Tic Tac Toe Game

In Partial Fulfillment of the Requirements for the Degree of

Bachelors of Engineering in
Computer Engineering

By:
Mr. Sumit Bhimte
(BE COMP A-06)
Mr. Saurabh Sonar
(BE COMP A-66)
Mr. Hrishikesh Hasabnis
(BE COMP A-25)

Under the Guidance of
Prof. Dnyaneshwar Choudhari
For the Academic Year
2020-2021

Pimpri Chinchwad College Of Engineering And Research,
Ravet, Pune 412101

PIMPRI CHINCHWAD EDUCATION TRUST'S

Pimpri Chinchwad College of Engineering and Research, Pune 412101

CERTIFICATE

This is to certify that the seminar report entitled

"Tic Tac Toe Game"

Submitted By:

Ms. Sumit Bhimte
Mr. Saurabh Sonar
Mr. Hrishikesh Hasabnis

Have successfully completed the seminar entitled**"Tic Tac Toe Game"** in the fulfillment of LP2 and this work has been carried out in my presence.

Date:

Place: Ravet,Pune

Prof. Dnyaneshwar Choudhari
Projet Guide
Department of Computer Engg.

Prof. Dr. Archana Chaugule
HOD(Computer Department)
PCCOER,Ravet

Prof.Dr.Tiwari.H.U
Principal,
PCCOER,Ravet

ACKNOWLEDGEMENT

It gives me great pleasure to present seminar on **“ Tic Tac Toe Game”**. In preparing this report number of hands helped me directly and indirectly. Therefore it becomes my duty to express my gratitude towards them.

I am very much obliged to subject guide **Prof. Dnyaneshwar Choudhari** in Computer Engineering Department, for helping me and giving me proper guidance. I will fail in my duty if I won’t acknowledge a great sense of gratitude to the Head of Department **Prof. Dr.Archana Chaugule** and the entire staff members in for their cooperation.

I am also thankful to my family for their whole hearted blessings are always for me support and constant encouragement towards the fulfillment of the work. I wish to record the help extended to be my friends in all possible ways and active support and constant encouragement.

Date:

Place: Ravet,Pune

Mr. Sumit Bhimte (A-06)
Mr. Saurabh Sonar (A-66)
Mr. Hrishikesh Hasabnis (A-25)

PURPOSE

The purpose of tic-tac-toe is to be the first player to get three in a row on a 3-by-3 grid or four in a row in a 4-by-4 grid.To start, one player draws a board, creating a grid of squares, usually 3-by-3 or 4-by-4.

In a 3-by-3 grid game, the player who is playing "X" always goes first. Players alternate placing Xs and Os on the board until either player has three in a row, horizontally, vertically, or diagonally or until all squares on the grid are filled. If a player is able to draw three Xs or three Os in a row, then that player wins. If all squares are filled and neither player has made a complete row of Xs or Os, then the game is a draw.

One of the game's best strategies involves creating a "fork," which is placing your mark in such a way that you have the opportunity to win two ways on your next turn. Your opponent can only block one, thereby, you can win after that.

ABSTRACT

Tic-Tac-Toe Game is a very popular game played by two participants on the grid of 3 by 3. A special symbol (X or O) is assigned to each participant to indicate that the slot is covered by the respective participant. The winner of the game is the participant who first cover a horizontal, vertical or diagonal row of the board having only their symbols. This Module proposed a winning strategy of Tic-Tac-Toe game and its computation is proved theoretically by the concepts of Theoretical Computer Science using multi-tape turing machine. This algorithm is designed for computer as a player in which computer act according to the intelligence of model to maximize the chances of success. The human player can makes its own choices. Any of the player can play first by their choice. The computation rules ensures selection of best slot for computer that will lead to win or prevent opponent to make a winning move.

Contents

List of Figures

Chapter 1

INTRODUCTION

Tic Tac Toe Game - The Tic-Tac-Toe game is a thought-provoking game which is performed on a panel of 3 by 3 grid consisting 9 boxes as shown in fig.1. Each participant is assigned a special symbol i.e. either X or O to demonstrate that the particular box is filled by the particular participant. The empty boxes on the panel are chosen by the participants to mark their symbol on the game panel alternatively. It is a game of perfect information which illustrates that the previously occurred actions are known to each participant at the time of decision

The main target of the participants in the game is to create a vertical, horizontal or a diagonal line on the game panel with assigned symbol. The participant who covers a row on the game panel with his assigned symbol first, will be declared as the winner. The game outcome is a draw when none of the boxes are empty and none of the sequences are formed i.e. vertical, horizontal or diagonal by any of the participant. The participants of this game have a special scheme to play it. However, the chances of win is more for the participant who has performed this earlier, in comparison with the new participant. If both of the participants perform smartly, then game outcome will be a draw.

An optimal strategy is required to play the game so that either the player win or game will be a draw and also play optimally with a sub-optimal opponent. The proposed strategy is implemented by using the concept of theoretical computer science. A mathematical model is designed using Multi-tape turing machine. There is a set of machines that read any possible input sequences by proposed model, process them as according to the logic design for machine and produce result accordingly. This algorithm is designed for computer as a player and human player can makes its own choices. Any of the player can become the first player. The computation rules helps computer in the selection of best slot that will lead to win or prevent opponent to make a winning move. The main objective of this research is to develop a strategy of Tic-TacToe game so that player always win the game or game will be a draw.

The proposed strategy is designed by keeping in mind each and every case in which previous strategies are failed so that this strategy should not fail in any circumstances. The game is played many times by considering every configuration of the board which results in either winning of the machine or a draw. The algorithm also succeeds in playing with a suboptimal player.

Rules Of Tic Tac Toe Game.

