



Spring-2017

Software Methods and Tools

Assignment-6

(JUnit)

Submitted by:

Moulika Chadalavada

16234180

## 1. Objective

The main aim of this assignment is to perform unit testing of Snake Game using JUnit. Different test cases are tested in JUnit to find out any error or bugs in the code. As part of this assignment testing class is created for Snake Game and tested below methods

```
private void updateGame()
```

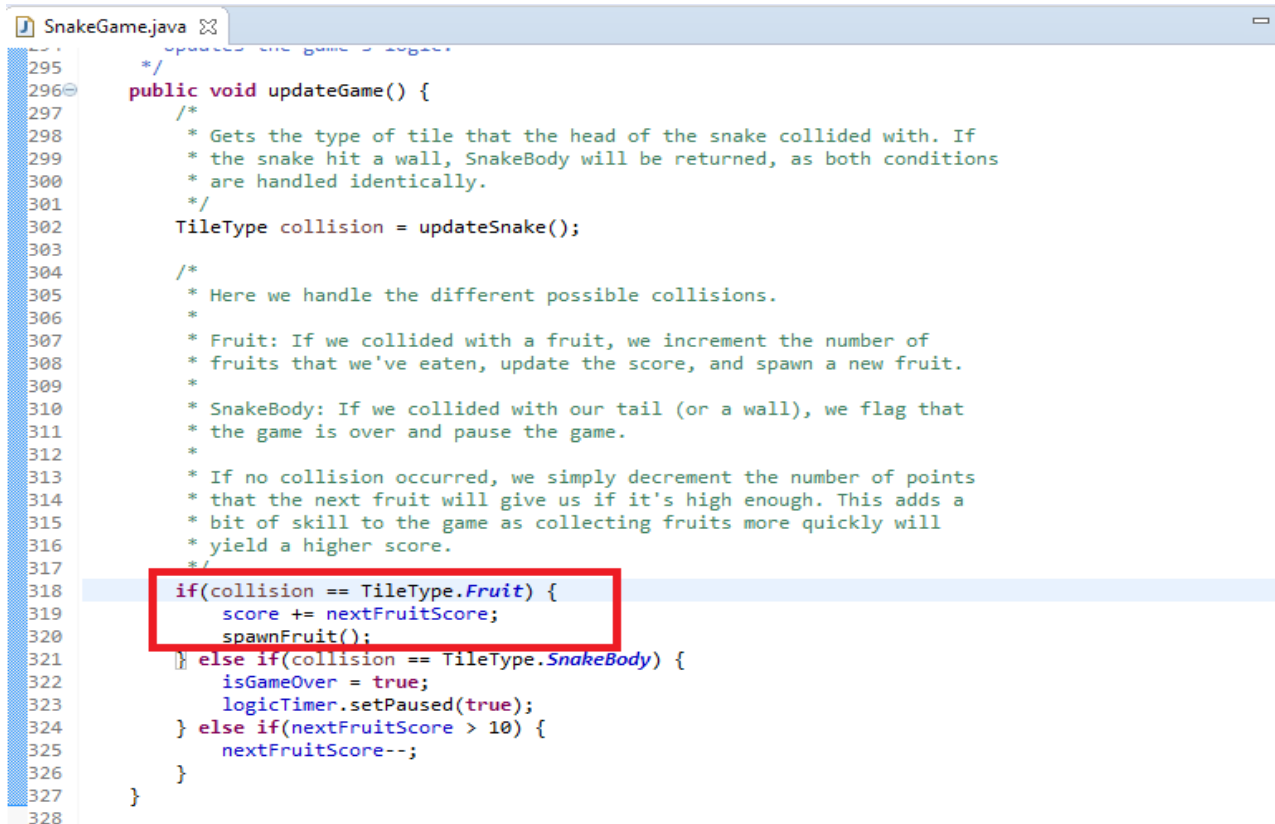
```
private void spawnFruit()
```

```
private TileType updateSnake()
```

In further sections, it briefly explained how the test cases are developed to find the error. Where exactly the error is present in the code.

