Phase 3 Report

Findings

This phase of the project posed an interesting challenge of breaking down our code and thoroughly analysing how each class works, not only in isolation, but in conjunction with all the other classes and packages too. To ensure the quality of our tests we followed practices as discussed in lecture as well as utilizing Line and Branch coverage as rough estimates for our code quality. We utilized an external framework (i.e. Mockito), to aid in testing some of our more intricate integration tests. Throughout this phase of the assignment we came across one or two major bugs in our original code that were brought to our attention through our testing. Notably, we had a bug where the player/enemies could clip through barriers at certain positions, but we've since updated our code to fix this issue, thanks to our test suite. Followed in this report is a breakdown of our test suite as well as test evaluation in terms of quality and coverage.

Test Suite Overview

EnemyTest

- Unit Tests:
    - testEnemyConstructor() - tests that enemies are constructed properly by testing x and y positions on the map, and ensuring that all movement is enabled.
    - testGetBounds() - tests the bounds of our enemy as drawn on the display.
    - testCheckFrozen() - tests to ensure enemy checkFrozen() method returns proper values.
- Integration Tests:
    - testCollisionCanLeft() - tests whether or not an enemy can move to the left. Initially test if the enemy can move with no barriers around and then test when an enemy collides with a barrier on the left.
    - testCollisionCanRight() - tests whether or not an enemy can move to the right. Initially test if the enemy can move with no barriers around and then test when an enemy collides with a barrier on the right.
    - testCollisionCanUp() - tests whether or not an enemy can move up. Initially test if the enemy can move with no barriers around and then test when an enemy collides with a barrier above.
    - testCollisionCanDown() - tests whether or not an enemy can move down. Initially test if the enemy can move with no barriers around and then test when an enemy collides with a barrier below.
    - testChasingHero() - tests whether or not the enemy moves towards the player.

Phase 3 Report

HealthBarTest

- Unit Tests:
    - testHealthBarConstructor() - tests the health bar for correct x and y positions.

HealthTest

- Unit Tests:
    - testHealthConstructor() - tests that the Health class initializes the health bar array list with 3 points.

PlayerTest

- Unit Tests:
    - testPlayerConstructor() - tests that the player is constructed properly by testing x and y positions on the map, and ensuring that all movement is enabled.
    - testGetBounds() - tests the bounds of the player as drawn on the display.
    - testCurrentTile(): - tests to ensure returning correct player position relative to the boards tiles.
- Integration Tests:
    - testCollisionCanLeft() - tests whether or not the player can move to the left. Initially test if the player can move with no barriers around and then test when the player collides with a barrier on the left.
    - testCollisionCanRight() - tests whether or not the player can move to the right. Initially test if the player can move with no barriers around and then test when the player collides with a barrier on the right.
    - testCollisionCanUp() - tests whether or not the player can move up. Initially test if the player can move with no barriers around and then test when the player collides with a barrier above.
    - testCollisionCanDown() - tests whether or not the player can move down. Initially test if the player can move with no barriers around and then test when an enemy collides with a barrier below.

EndPointTest

- Integration Tests:
    - insufficientKeyCount() - tests to ensure that the Main thread does NOT stop running if the player enters the end point without collecting all the required keys.

- sufficientKeyCount() - tests to ensure that the Main thread DOES stop running if the player enters the end point after having collected all the required keys.

TestLevelTest

- Unit Tests:
  - addGrass() - tests that an empty tile is added onto the board and that the correct id is used for displaying the image.
  - setEndPoint() - tests that an end point is being set into the correct tile on the board.
- Integration Tests:
  - addBombs () - tests that bomb hazards are being added to the level correctly. Ensures that bombs are added in the correct tiles and that those tiles contain the corresponding bombs that were set in them. Checks that the Main class bombs list is populated accordingly.
  - addRewards () - tests that all reward types are being added to the level correctly. Ensures that rewards are added in the correct tiles and that those tiles contain the corresponding rewards that were set in them. Checks that the Main class rewards list is populated accordingly.

TileTest

- Unit Tests:
  - TileConstructor() - tests that tiles are being instantiated correctly. Ensures that tile x and y positions, contents and width and height are set correctly.
  - getBounds() - tests to ensure correct tile boundaries are being returned.
  - hasContents() - tests to ensure tiles can both be empty (contains *null* item) or contain an item of some class. Ensures check method returns appropriate Boolean value.
  - getContents() - tests to ensure that accessor returns correct object from tile.

HealthBombTest

- Unit Tests:
  - testHealthBombConstructor() – tests that a health bomb is being placed on the correct tile in the board
- Integration Tests:
  - onHit() – checks whether an instance of the health bomb is being properly created and placed and that the player is interacting correctly with it. It initializes a bomb, player and map. It has two scenarios one where the bomb and player are placed in the same tile and one where they are not. In the scenario of them being in the same tile the tests goes ahead to check whether

the health of the player was reduced while in the other scenario the test ensures that the health remains the same.

