JAVA MINI-PROJECT

~VIC-VAC-VOE~

1.Description of Project:

TTT.java (aka TTT class) (The description of this one class gives a total, simple and a conclusive idea about out project)

~Prelude:

Since this program was done entirely in Eclipse, keeping in mind the need to create a GUI, there was no special need to go beyond the scope of one class. Though we could have created several classes to perform the same operation, to keep things simple and clean to understand, we have restricted ourselves to 1 class.

~Functions:

1. *public TTT()*
   This is a constructor which initialises the GUI.

2. *private void gameScore()*
   An utillity function to increase/set the game scores.

3. *private void choosePlayer()*
   An utility function which enables switching between players(ie X and O)

4. *public void reset()*
   A function which enables the user to perform a reset board operation at any point of the game.

5. *private void winningCond()*
   This is the function that works at the heart of the code. It checks the winning condition for both parties and updates the result. The check can be described as a brute force comparison for all possible winning conditions. If a winning condition passes, the result is updated, displayed and the board is reset, Otherwise, no operation is performed.

6. *private void initialise()*

This is the function that is called within the constructor. It initialises the GUI whose composition is described below:

~ 1 Frame
~ 3 Panels
~ 9+2 Buttons
~ 1+2 Labels
~ 2 TextBoxes

~Additional Information:

1. The Exit Functionality

   As in several popular platforms, the ability to exit the game whenever necessary has been provided

2. Dual Score board

   Keeping track of both the players score has been effectively implemented in our code.

3. Reset Ability

   The ability to reset at will gives the user power to change the game dynamics.

2. OOPs concepts involved:

~Abstraction:

All the data memebers are declared private and most of the functions are private as well

~Exception Handling:

Most of the errors that can incur in our program, are effectively handled by JVM itself.

~Inheritance:

The moment GUI came into play, along with it came the neccesity to perform inheritance at a vast scale. Though one might ask “how has inheritance been used when there is only one class in total?” The answer lies in the fact that the program uses GUI. Every button, label, Frame etc that has been used in the code is inherited from its appropriate parent classes and packages.

~PolyMorphism:

Here is the part were we arent entirely sure. But we are almost sure that **OVERRIDING** and **OVERLOADING** have been used within the GUI script, but we can surely say that they have not been used explicitly by us.(again, merely cos that would unnecessarily make a simple tic tac toe GUI ridiculously complicated and not user friendly). We are also however almost sure that we have NOT used run time polymorphism.

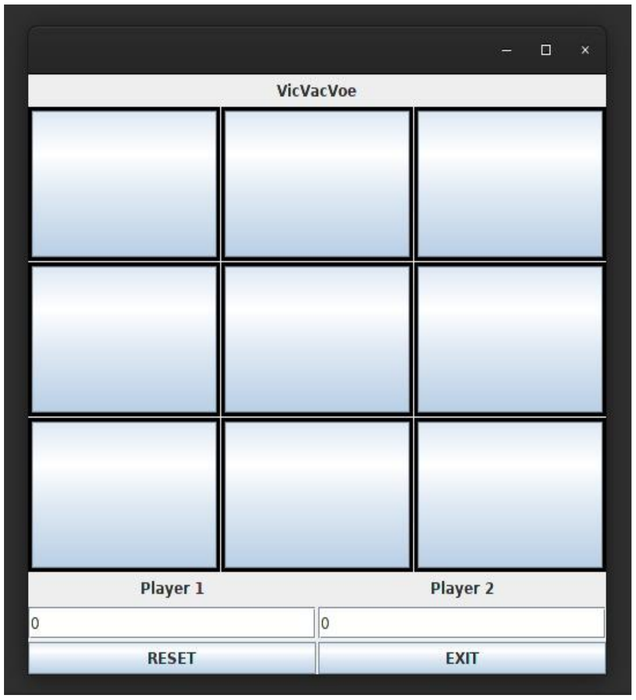
~Encapsulation:

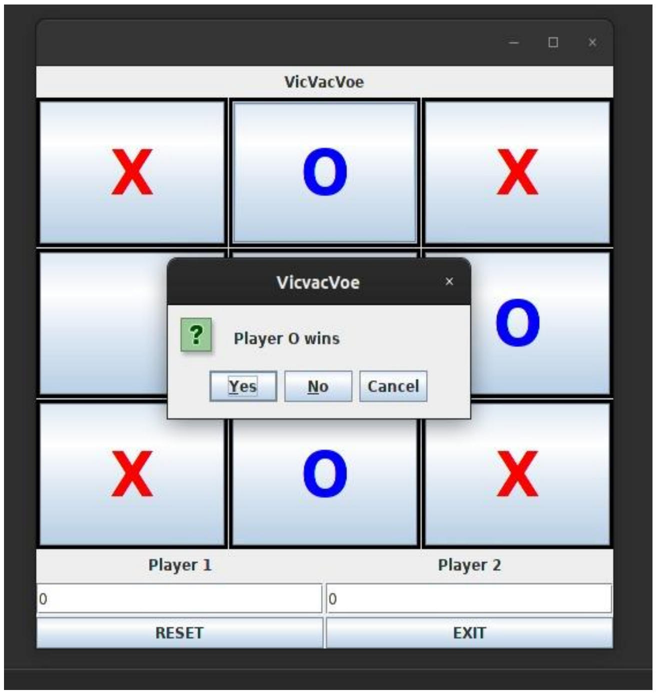
We think this is the most obvious part. We have used a class and that defines most of what encapsulation is.

