**Creating a video game in Java**



**Lecturer:**

**D4017\_Jude Joseph Lamug Martinez, MCS**

**Made by:**

**2702350724\_Rheynorich**

**Object Oriented Programming**

**Computer Science Faculty**

**Binus International University 2024/2025**

**Table of contents**

Table of contents………………………………………………………………………………………………………………….2

Abstract………………………………………………………………………………………………………………………………..3

Chapter 1 – Project Description…………………………………………………………………………………………….4

* 1. Introduction…………………………………………………………………………………………………………………..4
  2. Idea Inspiration………………………………………………………………………………………………………………4
  3. How it works………………………………………………………………………………………………………………….4

Chapter 2 – Modules used……………………………………………………………………………………………………5

Chapter 3 – The code……………………………………………………………………………………………………………6

Chapter 4 – Screenshots of Project……………………………………………………………………………………….8

Chapter 5 – What have I learned…………………………………………………………………………………………10

Chapter 6 – Source Code and Video Demo…………………………………………………………………………11

References………………………………………………………………………………………………………………………….12

**Abstract**

This is a document for my Object-oriented programming final project. In this project I decided to make a game using Java programming language. This document will explain the concept of the game and will give detailed explanation on how each of the classes and methods work altogether.

**Chapter 1 - Project Description**

* 1. **Introduction**

This project is coded with Java programming language. The game is a simple tic tac toe game in which players press on a 3x3 button plain. The players, X and O will take turns pressing the buttons in the 3x3 plain until either there are 3 X or O’s in a row.

* 1. **Idea Inspiration**

There was not a lot of effort in choosing this idea. I chose this game because it was the simplest to code and the fastest. I took this idea with the intention of getting some practice using the Java programming language. I didn’t want to take something that was hard to code and complex to understand as I am still a beginner.

* 1. **How it works**

The game will display a 3x3 plain of buttons. The code has an algorithm on which players go first. In my code, player X goes first, and which ever button the player presses will turn that button into X then the next player, O, whichever button O presses it will turn that button into an O. whoever has 3 X or Os in a row first win.

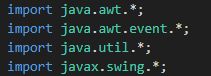
**Chapter 2 – Classes used**

**2.2 Classes**

In this code, there is only one class which is the Tic Tac Toe class. The reason there is only one class is because I followed a YouTube tutorial to code this game and, in the tutorial, the guy only puts all the constructors in one class.

**2.3 Imported modules**

In the code, I have imported 4 modules.



**Java.awt** – used to set colours, fonts, grid layout and border layout

**Java.awt.event** – it implements ActionListener which is an interface used to receive action events whenever a button is clicked on in the TicTacToe class

**Java.util** – this generates a random number to determine which player goes first in the TicTacToe class

**Javax.swing** – contains the modules used to display on the window. The modules are JFrame, JPanel, JLabel, JButton, JOptionpane.

**2.4 The code**

Let’s start from the top, **Variables:**

A screen shot of a computer code

Description automatically generated

This code is just all the variables that I have declared.

**Random** – is used to determine which players goes first

**Frame** – is used to declare the frame of the window

**Title\_panel** – is used to display the title of which player goes first. “X’s turn/O’s turn”

**Button\_panel** – is used to display and organize the buttons in which the player presses to select their place in their turn (X or O).

**Textfield** – is for the text displayed

**ScoreLabel** – is the text displayed to show player’s score

**Buttons** – are just the buttons players press

**Player1\_turn** – is a variable in Boolean (true or false) to determine which player goes first. If it’s true, player1 goes first. If false, then it is player2’s turn



This part is just setting up the display and declaring the decorations for the window and GUI of the game.

A screen shot of a computer screen

Description automatically generated

**actionPerformed**: Now this is the first logic of the game. It starts off by declaring what the button should do when it is pressed and if it is pressed, it will turn the button into X (Not O because the first player is X) then after the button is pressed and X has already marked it’s spot, it sets the player1\_turn to false to let player2 play and the else statement shows the opposite when player2 takes their turn to mark their spot on the buttons. After both players have their turn, it calls on the check() function to check if any one of the players have won the game

A computer screen shot of code

Description automatically generated

**firstTurn():** This function helps determine which players go first in playing. Thread.sleep(millis:2000) initiates a delay in the program so that the text label will not appear immediately.

A screen shot of a computer program

Description automatically generated

This function is all the possible combinations of which X and O can do to win. The code will cypher through all the possible combinations and if anyone of the combinations are met by either O or X then O or X will win the game. If none of the combinations are met, it will call on another function which is the checkTie() function. (The screenshot only shows part of the combination).

A computer screen shot of text

Description automatically generated

If any of the combination is met by either X or O, the buttons will turn green showing the proof of which the player got 3 X or Os in a row. Afterwards, they call on the function updateScore() to update the score and then disable the buttons after one of the players have won the game. Then it calls the function promptPLayAgain() to ask the player if they wish to play again or quit the game.

A screen shot of a computer code

Description automatically generated

**Tie**(): If none of the combinations are met, this function will be called to declare the game as a tie and will prompt the player to play again or quit the game

**disableButton**(): This functions is in a for loop which states that if all the buttons are pressed and taken by either X or O, the buttons will be set to false which prevents the players from pressing the button again to change the game.

**updateScore**(): this functions just updates the scoreboard

**resetBoard**(): it resets the entire board and calls on the firstTurn() function to continue the game.

Then the main function is called to start a new TicTacToe class to replay the game

**Chapter 3 – What did I learn?**

My experience: It wasn’t that hard to copy and rewrite the code from YouTube, and understanding the code is easy. Don’t get me wrong but I understand the code but when it comes to coding by myself without any help or any internet sources, I’m finished. When it comes to coding and using my own logic by myself, I’m cooked. But so far, I am understanding the coding slowly and even though I didn’t grasp the concept of Java as well as I did with python, I’m still learning it slowly.

I started making this project way before the deadline and I finished it before the mid-exams. It was that easy to finish. But it wasn’t as functional. Just by following the YouTube tutorial, it wasn’t enough. It was too simple. To tackle this issue, I added my own scoreboard and added a prompt to ask the players if whether they want to play again or quit the game.

Another thing I overlooked was that the YouTube tutorial made the game in one class. That was a problem when it came to working on the class diagram. I only had one class.

Chapter 4 – Links

Github link: