Software Quality Assignment 1

Tic Tac Toe Game

Kalapan Kannathasan - 100759041

SOFE 3980U: Software Quality

Date: March 03, 2022

Project Overview

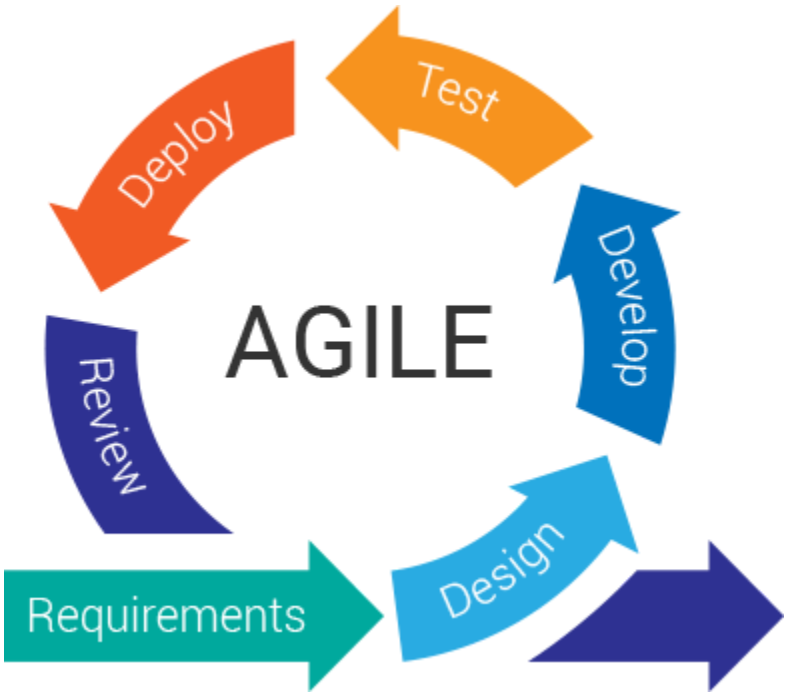
The game that was chosen for this assignment is Tic Tac Toe. The game will be made using Java on the IntelliJ ide. The testing framework that will be used is JUnit, which is one of the best open-source frameworks.

Game Description

The program that was made is a 3 by 3 Tic Tac Toe game. Tic Tac Toe is a popular 2 player game where one player is an “X” and the other is an “O”. The goal of the game is to get the player's shape in a specific pattern. The game is considered won when either player gets a pattern of 3 going up or down, 3 across, or 3 diagonal. If no player can complete any of these patterns and the board is filled, then the game is considered a draw.

Software Process

The software process that was chosen for this project is the Agile process model. The reason I chose the Agile process model is that it is the best option considering the timeframe of the project. The agile process also allows for later versions of the software to be developed easily as shown in the figure below. The main focus of the project is to make a simple Tic Tac Toe game more user-friendly and functional.



1. Requirements

This is the most important stage of the Agile process model as this is where everything for the project to be successful is noted and understood.

Game Requirements:

1. Check which player has played to alternate turns
2. Make sure a spot that is picked is not picked again
3. Check if the game is a draw
4. Check all winnable possibilities

Functionalities:

5. Display a 3 by 3 board
6. Update board live to keep players informed

2. Design

The goal of this stage is to determine the hardware and system requirements required to run the software. After consideration of different coding languages, it was concluded that Java was the best choice for this project. Java has the open-source framework JUnit which is one of the best testing frameworks to use.

- The game will have a “checkWinner” function which will be used to go through all the possible combinations required to win the game
    - Consists of a for loop that goes through all the winnable patterns
    - Consists of an if statement that checks if there is three “X” or three “O” in the pattern
- The game will have a “board” function which will draw out a visual board in the console which will be updated as the game progresses
    - Consists of multiple println which print out the board
- The program will have a “checkTurn” function that will decide the player turns
    - Consists of an if statement which checks which letter was placed to switch turns
- Consists of a main class will link all the functions together and print to the console the necessary texts

3. Development

In the development phase of the game, there were many attempts made to determine the most effective way to code the program. The process started with the creation of the “checkWinner” function. This function is used to determine which patterns on the board will need to be created to determine if the game has been won or not. This was quite difficult as there

are many different ways you can write out all the patterns but ultimately it was decided that a switch statement would be great for this application as it can be easily implemented with the board drawing function. The board drawing function was created next which consisted of println which was relatively easy. The final thing that was created was the main class which was used to tie in all the functions together. This was relatively easy as the previous functions already took out most of the code needed for the program. In this class, all the text which will be printed to the console was written along with the statement which will determine if the placement of the turns is valid.

