIGHT.

static class	MazeGame.GameEntity	Represents a basic game entity in the maze, such as Tom or Jerry. ATTRIBUTES - x, y: Integer coordinates of the entity in the maze. - color: Color used to represent the entity on the game panel. - direction: Direction of the entity's movement. OPERATIONS - GameEntity(int x, int y, Color color): Constructor to initialize the entity with coordinates and color. - draw(Graphics g): Method to draw the entity on the game panel. - getx(): Returns the x-coordinate of the entity. - gety(): Returns the y-coordinate of the entity. - getColor(): Returns the color of the entity.
static class	MazeGame.GamePanel	Represents the main panel where the game is drawn. OPERATIONS - paintComponent(Graphics g): Custom painting method for the game panel. - getPreferredSize(): Returns the preferred size of the game panel.
class	MazeGame.Jerry	Represents the character Jerry in the game, extending the GameEntity class. ATTRIBUTES - Inherits x, y, color from GameEntity. - direction: Direction representing the current movement direction of Jerry. OPERATIONS - Jerry(int x, int y): Constructor to initialize Jerry's position, color, and initial direction. - setDirection(Direction direction): Sets the movement direction of Jerry. - move(): Moves Jerry based on the current direction and validates the move. - getDirection(): Returns Jerry's current movement direction. - getColor(): Returns Jerry's current color.
class	MazeGame.Tom	Represents the character Tom in the game, extending the GameEntity class. ATTRIBUTES - Inherits x, y, color from GameEntity. - pathToJerry: Array of Vertex objects storing the calculated path to Jerry. - pathIndex: Integer index for the current position in the pathToJerry array. - flag: Integer flag used in movement logic. OPERATIONS - Tom(int x, int y): Constructor to initialize Tom's position and color. - getCurrentPosition(): Returns Tom's current position

	as a Vertex. - move(MazeMap mazeMap, Vertex jerryPosition): Moves Tom towards Jerry based on the calculated path. - calculatePathToJerry(MazeMap mazeMap, Vertex jerryPosition): Calculates the shortest path to Jerry. - followPathStepByStep(): Method for Tom to follow the calculated path step by step. - setPathToJerry(Vertex[] mockPath): Sets a mock path for Tom, used for testing.

Field Summary

Fields

Modifier and Type	Field	Description
`static MazeGame.Jerry`	`jerry`	
`boolean`	`jerryHasMoved`	
`static Timer`	`timer`	
`static MazeGame.Tom`	`tom`	
`static Timer`	`tomTimer`	

Constructor Summary

Constructors

Constructor	Description
`MazeGame(String path, Interface screen)`	MazeGame.MazeGame Constructor: Initializes the game window, loads the maze, sets up game entities, and starts the game timers.

Method Summary

All Methods Instance Methods Concrete Methods

Modifier and Type	Method	Description
`void`	`gameLoop()`	The main game loop
`Vertex`	`getJerryPositionAsVertex()`	returns the position of Jerry
`MazeMap`	`getMazeMap()`	returns the mazemap
`MazeGame.GamePanel`	`getPanel()`	returns the game panel
`void`	`loadMaze(String filePath)`	function to load maze to the game

Methods inherited from class java.lang.Object

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

Field Details

tom

```
public static MazeGame.Tom tom
```

jerry

```
public static MazeGame.Jerry jerry
```

timer

```
public static Timer  timer
```

tomTimer

```
public static Timer  tomTimer
```

jerryHasMoved

```
public boolean jerryHasMoved
```

Constructor Details

MazeGame

```
public MazeGame(String  path,
                Interface screen)
```

MazeGame.MazeGame Constructor: Initializes the game window, loads the maze, sets up game entities, and starts the game timers.

Method Details

getJerryPositionAsVertex

```
public Vertex getJerryPositionAsVertex()
```

returns the position of Jerry

Returns:

Vertex jerryPosition

getPanel

```
public MazeGame.GamePanel getPanel()
```

returns the game panel

Returns:

GamePanel panel

loadMaze

```
public void loadMaze(String  filePath)
```

function to load maze to the game

Parameters:

`filePath` - the file path to be passed into the game

gameLoop

```
public void gameLoop()
```

The main game loop

getMazeMap

```
public MazeMap getMazeMap()
```

returns the mazemap

Returns:

MazeMap mazeMap