CPEN522 - Group22 - Assignment1

Yonghan Zhou 93506160

January 2020

Exercise 1.2 Exploratory Testing

Test Charter 1 : Analyze the start/stop/exit function of JPacman
Actor : Normal user
Purpose : To evaluate if the start/stop/exit function works fine in JPacman game.
Setup : A MacBook Pro with 8GB memory and 2.3 GHz Intel Core i5 processor, macOS high sierra operating system, eclipse IDE.
Priority : High, because start/stop/exit function is used very frequently and is one of the most fundamental functions in this game.
Data : JPacman-framework package
Activities/Tours :
1. Run JPacman as a java application. we can see button panel where the "start" button and "exit" button are clickable but "stop" button is in gray.
2. Click the "start" button, the game begins. We can use the keyboard to control the Pac-man. The status bar shows "Playing". The "start" button turns to grey. The "stop" and "exit" buttons are clickable.
3. Click "stop", the game pauses. Everything in the play field stops running. The "start" button is available while "stop" is not.
4. Click "start" again. When the Pac-man is caught by the ghost, the game is over. The status bar shows "You have lost". Only "Exit" button is clickable on the button panel.
5. Click "exit", the play field would be closed.
Missing functions : No opening animation and no music after entering the game. The change of the status bar is not easy to notice, a pop-up will be better.

Test Charter 2 : Analyze the action of hitting the wall in Jpacman
Actor : Normal User
Purpose : To evaluate if the action of hitting the wall works fine in JPacman game.
Setup: A MacBook Pro with 8GB memory and 2.3 GHz Intel Core i5 processor, macOS high sierra operating system, eclipse IDE.
Priority : High, because this is an action that happens all the time through the game. It is importance for calculating the points.
Data: JPacman-framework package

Activities/Tours :
1. Run JPacman as a java application. The points are 0/1780. The Pac-man stays on a tile not a wall.
2. Move the Pac-man up until it hits the wall. The Pac-man cannot move any more and the points do not increase as well with new “up” command.
3. Move the Pac-man down until it hits the wall. The Pac-man cannot move any more and the points do not increase as well with new “down” command.
4. Move the Pac-man left until it hits the wall. The Pac-man cannot move any more and the points do not increase as well with new “left” command.
5. Move the Pac-man right until it hits the wall. The Pac-man cannot move any more and the points do not increase as well with new “right” command.
6. Randomly repeat step 2-5 to explore more walls in the play field until the end of the game.
Missing function : When walking forward and the left (or right) is the wall, turn left (or right), the face of Pac-man does not turn.

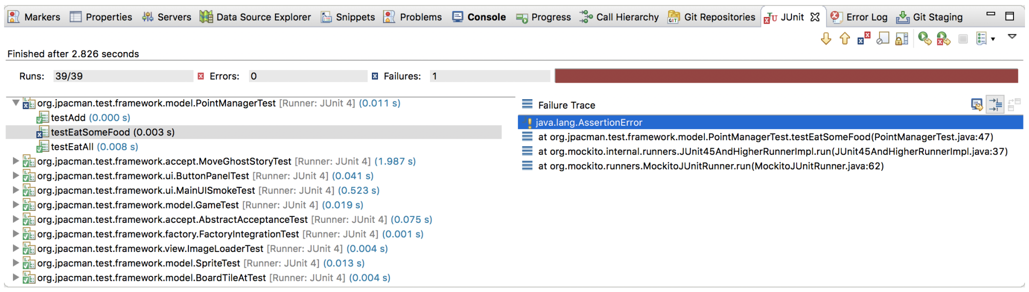
Test Charter 3 : Analyze the point manager in Jpacman
Actor : Normal User
Purpose : To evaluate if the point manager works fine in JPacman game.
Setup: A MacBook Pro with 8GB memory and 2.3 GHz Intel Core i5 processor, macOS high sierra operating system, eclipse IDE.
Priority : Median, because this is a function that will run through the whole game and will be used as the judgement of winning. But the point makes little sense when the player loses the game.
Data: JPacman-framework package
Activities/Tours :
1. Run JPacman as a java application. The point manager shows 0/1780.
2. Move the Pac-man up/down/right/left to eat the dots. The point manager shows points added by 10 for eating each dot.
3. Move the Pac-man up/down/right/left to the empty field. The point manager shows points staying the same.
4. When the Pac-man is caught by the ghost. it loses the game. The total points equal to the number of eaten dots times 10.
Missing function : No instruction on the point manager panel. For example, users do not know how many points they can get for each step without actually playing it.

