Assignment 1: Setting the Stage

Exercise 1.2. Exploratory Testing

Charter Name Name: JPACMAN Landmark Testing
Time/date: Sept 19/18 10:00 am
Duration: 10 minutes
Actor/tester: A player
Target/Purpose: Ending the Game/ Complete Game
Data/Resources:

- Eclipse 4.7.2 on Macbook Pro (Retina, 15-inch, Late 2013)
- JPacman Maven Project

Activities and Test Notes:

- Dying in the game by walking into the ghost → Successfully ended the game by walking into the ghost on a tile without food at 60/1780 points. The message “You have lost :-(“ is displayed, and all buttons except for exit are disabled.
- Winning the game (collect all food) → Successfully completed with 1780/1780 points. The message “You hav won :-)” is displayed, and all buttons except for exit are disabled.
- Exiting the game by clicking on the exit button → Successfully closes the window and ends the game.

Missing functionality: The ability to restart the game after the game ends/is completed, without closing and reloading the game.

Exercise 1.3. Running the Test Suite

Number of test classes: 10
Total number of test cases (all tests in all classes):

- Dynamically **39** test cases run.
- Another class also runs AbstractAcceptanceTest.testInitialSetup
- BoardTileAtTest.testTileAtDirection is called 7 times.
- In total there are **31** unique test cases.

Test case to modify: SpriteTest.relocateSprite()
Actions taken:

- Before the assert statements, let the sprite John occupy another tile with the same coordinates as tile “north” instead of “north”

```java
/**
 * Test what happens if we relocate a sprite.
 */
@Test
public void relocateSprite() {
    john.deoccupy();
    Tile north = new Tile(0, 1);
    john.occupy(new Tile(0, 1));

    // john now lives at north
    assertThat(north.topSprite(), equalTo(john));

    // but john doesn't live in the center anymore.
    assertFalse(center.containsSprite(john));
}
```

Figure 1.3.1. Code Modified to make relocateSprite() fail with the changed code highlighted

Findings:

- The test case fails as a result of making John occupy a different tile object
- This implies that tiles are not defined by their coordinates, and that multiple instances of the same object can be created and not be equal (north does not have John occupying it, but a tile in the same location does)

Exercise 1.4.

Test Class: MoveGhostStoryTest.java

Purpose: This class tests for the correctness of the ghosts' movement behaviour.

Details of Implementation:

- The class sets up a basic game board with empty tiles and food tiles. A player and ghost are also placed in the board, as shown in figure 1.4.1

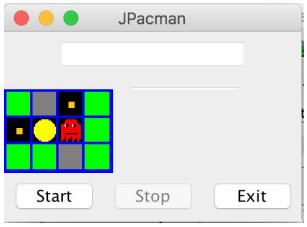

Figure 1.4.1. JPacman Test Game Board

- An empty tile and food tile is stored as a private variable in the class for reference in all the test cases.
- Unit tests are provided to determine if the ghost can move over empty tiles and tiles with food and appear on top of the tile.

- In the case of a food tile, a unit test is written to confirm that the ghost does not consume the food.
- There is a test to check if the ghost successfully kills the player if it moves into the player

Evaluation:

- The test cases provided are acceptable for checking the basic behaviour of the ghost
    - It covers each type of tile a ghost can move to
    - It covers the ghost killing the player
- The test cases can be easily broken by new test cases or a modified test game board
    - If any new test cases within the class modifies the value assigned to an empty tile or food tile, this may break other test cases
    - All test cases within this class expect a particular set up of the game board in the UI
        - All the tests in this class may break if the food tiles, empty tiles, and player were not located in the same tiles in figure 1.4.1.
        - If the board were to change dynamically, none of these test cases would hold.