



## Software Quality Assignment 2

Connect 4 - Game Development

Allan Santosh – 100557518

## Introduction

The game I chose to do for this assignment is Connect 4. This is a two-player game where the objective of the game is to be the first person to connect four of the same pieces horizontally, vertically or diagonally. The game starts by inputting player names first and then player one gets to drop the first piece in the grid followed by the second player. Each player plays turn by turn until one of them can get four of the same pieces together and then the game terminates and declares the winner.

### Sample Output

```
|   Connect 4   |
|       |       |       |       |       |
|       |       |       |       |       |
|       |       |       |       |       |
|       |       |       |       |       |
| 1   2   3   4   5   6  |
Welcome to Connect 4. Hope you enjoy the game.

Enter player 1 name : Batman
Enter player 2 name : Superman

Batman, your symbol is '0'. Enter column you want to mark : 1

|   Connect 4   |
|       |       |       |       |       |
|       |       |       |       |       |
|       |       |       |       |       |
|       |       |       |       |       |
| 0   |       |       |       |       |
| 1   2   3   4   5   6  |

Superman, your symbol is 'X'. Enter column you want to mark : 3

|   Connect 4   |
|       |       |       |       |       |
|       |       |       |       |       |
|       |       |       |       |       |
|       |       |       |       |       |
| 0   |       |       |       |       |
| 0   |       |       |       |       |
| 0   |       |       |       |       |
| 0   |       |       |       |       |
| 1   2   3   4   5   6  |

Batman, your symbol is '0'. Enter column you want to mark : 1

|   Connect 4   |
|       |       |       |       |       |
|       |       |       |       |       |
|       |       |       |       |       |
|       |       |       |       |       |
| 0   |       |       |       |       |
| 0   |       |       |       |       |
| 0   |       |       |       |       |
| 0   |       |       |       |       |
| 1   2   3   4   5   6  |

Superman, your symbol is 'X'. Enter column you want to mark : 4

|   Connect 4   |
|       |       |       |       |       |
|       |       |       |       |       |
|       |       |       |       |       |
|       |       |       |       |       |
| 0   |       |       |       |       |
| 0   |       |       |       |       |
| 0   |       |       |       |       |
| 0   |       |       |       |       |
| 1   2   3   4   5   6  |

Batman, your symbol is '0'. Enter column you want to mark : 1

|   Connect 4   |
|       |       |       |       |       |
|       |       |       |       |       |
|       |       |       |       |       |
|       |       |       |       |       |
| 0   |       |       |       |       |
| 0   |       |       |       |       |
| 0   |       |       |       |       |
| 0   |       |       |       |       |
| 1   2   3   4   5   6  |

Superman, your symbol is 'X'. Enter column you want to mark : 3

|   Connect 4   |
|       |       |       |       |       |
|       |       |       |       |       |
|       |       |       |       |       |
|       |       |       |       |       |
| 0   |       |       |       |       |
| 0   |       |       |       |       |
| 0   |       |       |       |       |
| 0   |       |       |       |       |
| 1   2   3   4   5   6  |

Batman, your symbol is '0'. Enter column you want to mark : 1

|   Connect 4   |
|       |       |       |       |       |
|       |       |       |       |       |
|       |       |       |       |       |
|       |       |       |       |       |
| 0   |       |       |       |       |
| 0   |       |       |       |       |
| 0   |       |       |       |       |
| 0   |       |       |       |       |
| 1   2   3   4   5   6  |

Batman has won the game !!!!
```

## Software Development Process

For this game, I decided to follow the agile development. This project was completed successfully in the time given. The reason I chose to follow the agile process is because during the coding phase my requirements changed. I was able to adapt the new changes and make it reflect on the code instantly. Due to the drawbacks of waterfall and incremental which doesn't allow me to do revisions on my work; and also, since the time frame given for this assignment was long, it doesn't fit the waterfall model and I decided to choose the agile development process.

Agile development is the best as it always allows for room for change. Especially games are always better with minor changes added to it, agile development is the way to go.

## Functionality

The simple command line interface game of connect 4 doesn't need additional libraries to be loaded. All the source files that are compiled can be found under the 'bin' folder. Since the program is written in Java, Java needs to be installed on the system. The testing is all done with Junit which was installed onto the Eclipse IDE.