4. Testing

The goal of the testing phase is to make sure everything is in working order before it is deployed. The goal was to obtain over a 75% coverage rate of all the methods in the TicTacToeGame.java file. The library chosen for this phase was JUnit as it provides the necessary functions and is also open-source.

The tests created are for certain aspects of the game to make sure the game functions as intended. Below is an explanation of the test case along with a screenshot of the code.

The first test case created was to make sure the TicTacToe board was empty. This was done by using assertNull() as this was combined with the null to make sure there was nothing on the board.

```
//test to make sure the board is empty
@Test
public void testEmptyBoard() {
    TicTacToeGame board1 = new TicTacToeGame();
    assertNull(board1.checkWinner(), object: null);
}
```

The second test case created was to make sure that “X” starts with the first move. This was done by using assertEquals().

```
//test to make sure x turn is set
@Test
public void testTurn() {
    TicTacToeGame board1 = new TicTacToeGame();
    board1.setTurn("X");
    assertEquals(board1.getTurn(), actual: "X");
}
```

The next couple of test cases were made to check if all the winning patterns work in the game. This was done by using assertEquals().

```
//test all the possible win conditions
@Test
public void testBoardCaseZero() {
    TicTacToeGame board1 = new TicTacToeGame();
    board1.setBoardCaseZero();
    assertEquals(board1.checkWinner(), actual: "X");
}
//test all the possible win conditions
@Test
public void testBoardCaseOne() {
    TicTacToeGame board1 = new TicTacToeGame();
    board1.setBoardCaseOne();
    assertEquals(board1.checkWinner(), actual: "X");
}
//test all the possible win conditions
@Test
public void testBoardCaseTwo() {
    TicTacToeGame board1 = new TicTacToeGame();
    board1.setBoardCaseTwo();
    assertEquals(board1.checkWinner(), actual: "X");
}
//test all the possible win conditions
@Test
public void testBoardCaseThree() {
    TicTacToeGame board1 = new TicTacToeGame();
    board1.setBoardCaseThree();
    assertEquals(board1.checkWinner(), actual: "X");
}
//test all the possible win conditions
@Test
public void testBoardCaseFour() {
    TicTacToeGame board1 = new TicTacToeGame();
    board1.setBoardCaseFour();
    assertEquals(board1.checkWinner(), actual: "X");
}
```

```java
//test all the possible win conditions
@Test
public void testBoardCaseFive() {
    TicTacToeGame board1 = new TicTacToeGame();
    board1.setBoardCaseFive();
    assertEquals(board1.checkWinner(), actual: "X");
}
//test all the possible win conditions
@Test
public void testBoardCaseSix() {
    TicTacToeGame board1 = new TicTacToeGame();
    board1.setBoardCaseSix();
    assertEquals(board1.checkWinner(), actual: "X");
}
//test all the possible win conditions
@Test
public void testBoardCaseSeven() {
    TicTacToeGame board1 = new TicTacToeGame();
    board1.setBoardCaseSeven();
    assertEquals(board1.checkWinner(), actual: "X");
}
```

Challenges:

The challenge faced during the test automation was trying to test the board for all winnable patterns. This was hard since I coded the “checkWinner” function with switch statements which made it hard to set the board to certain characters. This was overcome at the end by making multiple setBoardCase functions that manually assigned to the board the character “X”. Then this was referred to in the TicTacToeGameTest.java file which was checked if the program recognizes the pattern as a winnable solution.

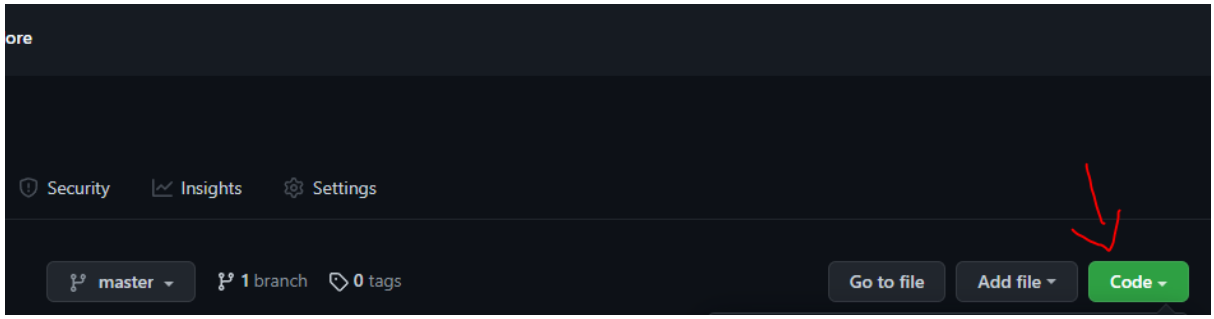
5. Deployment

This is the phase where the program is deployed. Since the game was made in Java and there is no live instance of the program, a GitHub repo has been created with all the necessary files required to run the game. Providing a link to a GitHub works in line with the Agile process model as this link can always be used to download and update the program after reviewing the game.