## 2. Problem found in Code

When Snake Game code is executed I found an error where **Fruit Eaten** count is not incrementing even though snake hit the food. In **updateGame()** method logic for incrementing fruit eaten count is missing as shown in below image. So, further JUnit test cases are used to find the problem. The Problem is found out in Test Case [TC 003](#)



```
SnakeGame.java
295  /*
296  public void updateGame() {
297      /*
298       * Gets the type of tile that the head of the snake collided with. If
299       * the snake hit a wall, SnakeBody will be returned, as both conditions
300       * are handled identically.
301       */
302      TileType collision = updateSnake();
303
304      /*
305       * Here we handle the different possible collisions.
306       *
307       * Fruit: If we collided with a fruit, we increment the number of
308       * fruits that we've eaten, update the score, and spawn a new fruit.
309       *
310       * SnakeBody: If we collided with our tail (or a wall), we flag that
311       * the game is over and pause the game.
312       *
313       * If no collision occurred, we simply decrement the number of points
314       * that the next fruit will give us if it's high enough. This adds a
315       * bit of skill to the game as collecting fruits more quickly will
316       * yield a higher score.
317       */
318      if(collision == TileType.Fruit) {
319          score += nextFruitScore;
320          spawnFruit();
321      } else if(collision == TileType.SnakeBody) {
322          isGameOver = true;
323          logicTimer.setPaused(true);
324      } else if(nextFruitScore > 10) {
325          nextFruitScore--;
326      }
327  }
328
```

### 3. Snake Game Test Cases

I have created below test cases out of which only one is failing. Every test case and its execution explained in detailed further.

Test Case No.	Test Case Name	Test Case Description	Methods Invoked	Pass/Fail
<a href="#">TC_001</a>	Increase Score when Snake Hits Food	In this test case, when Snake hits Food Then Score must be increased based On NextFruitScore	SnakeGame.java -> updateGame() spawnFruit()	Pass
<a href="#">TC_002</a>	Decrease NextFruitScore when Game starts Updating	In this test case we are checking whether NextFruitScore incrementing to 100 when spawnFruit is called and also checking if NextFruitScore is decrementing when game updates.	SnakeGame.java -> spawnFruit() updateGame()	Pass
<a href="#">TC_003</a>	Increase Fruit Eaten count when Snake hits Food	In this test case, when Snake hits Food then fruits eaten count is incremented by '1' for each Fruit.	SnakeGame.java -> updateGame() spawnFruit()	Fail
<a href="#">TC_004</a>	Pause Snake Game when Key Pressed is P	Initially isPaused value is False but when we set KeyPressed value to P then isPaused should be True	SnakeGame.java -> updateGame() isPaused()	Pass
<a href="#">TC_005</a>	Increase Snake Size when Snake Hits Food	In this test case, when Snake hits Food Then size of snake has to be increased	SnakeGame.java -> updateSnake()	Pass
<a href="#">TC_006</a>	Game Over when Snake Hits Wall or SnakeBody	In this test case, we set the attributes such that the Snake hits the Wall, so in updateGame() method the GameOver is set to True.	SnakeGame.java -> updateSnake() updateGame()	Pass

Among these test cases only one test case is Failed i.e. [TC\\_003](#) in which FruitEaten count is not getting incremented when Snake hits Fruit. This test is explained in detail in further sections.

Before proceeding with the test case development and execution, below code level changes are done in SnakeGame.java.

1. Changes the visibility of Private Methods to Public.
2. Created initSnakeGame() method because startGame() will directly execute the game instead of running test cases.

```
// 1.1vc starts
public void initSnakeGame() {
    this.random = new Random();
    this.snake = new LinkedList<>();
    this.directions = new LinkedList<>();
    this.logicTimer = new Clock(9.0f);
    this.isNewGame = true;

    logicTimer.setPaused(true);
}
// 1.1vc ends
```

### 3.1 TC\_001: Increase Score when Snake Hits Food

As part of functionality when Snake Hits Fruit the Score must be incremented based on NextFruitScore. In this test cases, initially I am fetching the position of Fruit and setting the same position to SnakeHead such that it hits the Fruit.

Once the positions are set successfully I am updating game using updateGame() method in which updateSnake() method is called internally and returns TileType as Fruit, such that updateGame() has to update Score. I have set game such that snake eats '5' fruits. This test case is passed as oldScore is not equal to actualScore.