- Rule R1- Under this rule, machine will analyze the immediate winning condition for player p that is 2 X and 1 empty slot w XXw , wXX , XwX and choose that empty slot, otherwise check for rule R2
- Rule R2- Here the machine will try to find immediate winning condition for player o that is 2 O and 1 empty slot w OOw , wOO , OwO and choose that empty slot to prevent player o from win, otherwise go for special case if first player is o or check for rule R4 and R5 if first player is p
- Special Case – It states that machine will check for sequence OXO

on any of the diagonal and remaining slots are empty then choose any middle slot of the board, otherwise rule R3 will be imposed

- Rule R3- Here the machine will check for sequences having 2 empty slots and 1 player o 's symbol O Oww , wwO , wOw and keep these empty slots to do action A
- Rule R4- This rule is implement to check for sequences consisting 2 empty slots and 1 player p 's symbol X Xww , wwX , wXw and keep these empty slots then rule R5 will be tested
- Rule R5- Under this rule, machine will try to find for sequences having 1 empty slot, 1 player o 's symbol O and 1 player p 's symbol X or 3 empty slots XOw , OXw , wXO , wOX , XwO , OwX , www and keep these empty slots then proceed for action A

1.1 Test Objectives

The general objective of this study is to create a Game which provides a numerous techniques to play the game of Tic-Tac-Toe. The specific objectives of this study are to:
(1) Include the 9 tiles to play that decides who wins the game
(2) Implement a Reset module that reset the whole game
(3) Implement a both Player winning count module that shows how many times payer wins.
(4) At last include Exit Button that exits game
(5) test functionality of the whole system.

1.2 Testing

Testing is the process of checking the functionality of an application to ensure it runs as per requirements. Unit testing comes into pic-

ture at the developers' level; it is the testing of single entity (class or method). Unit testing plays a critical role in helping a software company deliver quality products to its customers.
Unit Testing can be done in two ways.
Manual Testing.
Automated Testing.

1.2.1 Manual Testing

Executing a test cases manually without any tool support is known as manual testing.
Time-consuming and tedious Since test cases are executed by human resources, it is very slow and tedious.
Huge investment in human resources As test cases need to be executed manually, more testers are required in manual testing.
Less reliable Manual testing is less reliable, as it has to account for human errors.
Non-programmable No programming can be done to write sophisticated tests to fetch hidden information.

1.2.2 Automated Testing

Taking tool support and executing the test cases by using an automation tool is known as automation testing.
Fast Automation runs test cases significantly faster than human resources.
Less investment in human resources Test cases are executed using automation tools, so less number of testers are required in automation testing.
More reliable Automation tests are precise and reliable.
Programmable Testers can program sophisticated tests to bring

out hidden information.

1.3 JUnit

1.3.1 Introduction

JUnit is an open source Unit Testing Framework for JAVA. It is useful for Java Developers to write and run repeatable tests. Erich Gamma and Kent Beck initially develop it. It is an instance of xUnit architecture. As the name implies, it is used for Unit Testing of a small chunk of code. JUnit promotes the idea of "first testing then coding", which emphasizes on setting up the test data for a piece of code that can be tested first and then implemented. This approach is like "test a little, code a little, test a little, code a little." It increases the productivity of the programmer and the stability of program code, which in turn reduces the stress on the programmer and the time spent on debugging. Developers who are following test-driven methodology must write and execute unit test first before any code.

Once you are done with code, you should execute all tests, and it should pass. Every time any code is added, you need to re-execute all test cases and makes sure nothing is broken.

Features of JUnit

JUnit is an open source framework, which is used for writing and running tests.
Provides annotations to identify test methods.
Provides assertions for testing expected results.
Provides test runners for running tests.
JUnit tests allow you to write codes faster, which increases quality.
JUnit is elegantly simple. It is less complex and takes less time.
JUnit tests can be run automatically and they check their own results and provide immediate feedback. There's no need to manually comb

through a report of test results.
JUnit tests can be organized into test suites containing test cases and even other test suites.
JUnit shows test progress in a bar that is green if the test is running smoothly, and it turns red when a test fails.

Chapter 2

Functional Requirements

In software engineering, a functional requirement defines a system or its component. It describes the functions a software must perform. A function is nothing but inputs, its behavior, and outputs. It can be a calculation, data manipulation, business process, user interaction, or any other specific functionality which defines what function a system is likely to perform.

Functional software requirements help you to capture the intended behavior of the system. This behavior may be expressed as functions, services or tasks or which system is required to perform.A requirement is a description of the service that software must offer. A requirement can range from the high-level abstract statement of the sender's necessity to detailed mathematical functional requirement specifications.

Benefits of Functional Requirement

- Helps you to check whether the application is providing all the functionalities that were mentioned in the functional requirement of that application.
- A functional requirement document helps you to define the functionality of a system or one of its subsystems.
- Functional requirements along with requirement analysis help identify missing requirements.

- They help clearly define the expected system service and behavior.
- Errors caught in the Functional requirement gathering stage are the cheapest to fix.
- Support user goals, tasks, or activities.

Chapter 3

Non Functional Requirements

A non-functional requirement defines the quality attribute of a software system. They represent a set of standards used to judge the specific operation of a system. Example, how fast does the website load?

A non-functional requirement is essential to ensure the usability and effectiveness of the entire software system. Failing to meet non-functional requirements can result in systems that fail to satisfy user needs.

Non-functional Requirements allows you to impose constraints or restrictions on the design of the system across the various agile backlogs. Example, the site should load in 3 seconds when the number of simultaneous users are > 10000. Description of non-functional requirements is just as critical as a functional requirement.

Benefits of Non-Functional Requirement

- The non-functional requirements ensure the software system follow legal and compliance rules.
- They ensure the reliability, availability, and performance of the software system.
- They ensure good user experience and ease of operating the software.

- They help in formulating security policy of the software system.
- Non Functional Requirements offer key design specifications which support the architectural decisions made and in many ways can be seen as building blocks on which the system scalability depends.
- It also quantify what the investment is going to deliver and how far into the future can the application be expected to go in terms of meeting growth in end user workload.