## Challenges

One of the challenges I had with testing is the code itself. I didn't make object of different game instances. Instead I only made one static game board which didn't allow me to see what happened when I run methods that were meant for objects.

Finding test cases that had most coverage was hard. Considering if this project were to be expanded to a bigger grid, then a different test method needs to be implemented.

## How to Install

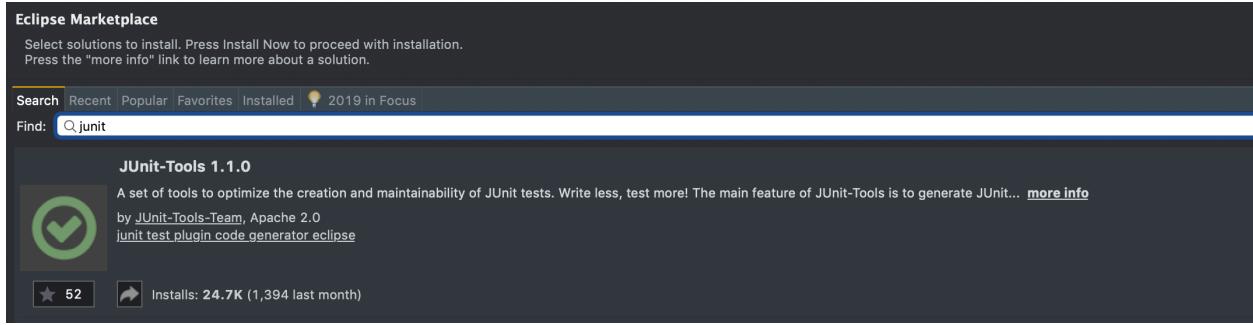
Step 1: Install Eclipse IDE from <https://www.eclipse.org/downloads/>

Step 2: Make a new folder in desktop and clone the GitHub repo using the command

```
git clone https://github.com/allansantosh/Connect-4
```

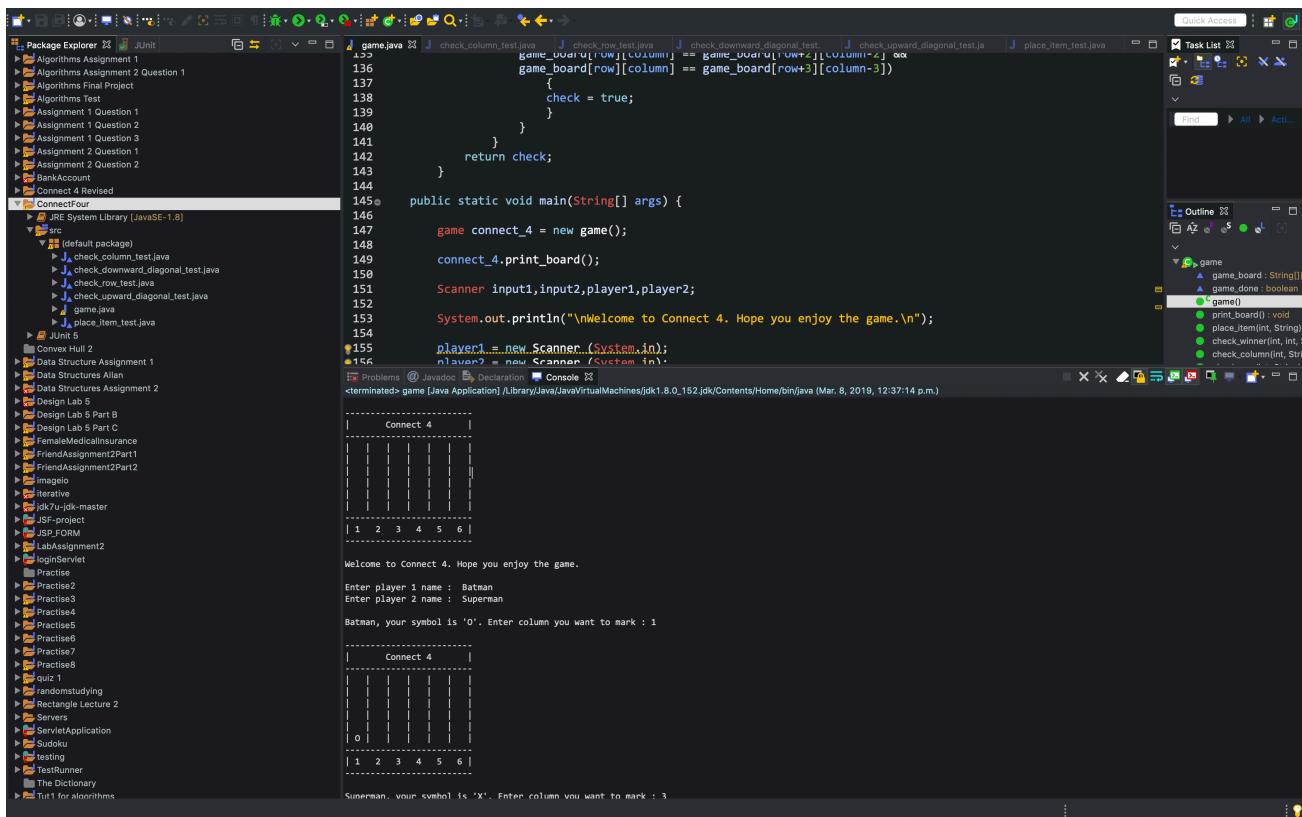
Step 3: Open Eclipse IDE. Press Help and open up Eclipse Market Place.

