Assignment 6: Two-Dimensional Arrays

Submit Assignment

Due Nov 9 by 5pm **Points** 10 **Submitting** a file upload
Available until Nov 14 at 9am

Programming Assignment 6

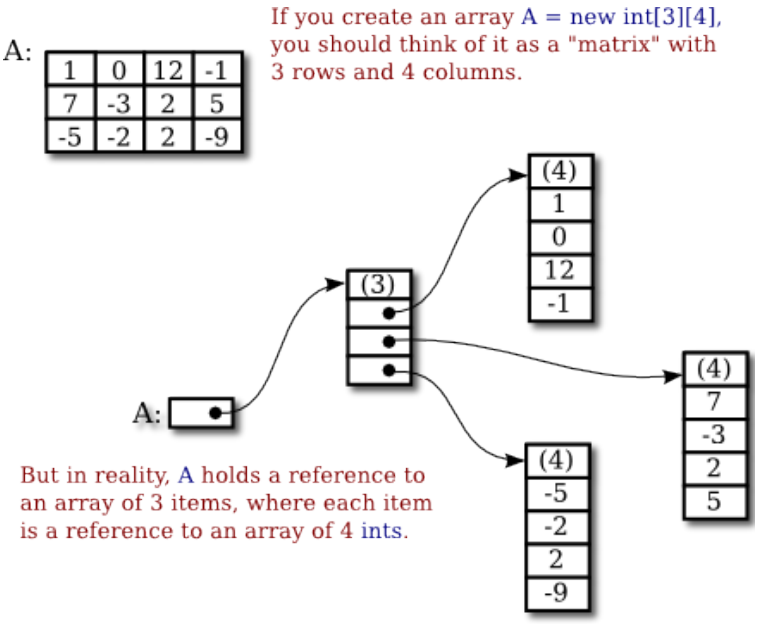
Tic-Tac-Toe using 2-D Array

In this assignment, you will design and implement the game of tic-tac-toe using a 2-D array.

Purpose
Demonstrate your ability to represent data using a 2-Dimensional array and program common operations for 2-D arrays.

Background Thought

Before we go any farther, there is a little surprise. Java does not actually have two-dimensional arrays. In a true 2D array, all the elements of the array occupy a continuous block of memory, but that's not true in Java. The syntax for array types is a clue: For any type BaseType, we should be able to form the type BaseType[], meaning "array of BaseType." If we use int[] as the base type, the type that we get is "int[][] meaning "array of int[]" or "array of array of int." And in fact, that's what happens. The elements in a 2D array of type int[][] are variables of type int[]. A variable of type int[] can only hold a reference to an array of int. So, a 2D array is really an array of references, where each reference can refer to a one-dimensional array. Those one-dimensional arrays are the rows of the 2D array.



Instructions/Design

In a game of tic-tac-toe, 2 players take turns marking an available cell in a 3x3 grid with their respective tokens (either X or O). When one player has placed 3 tokens in a horizontal, vertical, or diagonal row on the grid, the game is over and that player has won. A draw (no winner) occurs when all the cells on the grid have been filled with tokens and neither player has achieved a win. Create a program for playing tic-tac-toe.

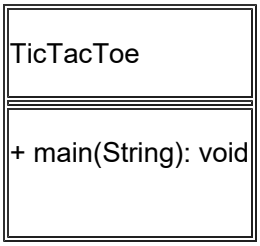
The program prompts 2 players to enter an X token and O token alternately. Whenever a token is entered, the program redisplays the board on the console and determines the status of the game (win, draw or continue).

YOU MUST USE THE FOLLOWING TWO CLASSES IN YOUR SOLUTION:

TicTacToe Class

This class is the driver program. It contains one method: main() which uses the GameBoard class to play the game of Tic-Tac-Toe. It also contains a flag that is set when the game is over (win or draw).