Chapter 4

Block Diagram

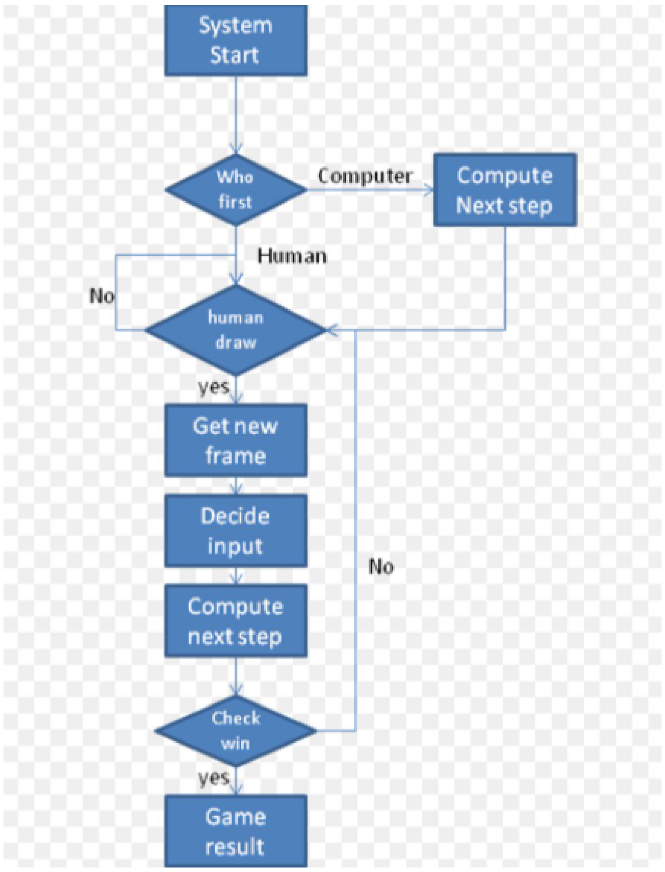


Figure 4.1: Block Diagram For Tic-Tac-Toe Game

Chapter 5

Output Screenshots

5.1 Pass/Fail criteria

This section specifies generic pass/fail criteria for the tests covered in this plan. They are supplemented by pass/fail criteria in the test design specification.

5.1.1 Component Pass/Fail criteria

Tests executed on components only pass when they satisfy the signatures, constraints, and interfaces dictated by the Object Design Specification for that component. This includes positive tests, negative and stress tests, and boundary tests.

Object Creation

The Tic-Tac-Toe Game java program is created and its object is created in the junit testcase.This testing whether the object is created successfully is tested.

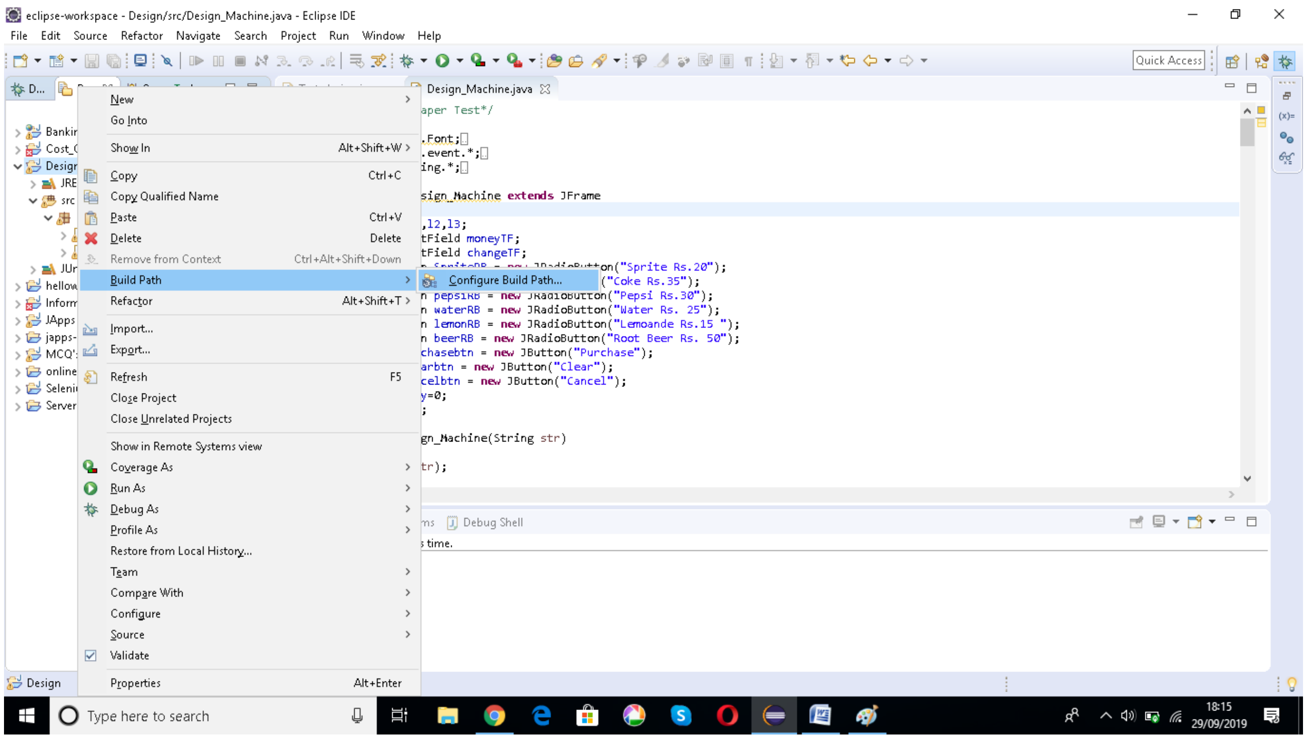
Reset Button

This check verifies whether whole game reset or not.