Step 4: Install Junit from Eclipse Market Place



Step 5: Open File → New Project → point to the new folder in the desktop where you cloned the GitHub Repo.

Step 6: The project should open as follows.



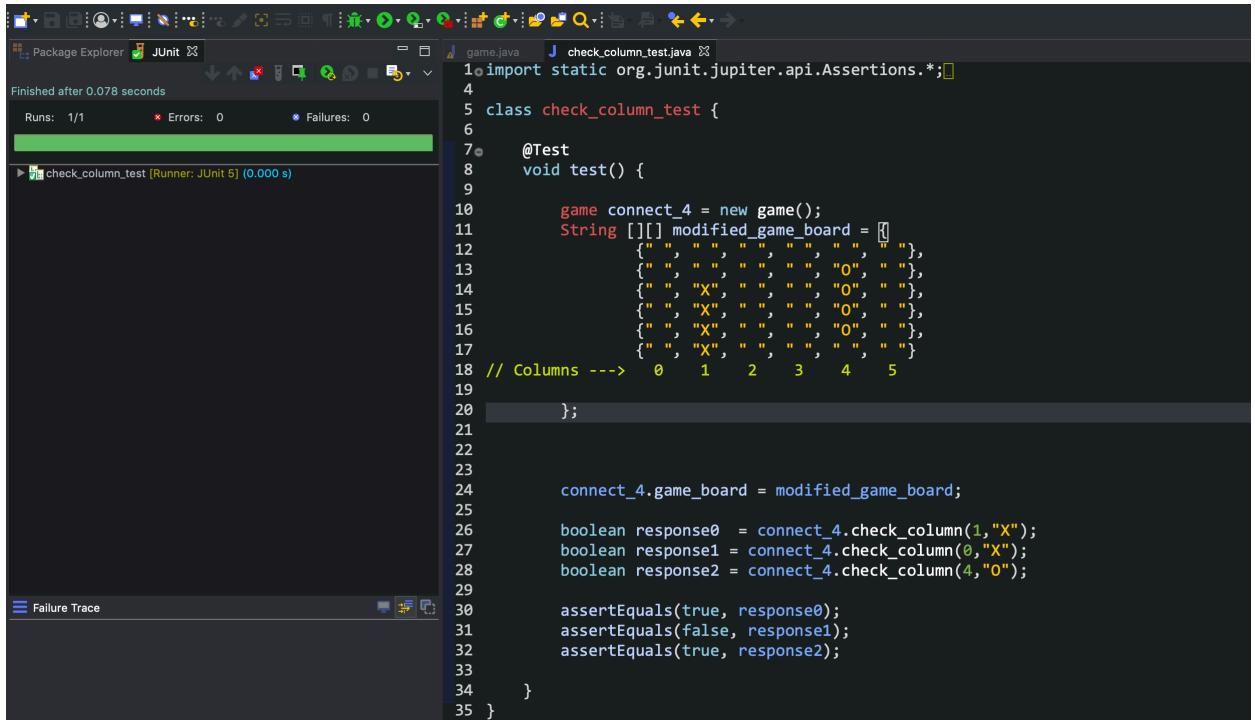
Step 7: Press the green run button to compile and run the game and play the game.