Test Case Description	Expected Output	Actual Output	Result
Check if Score is updating when Snake Hits Fruit 5 times.	! (oldScore = 0)	actualScore = 500	Pass

**Following is code for executing test case:**

```
@Test
/*
 * This test case checks is the Score is getting incremented if Snake hits
 * Fruit
 */
public void testUpdateScore() {

    int fruitcnt = 5;

    /*
     * First we are getting the current Fruit Position by executing
     * spawnFruit()
     */
    int oldScore = snakeGame.getScore();

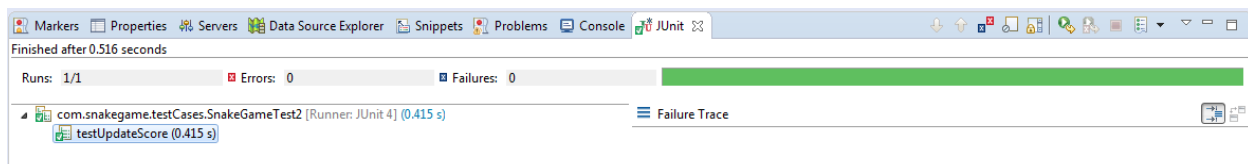
    for (int i = 0; i < fruitcnt; i++) {
        snakeGame.spawnFruit();
        TileType fruitTileType = snakeGame.board.getTile(snakeGame.fruitTilePositionX,
            snakeGame.fruitTilePositionY);
        System.out.println(fruitTileType);

        /*
         * Further we are setting the coordinates of Fruit to SnakeHead
         * position Such that SnakeHead position is equal to Fruit Position
         */
        head = new Point(snakeGame.fruitTilePositionX, ++snakeGame.fruitTilePositionY);
        snakeGame.snake.add(head);
        snakeGame.directions.add(Direction.North);

        /*
         * Once the position of Snake is update we are executing
         * updateGame() and further updateSnake() should return TileType as
         * Fruit And in updateGame Score count has to be incremented based
         * on NextFruitScore.
         */
        snakeGame.updateGame();
        actualScore = snakeGame.getScore();
        System.out.println(snakeGame.updateSnake()); // returns Fruit
    }

    System.out.println(snakeGame.getScore()); // Returns 500 as Snake ate 5
                                                // Fruits

    assertEquals(oldScore, actualScore); // 0 not equal to 500
}
```

**Test Case Output:**

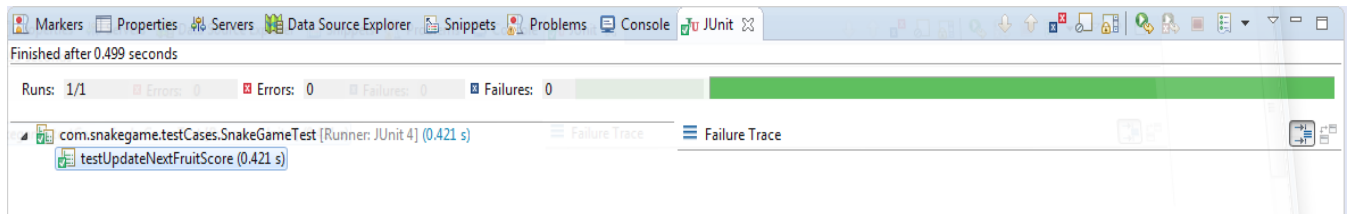
### 3.2 TC\_002: Decrease NextFruitScore when Game Starts Updating

As per functionality initially the NextFruitScore is '0' when spawnFruit() is executed it has to be increase to 100. Also in spawnFruit() based on coordinated set for Fruit the TileType should be Fruit. So as nextFruitScore is updated to 100 and TileType is returning as Fruit so this case is passed.

Also in this test case I am updating game (updateGame()) 10 times such that NextFruitScore should decrement 10 times. As the ActualFruitScore and ExpectedFruitScore is 90 test case is passed.

Test Case Description	Expected Output	Actual Output	Result
Check if TileType returned is Fruit after setting Fruit coordinates in spawnFruit()	TileType = Fruit	TileType = Fruit	Pass
Check if Fruit coordinates position is Less than Board dimension (25 x 25)	COL_CNT + 1 ROW_CNT + 1	X_FruitPosition < 26 Y_FruitPosition < 26	Pass
Check if NextFruitScore is decremented 10 times when updateGame() executed 10 times.	NextFruitScore = 90	NextFruitScore = 90	Pass