Quit Button Verification

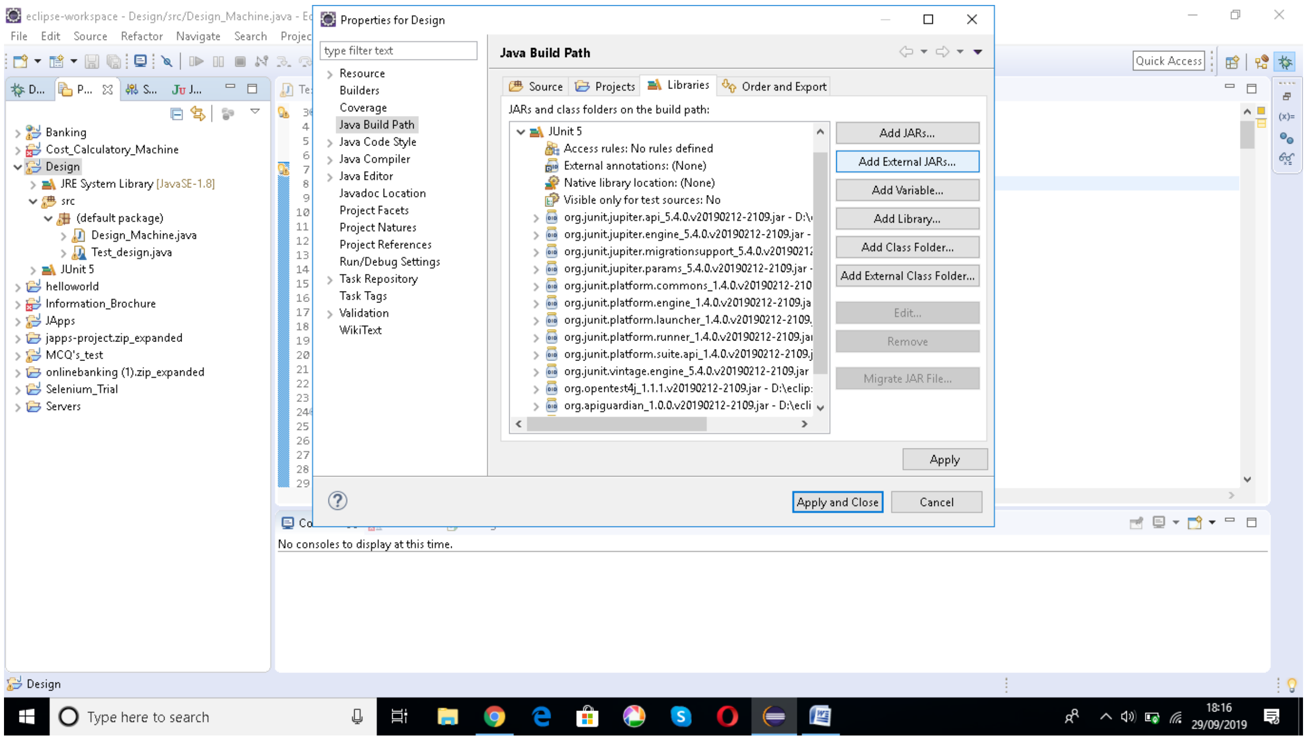
This check verifies whether Game quits or not.

5.1.2 Screenshots



```
aper Test*/

.Font;
.event.*;
ing.*;

sign_Machine extends JFrame

,l2,l3;
tField moneyTF;
tField changeTF;
                          on("Sprite Rs.20");
                          ("Coke Rs.35");
n pepsiRB = new JRadioButton("Pepsi Rs.30");
n waterRB = new JRadioButton("Water Rs. 25");
n lemonRB = new JRadioButton("Lemoande Rs.15 ");
n beerRB = new JRadioButton("Root Beer Rs. 50");
chasebtn = new JButton("Purchase");
arbtn = new JButton("Clear");
celbtn = new JButton("Cancel");
y=0;
;

gn_Machine(String str)

tr);
```

Figure 5.1: Junit Build Path



Java Build Path

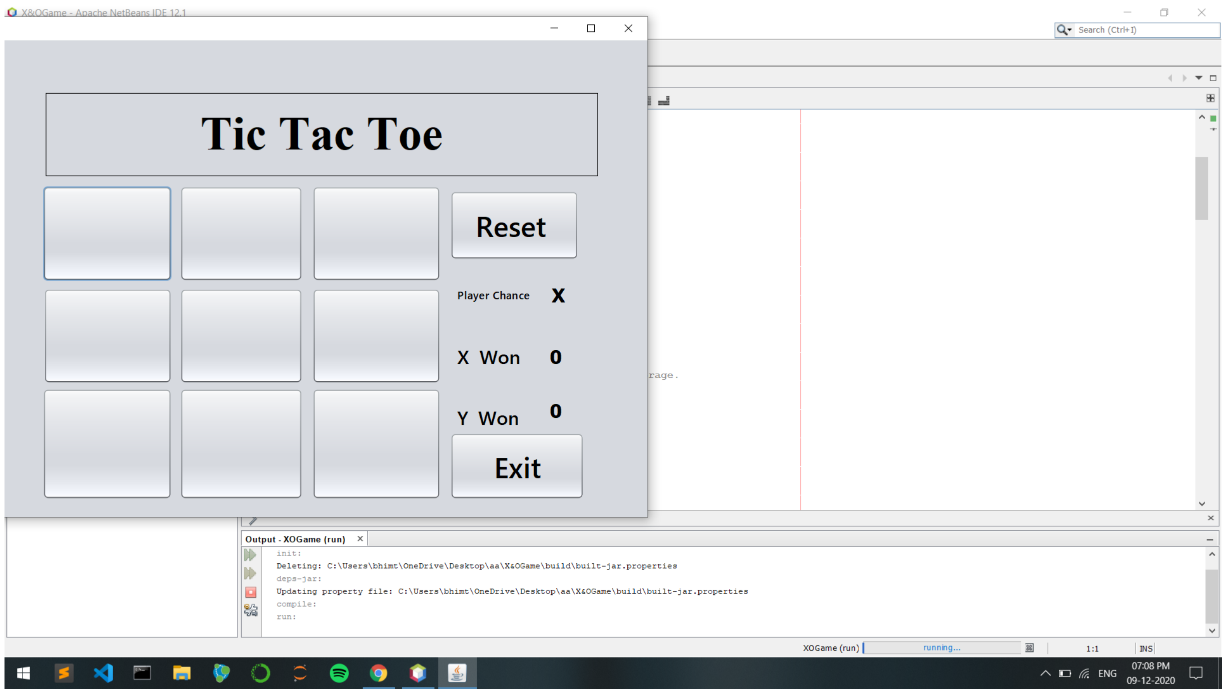
JARs and class folders on the build path:

```
JUnit 5
    Access rules: No rules defined
    External annotations: (None)
    Native library location: (None)
    Visible only for test sources: No
  org.junit.jupiter.api_5.4.0.v20190212-2109.jar - D:\
  org.junit.jupiter.engine_5.4.0.v20190212-2109.jar -
  org.junit.jupiter.migrationsupport_5.4.0.v2019021
  org.junit.jupiter.params_5.4.0.v20190212-2109.jar -
  org.junit.platform.commons_1.4.0.v20190212-210
  org.junit.platform.engine_1.4.0.v20190212-2109.ja
  org.junit.platform.launcher_1.4.0.v20190212-2109.
  org.junit.platform.runner_1.4.0.v20190212-2109.ja
  org.junit.platform.suite.api_1.4.0.v20190212-2109.
  org.junit.vintage.engine_5.4.0.v20190212-2109.jar
  org.opentest4j_1.1.1.v20190212-2109.jar - D:\eclip
  org.apiguardian_1.0.0.v20190212-2109.jar - D:\ecli
```

```
Banking
Cost_Calculatory_Machine
Design
  JRE System Library [JavaSE-1.8]
  src
    (default package)
      Design_Machine.java
      Test_design.java
  JUnit 5
helloworld
Information_Brochure
JApps
japps-project.zip_expanded
MCQ's_test
onlinebanking (1).zip_expanded
Selenium_Trial
Servers
```

