Author of the Component: Basil Issac

Inspection Team Members: Bradley Robinson, Kanishka Garg and Shubham Jain

Date: 30th November 2013

Testing Component/Method/Function Name: Public void paint(Graphics g)

I/P Data: displayWelcomeScreen, displayMainScreen, displaySelectLevel, displayMap

O/P: Screens for the game shall be displayed according to the input received

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*Code\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

**public** **void** paint(Graphics g){**super**.paint(g);

**if**(!win){

**if**(displayWelcomeScreen){

g.drawImage(m.getWelcomeScreen(), 0, 0, 640, 640, **null**);

}

**else**{

**if**(displayMainScreen){

g.drawImage(m.getMainScreen(), 0, 0, 640, 640, **null**);

}

**else**{

**if**(displaySelectLevel){

g.drawImage(m.getSelectLevelScreen(), 0, 0, 640, 640, **null**);

}

**else** **if**(displayHighScore){

g.drawImage(m.getScoreScreen(), 0, 0, 640, 640, **null**);

g.setFont(**new** Font("Arial", Font.BOLD, 18));

g.setColor(Color.white);

**int** verticalSpace = 150;

**for**(**int** i = 0; i < mapHeight; i++){

g.drawString(m.getScores(i), 5\*32, verticalSpace+i\*32);

}

}

**else**{

**if**(displayMap){

**for**(**int** i = 0; i< mapHeight; i++){

**for**(**int** j = 0; j< mapWidth; j++) **if**(m.getMap(i,j).equals("f"))

g.drawImage(m.getFinish(), i \* 32, j \* 32, 32, 32, **null**); **if**(m.getMap(i,j).equals("g"))

g.drawImage(m.getGrass(), i \* 32, j \* 32, 32, 32, **null**);

**if**(m.getMap(i,j).equals("w"))

g.drawImage(m.getWall(), i \* 32, j \* 32, 32, 32, **null**); **if**(m.getMap(i,j).equals("e"))

g.drawImage(m.getWater(), i \* 32, j \* 32, 32, 32, **null**); **if**(m.getMap(i,j).equals("l"))

g.drawImage(m.getLadder(), i\*32, j\*32, 32, 32, **null**);

**if**(m.getMap(i,j).equals("p"))

g.drawImage(m.getFirepole(), i\*32, j\*32, 32, 32, **null**);

**if**(m.getMap(i,j).equals("o"))

g.drawImage(m.getPitfall(), i\*32, j\*32, 32, 32, **null**);

//draws player g.drawImage(p.getPlayer(),p.getTileX() \* 32, p.getTileY() \* 32, 32, 32, **null**);

//draws villain g.drawImage(v.getVillain(),v.getVillainX() \* 32, v.getVillainY() \* 32, 32, 32, **null**);

//draws heart

g.drawImage(h1.getHeart(), 0 \* 32, 19 \* 32, 32, 32, **null**);

g.drawImage(h2.getHeart(), 1 \* 32, 19 \* 32, 32, 32, **null**);

g.drawImage(h3.getHeart(), 2 \* 32, 19 \* 32, 32, 32, **null**);

//draws Floor, Level, Score

g.setFont(**new** Font("Arial", Font.BOLD, 28));

g.setColor(Color.white);

g.drawString("Floor: "+floor, 16\*32, 1\*32);

g.drawString("Level: "+level, 0\*32, 1\*32);

g.drawString("Score: "+score, 8\*32, 1\*32);

//draws Timer

g.setFont(**new** Font("Arial", Font.BOLD, 28));

g.setColor(Color.yellow);

**long** minutes = (time/3600);

**long** seconds = (time/60)%60;

g.drawString("Time Left: "+Long.toString(minutes)+"m"+Long.toString(seconds)+"s", 12\*32, 20\*32);