Exercise 1.3 : Running the Test Suite

There are 10 test classes and 39 test cases.

I modified the `testEatSomeFood` in `org.jpacman.test.framework.model.PointManagerTest` class by changing the value of `pointeToEat` from `totalPoints/2` to `totalPoints`.

Then test case `testEatSomeFood` would fail and be indicated by Junit. The error message is ”java.long.AssertionError”.



```
Finished after 2.826 seconds

Runs: 39/39          Errors: 0          Failures: 1

▼ org.jpacman.test.framework.model.PointManagerTest [Runner: JUnit 4] (0.011 s)
      testAdd (0.000 s)
      testEatSomeFood (0.003 s)
      testEatAll (0.008 s)
▶ org.jpacman.test.framework.accept.MoveGhostStoryTest [Runner: JUnit 4] (1.987 s)
▶ org.jpacman.test.framework.ui.ButtonPanelTest [Runner: JUnit 4] (0.041 s)
▶ org.jpacman.test.framework.ui.MainUISmokeTest [Runner: JUnit 4] (0.523 s)
▶ org.jpacman.test.framework.model.GameTest [Runner: JUnit 4] (0.019 s)
▶ org.jpacman.test.framework.accept.AbstractAcceptanceTest [Runner: JUnit 4] (0.075 s)
▶ org.jpacman.test.framework.factory.FactoryIntegrationTest [Runner: JUnit 4] (0.001 s)
▶ org.jpacman.test.framework.view.ImageLoaderTest [Runner: JUnit 4] (0.004 s)
▶ org.jpacman.test.framework.model.SpriteTest [Runner: JUnit 4] (0.013 s)
▶ org.jpacman.test.framework.model.BoardTileAtTest [Runner: JUnit 4] (0.004 s)
```

```
Failure Trace
java.lang.AssertionError
at org.jpacman.test.framework.model.PointManagerTest.testEatSomeFood(PointManagerTest.java:47)
at org.mockito.internal.runners.JUnit45AndHigherRunnerImpl.run(JUnit45AndHigherRunnerImpl.java:37)
at org.mockito.runners.MockitoJUnitRunner.run(MockitoJUnitRunner.java:62)
```

Exercise 1.4

Look into MoveGhostStoryTest class that tests ghost movements.
Before the testing, the class run `setUp` to create a simpler map with player on the left of the ghost, player, etc with testing resources. And also create one emptyTile at the location (2,2) which is one step down to the ghost and one foodTile at the location (2,0) which is one step up to the ghost.

Four moving functions of ghost are tested here.
1. Test that a ghost can move towards an empty tile.
Make the ghost moves down one step. And check the tile that the ghost locates is empty or not, using `assertEquals`.
2. Test that a ghost can move over food.
Make the ghost moves up one step. And check the tile that the ghost locates is food or not, using `assertEquals`. Then check the Spritetype of the top sprite of this foodtile is ghost or not with `assertEquals`
3. Test that a ghost can leave the food once he's on it.
If the second test success, move the ghost down, i.e., move away from the foodtile. Check whether the Spritetype of food tile is food or not.
4. Test that the ghost causes the player's death if the ghost bumps into the player. Make the ghost moves left one step. Check the player is alive or not using `assertFalse`.

Improvement : Test class don't test that a ghost can't keep moving if the forward is wall. The name of test function is not good enough. Take the first one as an example. The name test_S3 _2_GhostFood is hard to understand what S3, 2 and GhostFood are.