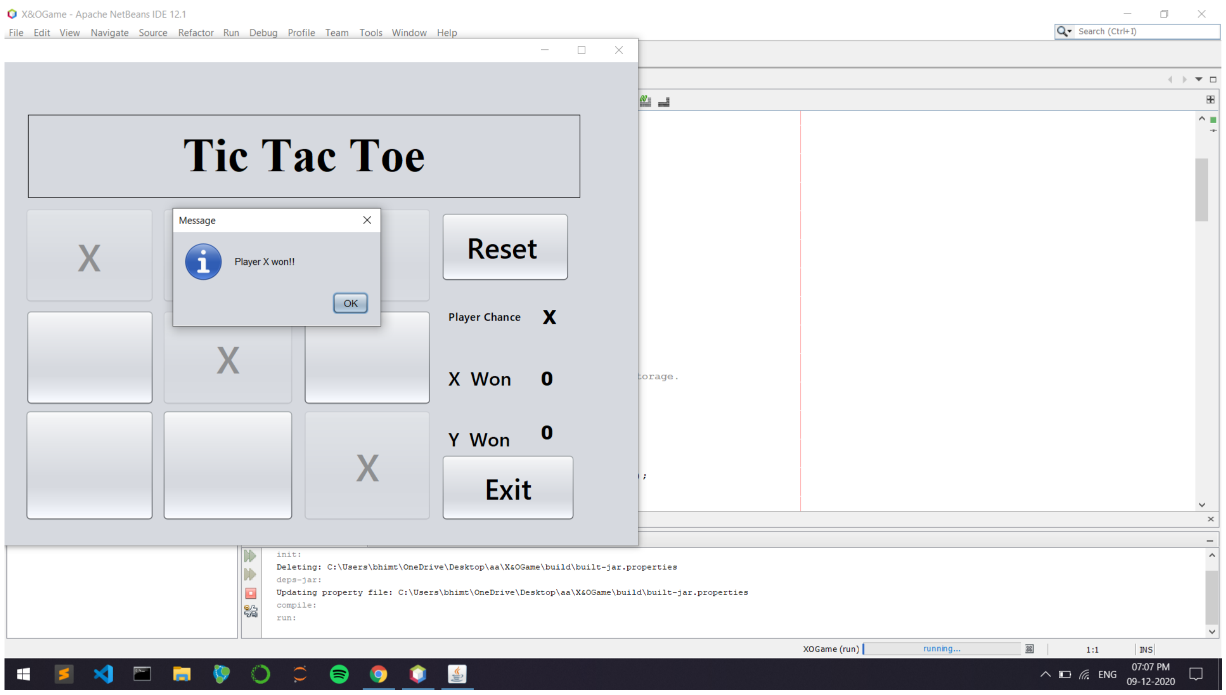
No consoles to display at this time.

Figure 5.2: Import Junit Jar Files



```
init:
Deleting: C:\Users\bhimt\OneDrive\Desktop\aa\X&OGame\build\built-jar.properties
deps-jar:
Updating property file: C:\Users\bhimt\OneDrive\Desktop\aa\X&OGame\build\built-jar.properties
compile:
run:
```