Step 8: In the project explorer, the test cases are also available. Double click on the test case java files and press the green button to run the test cases.

## Running test cases

*Note the left side of the images shows if the tests passed or failed.*

## 1. Checking columns if there is 4 of a kind.

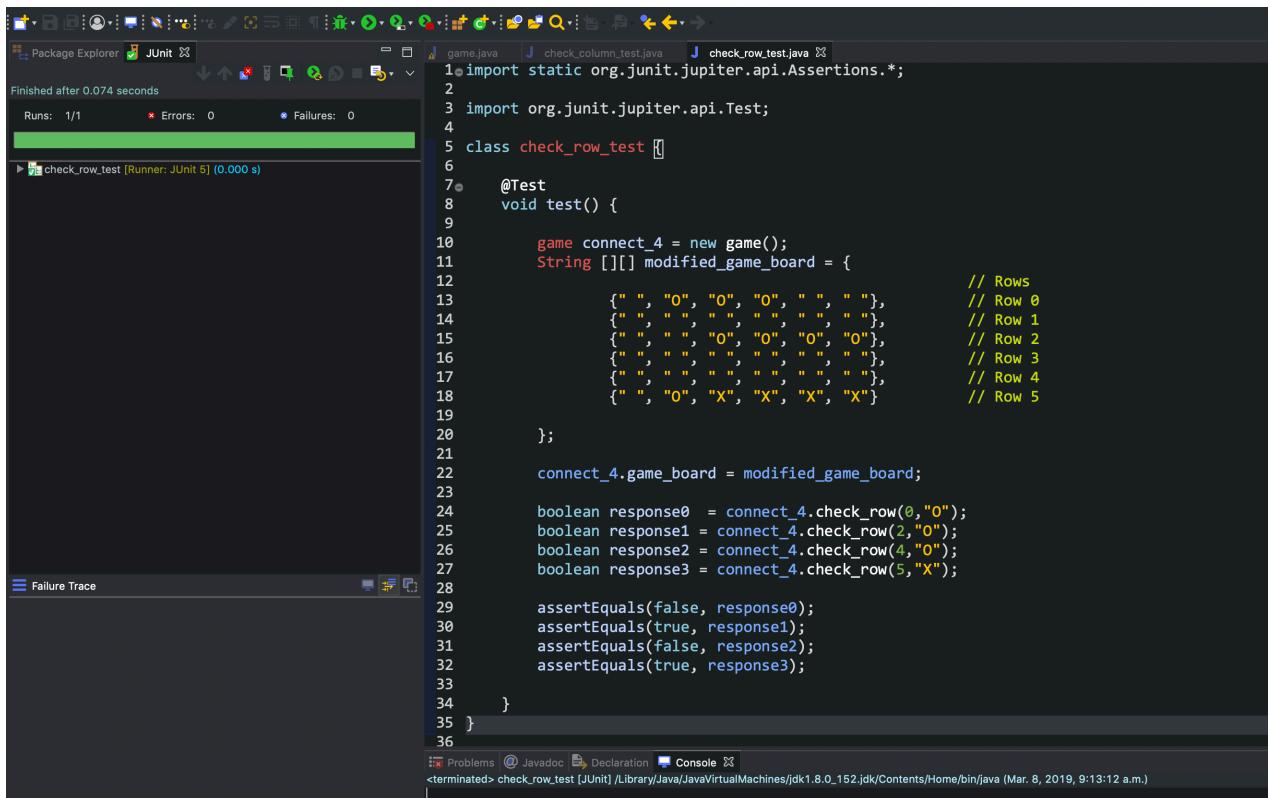


The screenshot shows the Eclipse IDE interface with the JUnit perspective selected. The top status bar indicates "Finished after 0.078 seconds". The left sidebar shows the "Package Explorer" and "JUnit" view, which lists the test class "check\_column\_test". The main editor area contains the following Java code:

```
1 import static org.junit.jupiter.api.Assertions.*;
2
3 class check_column_test {
4
5     @Test
6     void test() {
7
8         game connect_4 = new game();
9         String [][] modified_game_board = [
10             {" ", " ", " ", " ", " ", " "},
11             {" ", " ", " ", " ", "0", " "},
12             {" ", "X", " ", " ", "0", " "},
13             {" ", "X", " ", " ", "0", " "},
14             {" ", "X", " ", " ", "0", " "},
15             {" ", "X", " ", " ", "0", " "}
16         ]; // Columns ----> 0 1 2 3 4 5
17
18         connect_4.game_board = modified_game_board;
19
20         boolean response0 = connect_4.check_column(1, "X");
21         boolean response1 = connect_4.check_column(0, "X");
22         boolean response2 = connect_4.check_column(4, "0");
23
24         assertEquals(true, response0);
25         assertEquals(false, response1);
26         assertEquals(true, response2);
27     }
28
29 }
30
31
32
33
34 }
```

