ICS4U Module 5: Note

Two-Dimensional Arrays

- Two-dimensional arrays are used to represent data that corresponds to a grid. Such as:
  - A checkerboard
  - The streets in a city
  - Seats in a theatre
  - A tic-tac-toe board – like this representation:

	0	1	2
0	X	O	
1		O	X
2			

Declaration

Form:

```
<type>[][] <name>;
<name> = new <type>[<num>][<num>];
```

Notice the TWO sets of square brackets

Example:

```
String[][] tttBoard = new String[3][3];
```

Three rows, three columns

Length Propery

```
rows = tttBoard.length;
cols = tttBoard[0].length;
```

length of array is the # of rows

length of the first row is the # of cols

Accessing Elements

```
tttBoard[1][2] = "X";
```

assigns the element in the 2nd row, 3rd column to „X“

Nested for Statements: often used to access the elements because one loop counter indicates the rows, and the other indicates the columns

This nested for loop displays the contents of the 2D array

```
for (int row = 0; row < tttBoard.length; row++) {
      for (int col = 0; col < tttBoard.length; col++) {
            System.out.print(tttBoard[row][co  l]);
      }
      System.out.println();
}
```

Tic-Tac-Toe client

```
/* TicTacToe.java from Module 5
 * Plays a game of tic-tac-toe between two users.
 */

 /**
 * Tic-tac-toe is played.
 */

 public class TicTacToe {

      public static void main(String[] args) {
            TTT TTTGame = new TTT();
```

```
                  TTTGame.play();
      }
}
```

Tic-Tac-Toe Class

```

/**
 * TTT class.
 */

import java.util.Scanner;

public class TTT {
      private String[][] tttBoard;
      private String player1, player2;


      /**
       * constructor
       * pre: none
       * post: tttBoard has been initialized. player1 is X and player2 is O.
       */
      public TTT() {
            player1 = "X";
            player2 = "O";
            tttBoard = new String[3][3];
            for(int row = 0; row < tttBoard.length; row++) {
                  for (int col = 0; col < tttBoard[0].length; col++) {
                        tttBoard[row][col] = " ";
                  }
            }
      }


      /**
       * Plays a game of tic-tac-toe with two users, keeping track
       * of player (X or O) turns. player1 goes first.
       * pre: none
       * post: A game of tic-tac-toe has been played.
       */
      public void play() {
            String currPlayer = player1;
            int movesMade = 0;

            do {
                  displayBoard();
                  makeMove(currPlayer);
                  movesMade += 1;
                  if (currPlayer == player1){
                        currPlayer = player2;
                  } else {
                        currPlayer = player1;
                  }
            } while (movesMade <= 9 && winner() == " ");
            displayBoard();
            System.out.println("Winner is " + winner());
      }


      /**
```

```
         * Displays the board.
         * pre: none
         * post: The tic-tac-toe board has been displayed.
         */
        private void displayBoard() {
                for(int row = 0; row < tttBoard.length; row++) {
                        for (int col = 0; col < tttBoard[0].length; col++) {
                                System.out.print("[" + tttBoard[row][col] + "]");
                        }
                        System.out.println();
                }
        }


        /**
         * Prompt user for a move until a valid move has been made.
         * pre: none
         * post: A mark has been made in an empty tic-tac-toe board square.
         */
        private void makeMove(String player) {
                Scanner input = new Scanner(System.in);
                boolean validMove = false;
                int row, col;

                do {
                        System.out.print("Enter row number (0, 1, 2): ");
                        row = input.nextInt();
                        System.out.print("Enter column number (0, 1, 2): ");
                        col = input.nextInt();
                        if ((row >= 0 && row < tttBoard.length &&
                                col >= 0 && col < tttBoard[0].length) &&
                                tttBoard[row][col].equals(" ")) {
                                tttBoard[row][col] = player;
                                validMove = true;
                                } else {
                                        System.out.println("Invalid move.  Try
again.");
                                }
                } while (!validMove);
        }


        /**
         * Determine winner. Return " " if no winner.
         * pre: none
         * post: X, O, or " " has been returned as the winner.
         */
        private String winner() {

                /* test rows */
                for (int row = 0; row < tttBoard.length; row++) {
                        if (tttBoard[row][0].equals(tttBoard[row][1]) &&
tttBoard[row][1].equals(tttBoard[row][2]) &&
                                !(tttBoard[row][0].equals(" "))) {
                                        return(tttBoard[row][0]);
                        }
                }

                /* test columns */
```

```
            for (int col = 0; col < tttBoard[0].length; col++) {
                  if (tttBoard[0][col].equals(tttBoard[1][col]) &&
tttBoard[1][col].equals(tttBoard[2][col]) &&
                        !(tttBoard[0][col].equals(" "))) {
                              return(tttBoard[0][col]);
                  }
            }

            /* test diagonal */
            if (tttBoard[0][0].equals(tttBoard[1][1]) &&
tttBoard[1][1].equals(tttBoard[2][2]) &&
                  !(tttBoard[0][0].equals(" "))) {
                  return(tttBoard[0][0]);
            }

            /* test other diagonal */
            if (tttBoard[0][2].equals(tttBoard[1][1]) &&
tttBoard[1][1].equals(tttBoard[2][0]) &&
                  !(tttBoard[0][2].equals(" "))) {
                  return(tttBoard[0][2]);
            }

            return(" ");
      }
}
```