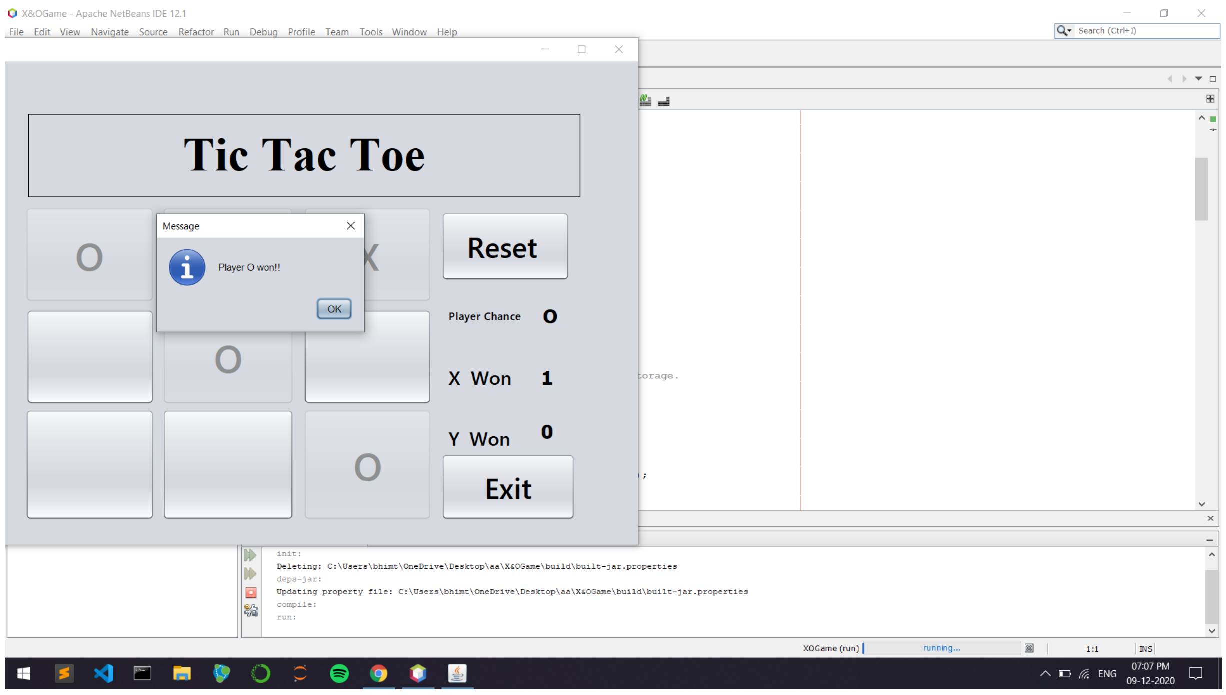
Figure 5.3: Application view



Player X won!!

```
init:
Deleting: C:\Users\bhimt\OneDrive\Desktop\aa\X&OGame\build\built-jar.properties
deps-jar:
Updating property file: C:\Users\bhimt\OneDrive\Desktop\aa\X&OGame\build\built-jar.properties
compile:
run:
```

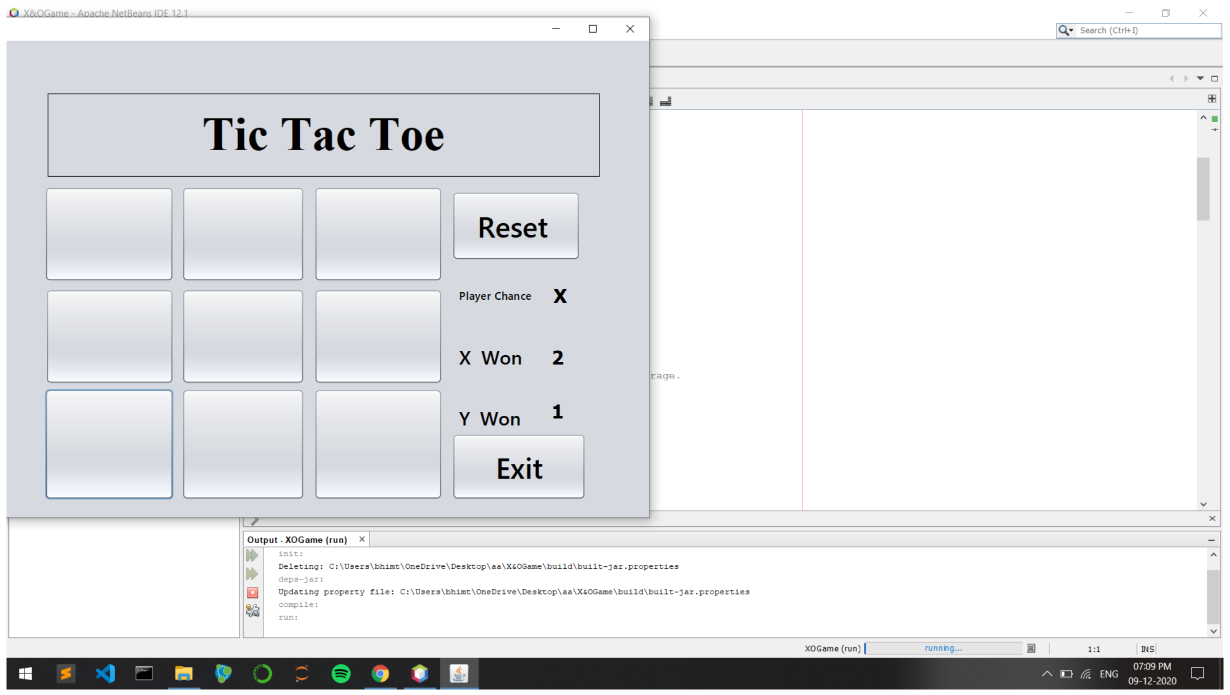
Figure 5.4: Player X Wins



Player O won!!