The code defines a test method named "test" that creates a new instance of the "game" class and sets its "game\_board" to a modified version of the original board. The modified board has a sequence of four 'X's in column 1 and four '0's in column 4. The test then checks the results of calling the "check\_column" method for each column, asserting that it returns true for column 1 ('X') and column 4 ('0'), and false for columns 0 ('X') and 2-3 (' ').

## 2. Checking if the rows had 4 of a kind.

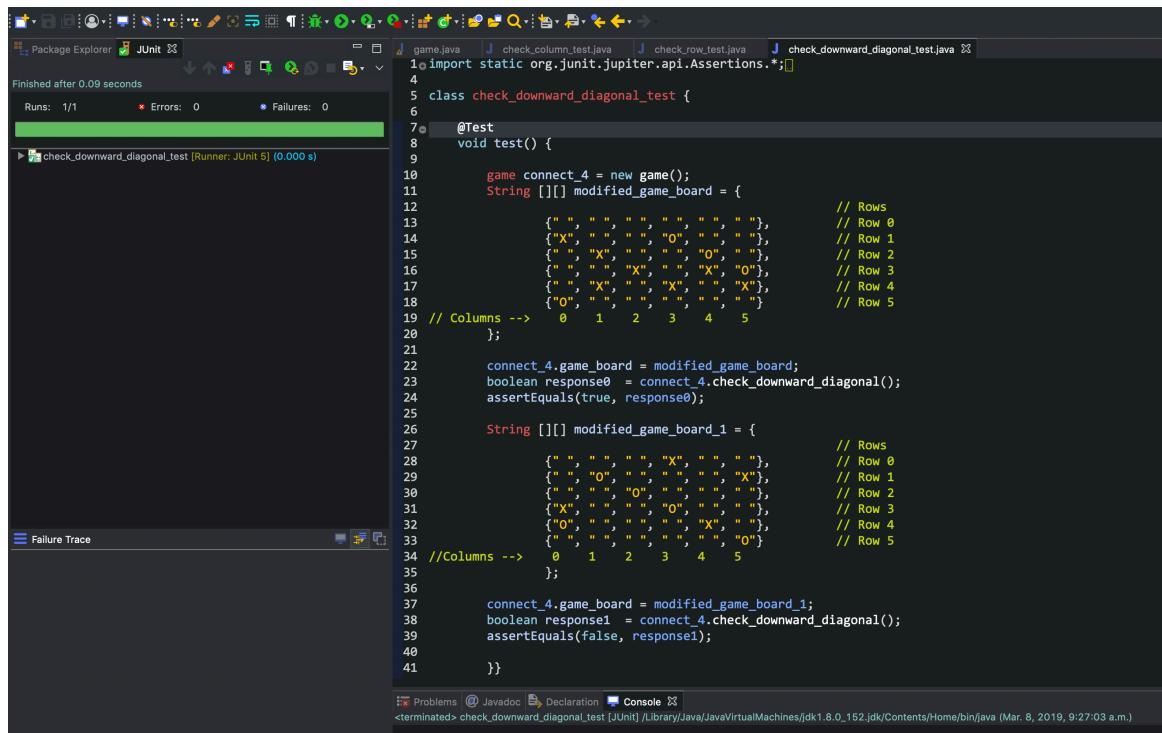


The screenshot shows the Eclipse IDE interface with the JUnit perspective selected. The top status bar indicates "Finished after 0.078 seconds". The left sidebar shows the "Package Explorer" and "JUnit" view, which lists the test class "check\_row\_test". The main editor area contains the following Java code:

```
1 import static org.junit.jupiter.api.Assertions.*;
2
3 import org.junit.jupiter.api.Test;
4
5 class check_row_test {
6
7     @Test
8     void test() {
9
10        game connect_4 = new game();
11        String [][] modified_game_board = {
12            {" ", "0", "0", "0", " ", " "}, // Row 0
13            {" ", " ", " ", " ", " ", " "}, // Row 1
