Lab 3

NOTE: For this lab, you will get 5 points for a valid attempt if turned in by the due date.

The main new component in this lab, multidimensional arrays, are in Chapter 7 and were reviewed in class.

TIP: Remember, don't be overwhelmed by all the details. Simply focus on one line at a time as the instructions walk you through the creation of the app.

\*\*\* Instead of "work" or an "assignment"... THINK OF THIS AS A FUN PROGRAMMING CHALLENGE! Never forget, coding is fun! Seriously, that's like in the Bill of Rights or something like that. \*\*\*

1. **SUMMARY**
   1. Build an app that randomly creates a 10 by 10 grid of 0s and 1s where the 0s represent open spaces and 1s represent blocked spaces (like we did in class), and then see if user can navigate from position 0,0 to where either the first index or second index equal 9 w/o hitting a 1. In other words the user must move clear across the grid without hitting a 1 or the game is over.
   2. This lab will involve the following new features:
      1. java.util.SecureRandom class
      2. Two-dimensional arrays
2. **DETAILS**
   1. There won't be as much detail on the grid creation part of this as we already created an example of the grid in class together.
   2. Classes
      1. **GameGrid\_Test**
         1. Will have main method and will instantiate instance of GameGrid and call run... that's it.
      2. **GameGrid**
         1. This will have one public "run" method and this will be called by the GameGrid\_Test class's main method. ALL CODE can be in the "run" method.int fi
            1. Create game grid: This method will create a game grid (two-dimensional int array) and fill each cell with 1 or 0 randomly using the Random class.

There should be a **LESS CHANCE** to have a 1 fill the spot than a 0.

Reminder Tip:

Set an int variable called iWallChance to a number like 30, and then use the bound parameter of the SecureRandom class’s nextInt method to control the range of possible numbers just like we did in class.

Then you'd compare the returned random number to see if it's less than your iWallChance variable.

If it was less, you'd set that cell to 1... if not, then the cell would be set to 0.

* + - * 1. Change the location [0][0] of the grid to 0 so that the first space is open and not a wall because that's where we'll start our user.
        2. Create two int variables for the user's position... iUserRow and iUserCol... and set each to 0.
        3. Create a boolean variable for the following loop to know when to exit.
        4. Create while loop for moving: This loop will keep asking user whether to move down or to the right.

If the user chooses down, then that will affect the first array value (row); and if the user chooses right, that will affect the second array value.

With each user answer, adjust the iUserRow or iUserCol positions depending whether down or right by increasing the correct variable by 1.

Check if this new user’s y and x position in the grid array (e.g. aiGrid[iUserRow][iUserCol]) equals a 1. If it's a 1, tell the user they failed and exit loop.

If the value in the array at the user’s position was not a 1, then check if either iUserRow or iUserCol is now a 9 (if you did a 10 by 10 grid array).

If one of them is 9, they have reached either the right side or bottom and tell them they have won and exit loop.

* + - * 1. After loop is over, then print out the grid as we did in class (nested for loop (which is a for loop within a for loop)).

With each cell, before you print out the 0 or 1 from the array position, check if that cell’s value (y and x from your for loop counters) equals the user position (iUserRow and iUserCol).

If it does, print out a capital X in that spot instead of what’s in the array at that spot.

Otherwise, simply print out the value of that spot in the array.

Printing tip: Remember in class that we used System.out.print... within the inner for loop for printing so no line break was done. Then use System.out.println(""); after the closing curly bracket of the inner for loop but before the closing curly bracket of the outer for loop. That way each row will be printed out on a separate line for a nice readable layout.

1. **TURNING IN LAB:** 
   1. Turn your lab into Canvas.
   2. IMPORTANT: DO NOT ZIP your folders/files please.
   3. Turn in just the two raw java files for this lab.