LAB #7 – Tic-Tac-Toe: 2-D arrays & OOP

You will learn to write an object-oriented program (OOP) with a class acting as a blueprint for playing Tic-Tac-Toe. In this program, you will need to think about the states and behavior for playing tic-tac-toe. You will write the TicTacToe class and create an instance of the class, i.e. a TicTacToe object, in the PlayGames client. Write a TicTacToe class containing the constructors, attributes, and methods for playing tic-tac-toe and a PlayGames client that uses the class to play the tic-tac-toe game.

1. Create a TicTacToe.java file containing the proper states for playing the game. For example, you will need two players and a board. You will probably need other class attributes/properties for playing the game, but this is an exercise left for you as part of Lab #7 and Assignment #5.

   ```java
   public class TicTacToe {
       char board[][];
       char player1;
       char player2;
       ...
   }
   ```

2. You will need to have two constructors, one which is a default constructor, TicTacToe(), and another constructor that takes the input for player 1 and player 2 choices as its parameters, TicTacToe(char player1, char player2). As you can see in this example, you will need to have some method for initializing the board to spaces to start. You will probably need to construct the other class attributes/properties for playing the game, but this is an exercise left for you as part of Lab #7 and Assignment #5, and you will need to fill in the details of the TicTacToe(char player1, char player2) constructor.

   ```java
   public TicTacToe() {
       player1='x';
       player2='o';
       board = new char[3][3];
       initializeBoard();
       ...
   }
   public TicTacToe(char player1, char player2) {
       ...
   }
   ```

3. Now, create methods for the tic-tac-toe that allow the board to be initialized to spaces, initializeBoard(), the players to pick their pieces, i.e. ‘X’ or ‘O’, determinePlayerChoice(), the board to be filled with the players’ choice, fillBoard(), the board to be printed to the screen after each user’s turn,
printBoard(), the board to be checked if it is full, isFull(), a check to determine a winner, checkForWinner(), and the results of the game to be printed to the screen, printWinnerResults().

Extras:
- Begin implementing error handling for:
  - Player chooses a position that is not on the board, i.e. row 5, column 2.
  - Player chooses a position that is already occupied.
  - Player doesn’t enter an ‘X’ or an ‘O’.