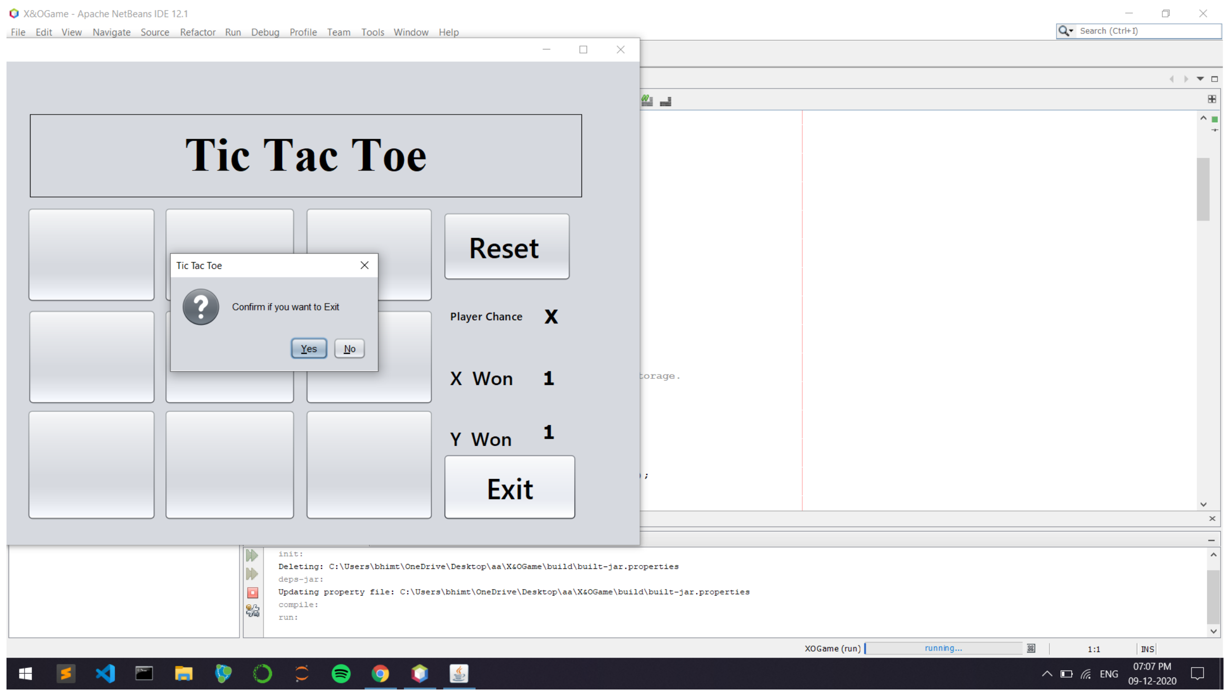
```
init:
Deleting: C:\Users\bhimt\OneDrive\Desktop\aa\X&OGame\build\built-jar.properties
deps-jar:
Updating property file: C:\Users\bhimt\OneDrive\Desktop\aa\X&OGame\build\built-jar.properties
compile:
run:
```

Figure 5.5: Player O Wins



```
init:
Deleting: C:\Users\bhimt\OneDrive\Desktop\aa\X&OGame\build\built-jar.properties
deps-jar:
Updating property file: C:\Users\bhimt\OneDrive\Desktop\aa\X&OGame\build\built-jar.properties
compile:
run:
```

Figure 5.6: Player Winning Count

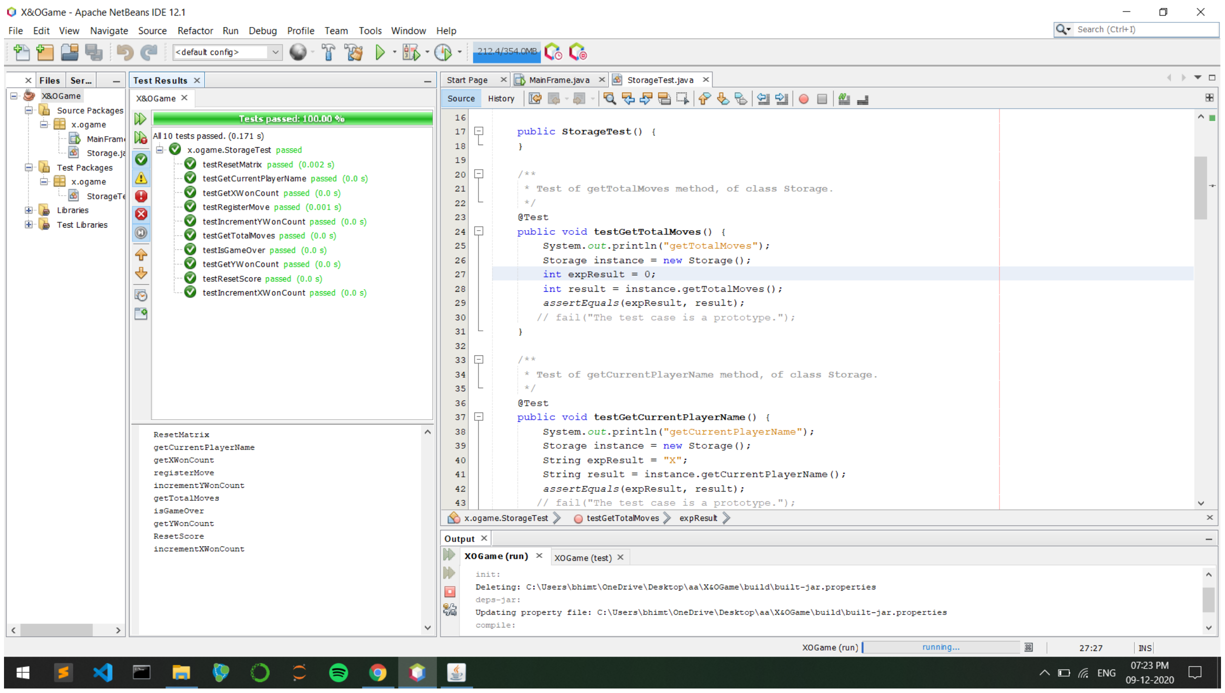

X&OGame - Apache NetBeans IDE 12.1
File Edit View Navigate Source Refactor Run Debug Profile Team Tools Window Help
Tic Tac Toe
Reset
Tic Tac Toe
Confirm if you want to Exit
Yes
No
Player Chance X
X Won 1
Y Won 1
Exit
XOGame (run) running...

```
init:
Deleting: C:\Users\bhimt\OneDrive\Desktop\aa\X&OGame\build\built-jar.properties
deps-jar:
Updating property file: C:\Users\bhimt\OneDrive\Desktop\aa\X&OGame\build\built-jar.properties
compile:
run:
```