14            {" ", "0", "0", "0", "0", "0"}, // Row 2
15            {" ", " ", " ", " ", " ", " "}, // Row 3
16            {" ", " ", " ", " ", " ", " "}, // Row 4
17            {" ", "0", "X", "X", "X", "X"} // Row 5
18        };
19
20        connect_4.game_board = modified_game_board;
21
22        boolean response0 = connect_4.check_row(0, "0");
23        boolean response1 = connect_4.check_row(2, "0");
24        boolean response2 = connect_4.check_row(4, "0");
25        boolean response3 = connect_4.check_row(5, "X");
26
27        assertEquals(false, response0);
28        assertEquals(true, response1);
29        assertEquals(false, response2);
30        assertEquals(true, response3);
31
32    }
33
34 }
35
36 }
```

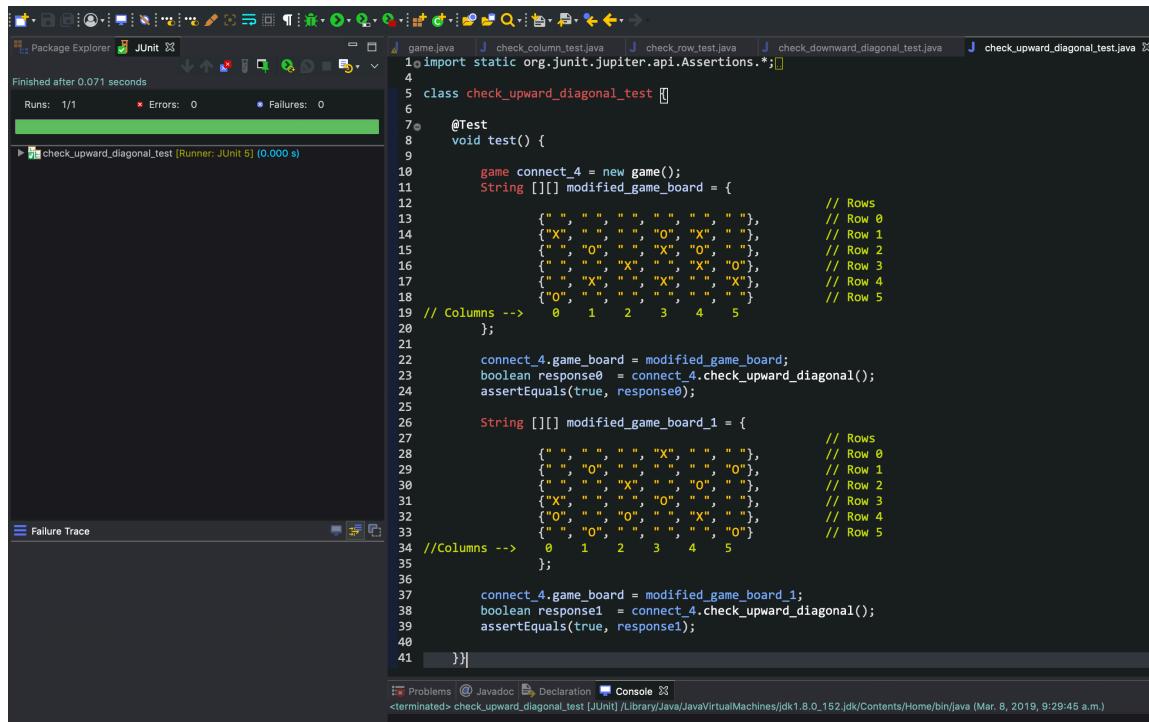
The code defines a test method named "test" that creates a new instance of the "game" class and sets its "game\_board" to a modified version of the original board. The modified board has a sequence of four '0's in row 2 and a sequence of four 'X's in row 5. The test then checks the results of calling the "check\_row" method for each row, asserting that it returns false for row 0 ('0'), true for row 2 ('0'), false for row 4 ('0'), and true for row 5 ('X').