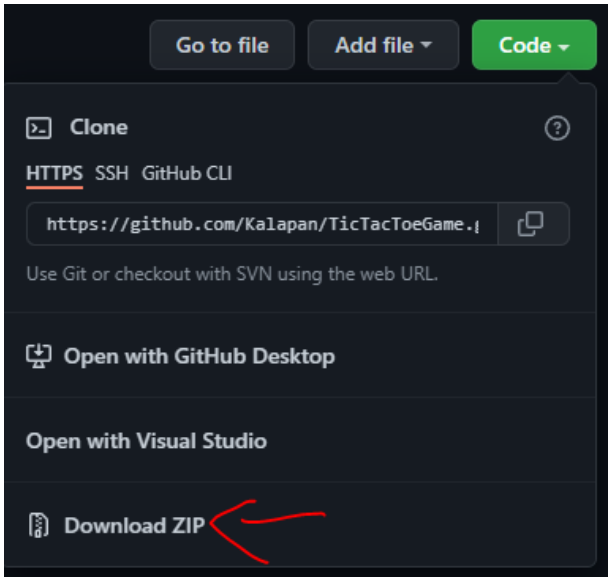
To download and run the game, please click on the GitHub link below:
GitHub Repo: https://github.com/Kalapan/TicTacToeGame

Download and Run Instructions

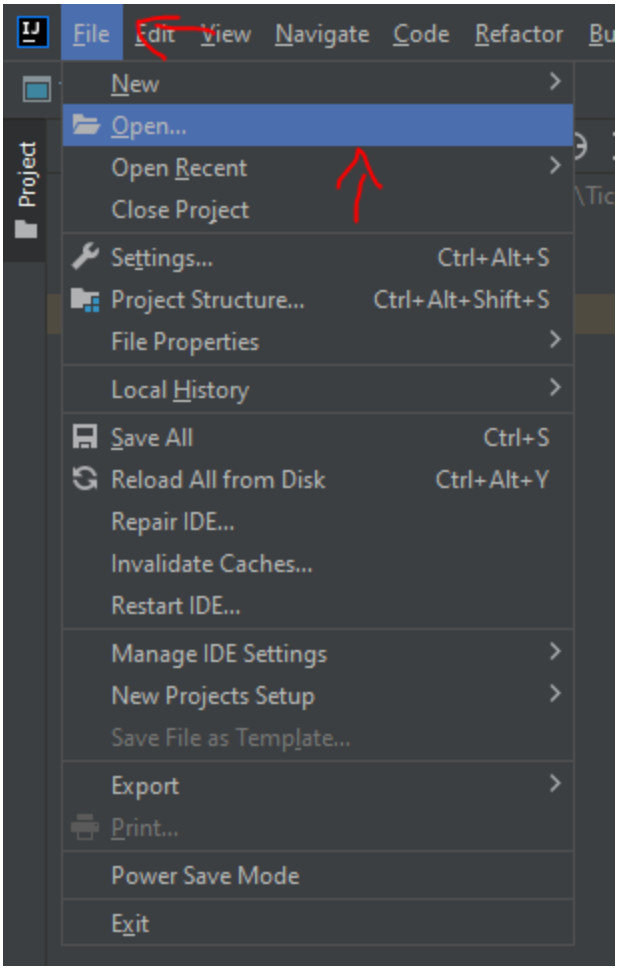
Head over to the GitHub link provided above and click on the green code button as shown below.



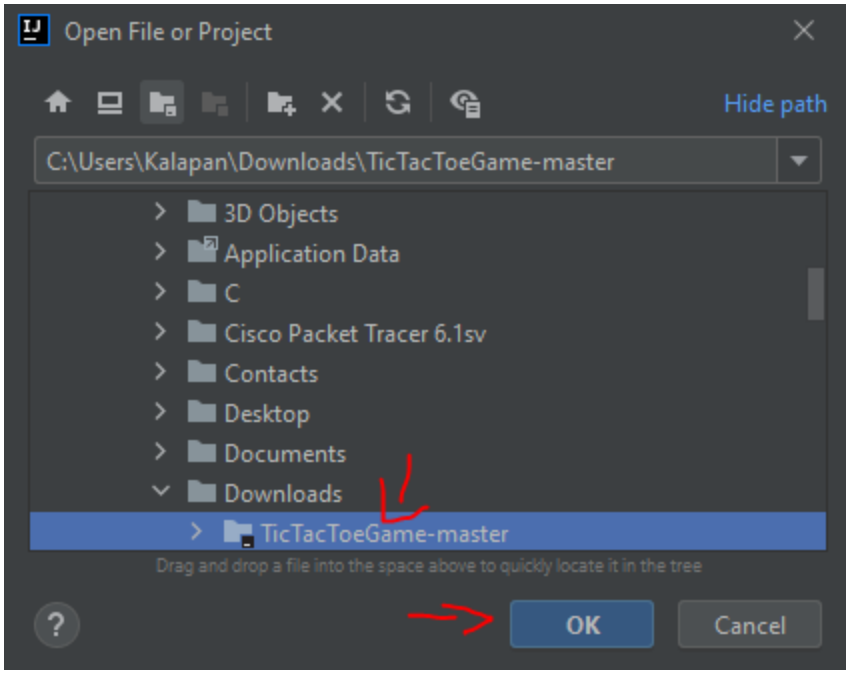
After clicking the green code button, a drop-down menu will appear where you will need to click "Download ZIP".