ScoreBombTest

- Unit Tests:
  - testScoreBombConstructor() - checks whether the score bomb is being placed in the correct tile.
- Integration Tests:
  - onHit() – checks the interaction between a player and a score bomb. Firstly, it initializes two sets of players and score bombs one with the case where they are not placed in the same tile and one in which they are The first case checks if the player's score stayed the same and the second case checks whether it reduced. If both cases perform as expected, the test is passed.

FreezeTest

- Unit Tests:
  - testFreezeConstructor() – tests that a freeze reward is placed correctly on a given tile of the board
- Integration Tests:
  - onHit() – initializes an instance of a freeze reward, player and test level to ensure that when both the player and the reward are placed in the same tile that the player's score increases.

HealthRewardTest

- Unit Tests:
  - testHealthRewardConstructor() – checks whether the health reward is being placed in the correct tile.
- Integration Tests:
  - onHit() – initializes a player, map, and a health reward to see whether placing an interaction between the player and the health reward leads to the player's health increasing. Therefore, if the player and the reward are placed in the same tile and the player's health increases, then the test is passed.

KeyRewardTest

- Unit Tests:
  - testKeyRewardConstructor() – tests that a key reward is placed correctly in the specified tile.
- Integration Tests:
  - onHit() – this test initializes a player, freeze reward and map to see whether placing a key reward in the same tile as the player leads to an increase in key count and score for the player. If the above requirements are met, then the test is passed.

Phase 3 Report

KeyTest

- Integration Tests:
  - testKeyPressedLeft() – passes a mockKeyEvent to simulate a left key press and asserts whether the Player is moving to the left.
  - testKeyPressedRight() - passes a mockKeyEvent to simulate a right key press and asserts whether the Player is moving to the right.
  - testKeyPressedUp() - passes a mockKeyEvent to simulate a up key press and asserts whether the Player is moving up.
  - testKeyPressedDown() - passes a mockKeyEvent to simulate a down key press and asserts whether the Player is moving down.
  - testKeyPressedPause() – passes a mockKeyEvent to simulate a 'p' key press and asserts whether the Main thread has paused.
  - testKeyReleasedLeft() – passes a mockKeyEvent to simulate releasing the left key after press and asserts whether the Player has stopped moving to the left.
  - testKeyReleasedRight() - passes a mockKeyEvent to simulate releasing the right key after press and asserts whether the Player has stopped moving to the right.
  - testKeyReleasedUp() - passes a mockKeyEvent to simulate releasing the up key after press and asserts whether the Player has stopped moving up.
  - testKeyReleasedDown() - passes a mockKeyEvent to simulate releasing the down key after press and asserts whether the Player has stopped moving down.

Test Quality and Coverage

Class	Line Coverage	Branch Coverage
Enemy	96%	84%
Health	88%	100%
HealthBar	100%	100%
KeyBar	100%	100%
Player	90%	60%
Score	100%	100%
EndPoint	100%	100%
TestLevel	100%	100%
Tile	85%	80%
Map	100%	100%

Phase 3 Report

Bomb	100%	100%
HealthBomb	100%	100%
ScoreBomb	100%	100%
Reward	100%	100%
Freeze	92%	0%
HealthReward	100%	100%
KeyReward	100%	100%
Key	100%	81%
Main	75%	50%

*Coverage calculated using Intellij IDEA coverage runner

Not Covered:

- **Enemy:**
  - update(): calls collision and checkFrozen functions, both of which are tested.
- **HealthBar:**
  - update(): function calls external function to redraw graphics.
- **KeyBar:**
  - update(): makes external function call to redraw graphics.
- **Player:**
  - setMovingLeft/Right/Up/Down(): function sets boolean values.
  - update(): makes external function call to redraw graphics.
- **Score:**
  - getPanel(): function simply returns a JPanel.
  - getComponent(): function simply returns a JLabel.
- **TestLevel:**
  - Constructor: makes calls to helper functions in TestLevel that have been fully tested.
  - update(): calls Tile.update() function which makes external function calls to redraw graphics.
- **Tile:**
  - update(): makes external function calls to redraw graphics.
- **Map:**
  - Constructor: runs a switch case and loads a manually inputted integer map of the level design.
  - getMap(); returns the manually inputted integer map.
- **HealthBombTest:**
  - update() – responsible for redrawing the graphics.
- **ScoreBombTest:**
  - update() – responsible for redrawing the graphics.
- **FreezeTest:**
  - update() – responsible for redrawing the graphics.

Phase 3 Report

- **HealthRewardTest:**
  - update() – responsible for redrawing the graphics.
- **KeyRewardTest:**
  - update() – responsible for redrawing the graphics.
- **KeyTest:**
  - keyTyped() –unused function.
- **Main:**
  - The Main class is our driver program which executes functions from all the other classes which are covered by our test suite. Aside from calls to our own functions the Main class also contains simple setter and getter functions as well as external functions which come from java libraries; therefore no specific tests were run for the Main class.