#### Test Case Output:



**Following is code for executing test case:**

```

@Test
/*
 * This test case executes spawnFruit() method and checking whether
 * NextFruitScore is updating from 0 to 100 and also checking if TileType is
 * getting set to Fruit for given Position. Further we are also checking if
 * NextFruitScore is getting decremented when snake game is getting updated
 * '10' times
 */
public void testUpdateNextFruitScore() {

    snakeGame.resetGame();

    int X_Board_Col_Cnt = snakeGame.board.COL_COUNT + 1;
    int Y_Board_Row_Cnt = snakeGame.board.ROW_COUNT + 1;

    /*
     * spawnFruit() method is called such that the nextFruitScore has to be
     * updated to 100 and TileType has to be set to Fruit
     */
    snakeGame.spawnFruit();

    /*
     * Getting Fruit Position and TileType
     */
    int X_FruitPosition = snakeGame.fruitTilePositionX;
    int Y_FruitPosition = snakeGame.fruitTilePositionY;

    TileType fruitTileType = snakeGame.board.getTile(X_FruitPosition, Y_FruitPosition);

    /*
     * Checking whether the coordinate positions set is greater than Board
     * Dimensions
     */
    assertEquals(X_Board_Col_Cnt, X_FruitPosition);
    assertEquals(Y_Board_Row_Cnt, Y_FruitPosition);

    assertEquals(TileType.Fruit, fruitTileType); // fruitTileType should return Fruit

    /*
     * Updating Game 'n' number of times i.e using updateGame() method As
     * currentFruitScore > 10 the actualScore should be decremented n times
     */

    int expectedFruitScore = snakeGame.getNextFruitScore() - 10;

    for (int i = 0; i < 10; i++) {
        snakeGame.updateGame();
    }
    int actualFruitScore = snakeGame.getNextFruitScore(); // must return 90

    assertEquals(expectedFruitScore, actualFruitScore); // Both has to be 90 as game updated 10 times
}

```

### 3.3 TC\_003: Increase Fruit Eaten count when Snake Hits Food

As part of functionality when Snake Hits Fruit the FruitEaten count must be incremented. In this test cases, initially I am fetching the position of Fruit and setting the same position to SnakeHead such that it hits the Fruit. Once the positions are set successfully I am updating game using updateGame() method in which updateSnake() method is called internally and returns TileType as Fruit, such that updateGame() has to update FruitsEaten count. I have set game such that snake eats '5' fruits. But this test case is failed as expectedFruitsEaten is not equal to actualFruitsEaten, as FruitsEaten score is not getting incremented in updateGame().

Test Case Description	Expected Output	Actual Output	Result
Check if FruitEaten count is increasing When Snake hits 5 Fruits.	expectedFruitsEaten= 5	actualFruitsEaten= 0	Fail

Following is code for executing test case:

```
@Test
/*
 * This test case checks is the Fruits Eaten count is getting incremented if
 * Snake hits Fruit
 */
public void testUpdateFruitEaten() {

    int actualFruitsEaten = 0;
    int fruitcnt = 5;

    int expectedFruitsEaten = snakeGame.getFruitsEaten() + fruitcnt; // fruitcnt=5
    /*
     * First we are getting the current Fruit Position by executing
     * spawnFruit()
     */
    for (int i = 0; i < fruitcnt; i++) {

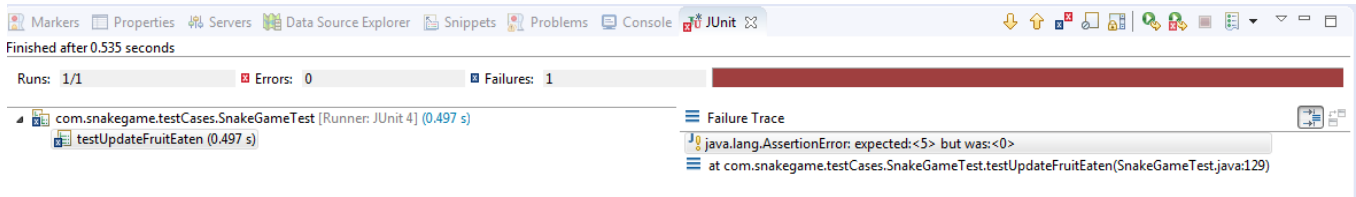
        snakeGame.spawnFruit();

        /*
         * Further we are setting the coordinates of Fruit to SnakeHead
         * position Such that SnakeHead position is equal to Fruit Position
         */
        head = new Point(snakeGame.fruitTilePositionX, snakeGame.fruitTilePositionY);
        snakeGame.snake.add(head);
        snakeGame.directions.add(Direction.North);
        snakeGame.board.setTile(head, TileType.Fruit);

        /*
         * Once the position of Snake is update we are executing
         * updateGame() and further updateSnake() should return TileType as
         * Fruit And in updateGame FruitEaten count has to be incremented.
         */
        snakeGame.updateGame();
        actualFruitsEaten = snakeGame.fruitsEaten;
        System.out.println(snakeGame.updateSnake()); // returns Fruit
    }

    assertEquals(expectedFruitsEaten, actualFruitsEaten); // Fails as count not increasing
}
```



**Test Case Output:****3.4 TC\_004: Pause Snake Game when Key Pressed is P**

As per logic of the game the game should Pause at any point of time when KeyPressed by player is 'P'. So, in this test case initially when the game is started isPaused is 'False' as game starts executing, further I am setting KeyPressed to 'P' such that same has to be paused (isPaused = True). As isPaused value is updated to True this test case is passed.