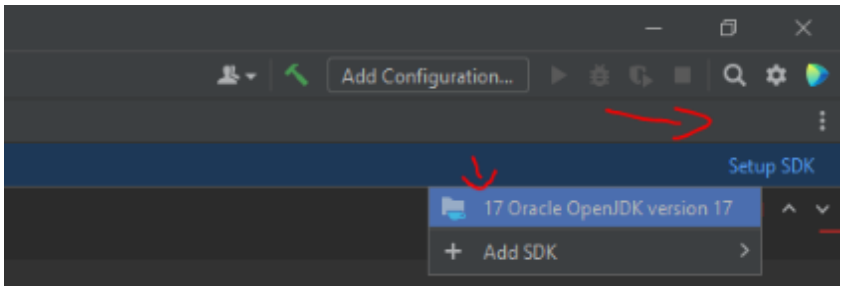
This will download a zip file to your computer. After downloading this zip file, unzip the file and open up an ide you want to use. For this example, I will use IntelliJ. After opening IntelliJ, click on "File" in the top left corner and click "Open" as shown below.



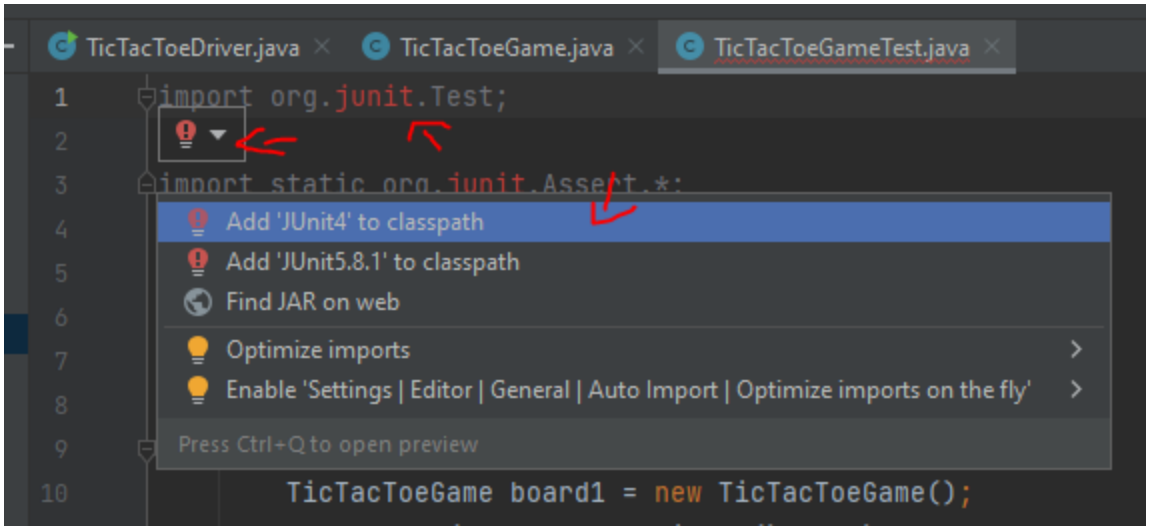
After clicking “Open”, navigate to your downloads file and click on the downloaded file from GitHub and click “OK”.