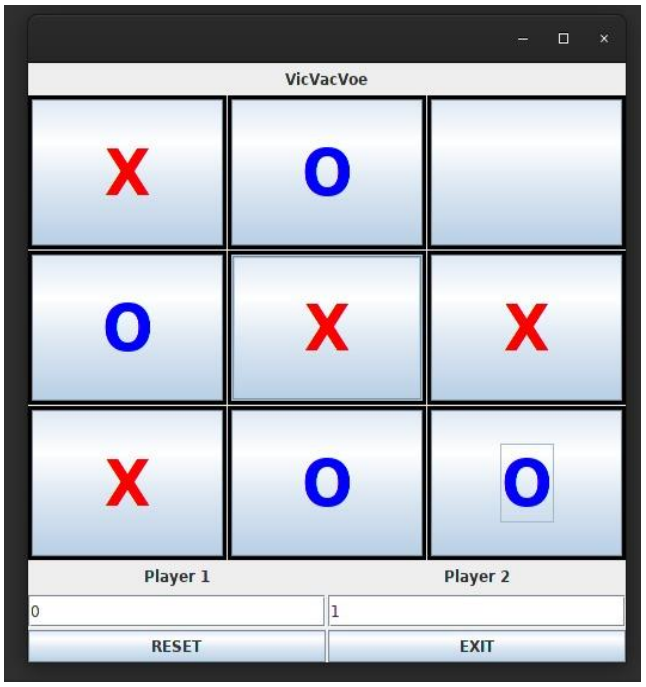
~GUI:

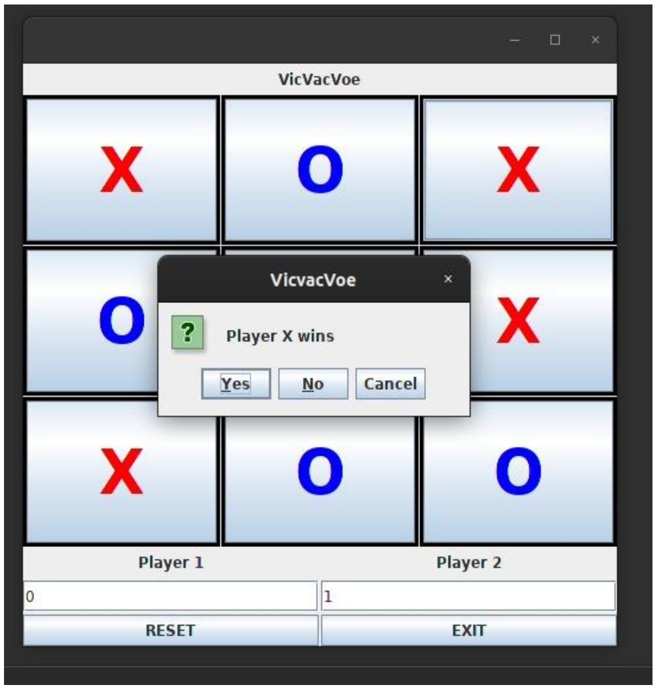
We don’t know if this comes under the banner “OOPs concepts”. But we do know, that by doing the project we learnt a LOT about how GUI works in JAVA, and we just thought we should put this here, as a symbolism to how grateful we are for learning this new and curious concept by means of this project.

3. Output Screenshots:









4. Learning Outcomes:

~We don’t know if this comes under the banner “OOPs concepts”. But we do know, that by doing the project we learnt a LOT about how GUI works in JAVA, and we just thought we should put this here, as a symbolism to how grateful we are for learning this new and curious concept by means of this project.

~We certainly did learn using several oops concepts in a new and refreshing way by doing this manner. This includes Polymorphism, Encapsulation, Inheritance, etc.

~Team work and cooperation came along side as a benefit while doing this project and certainly did, we think, give us a heads up as to what to expect out of out next project together as a team.

~Error Handling and Debugging is a learning that accompanies one with every simple code he/she types. Needless to say, that this project helped us polish our skills in the same.

5. Class diagram: