CSCI 1101 Computer Science II

Assignment No. 2

Date Given: Friday, February 10, 2017

Due: Monday, February 27, 2017, 11.55 p.m.

Submission: On Brightspace

Please follow all the instructions carefully. You must implement the classes as specified in the UML diagram. You may add other methods if needed, however, all the methods and the variables indicated in the UML diagram must be implemented. You must submit one zip file containing the source codes (.java files) and a text document with sample outputs. All your programs must be nicely formatted and commented.

Develop the code in a step-by-step manner. Each correctly developed code piece will get you points.

The objective of this assignment is to simulate a game of tic tac toe. For those of you who are not familiar with the game, you can check out the following website:
(http://www.exploratorium.edu/brain_explorer/tictactoe.html).
Figure 1 below shows some sample tic toe toe boards with some input.

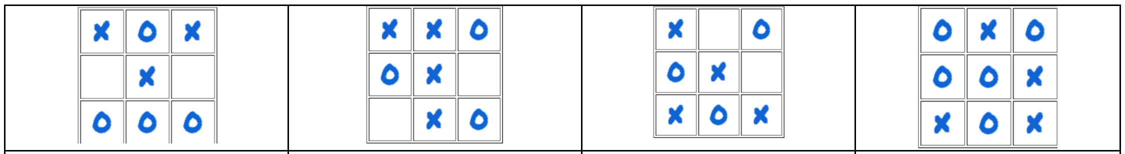
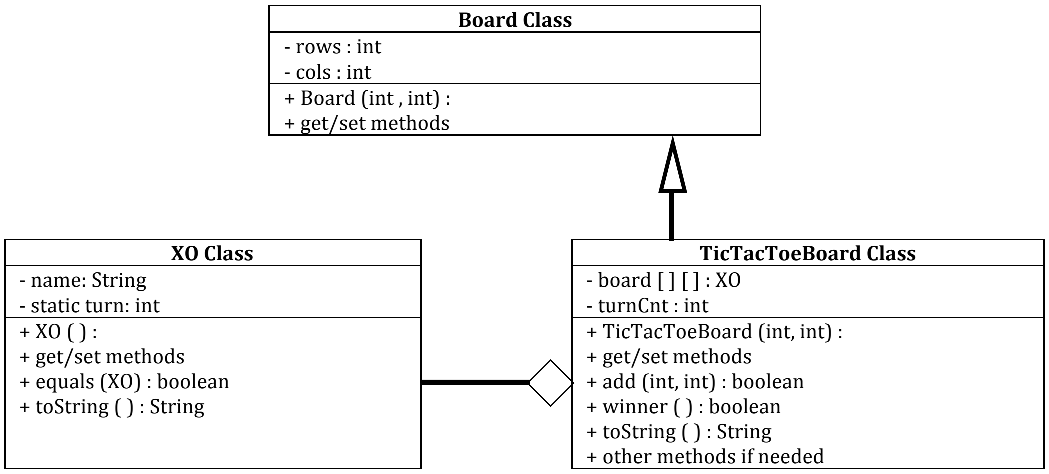

Win (O's Row) | Win (X's Column) | Win (X's Diagonal) | No winner (Tie)

Fig. 1 Tic Tac Toe Boards and ways to win or tie (i.e., no winner)

You will be using the concepts of inheritance and aggregation in object-oriented programming. You will also need to use 2-d arrays to represent the grid of the tic tac toe board. 2-d arrays are similar to 1-d arrays except that they use two indices. There is a small tutorial on 2-d arrays at the end of this assignment. Check it out.

There are three main classes (Board, TicTacToeBoard, XO) and one additional class called Play that will use and run the classes (that is, play the game). See the UML diagram below to see how the classes fit together.



XO Class - this class will hold information about an "X" or an "O" object.
This class should have two attributes: a name (which is a String “X” or “O”) and a static int variable that is either set to 0 or 1 (it should be initialized to 1). You will use this variable to determine which name to give the object when it is created (e.g., if it is 1, set the name to "X").

You need to implement a no args constructor that sets the name of the object to either X or O based on the static variable (and update the variable). You also need to implement:

- the respective get and set methods
- an equals method to see if this XO object is the same as another XO object
- a toString method that returns the name of the XO object

Board class – this is a class used to implement a variety of different boards for different games (e.g., a chess board, checkers board, and in our case a tic tac toe board). This class has two private attributes: the number of rows and number of columns of the board.

It also has the following methods:

- It has a constructor that takes in and initializes the rows and columns
- The appropriate get/set methods
  You may add other methods if required.

TicTacToeBoard
This class extends the Board class.
It has two private variables:

- A 2D array made up of XO objects with the size of Boards' rows and columns
- An int that keeps track of how many turns (e.g., how many X's and O's have currently been placed on the board.)