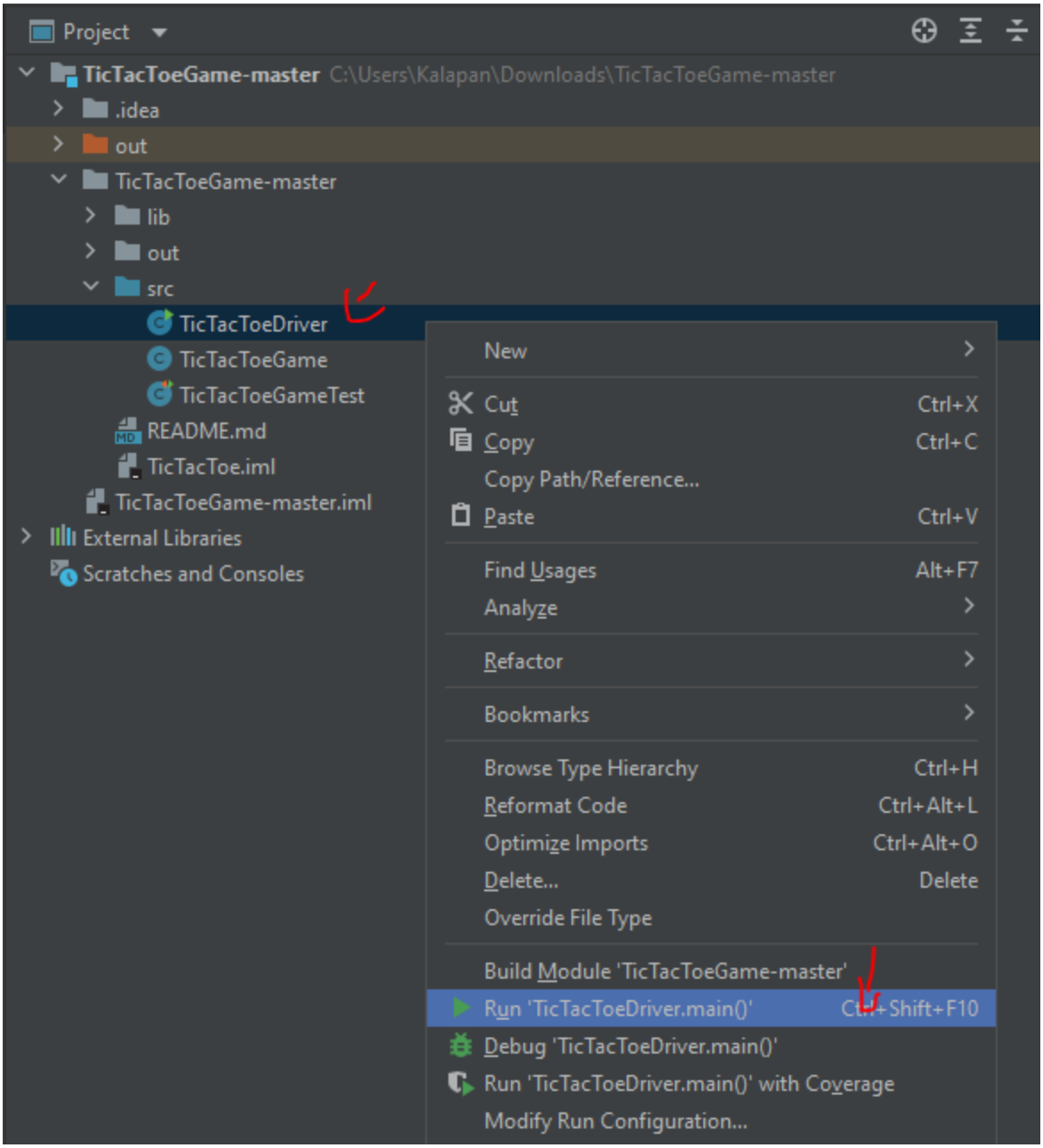
Upon opening the file, click the “src” dropdown and double click on all three java classes under “src”. Note, an error may occur stating “Project JDK is not defined”. If this happens, make sure you are on the “TicTacToeDriver” class and click “Setup SDK” which is in the top right corner. Then click the first option.



