Tutorial 2: Program Flow, Arrays, ArrayLists, Vectors, loops

1. Tutorial Objectives

This tutorial builds on concepts introduced in the previous weeks and assumes you are now comfortable with last weeks ideas and able to perform standard operations in Java such as returning values from methods, modifying arrays through looping and so on.

There are always three levels of difficulty of the tutorial exercises marked with asterisks (*) as follows:

* Easy   ** Medium   *** Hard

It is important to do the tutorial exercises in order as they often rely on a previous question to be completed first. Hints will be marked in italics.

You are expected to be able to finish the easy and medium exercises without any help. However, never hesitate to ask the demonstrator if you have any questions.

2. Exercises

** Exercise 1   Program Flow and Booleans*

Program functions can have multiple return statements that exit the function at different points in the program flow. However, it is usually less readable and prone to bugs and we encourage you to make sure your functions have only single return statement at the end.

```
/**
 * Returns true if given integer is even, false otherwise
 * @param number
 * @return
 */
 public static boolean isEven (int number) {
         if (number % 2 == 0)
            return true ;
         else
            return false ;
            }
```

1. Write a main function which will read a number from console and display result of running *isEven(...)*.
2. Rewrite the *isEven(...)* function so that it has only a single return statement but produces the same output as before.

   Hint: *Create a local variable isEven to store the result.*

3. Note, that setting a local *boolean* variable to *true* if a statement is *true* and *false* otherwise is kind of redundant. Instead, replace the *if-statement* with a single line of code which will assign the local *boolean* variable to the condition of the *if-statement* itself.

   Hint: *So that you are not surprised if you ever come across an inlined if-statement notation like this one: boolean isEven = (number % 2 == 0) ? true : false;, try to understand what it says. It evaluates a conditional question, in this case (number % 2 == 0) and sets the result accordingly. This is just an example of inlined syntax, for this particular case you actually do not need the ? ... part of the code as what is inside the brackets already evaluates to a boolean value.*

4. Rewrite the *isEven(...)* function so that it does not require a local variable, yet it still produces the same result as before.

* Exercise 2 Array Element Removal

Sometimes it is useful to remove a specific element from an array. While arrays in Java are fixed size (so we cannot change their length after they're created) it is possible to create a method that makes it look otherwise.

Write a method that takes a *String[]* and *int* and returns the *String[]* that is the same except it doesn't contain this one element of specified index. For instance, after the following:

```
String[] arr = {"The ", "quick ", "brown ", "fox ", "jumps ",
                "over ", "the ", "lazy ", "dog."};
arr = remElement(arr, 7);
```

arr should contain the sentence "*The quick brown fox jumps over the dog*".

Hint: *As the array's length cannot be changed, you will have to create a new one and copy the appropriate elements into it.*

While possible, this is not the best solution. In general, if you're doing a lot of element removal arrays are probably not the best data structures for you to use. Try to find information about some Java classes that make such operations easier and more efficient.

** Exercise 3 ArrayList and random numbers

We will practice the use of `ArrayList` in this exercise.

- Create an *ArrayList aList* to store 100 random integer values (range 0-100).
- Print out the values in the *aList*.
- Take all the even numbers in *aList* and calculate the sum of the numbers.
- Remove all the odd elements in *aList*. Add them back to the end of the List.
- Create an *Array* of integers out of the new *aList*.
- Use *Arrays.toString* to print out the values of the array.

** Exercise 4

Use java vectors to write a java program that will read a sequence of positive real numbers entered by the user and will print the same numbers in sorted order from smallest to largest. The user will input a zero to mark the end of the input. Assume that at most 50 positive numbers will be entered.

*** Exercise 5 Pascal's Triangle

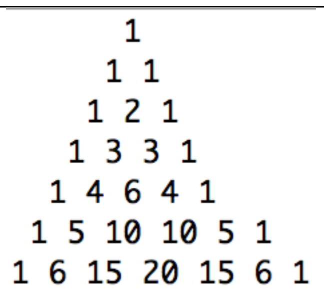
One of the most interesting Number Patterns is Pascal's Triangle (named after Blaise Pascal, a famous French Mathematician and Philosopher).

Write a method to print the following pattern (Figures 1) using nested-loops in a class called *PrintTriangles*. The signature of the methods is:

```
public static void printPascalTriangle(int numRows)
```

Write the *main()* which prompts the user for the *numRows* and ouputs the pattern.

Hint: Please note that the triangle in (Figure 1) is generated using the following simple formula:
https://en.wikipedia.org/wiki/Pascal%27s_triangle#/media/File:PascalTriangleAnimated2.gif



```
      1
     1 1
    1 2 1
   1 3 3 1
  1 4 6 4 1
 1 5 10 10 5 1
1 6 15 20 15 6 1
```

Figure 1: Pascal Triangle

Optional

Read the following description of other patterns of the Pascal triangle and implement some of them.

https://en.wikipedia.org/wiki/Pascal%27s_triangle

~~~ 000~~~
~~~