Test Case Description	Expected Output	Actual Output	Result
Check if Game is getting Paused when KeyPressed is set to True	True	True	Pass

**Following is code for executing test case:**

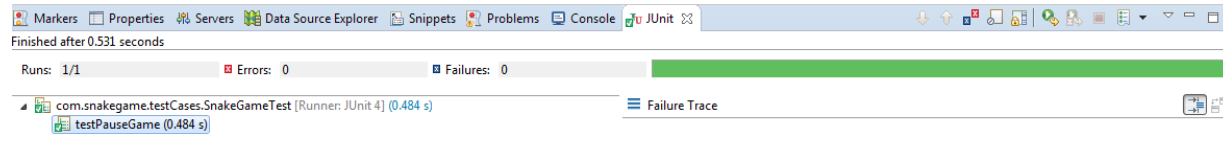
```

@Test
/*
 * This test case checks whether the Game is paused when Key Pressed = 'P'
 */
public void testPauseGame() {
    head = new Point(BoardPanel.COL_COUNT / 2, BoardPanel.ROW_COUNT / 2);
    snakeGame.snake.add(head);
    snakeGame.board.setTile(head, TileType.SnakeHead);
    snakeGame.directions.add(Direction.North);

    System.out.println(snakeGame.isPaused());
    snakeGame.setFocusable(true);
    snakeGame.requestFocus();
    snakeGame.addKeyListener(new KeyAdapter() {
        @Override
        public void keyPressed(KeyEvent e) {
            e.setKeyCode(KeyEvent.VK_P);
            if (e.getKeyCode() == KeyEvent.VK_P) {
                isPaused = snakeGame.isPaused();
            }
        }
    });
    snakeGame.updateGame();

    assertTrue(isPaused); // should return True
}

```

**Test Case Output:****3.5 TC\_005: Increase Snake Size when Snake Hits Food**

As per functionality the snake size must increase as the game starts update when hits Fruit. In this test-case initially I am setting the default size of snake to '6', next I am setting default positions of Fruit using spawnFruit() and setting coordinates such that Snake hits Fruit and Snake size incremented by 1. As expected size of snake equals to actual size this test case is passed.

Test Case Description	Expected Output	Actual Output	Result
Check if Fruit Size increases if Snake hits Fruits	sizeOld+1 = 7 (sizeOld = 6)	sizeNew = 7	Pass

**Following is code for executing test case:**

```

@Test
/*
 * This test case is used to check if Snake size increases as Snake hits
 * Fruit
 */
public void testUpdateSnakeSize() {

    int sizeOld=0;

    /*
     * Start Updating The Snake such that Snake default size is set to 6
     */
    for (int i = 0; i < 5; i++) {
        snakeGame.updateGame();
        sizeOld = snakeGame.snake.size();
    }

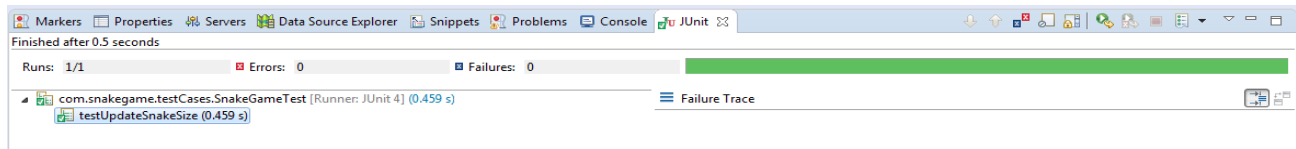
    snakeGame.spawnFruit();

    /*
     * Further we are setting the coordinates of Fruit to SnakeHead position
     * Such that SnakeHead position is equal to Fruit Position
     */
    head = new Point(snakeGame.fruitTilePositionX, ++snakeGame.fruitTilePositionY);
    snakeGame.snake.add(head);
    snakeGame.directions.add(Direction.North);

    /*
     * Once the position of Snake is update we are executing updateGame()
     * and further updateSnake() should return TileType as Fruit And in
     * updateGame Score count has to be incremented based on NextFruitScore.
     */
    snakeGame.updateGame();
    snakeGame.updateSnake();// returns Fruit
    sizeNew = snakeGame.snake.size();

    /*
     * We are checking whether Snake Size is Increasing
     */
    assertEquals(sizeOld + 1, sizeNew);
}