Methods

- The constructor takes in two ints representing the rows and columns. It initializes the 2D array to the size of rows and columns, and sets count of turns to 0.
- There are appropriate get and set methods
- It also implements. the following methods:
  - The add method will add a new XO object onto the board (add to the 2D array). The method takes in two numbers representing which row and column you want to add the object. It returns a boolean if the object was successfully added. It will need to make sure that the passed in rows and columns are within the size restrictions of the board. It will also need to make sure that the place on the board doesn't already have an XO object.
  - The winner method checks to see if there is a winner (returns true or false). The method will need to check for wins by rows, columns and across the diagonals. See Fig. 1 for the winning combinations.
  - A toString method that prints out what the current board looks like. The toString method should print out the board similar to the sample output (at the end of the assignment). And as XOs are added to the board, these should also be printed.

You may add other methods in this class (e.g., to check for the different winning combinations).

Play class
This class will allow users to play the game (e.g., the demo class). You will create a new TicTacToeBoard. You should use a Scanner object to get the names of your two players and to get the locations of their respective X and O's on the board. The game should continue until there is a winner or there is a tie (ie., no more places to put an XO). Make sure you do proper error checking. See below for a sample output of a game. Your output should show the different conditions.

```
----jGRASP exec: java Play
Player 1 name: Bob
Player 2 name: Sara

Bob - you are X's and you go first.

                    Col
                    1         2         3
Row       1

          2

          3

Please input a row# between 1-3 and column# between 1-3: 2 2
                    Col
                    1         2         3
Row       1

          2                   X

          3

Sara please input a row# between 1-3 and column# between 1-3: 1 1
                    Col
                    1         2         3
Row       1         O

          2                   X
Bob please input a row# between 1-3 and column# between 1-3: 1 2
                    Col
                    1         2         3
Row       1         O         X

          2                   X

          3

Sara please input a row# between 1-3 and column# between 1-3: 1 3
                    Col
                    1         2         3
Row       1         O         X         O

          2                   X

          3

Bob please input a row# between 1-3 and column# between 1-3: 3 2
                    Col
                    1         2         3
Row       1         O         X         O

          2                   X

          3                   X

Bob you win!!
```

Short Tutorial on 2-d arrays

2-d arrays are very similar to 1-d arrays except that they have two subscripts or indices, one representing the row number and the other representing the column number.

For example,
```
int[][] a = new int[5][5];
```

creates a 2-d array with 5 rows and 5 columns.

You can process the 2-d array in the same manner. Instead of one for loop, we use a nested for loop. For example,

```
for(i=0;i<5;i++)
{
      for(j=0;j<5;j++)
            System.out.print(a[i][j] + "\t");
      System.out.println();
}
```

will print the contents of the array as a 5X5 matrix.

The boxes will be numbered as follows:

	Column 0	Column 1	Column 2	Column 3	Column 4
Row 0	a[0][0]	a[0][1]	a[0][2]	a[0][3]	a[0][4]
Row 1	a[1][0]	a[1][1]	a[1][2]	a[1][3]	a[1][4]
Row 2	a[2][0]	a[2][1]	a[2][2]	a[2][3]	a[2][4]
Row 3	a[3][0]	a[3][1]	a[3][2]	a[3][3]	a[3][4]
Row 4	a[4][0]	a[4][1]	a[4][2]	a[4][3]	a[4][4]

If you want to print all the elements in Column 2, for example:
```
for(i=0;i<5;i++)
            System.out.println(a[i][2]);
```

Similarly, the following will print the elements in Row no.3
```
for(j=0;j<5;j++)
            System.out.print(a[3][j] + "\t");
```
The following program creates a 2-d array, reads 25 integers from the keyboard, prints it as a 5X5 matrix and finds the sum of all elements. Note that the 25 integers can be entered all on a single line when they are read from the keyboard. Try it out.
```
import java.util.Scanner;
public class TwoDArray
{
      public static void main(String[] args)
      {
            int[][] a = new int[5][5];
            int i,j, sum=0;
            Scanner keyboard = new Scanner(System.in);
```

```
          for(i=0; i<5;i++)
               for (j=0;j<5; j++)
                    a[i][j]= keyboard.nextInt();

          for(i=0;i<5;i++)
          {
               for(j=0;j<5;j++)
                    System.out.print(a[i][j] + "\t");
               System.out.println();
          }

          for(i=0;i<5;i++)
               for(j=0;j<5;j++)
                    sum+=a[i][j];
          System.out.println("The sum of all elements is: " + sum);
     }

}
```

Array initializer expression for 2-d arrays

A 2-d array can be created using an array initializer expression. For example:

```
int[][] numbers = {{1,2,3,4},{5,6,7,8},{9,10,11,12}};
```

creates the following array

```
1    2    3    4
5    6    7    8
9    10   11   12
```

Length field in a 2-d array

A 2-d array has a length field that holds the number of rows, and each row has a length field that holds the number of columns.

The following program uses the length fields of a 2d array to display the number of rows, and the number of columns in each row. Try it out.

```
public class Lengths
{
     public static void main(String[] args)
     {
          int[][] numbers = {{1,2,3,4},{5,6,7,8},{9,10,11,12}};
          System.out.println("The number of rows is " + numbers.length);
          for(int index=0; index<numbers.length; index++)
               System.out.println("The number of columns in row " +
     index + " is " +  numbers[index].length);
     }
}
```