### 3. Checking the downward diagonal had 4 of a kind.



```
import static org.junit.jupiter.api.Assertions.*;  
  
class check_downward_diagonal_test {  
  
    @Test  
    void test() {  
  
        game connect_4 = new game();  
        String [][] modified_game_board = {  
            {" ", " ", " ", " ", " ", " "},  
            {"X", " ", "O", " ", " ", " "},  
            {" ", "X", " ", "O", " ", " "},  
            {" ", "X", "X", " ", "O", " "},  
            {" ", "X", "X", "X", " ", "O"},  
            {" ", "X", "X", "X", "X", " "}  
        };  
  
        connect_4.game_board = modified_game_board;  
        boolean response0 = connect_4.check_downward_diagonal();  
        assertEquals(true, response0);  
  
        String [][] modified_game_board_1 = {  
            {" ", " ", "X", " ", " ", " "},  
            {" ", "O", " ", "X", " ", " "},  
            {" ", "X", "O", " ", "X", " "},  
            {"X", " ", "O", " ", "X", " "},  
            {"X", "X", "O", " ", "X", " "},  
            {"O", "X", "X", "O", " ", " "}  
        };  
  
        connect_4.game_board = modified_game_board_1;  
        boolean response1 = connect_4.check_downward_diagonal();  
        assertEquals(false, response1);  
    }  
}
```

### 4. Checking if the upward diagonal had 4 of a kind.



```
import static org.junit.jupiter.api.Assertions.*;  
  
class check_upward_diagonal_test {  
  
    @Test  
    void test() {  
  
        game connect_4 = new game();  
        String [][] modified_game_board = {  
            {" ", " ", " ", " ", " ", " "},  
            {"X", " ", "O", "X", " ", " "},  
            {" ", "O", " ", "X", "O", " "},  
            {" ", "X", " ", "O", "X", "O"},  
            {" ", "X", "X", " ", "O", "X"},  
            {"O", "X", "X", "X", " ", "O"}  
        };  
  
        connect_4.game_board = modified_game_board;  
        boolean response0 = connect_4.check_upward_diagonal();  
        assertEquals(true, response0);  
  
        String [][] modified_game_board_1 = {  
            {" ", " ", "X", " ", " ", " "},  
            {" ", "O", " ", " ", "O", " "},  
            {" ", "X", "O", " ", " ", " "},  
            {"X", " ", "O", " ", " ", " "},  
            {"O", " ", "O", "X", " ", " "},  
            {" ", "O", " ", " ", "O", " "}  
        };  
  
        connect_4.game_board = modified_game_board_1;  
        boolean response1 = connect_4.check_upward_diagonal();  
        assertEquals(false, response1);  
    }  
}
```

5. Checking if the player places a piece in the grid, does it get placed.

The screenshot shows the Eclipse IDE interface with the following details:

- Top Bar:** Shows various icons for file operations, search, and navigation.
- Package Explorer:** Displays several Java files: game.java, check\_column\_test.java, check\_row\_test.java, check\_downward\_diagonal\_test.java, and check\_upward\_diagonal\_test.java.
- JUnit View:** Shows a green bar indicating "Finished after 0.086 seconds" with "Runs: 1/1", "Errors: 0", and "Failures: 0".
- Code Editor:** Displays the `place_item_test` class:

```
import static org.junit.jupiter.api.Assertions.*;  
import org.junit.jupiter.api.Test;  
  
class place_item_test {  
    @Test  
    void test() {  
        game connect_4 = new game();  
        connect_4.print_board();  
        connect_4.place_item(1,"X");  
        connect_4.place_item(1,"X");  
        connect_4.place_item(1,"X");  
        connect_4.place_item(1,"X");  
        connect_4.print_board();  
        assertEquals(true,connect_4.check_column(1,"X"));  
    }  
}  
// Column 1 is technically column 2 as array starts from index 0
```
- Console View:** Shows the command: `<terminated> place_item_test [JUnit] /Library/Java/JavaVirtualMachines/jdk1.8.0_152.jdk/Contents/Home/bin/java (Mar. 8, 2019, 9:52:18 a.m.)`.
- Failure Trace:** Shows two board states for the Connect 4 game:

  - Initial State:** A 6x6 grid labeled "Connect 4" with columns numbered 1 to 6. All columns are empty (dashed lines).
  - Final State:** A 6x6 grid labeled "Connect 4" with columns numbered 1 to 6. The first column contains four "X"s at rows 1, 2, 3, and 4, while other columns are empty.