Author of the Component: Bradley Robinson

Inspection Team Members: Kanishka Garg, Basil Issac, Shubham Jain

Date: 29th November 2013

**UNIT TESTS**

Component/method/function signature: public void playerMovesSpecialCell ()

I/P: The character in the Map file.

O/P: Processing done in the system/game on the basis of the data in the Map file.

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*code snippet\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

public void playerMovesSpecialCell(){

if(m.getMap(p.getTileX(), p.getTileY()).equals("o")){

decreaseHealth();

p.resetPlayer();

v.resetVillian();

System.out.println("Health Decreases");

}

if(m.getMap(p.getTileX(), p.getTileY()).equals("f")){

win = true;

System.out.println("WON LEVEL: "+level);

level++;

score+=100;

floor=1;

m.openFile("Map"+level+"-1");

p.resetPlayer();

v.resetVillian();

}

if(m.getMap(p.getTileX(), p.getTileY()).equals("p")){

System.out.println("PLAYER IS MOVING FLOOR DOWN");

floor--;

m.openFile("Map1-"+floor);

v.resetVillian();

}

//l is ladder

if(m.getMap(p.getTileX(), p.getTileY()).equals("l")){

System.out.println("PLAYER IS MOVING FLOOR UP");

floor++;

m.openFile("Map1-"+floor);

v.resetVillian();

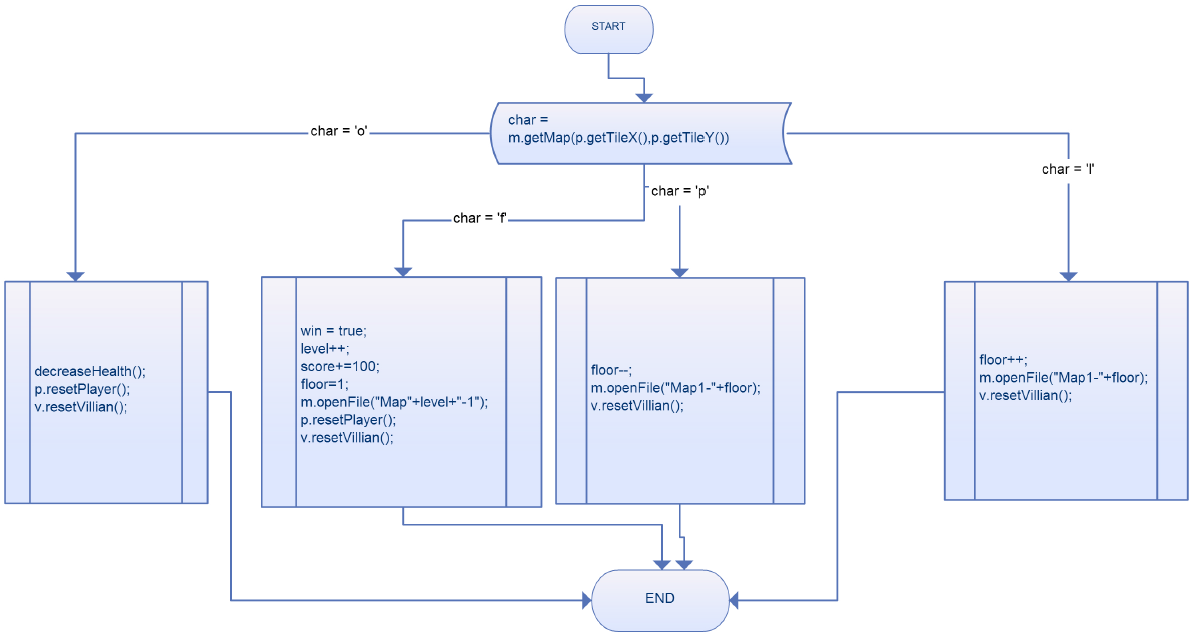
}

}

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*end\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

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.



|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **TEST CASE** | **char** | **Expected Output** | **Actual Output** | **Result** |
| TC01 | ‘o’ | The Player’s health should be decreased and The Player and Villain should be reset to initial position. | Player loses health;  Player position reset;  Villain position reset; | PASS |
| TC02 | ‘f’ | The Player shall win that level, level should increase by 1, score shall increase by 100, floor shall increase by 1, and the Player and Villain shall reset after opening the new file. | Player finishes particular level, displays a winning screen and moves to next level. | PASS |
| TC03 | ‘p’ | The Player shall move to the lower floor by decreasing the floor number by 1 and Villain shall reset after opening the new file of the floor. | Player moves to the floor below. | PASS |
| TC04 | ‘l’ | The Player shall move to the upper floor by decreasing the floor number by 1 and Villain shall reset after opening the new file of the floor. | Player moves to the floor above. | PASS |

**2. 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:

**Char**: {‘o’,’f’,’p’,’l’,’g’,’w’,’e’}🡨 Valid

{0-9}, {special symbols}, {other characters other than above 7}🡨 Invalid

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

We have already included the test cases for valid inputs in the path testing and the invalid case occurs when any Invalid character is included in the map file in which case there can be umpteen number of test cases.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **TEST CASE** | **char** | **Expected Output** | **Actual Output** | **Result** |
| TC05 | ‘g’ | The Player shall move to the next cell. | Player moves on next cell having grass. | PASS |
| TC06 | ‘w’ | The Player shall not be able to move beyond. | Player cannot move since wall is there. | PASS |
| TC07 | ‘e’ | The Player shall not be able to move beyond. | Player cannot move since water is there. | PASS |
| TC08 | ‘4’ | The InvalidCharacterException shall be raised. | Exception thrown. | PASS |
| TC09 | ‘#’ | The InvalidCharacterException shall be raised. | Exception thrown. | PASS |
| TC10 | ‘\*’ | The InvalidCharacterException shall be raised. | Exception thrown. | PASS |

3. **Boundary Testing:**

This type of testing is special case of equivalence testing which checks the code at boundary of the equivalent classes. The boundary cases are already covered by **Path** and **Equivalence Testing.**