Figure 5.7: Exit Button clicked

Chapter 6

Junit Test Cases Screenshot



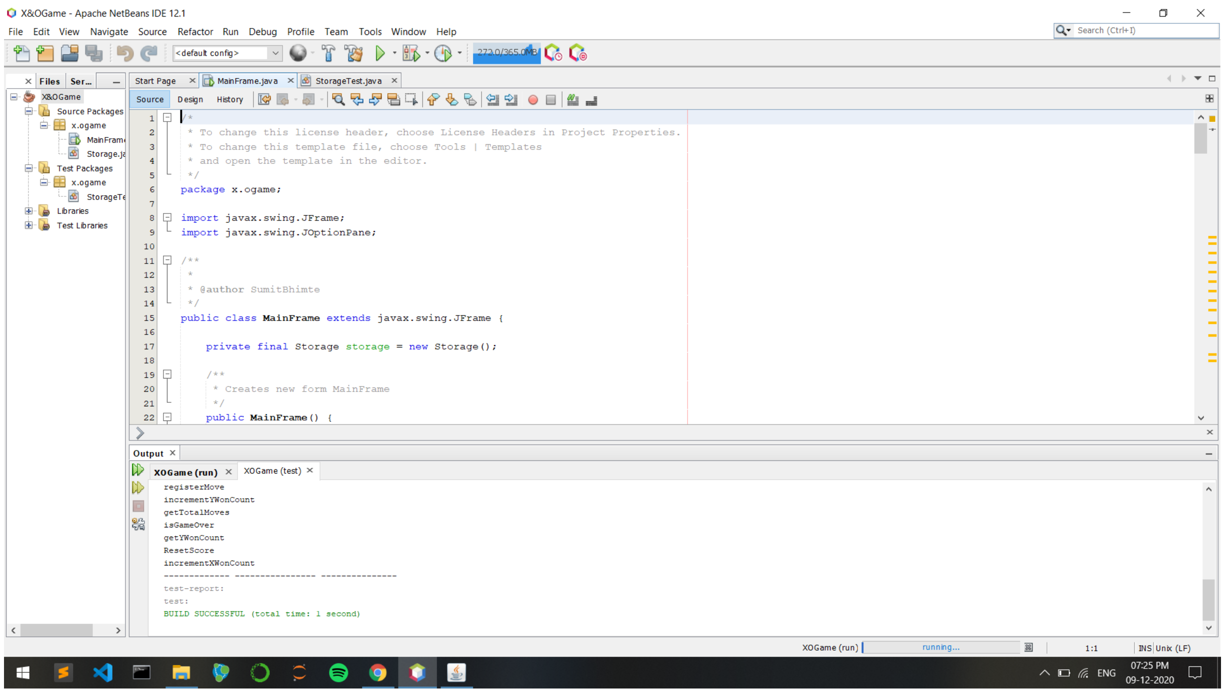
```
Tests passed: 100.00 %
All 10 tests passed. (0.171 s)
x.ogame.StorageTest passed
    testResetMatrix passed (0.002 s)
    testGetCurrentPlayerName passed (0.0 s)
    testGetXWonCount passed (0.0 s)
    testRegisterMove passed (0.001 s)
    testIncrementYWonCount passed (0.0 s)
    testGetTotalMoves passed (0.0 s)
    testIsGameOver passed (0.0 s)
    testGetYWonCount passed (0.0 s)
    testResetScore passed (0.0 s)
    testIncrementXWonCount passed (0.0 s)
```

```
ResetMatrix
getCurrentPlayerName
getXWonCount
registerMove
incrementYWonCount
getTotalMoves
isGameOver
getYWonCount
ResetScore
incrementXWonCount
```

```java

    public StorageTest() {
    }

    /**
     * Test of getTotalMoves method, of class Storage.
     */
    @Test
    public void testGetTotalMoves() {
        System.out.println("getTotalMoves");
        Storage instance = new Storage();
        int expResult = 0;
        int result = instance.getTotalMoves();
        assertEquals(expResult, result);
       // fail("The test case is a prototype.");
    }

    /**
     * Test of getCurrentPlayerName method, of class Storage.
     */
    @Test
    public void testGetCurrentPlayerName() {
        System.out.println("getCurrentPlayerName");
        Storage instance = new Storage();
        String expResult = "X";
        String result = instance.getCurrentPlayerName();
        assertEquals(expResult, result);
       // fail("The test case is a prototype.");
```

```
init:
Deleting: C:\Users\bhimt\OneDrive\Desktop\aa\X&OGame\build\built-jar.properties
deps-jar:
Updating property file: C:\Users\bhimt\OneDrive\Desktop\aa\X&OGame\build\built-jar.properties
compile:
```

Figure 6.1: Test cases Execution.



```java
/*
 * To change this license header, choose License Headers in Project Properties.
 * To change this template file, choose Tools | Templates
 * and open the template in the editor.
 */
package x.ogame;

import javax.swing.JFrame;
import javax.swing.JOptionPane;

/**
 *
 * @author SumitBhimte
 */
public class MainFrame extends javax.swing.JFrame {

    private final Storage storage = new Storage();

    /**
     * Creates new form MainFrame
     */
    public MainFrame() {
```

```
registerMove
incrementYWonCount
getTotalMoves
isGameOver
getYWonCount
ResetScore
incrementXWonCount
------------ --------------- --------------
test-report:
test:
BUILD SUCCESSFUL (total time: 1 second)
```

Figure 6.2: Test cases displayed on console.

Chapter 7

Test cases

MCQ Test Series Application			
Sr no.	Test Case	Result	Function
1	Object Creation	Pass	assertNotNull()
2	Player X Wins	Pass	assertEquals()
3	Player O Wins	Pass	assertEquals()
4	Player Winning Count	Pass	assertSame()
5	Reset Button	Pass	assertNotSame()
6	Exit Button Checked	Pass	assertFalse()

Chapter 8

Conclusion

The window based application-Tic-Tac-Toe Application is successfully implemented and tested using junit.