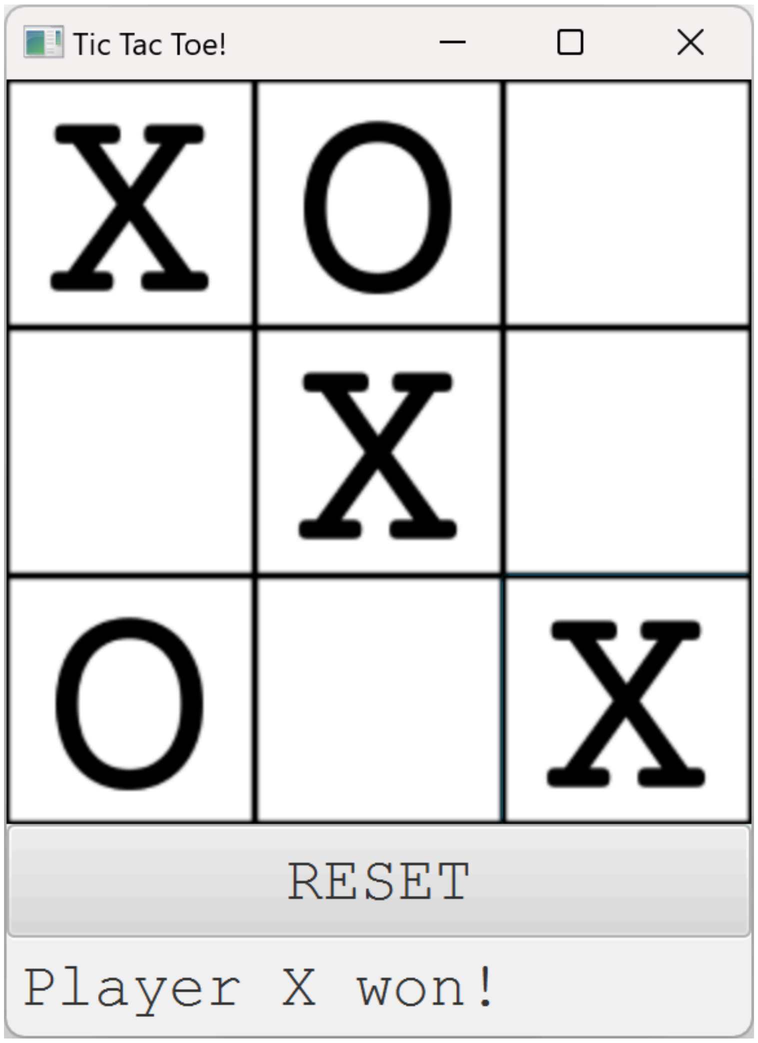
Tic Tac Toe

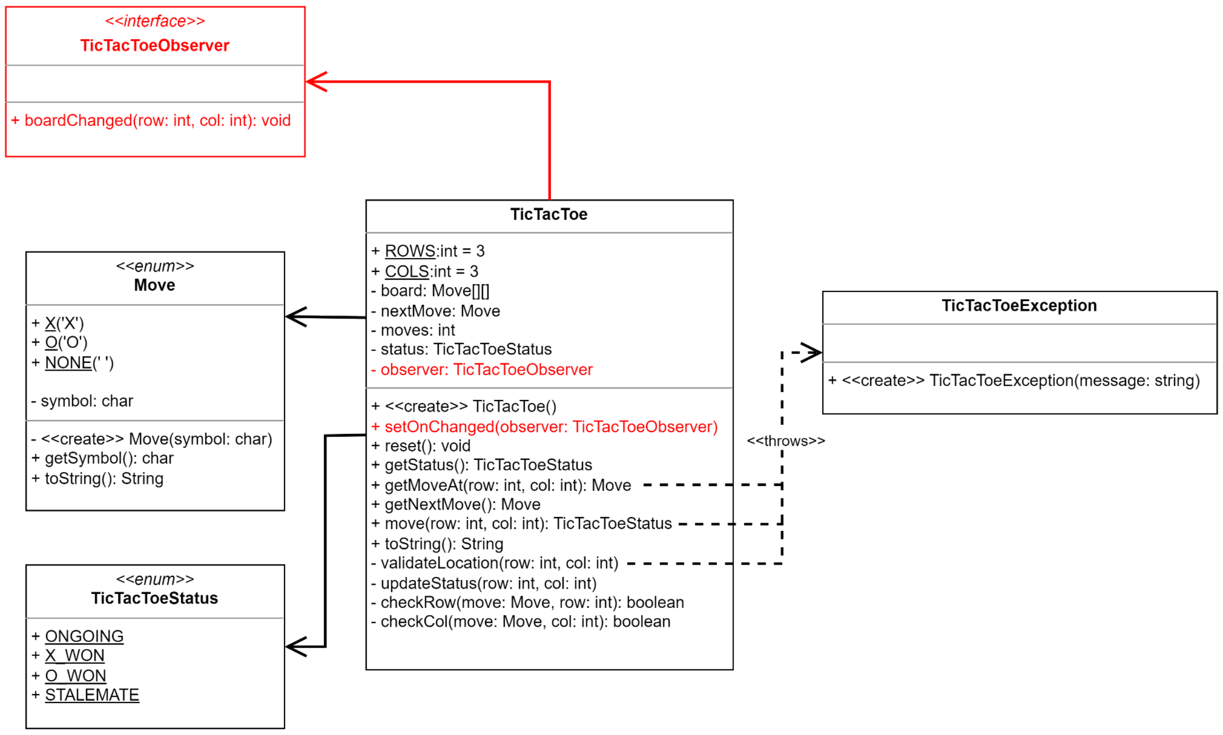
Goals of the Assignment

The goal of this assignment is to write a JavaFX GUI that players can use to play a game of Tic Tac Toe. As always, you are expected to practice good software engineering, including the Git workflow. You ***do not*** have to write JUnit unit tests for your code on this assignment. Read this document ***<u>in its entirety</u>*** before asking for help from the course staff.



```
Player X won!
```

An example user interface. You <u>do not</u> need to make your GUI look like this! Be creative!



<<interface>>
TicTacToeObserver
+ boardChanged(row: int, col: int): void
TicTacToe
+ ROWS:int = 3
+ COLS:int = 3
- board: Move[][]
- nextMove: Move
- moves: int
- status: TicTacToeStatus
- observer: TicTacToeObserver
+ <<create>> TicTacToe()
+ setOnChanged(observer: TicTacToeObserver)
+ reset(): void
+ getStatus(): TicTacToeStatus
+ getMoveAt(row: int, col: int): Move
+ getNextMove(): Move
+ move(row: int, col: int): TicTacToeStatus
+ toString(): String
- validateLocation(row: int, col: int)
- updateStatus(row: int, col: int)
- checkRow(move: Move, row: int): boolean
- checkCol(move: Move, col: int): boolean
<<enum>>
Move
+ X('X')
+ O('O')
+ NONE(' ')
- symbol: char
- <<create>> Move(symbol: char)
+ getSymbol(): char
+ toString(): String
<<enum>>
TicTacToeStatus
+ ONGOING
+ X_WON
+ O_WON
+ STALEMATE
TicTacToeException
+ <<create>> TicTacToeException(message: string)
<<throws>>

