Introduction to OOP and Text I/O

The key topics in this week are the introduction to Object-Oriented Paradigm, the interactive input via *Scanner* class, and text output via *System.out*.

Task

1. Understand OOP
2. Practice using objects and classes
3. Practice Text IO and text processing
4. Learn how to define classes
5. Practice writing classes
6. Practice writing methods

Part A

These exercises can be done on paper and do not require the use of a computer.

1. Answer the following questions:
   a. What is the difference between a class and an object?
   b. What the differences between a variable of primitive type and a variable of reference type?
   c. In the same class, can two methods have the exact same name? Do the method signatures / headers have to have any difference in them?

2. Write signatures/headers for the following method specifications:
   a. The method is called `findAvg`. It takes in an `int[]` parameter and returns a `double`.
   b. The method is called `findMode`. It takes in an `int[]` and returns an `int`.
   c. The method is called `subString`. It takes in a `String`, and two `int` formal parameters that specify the start position and length of the substring. The method should return a `String`.

3. `BankAccount` is a defined class. Explain the difference between

```
BankAccount b;
```

and

```
BankAccount b = new BankAccount(5000);
```

Part B

Exercise 1: Simple Text Processing

(1) **Count the words**: Write a program to read a paragraph; count the number of a given word in the paragraph; display the word and its count.

(2) **Validate names**: Write a program to read and validate names that are in form of given name(s) and surname, all along a line. The names are composed of letters and cannot include numbers and punctuation characters. If the input string is not a valid name, display the string and then comment that this is not a valid name; otherwise, add the valid name into the array of strings and then print the whole list of names in the array.

Exercise 2: Pattern/Format matching!

(1) **Date format**: Write a program to check whether an input date matches the format of "(d)d-

(m)m-(yy)yy". If an input date matches the format, display "true" on the console; otherwise display "false". For example: an input date 11-07-2014 or 2-2-14 matches the format, and "true" will be displayed on the console as the output.

(2) **Salary**: read a list of salary amounts that start with a dollar sign "$" and followed by a non-negative number, save the valid salary inputs into an array and sort the salary in ascending order, calculate the average of the salary inputs, and count the number of inputs less than /greater than the average.

Exercise 3:

Stage 1:

Start a new Java Project named lab04-Ex3. It will ultimately have two source code files.
First, define a new class named City, which will be used to represent basic facts about various cities from around the world.

1. Ensure the class contains definitions for the following attributes (see table on next page for example data):
   - The name of the city
   - The country in which the city is located
   - The population size of the city
   - The Latitude and Longitude of the center of the city
2. Ensure the class has a parameterized constructor that accepts values for assigning to each of the attributes.
3. Ensure that the class defines accessors for all attributes.
4. Ensure that there is a mutator for the population size attribute only – no other attributes are to have a mutator. (Why do you think this is being done?)
5. Write a method named toString() which requires no parameters and that will return a String describing details about the City, in the manner as shown in the following example (all in one line):

Boston (U.S.A.) [42 N, 71 W]: 4590000

Stage 2:

Create a second class in this project, name CityMain, which will be used to test that the City class (from part 1 of this task) is functioning correctly.

1. Define 5 variables to be of the City class type. (If you are comfortable using arrays, you may use an array instead).
2. Assign to each variable, a new City object by hard-coding the values in the code using data from the following table.

City	Country	Latitude	Longitude	Population
Boston	U.S.A.	42 North	71 West	4590000
Sydney	Australia	33 South	151 East	4627345
Johannesburg	South Africa	26 South	28 East	1009035
Kuala Lumpur	Malaysia	3 North	101 East	1627172
Rio de Janeiro	Brazil	22 South	43 West	6323037

3. Ensure the toString() method works for some cities.
4. Using a range of print statements, display individual pieces of information about different cities 'attributes.
5. Ensure that the mutator method works to actually change the population for at least one city. Then use another print statement to show the new value was accepted.