}

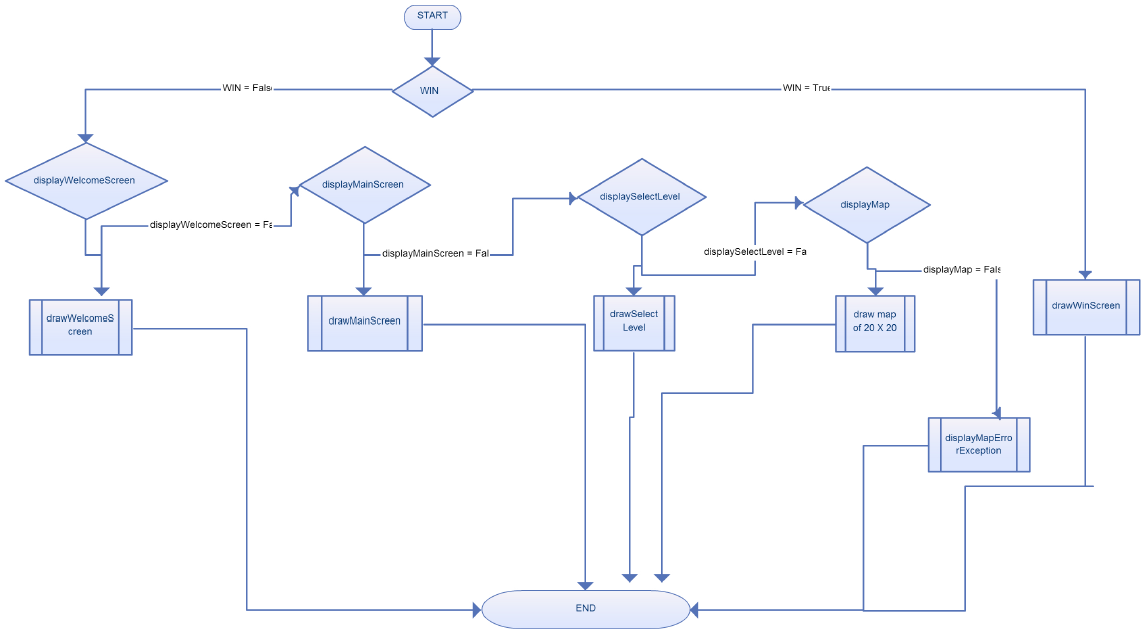
**else**

System.out.println(“Exception Error Thrown”);

}

Testing Plan:

1. **Path Testing**: This type of testing explores the all the possible paths that our code can traverse. So, Inputs are given to test the successful execution of each and every path.



**INPUT**

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **TEST CASE** | **WIN** | **displayWelcome Screen** | **displayMain**  **Screen** | **displaySelect**  **Level** | **displayMap** | **Expected Output** | **Actual Output** | **Result** |
| TC01 | False | True | False | False | False | The Welcome screen shall be displayed. | The Welcome screen gets displayed. | PASS |
| TC02 | False | False | True | False | False | The Main Screen shall be displayed. | The Main Screen gets displayed | PASS |
| TC03 | False | False | False | True | False | The Select Level screen shall be displayed. | The Select Level screen gets displayed. | PASS |
| TC04 | False | False | False | False | True | The High Score screen shall be displayed. | The High Score screen gets displayed. | PASS |
| TC05 | True | False | False | False | False | The Win screen shall be displayed. | The Win screen gets displayed. | PASS |
| TC06 | False | False | False | False | False | The displayMapError Exception shall be raised and displayed. | Exception Throwned | PASS |

1. **Equivalence Testing:**

**Step 1: Identification of the equivalence classes**

Divide the input variables into valid and invalid values.

Variables for this method along with the possible values:

WIN: {True,False}

displayWelcomescreen: {True,False}

displayMainscreen: {True,False}

displaySelectLevel: {True,False}

displayHighScores: {True,False}

**Step 2: Selection of the test inputs**

There can be 32 possible test cases in which 6 are valid cases and already covered in the **path testing** and we will test for a few of the rest 26 test cases in which case the code will raise a InvalidInputException.

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **TEST CASE** | **WIN** | **displayWelcome Screen** | **displayMain**  **Screen** | **displaySelect**  **Level** | **displayMap** | **Expected Output** | **Actual Output** | **Result** |
| TC07 | False | True | **T**rue | False | False | The InvalidInput Exception shall be raised and displayed. | Exception Throwned | PASS |
| TC08 | False | True | **T**rue | False | True | The InvalidInput Exception shall be raised and displayed. | Exception Throwned | PASS |
| TC09 | True | True | **T**rue | False | False | The InvalidInput Exception shall be raised and displayed. | Exception Throwned | PASS |
| TC10 | False | True | **T**rue | True | True | The InvalidInput Exception shall be raised and displayed. | Exception Throwned | PASS |
| TC11 | False | False | **False** | True | True | The InvalidInput Exception shall be raised and displayed. | Exception Throwned | PASS |
| TC12 | False | False | **T**rue | True | True | The InvalidInput Exception shall be raised and displayed. | Exception Throwned | PASS |
| TC13 | True | True | **T**rue | True | True | The InvalidInput Exception shall be raised and displayed. | Exception Throwned | PASS |
| TC14 | False | False | **T**rue | False | True | The InvalidInput Exception shall be raised and displayed. | Exception Throwned | PASS |

1. **Boundary Testing:** This type of testing is special case of equivalence testing which checks the code at boundary of the equivalent classes. In this case, we have Boolean type of inputs and have already covered the boundary cases in the path testing test cases. So, the test cases for Boundary testing are same as Path testing.