

# Programming I

## Final lab

**Instructions:** For programming exercise 1, turn in your modified BaseBoard.java file (with the `drawString` method) and a separate file with a main method that invokes `drawString`. For programming exercise 2, turn a single .java file containing all 6 methods and main. For programming exercise 3, turn in your modified BaseBoard.java file and a separate file with a main method that drives the animation.

### Programming Exercise 1 (3 points)

For this exercise you are to create a method in the `Board` class (in `BaseBoard.java`) that *draws* a `String` (read the explanation on `Strings` below) on a `Board` object by placing each of its characters in a cell, moving from left to right. The coordinate of the first character in the `String` in the `Board` is specified using parameters. The method should check as it draws the `String` if it will exceed the boundaries of the `Board` object, in which case it will clip any overflowing characters. The header for the method should be:

```
public void drawString(String str, int x, int y)
```

`str` is the `String` to be drawn on the `Board` object, and parameters `x` and `y` represent the starting position of the `String`. To test your method, create a separate class containing a `main` method with the code:

```
Board b = new Board(10,10);  
b.drawString("Hello", 2, 0);  
b.drawString("New Zealand", 5, 5);
```

The `Board` object should look like:



### Strings

A `String` in Java is a data type that represents a sequence of characters. A `String` value (or a `String` literal) must be surrounded by double quotes. Examples of `String` literals are:

```
"John Doe !@#$%^&*("  
"a"  
"12345" // this is not treated as a numeric value
```

Although `String` is a class in Java, we can declare and initialize a `String` variable just as we declare integers or doubles:

```
String country = "Tanzania";  
String zipCode = "11220";
```

```
String phone = "203-333-1122";
```

To obtain the number of characters in a `String`, we can invoke the method `length` on a `String` variable:

```
int len = country.length(); // country is the variable declared above
System.out.println(country + " has " + len + " characters in it.");
```

The code's output will be:

```
Tanzania has 8 characters in it.
```

Given a `String` variable, it's often useful to be able to extract the individual characters from it. To make that possible, Java associates a position with each of the characters in a `String`. Given `String` literal "Germany", the character 'G' is at position 0, 'e' at position 1, and 'y' at position 6 (Notice that the position of the last character in a `String` is one less than its length, which is 7 in this case). To extract a character out of a `String`, we can the method `charAt` which takes as a parameter a position and returns the character at that position. Consider this code:

```
String band = "Smashing Pumpkins";
char ch = band.charAt(3);
System.out.println("The fourth character in "+ band + " is " + ch);
```

The second statement invokes `charAt` with argument 3 (the 4<sup>th</sup> character), which returns 's'. The return value is stored in `ch` before it's printed out in the following statement.

Given a `String` variable, we can extract all of its characters one at a time (that's what you'll need to do for exercise 2), by using a loop with the variable being the position in the `String`. At every iteration, the position is incremented by one to extract the next character in the `String`. The following code prints out every character on a separate line:

```
String band = "Coldplay";
int len = band.length; // obtain the length of the String

// loop as long as position is one less than the length of the String
for (int position = 0; position < len; position++)
{
    char ch = band.charAt(position);
    System.out.println(ch);
}
```

The code's output will be:

```
C
o
l
d
p
l
a
y
```

## Programming exercise 2

For this exercise, you are to create 6 methods that perform operations on a two-dimensional array (a table) of integers and incorporate them as part of a program that demonstrates that the methods work properly. Your program must use the method headers and the main method provided. The array operations you will implement are as follows:

- 1) Display the contents of the array (1 point):

Method header: `public static void printTable(int[][] table)`

This method prints each row in the table on a separate line, where the elements are separated by a space.

- 2) Put random integers in the table (1 point):

Method header: `public static void putRandomValues(int[][] table, int max)`

This method creates random integers in the range 0 to `max` and places them in the table.

- 3) Sum elements (1 point):

Method header: `public static int sumTable(int[][] table)`

This method sums all the values in the table. The method takes a two-dimensional array as a parameter and returns the sum.

- 4) Mirror table elements (2 points):

Method header: `public static void mirrorTableVertical(int[][] table)`

This method mirrors the elements in the left half of a table onto the right half. For instance, given a table with the values

```
1 2 3 4
5 6 3 4
7 2 3 4
```

the method will transform it to:

```
1 2 2 1
5 6 6 5
7 2 2 7
```

- 5) Shuffle rows (2 points)

Method header: `public static void shuffleRows(int[][] table)`

This method repeatedly exchanges the contents of two randomly selected rows (i.e. it's the order of the rows that's changed, not their contents).

For instance, a possible transformation of the table with the values

```
5 4 2
2 4 3
9 2 5
```

is

```
2 4 3
9 2 5
5 4 2
```

6) Sort elements in increasing order (2 points) -- **challenge question**

Method header: `public static void sort(int[][] table)`

This method rearranges the elements so that the smallest is in the top left cell and the largest is in the bottom right cell. Use the selection sort method developed in class as a starting point.

For instance, the table

```
8 4 3
6 5 2
9 1 7
```

would be transformed to

```
1 2 3
4 5 6
7 8 9
```

Finally, the main method in your program should look like:

```
public static void main(String[] args)
{
    int max = 100;

    Scanner input = new Scanner(System.in);
    System.out.print("Enter the number of table columns: ");
    int cols = input.nextInt();

    System.out.print("Enter the number of table rows: ");
    int rows = input.nextInt();

    // create a table accordingly
    int [][] table = new int[rows][cols];

    putRandomValues(table, max);

    System.out.println("Here are some random values in the table: ");
    printTable(table);

    System.out.println("The sum of table values is " + sumTable(table));

    mirrorTableVertical(table);
}
```

```
System.out.println("The table, mirrored vertically: ");
printTable(table);

shuffleRows(table);
System.out.println("After shuffling some rows: ");
printTable(table);

sort(table);
System.out.println("Arranging the elements in increasing order: ");
printTable(table);
}
```

### **Programming exercise 3** (7 points)

Create a board animation that draws on the various programming concepts and techniques that you learned about in this course. Your challenge is to be creative and demonstrate that you can apply your programming knowledge to create a dazzling animation. You can think of your animation as a sequence of visual effects that can each be implemented as a separate method in the Board class. The main method of your program will invoke these methods in a meaningful order with potentially varying arguments in order to create the desired animation. Your animation must at least include 3 distinct effects. Your animation will be evaluated based on the following:

- complexity of the effects
- theme of the animation (e.g. Does the animation flow or are the effects unrelated?)
- variety of programming concepts utilized (e.g. randomness, color, text, nested loops, user interactivity, etc)

Before each method you implement in the Board class, add a comment block describing the effect and the parameters (see the comment block for mouseClicked).