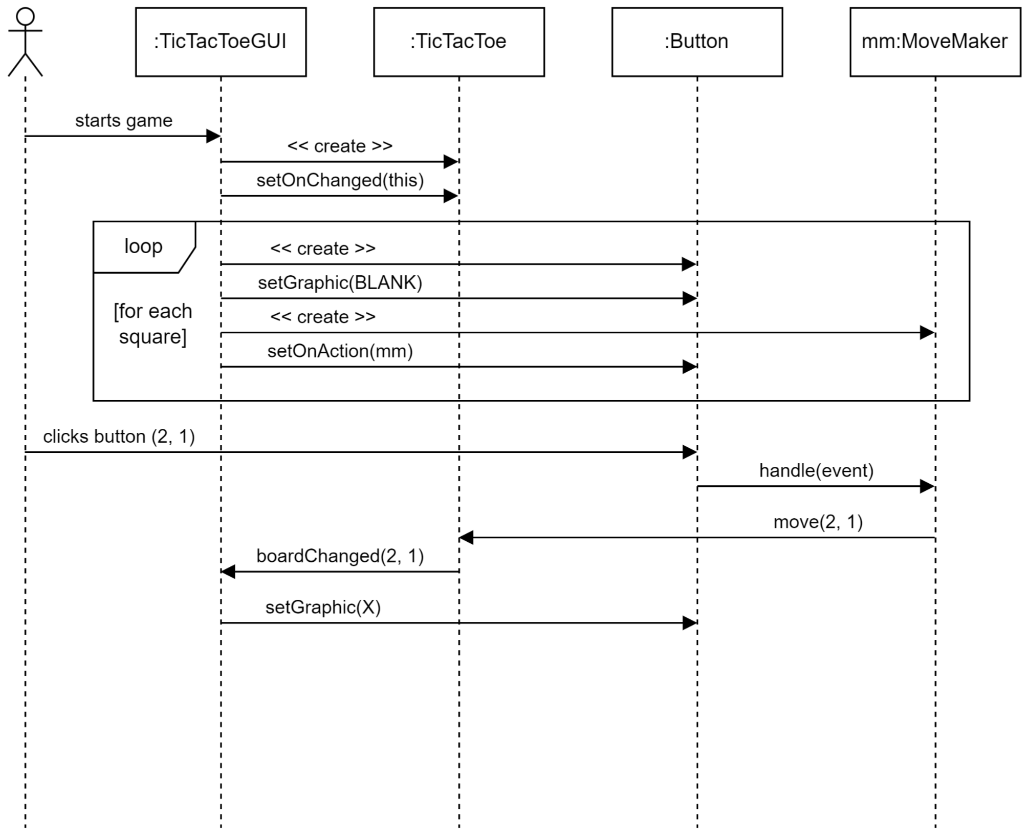
*The Tic-Tac-Toe model after it has been **modified** to implement the Observer Pattern.*

Activities

1. You have been provided with code that implements a Tic Tac Toe model and a simple command-line interface. If you have not done so already, familiarize yourself with the following classes:
    a. `ttt.model.TicTacToeException`
    b. `ttt.model.Move`
    c. `ttt.model.TicTacToeStatus`
    d. `ttt.model.TicTacToe`
    e. `ttt.view.TicTacToeCLI` - it is recommended that you run this class and play a game or two before continuing.

2. You will need to implement the Observer Pattern in the Tic Tac Toe model (see the UML class diagram above for suggestions). This should include:
    a. A new ***observer interface*** must be implemented by any class that wishes to be notified whenever the board changes, e.g. an X or an O is played in a square. Its single method should declare parameter(s) sufficient to determine the specific location of the move that was made.
    b. Update the TicTacToe class to support registering and notifying an observer.
        i. Look through the code and make sure to notify the observer whenever the board changes. Hint: don't forget the `reset` method!

ii. The sequence diagram below illustrates a possible implementation of the Observer pattern in the Tic Tac Toe game.



3. Next, you will create a JavaFX GUI of your own design that can be used to play TicTacToe using the provided model with your Observer Pattern modifications. While you are encouraged to flex your creative muscles, your GUI must meet at least the following requirements:
    a. The board must be displayed at all times. There must be a clear visual distinction between blank cells, cells with X's and cells with O's on the board.
        i. You are encouraged to find or make your own images. You do not have to use Xs and Os!
        ii. While you are free to incorporate animation and sound, neither is required.
    b. There must be a means to display status and feedback messages to the user, for example:
        i. When a new game has started.
        ii. When an invalid move is made.
        iii. When a move is made.
        iv. When the game is over.
    c. There must be a means for a player to select a square in which to move.
    d. There must be a means for a player to restart the game at any time.
    e. You ***do not*** need to write JUnit unit tests for your application.