Objectives:
More practice using two dimensional arrays to process values of different types.

Preparation

Review the code and modify it as described below.

```java
public class TwoDArray
{
    public static void main (String[] args)
    {
        int[][] table = new int[5][10];
        // Load the table with values
        for (int row=0; row < table.length; row++)
            for (int col=0; col < table[row].length; col++)
                table[row][col] = row * 10 + col;
        // Print the table
        for (int row=0; row < table.length; row++)
        {
            for (int col=0; col < table[row].length; col++)
                System.out.print (table[row][col] + "	");
            System.out.println();
        }
    }
}
```

1. Run this program and observe what it does.
2. The output produced is shown below. Circle the entries for `table[0][5]` and for `table[3][2]`

```
  0  1  2  3  4  5  6  7  8  9
10 11 12 13 14 15 16 17 18 19
20 21 22 23 24 25 26 27 28 29
30 31 32 33 34 35 36 37 38 39
40 41 42 43 44 45 46 47 48 49
```

3. Modify the program so that the output rows and columns are labeled. The output should now look EXACTLY like this:

```
#   0  1  2  3  4  5  6  7  8  9
0  10 11 12 13 14 15 16 17 18 19
1  20 21 22 23 24 25 26 27 28 29
2  30 31 32 33 34 35 36 37 38 39
3  40 41 42 43 44 45 46 47 48 49
```

4. Modify the dimensions of the array – make it 3 rows by 4 columns and run the program again. The output should still look OK without having to change anything else in the program. If necessary, adapt your program so that it works with any reasonable dimensions (note that there is a limit to how many columns can be displayed across on one line, so it is not expected to work well with large values for the number of columns).

Submit your updated version of `TwoDArray.java` through Blackboard by 8:00am the morning of the Lab.
A. 2D array of double

1. Make a new version of TwoDArray.java that creates an 2D array of 5x5 values of type double, set to random values in the range 0...1 (use Math.random()).

2. Examine your code: you should still have two nested loops (one to initialize the 2D array and one to print its contents). Add a third nested loop to process the array entries one more time, this time counting how many are greater than 0.5 (since the entries were generated randomly, it should be about half of them), and some code to display the count. Run the program a few times to note the counts below:


3. Increase the array size to a 100x100 and comment out or delete the code that outputs the array (ie, we want to avoid huge amount of output – we are only interested in the number of elements greater than 0.5, at this point). Run the program a few more times and note the counts again:


4. Incorporate a timer in your program. How much longer does it take to run for different size arrays? You can try 200x200, 300x300, 500x500, 1000x1000, etc and observe the times in milliseconds.

Array size: 
Milliseconds: 


At some point, the program will run out of space.
How large an array can your computer handle? 
Is it the same for your lab partner? 

Do you sense that there is a formula for the approximate number of milliseconds required for an NxN array, in terms of N?
A. 2D array of boolean

1. Make a new version of TwoDArray.java that creates a 2D array of 3x3 values of type boolean. Use an initializer list to instantiate and initialize the array as follows:

```java
boolean[] table = {
    {false, true, false},
    {true, false, true},
    {true, true, false}
};
```

Verify that when the array gets printed, with the correct values before proceeding.

2. Change the way that the array is displayed, so that instead of printing true/false, it prints an asterisk just in the true positions. For the values above, it should print the table as follows:

```
# | 0   1   2
---+-------
0 |   *   *
1 | *   *   *
2 |   *   *
```

3. Modify the initialization of the array, so as to obtain the following configuration:

```
# | 0   1   2   3
---+-------+-
0 |   *   *   *
1 | *   *   *
2 |   *   *
3 |   *   *
```

4. Suppose indices represent people and that the value at row i, column j of this 2D array is true just in case i likes j, and false otherwise. For example, person 3 likes persons 1 and 2, but does not like person 3(self). Write some code to output all the matches contained in the array; a match means that person x likes person y and vice-versa. In the above example we have two matches: (0,1) and (1,2).

**Hints:**
- Use a nested loop to go through all possible combinations of people,
- An if statement can checks whether a particular combination represents a match
- Try to avoid counting the matches twice!

3. (Optional) Experiment with a large array of this kind representing a large number of users. Set the values randomly and explore the number of matches obtained.
Lab 12 Comments

Comments on this lab, please:

What was the most valuable thing you learned in this lab?

What did you like best about this lab?

Was there any particular problem?

Do you have any suggestions for improving this lab as an effective learning experience?