Author of the Component: Shubham Jain

Inspection Team Members: Basil Issac, Bradley Robinson, Kanishka Garg

Date: 30th November 2013

**UNIT TESTING**

Component/method/function signature: public void VM\_AI(Villain v)

I/P for the method: An object of type Villain

O/P: Move made by the Villain character according to the move made by the Player character object

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

/\*Mummy AI \*/

**public** **void** VM(Villain v){

**int** deltaX = p.getTileX() - v.getVillainX();

**int** deltaY = p.getTileY() - v.getVillainY();

deltaX = -deltaX;

deltaY = -deltaY;

System.*out*.println("DeltaX = "+deltaX);

System.*out*.println("DeltaY = "+deltaY);

**if**(Math.*abs*(deltaX) < Math.*abs*(deltaY))

{

**if**(deltaX < 0)

moveRight(deltaX, deltaY);

**else**

{

**if**(deltaX > 0)

moveLeft(deltaX, deltaY);

**else**

{

**if**(deltaY > 0)

moveUp(deltaX, deltaY);

**else**

moveDown(deltaX, deltaY);

}

}

}

**else** **if**(Math.*abs*(deltaX) > Math.*abs*(deltaY))

{

**if**(deltaY < 0)

moveDown(deltaX, deltaY);

**else**

{

**if**(deltaY > 0)

moveUp(deltaX, deltaY);

**else**

{

**if**(deltaX > 0)

moveLeft(deltaX, deltaY);

**else**

moveRight(deltaX, deltaY);

}

}

}

**else**

{

System.*out*.println("Exception Error Thrown");

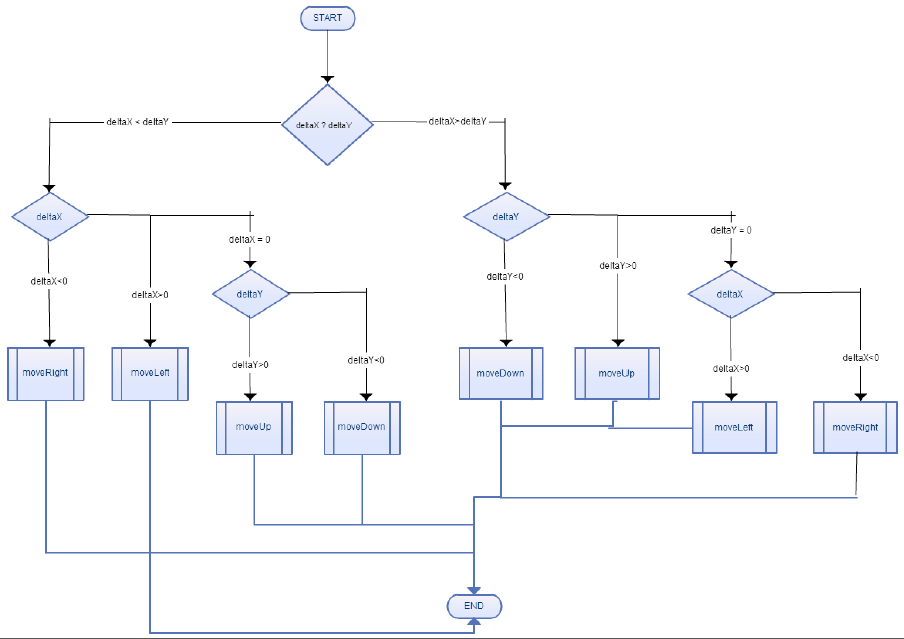
}

}

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*end of method\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

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 | deltaX | deltaY | Expected Output | Actual Output | Result |
| TC01 | -2 | 4 | Villain character should move to the right. | Mummy moves right | PASS |
| TC02 | 4 | 8 | Villain character should move to the left. | Mummy moves left | PASS |
| TC03 | 0 | 10 | Villain character should move to the up. | Mummy moves up | PASS |
| TC04 | 0 | -8 | Villain character should move to the down. | Mummy moves down | PASS |
| TC05 | 3 | -2 | Villain character should move to the down. | Mummy moves down | PASS |
| TC06 | 5 | 2 | Villain character should move to the up. | Mummy moves up | PASS |
| TC07 | 10 | 0 | Villain character should move to the left. | Mummy moves left | PASS |
| TC08 | -8 | 0 | Villain character should move to the right. | Mummy moves right | 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:

deltaX:{-infinity, 0}, {0, infinity}

deltaY:{-infinity, 0}, {0, infinity}

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

We have already included the test cases for valid inputs in the path testing and the only possible invalid case in case deltaX = 0 and deltaY = 0, in which case an exception gets raised. Other invalid cases can occur when deltaX and deltaY takes very large values and the logic fails due to overflow errors in hardware.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **TEST CASE** | **deltaX** | **deltaY** | **Expected Output** | **Actual Output** | **Result** |
| TC09 | 0 | 0 | VillainandPlayerPositionException shall be raised. | As expected | PASS |
| TC10 | Any long int  (like, 999880098) | -2 | IntegerOutofBound Exception shall be raised. | As expected | PASS |
| TC11 | 9 | 111009832388 | IntegerOutofBound Exception shall be raised. | As expected | PASS |
| TC12 | 98190091809 | 98190091809 | IntegerOutofBound Exception shall be raised. | As expected | PASS |

1. **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 TC 03, 04, 07, 08 in **Path Testing** and TC 09 in **Equivalence Testing.**