**Test Document**

**Test Case #1**

**Name:** Player spawning

**Type:** Unit (Player class)

**Purpose:** Make sure the player is correctly spawned when starting the game.

**Input:** Start game.Call method *draw()*.

**Results:** Player is drawn at the centre of the canvas.

**Test Case #2**

**Name:** Aiming

**Type:** Unit (Player class)

**Purpose:** Make sure the player aims towards the mouse cursor at all times.

**Input:** Move mouse. Using trigonometric equations from the Math class, determine the position of the player in correspondence to the cursor and call the method *draw()*.

**Results:** Player should follow the cursor.

**Test Case #3**

**Name:** Player Movement

**Type:** Unit (GameStage class)

**Purpose:** Make sure the player moves in accordance to the WASD keys.

**Input:** W, A, S, and D are the possible inputs.

**Results:** Player should move up if W is pressed, move down if S is pressed, move left if A is pressed, and move right if D is pressed.

**Test Case #4**

**Name:** Shooting

**Type:** Unit (GameStage class)

**Purpose:** Make sure the player shoots towards the mouse cursor when clicked.

**Input:** Click is the possible input. Gets the x and y coordinates using the event object.

**Results:** Creates a PBullet object, and draws it.

**Test Case #5**

**Name:** Saving path of Barrier

**Type:** Unit (GameStage class)

**Purpose:** Make a barrier that follows the path of the mouse when dragged.

**Input:** Drag is the possible input. Makes a new object of the Coordinates class that has x and y coordinates. Adds the object to an arrayList of coordinates.

**Results:** The path of the barrier should follow the path of the mouse when dragged.

**Test Case #6**

**Name:** Drawing Barrier

**Type:** Unit (Barrier class)

**Purpose:** Make sure the barrier drawn reflects the coordinates saved in the arrayList.

**Input:** Stop clicking the mouse.Set stroke colour to red using gc object. Stroke multiple small lines and join them using strokeLine.

**Results:** The barrier should be drawn on the same path that the mouse was dragged on.

**Test Case #7**

**Name:** Clearing Barrier

**Type:** Unit (Barrier class)

**Purpose:** Make sure the barrier is cleared when the *clear()* method is called.

**Input:** Time for barrier expires. Set stroke colour to white using gc object. Stroke multiple small lines and join them using strokeLine. Clear the arrayList of coordinates.

**Results:** The barrier should disappear from the canvas.

**Test Case #8**

**Name:** Advancing to the next level

**Type:** Unit (Boss class)

**Purpose:** Change the stage.

**Input:** Start a game and eliminate the boss.

**Results:** The colour of the boss is changed, all the existing boss bullets are removed, and the time and speeds are reset.

**Test Case #9**

**Name:** Random Movement of Boss

**Type:** Unit (Boss class)

**Purpose:** Make sure the boss moves randomly whenever the *move1()* method is called.

**Input:** Start level. When the thread is run a multiple of 100 times, the *move1()* method is called. *move1()* gives the xSpeed and ySpeed of the boss random values.

**Results:** The boss should move randomly every so often.

**Test Case #10**

**Name:** Attack 1

**Type:** Unit (Boss class)

**Purpose:** Make sure that the boss disperses balls in all directions.

**Input:** Start first level. When the thread is run a multiple of 100 times, the *atk1()* method is called. atk1() creates a Bullet object and gives it an xSpeed and ySpeed based on the angle that is passed to it.

**Results:** The first boss of the game should shoot balls in all directions.

**Test Case #11**

**Name:** Bullet spawning

**Type:** Unit (Bullet class)

**Purpose:** Make sure the bullet is correctly spawned when shooting.

**Input:** Start game. When mouse is clicked and existing bullets do not exceed 5, the bullet’s *draw()* method is called.

**Results:** The bullet should spawn in the direction of the cursor.

**Test Case #12**

**Name:** Random Movement of Boss

**Type:** Unit (Boss class)

**Purpose:** Make sure the boss moves randomly whenever the *move1()* method is called.

**Input:** Start level. When the thread is run a multiple of 100 times, the *move1()* method is called. *move1()* gives the xSpeed and ySpeed of the boss random values.

**Results:** The boss should move randomly every so often.

**Test Case #13**

**Name:** Decrease Boss health

**Type:** Integration (Health, Boss, and GameStage class)

**Purpose:** Make sure the boss loses health when hit by the player’s bullet.

**Input:** Start level. Hit the boss with one bullet.

**Results:** The boss should lose 10 health.

**Test Case #14**

**Name:** Eliminate boss bullets that come into contact with the barrier.

**Type:** Integration (Barrier, Boss, and GameStage class)

**Purpose:** Make sure the boss’s bullets are removed when they touch the barrier.

**Input:** Start level. Draw barrier. Boss’s bullet hits barrier.

**Results:** The boss’s bullet should disappear and be removed from the arrayList.

**Test Case #15**

**Name:** New Game

**Type:** Integration (Player, Boss, Barrier, GameStage, Bullet, and PBullet class)

**Purpose:** Make sure the player, boss, bullets, and barriers are functional when starting the game.

**Input:** Run game, Click “New Game”.

**Results:** The game should start and the objects should perform as outlined.