Here is the UML:



Here is the algorithm:

```
displayBoard
while (keepPlaying)
    makeAMove X
    diplayBoard
    if (isWon X)
        X player won
    else if (isDraw)
        No winner
```

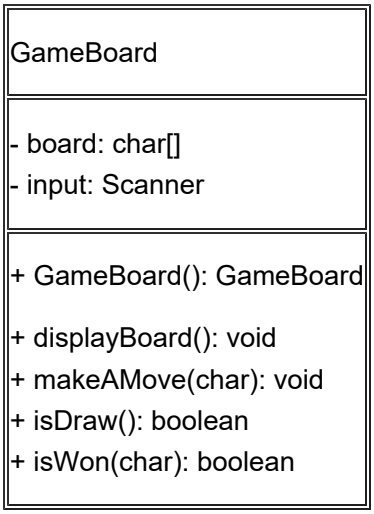
```
    if (keepPlaying) //no need to continue if X won/draw
        displayBoard
        makeAMove O
        if (isWon O)
            O player won
        else if (isDraw)
            No winner
```

GameBoard Class

This class holds the reference to the 2-D array representing the board and contains all the methods needed to display the board, have one user make a move, check if there is a winner, and check if there is a draw.

The constructor should use an initializer list to initialize each cell to a character of your choice (I used '$' but you can use anything even blank ' ').

Here is the UML:



The ONLY classes needed are: **GameBoard** and the driver**: TicTacToe**

Input

One user acts as BOTH player X and player O and is prompted to enter the row and then the column they want. Example (user input in bold):

Enter a row for player X: ***1***
Enter a column for player X: ***2***

Hint:
Do one method at a time and make it work before moving on to the next method. Start with displaying the board.

Sample Output

Your output need not look exactly link mine.
To see your output, I will attempt to download and then unzip your Assignment6.zip file from the file you submit onto Canvas.

I will then run your program on my local machine using the command:
`java TicTacToe`

```
---------------
| $ | $ | $ |
---------------
| $ | $ | $ |
---------------
| $ | $ | $ |
---------------
Enter a row for player X: 1
Enter a column for player X: 1

---------------
| $ | $ | $ |
---------------
| $ | X | $ |
---------------
| $ | $ | $ |
---------------
Enter a row for player O: 1
Enter a column for player O: 1
This cell is already occupied. Try a different cell
Enter a row for player O: 0
Enter a column for player O: 0

---------------
| O | $ | $ |
---------------
| $ | X | $ |
---------------
| $ | $ | $ |
---------------
Enter a row for player X: 0
Enter a column for player X: 1

---------------
| O | X | $ |
---------------
| $ | X | $ |
---------------
| $ | $ | $ |
---------------
Enter a row for player O: 1
Enter a column for player O: 0

---------------
```

```
| O | X | $ |
---------------
| O | X | $ |
---------------
| $ | $ | $ |
---------------
Enter a row for player X: 0
Enter a column for player X: 2

---------------
| O | X | X |
---------------
| O | X | $ |
---------------
| $ | $ | $ |
---------------
Enter a row for player O: 2
Enter a column for player O: 0

---------------
| O | X | X |
---------------
| O | X | $ |
---------------
| O | $ | $ |
---------------
O player won
```

Note:
User is NOT allowed to choose a cell that is occupied with a X or 0 already.
This check - using a loop - is in makeAMove()

Submitting Your Work FOR CREDIT

Once you have the code working on your own machine, submit your assignment on Canvas (to avoid late penalty submission must be completed no later than 5pm on due date). But first...

a) ***zip*** your source code and byte code files into an archive named **Assignment6.zip**

- **GameBoard.java** and **GameBoard.class**
- **TicTacToe.java** and **TicTacToe.class**

b) At bottom of this page browse for and then upload **Assignment6.zip** to submit.
NOTE: All group members MUST submit the same zipped for to receive a grade.

c) Don't forget all classes/methods need javadoc comments and you need an algorithm at top of driver program in comment

Note: If you do not submit assignment successfully using **Canvas**, I cannot give you a grade. I cannot give you a grade unless your code compiles.
To avoid late penalty, submission must be completed by 5pm on due date. GIVE YOURSELF TIME TO SUBMIT.

Code must compile to be given any credit.