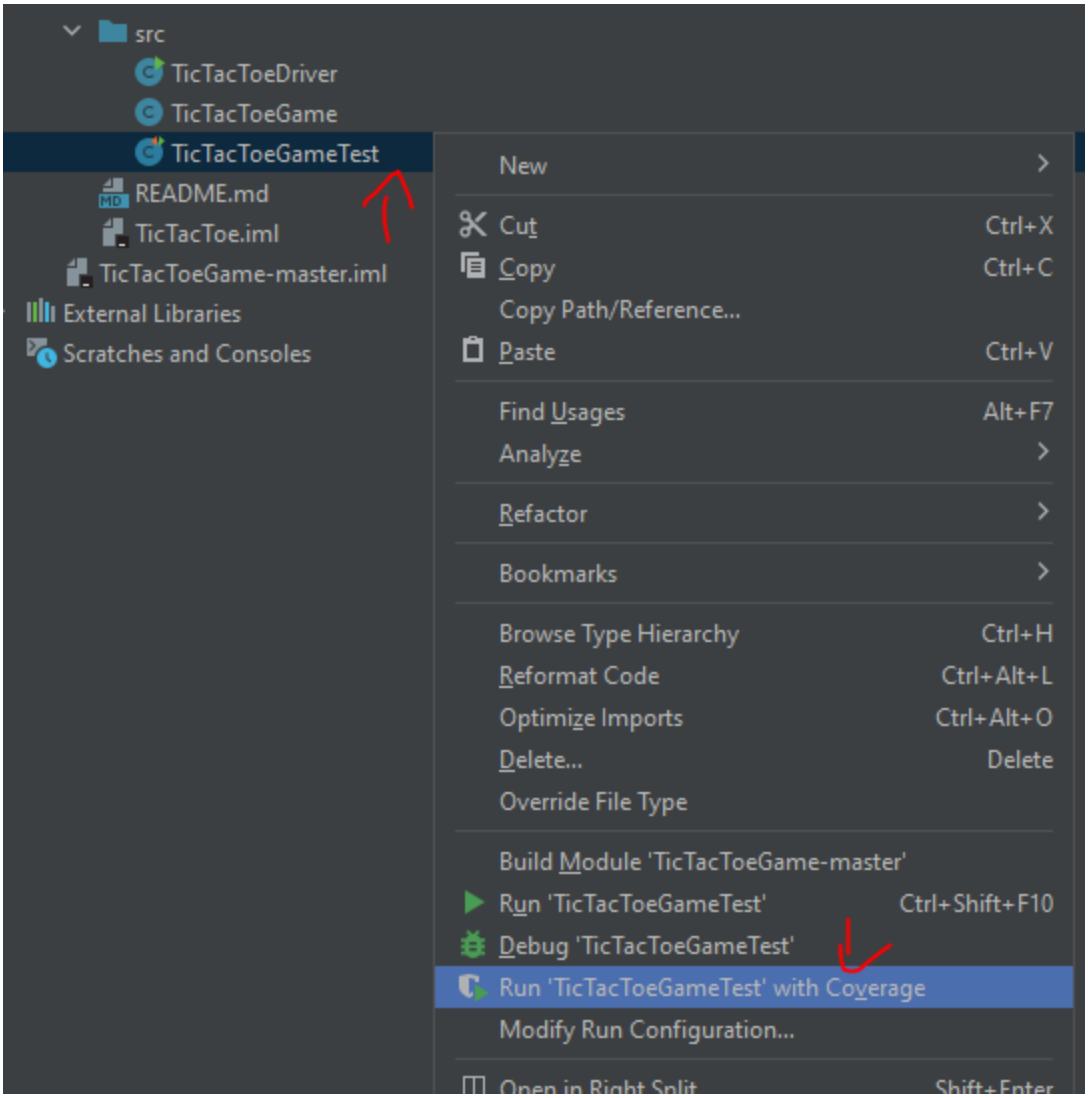
There may also be another error on the “TicTacToeGameTest” java class. If you have the error, click on “junit” then click the light bulb which pops up, and click “Add ‘JUnit4’ to classpath” as shown below.



Finally, click “OK” on the pop-up window which will fix all errors. Now to run the program, right-click on “TicTacToeDriver” from the left drop-down menu and click “Run ‘TicTacToeDriver.main()’”. Please refer to the below image.

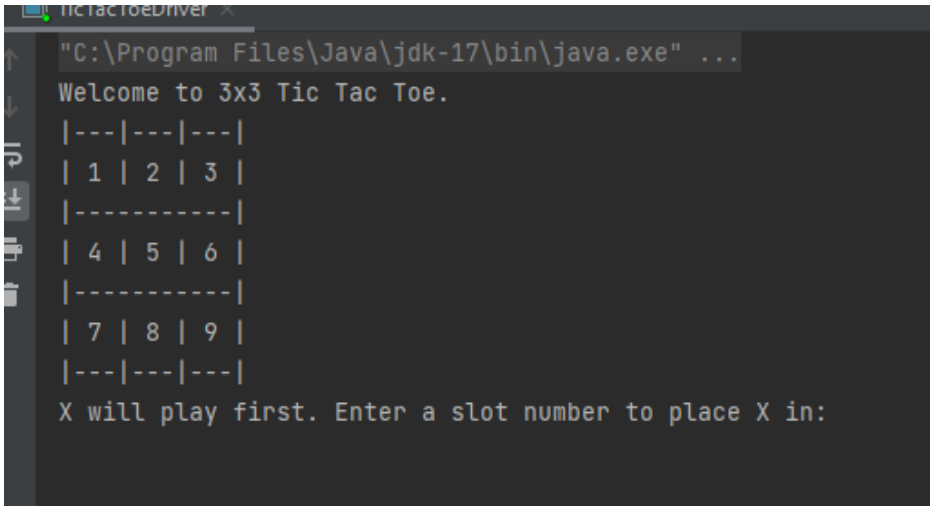


To run the test cases, right-click on “TicTacToeGameTest” and click “Run TicTacToeGameTest’ with Coverage”.