```

**Test Case Output:****3.6 TC\_006: Game Over when Snake Hits Wall or SnakeBody**

As part of this test case I have setup default elements for Snake, and updated Game. Further added Direction to South such that the Snake hits Wall and the TileType is updated to SnakeBody. According to Game logic the Game should over when collision is SnakeBody.

Test Case Description	Expected Output	Actual Output	Result
In this test case, we set the attributes such that the Snake hits the Wall, so in updateGame() method the GameOver is set to True.	TileType = SnakeBody	TileType=SnakeBody	Pass
	GameOver = true	GameOver = true	Pass

Following is code for executing test case:

```

@Test
/*
 * This test case checks if the Game is getting ended if Snake hits Wall or
 * SnakeBody
 */
public void testUpdateSnakeCollision() {

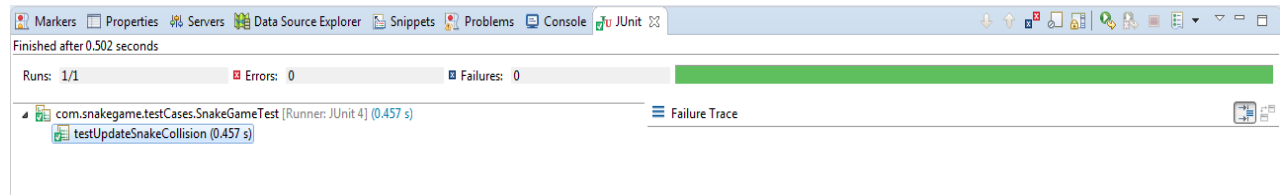
    /*
     * Here we are setting head coordinates such that it is greater than
     * Board coordinates so once execute updateGame() updateSnake() should
     * return SnakeBody
     */
    head = new Point(25, 25);
    snakeGame.snake.add(head);
    snakeGame.directions.add(Direction.North);
    snakeGame.updateGame();

    assertEquals(TileType.SnakeBody, snakeGame.updateSnake());

    /*
     * As TileType is SnakeBody the GameOver should be True
     */

    assertTrue(snakeGame.isGameOver());
}

```

**Test Case Output:****4. Functionalities Tested**

As discussed the main methods that I testes as `updateGame()` , `updateSnake()` , `spawnFruit()`

1. **spawnFruit ()**: This method basically sets the fruits position at any point of time.

**Functionalities Tested:**

- I have checked whether the TileType getting set in this method is 'Fruit'. ([TC\\_002](#))
- I have checked whether coordinated of Fruit is less than Board dimensions. ([TC\\_002](#))

2. **updateSnake()**: This method sets the TileType (Fruit/SnakeBody/SnakeHead) based on which the Game is updated.

**Functionalities Tested:**

- I checked whether this method is returning SnakeBody when Snake Head coordinates are set to Board dimensions (25 x 25) ([TC\\_006](#))
- I checked if TileType is set to Fruit when Snake hits Fruit ([TC\\_002](#))
- I checked if Snake size is getting incremented when it hits Fruit ([TC\\_005](#))

3. **updateGame()**: This method updates the Game which internally calls `updateSnake()` method.

**Functionalities Tested:**

- I checked if Score is getting incremented when Snake Hits Fruit i.e. TileType is Fruits. Also Score increments based on NextFruitScore. ([TC\\_001](#))
- I checked if NextFruitScore is decrementing when game is getting updates i.e. Snake Moves ([TC\\_002](#))
- I checked if FruitEaten count is incrementing when Snake Hits Fruit i.e. TileType is Fruits. ([TC\\_003](#))
- I checked if Game is paused when Key Pressed is 'P'. ([TC\\_004](#))
- I checked if GameOver is True when TileType is SnakeBody. ([TC\\_006](#))

## 5. Screenshot of the result after running Test Suite

Below is the screenshot for the Snake Game JUnit Test Suite that displays all the test methods that are written as part of this Assignment and it also shows that test method **testUpdateFruitEaten** is failed as FruitEaten count is not getting incremented.