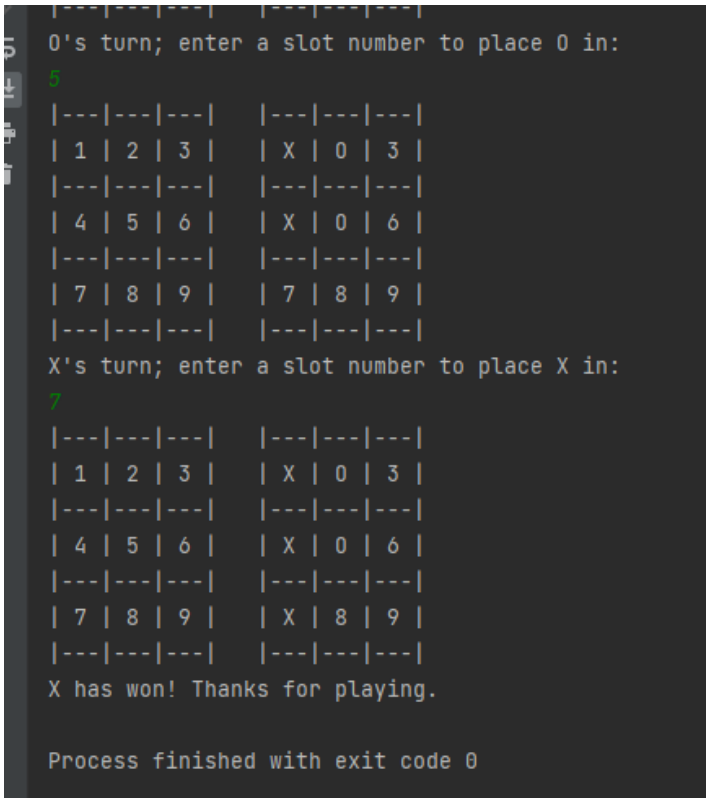

src
TicTacToeDriver
TicTacToeGame
TicTacToeGameTest
README.md
TicTacToe.iml
TicTacToeGame-master.iml
External Libraries
Scratches and Consoles
New
Cut Ctrl+X
Copy Ctrl+C
Copy Path/Reference...
Paste Ctrl+V
Find Usages Alt+F7
Analyze
Refactor
Bookmarks
Browse Type Hierarchy Ctrl+H
Reformat Code Ctrl+Alt+L
Optimize Imports Ctrl+Alt+O
Delete... Delete
Override File Type
Build Module 'TicTacToeGame-master'
Run 'TicTacToeGameTest' Ctrl+Shift+F10
Debug 'TicTacToeGameTest'
Run 'TicTacToeGameTest' with Coverage
Modify Run Configuration...
Open in Right Split Shift+Enter

ScreenShots

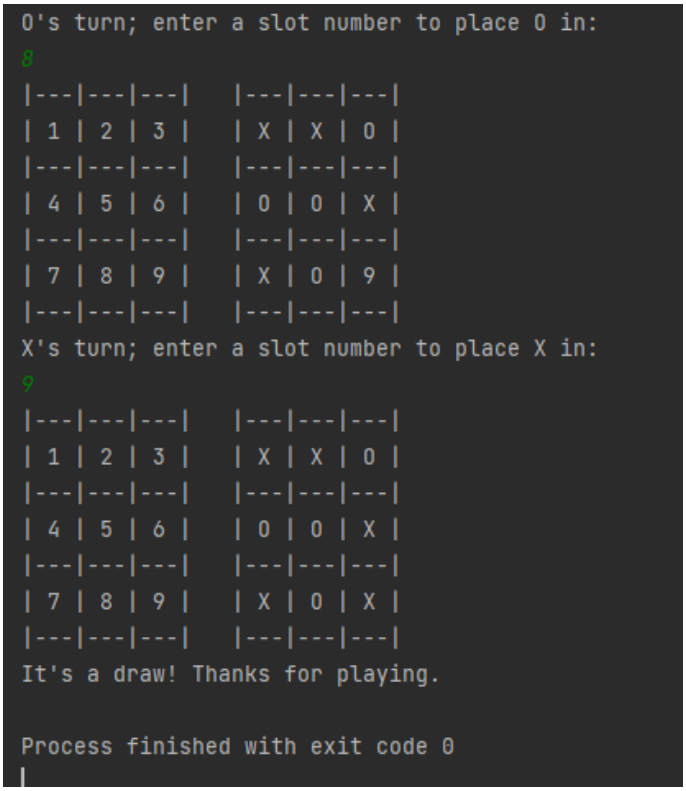
Startup Screen:



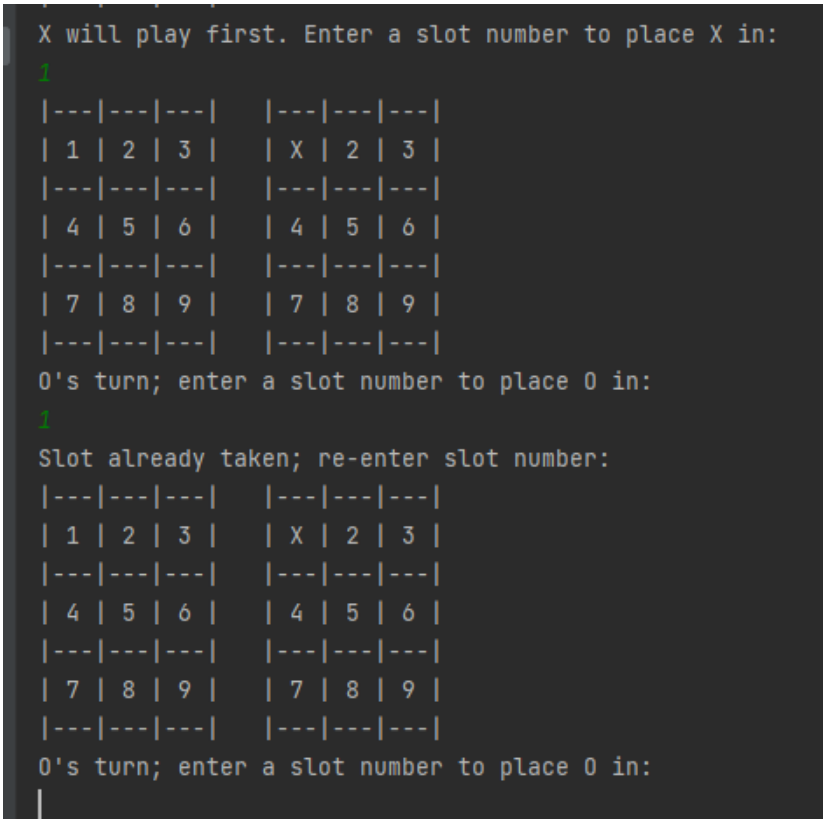
Winner Screen:



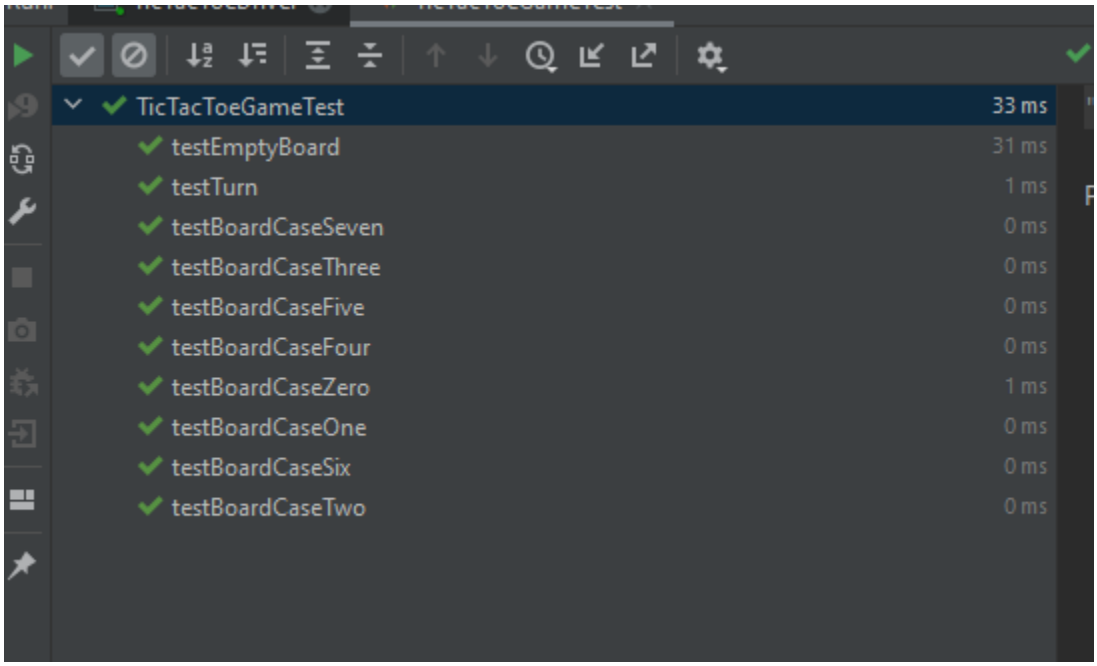
Draw Screen:



Invalid Input Screen:



All Test Cases Passed:



Test Coverage: