Level	M1
Course	Java introduction
Subject	Lab Work 2
Duration	

***Objective**: Practice Java programming*

- *Create classes, objects*
- *Use some keywords knowing the reason*
- *Define methods*

Exercise 1:

A group of friends decide to run the Lille Marathon. They register to the Course Organization and obtain their Runner number.
Their names and times (in minutes) are below:

Name	Time (minutes)
Nicolas	341
Annie	273
Paul	278
Pierre	329
Corinne	445
Emilie	402
Sonia	388
Stephane	275
John	243
Thomas	334
Sabine	412
Perrine	393
Sebastien	299
Jeanne	343
Amelie	317
Alain	265

A. Create appropriate classes (think Class/object and structure things like they are in reality)
B. Implement methods (think generic, modular and reusability)
C. Find the fastest runner. Print the name and his/her time (in minutes).
D. Find the slowest runner. Print the name and his/her time (in minutes).
E. Find the top 3 runners. Print the names and their time (in minutes).

Exercise 2: Periodic table of chemical elements

The periodic table of chemical elements classifies and displays all chemical elements. Each chemical element has a unique name, symbolic name and atomic number (number of protons). Chemical elements are grouped together by common characteristics (alkali metal, poor metal ...) called the chemical series.

Examples of chemical elements:

- H: Hydrogen with atomic number 1.
- O: Oxygen with atomic number 8.
- K: Potassium with atomic number 19. It is an alkali metal.
- Zn: Zinc with atomic number 30. It is a transition metal.
- Ga: Gallium with atomic number 31. It is a post-transition metal.

We consider the following three types of chemical series:

- *Alkali metals* are all chemical elements with anatomic number equals to 3, 11, 19, 37, 55 or 87
- *Transition metals* are all chemical elements with an atomic number from 21 to 30, 39 to 48, 72 to 80, and 104 to 112 or equal to 57 or 89.
- *Post-transition Metals* are all chemical elements with an atomic number from 30 to 31, from 48 to 50 or from 80 to 83.

Design and implement a class ChemicalElement.

The class should contain methods to retrieve, for a specific chemical element, its name, its symbolic name, its atomic number, and which type of *chemical series* it has.
You should create three different methods to retrieve the type of *chemical series* (one for *Alkali metals*, one for *Transition metals* and one for *Post-transistion Metals*). Each method should return a boolean. Implement these three methods without using any *if* statement:

- one method should use a switch statement
- one method should use a single boolean expression
- one method should use a static boolean-array where the index is the atomic number.

You can initialize the array in the static initializer of the class:

static { ...

}

Make sure that you choose for each of these three methods the best of the above implementations. How do these variants differ with respect to the maintainability and performance of the program?

Add five constants for the above five examples of chemical elements to the class (H, O, K, Zn, Ga).

The values of a chemical element must be unmutable: once a chemical element is constructed, its must not be possible to change its internal state.

Write a program to test the class.

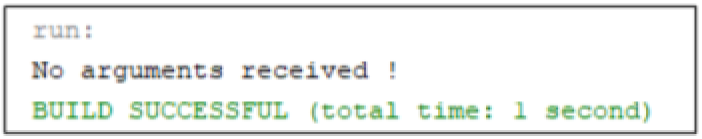
Exercise 3:

Introduction

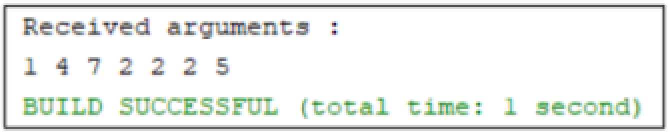
This lab consists in incrementally writing a program that draws (in text mode) a histogram illustrating the repartition of integers provided in command line argument.
At the end of the lab, the drawing of this histogram will be customizable: its orientation (horizontal/vertical) and the symbol used to draw bars could be tuned by options.

1- Text display

Create a java program named *Histogram* that simply displays the values it receives as command line arguments.



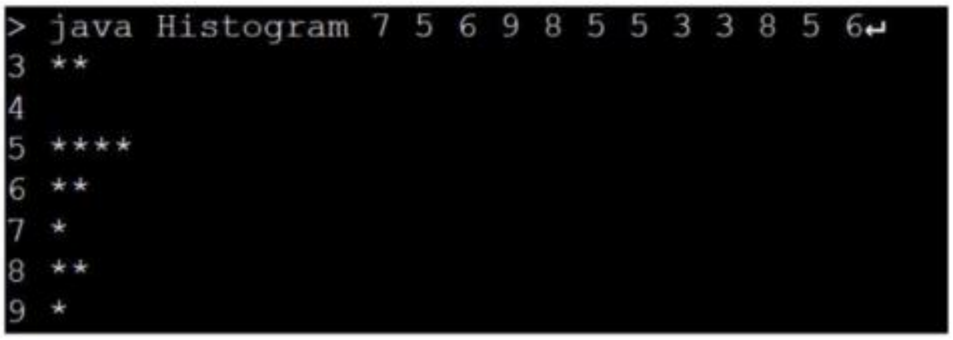
```
run:
No arguments received !
BUILD SUCCESSFUL (total time: 1 second)
```



```
Received arguments :
1 4 7 2 2 2 5
BUILD SUCCESSFUL (total time: 1 second)
```

2- Bar display

Modify your program to add the following feature: the program draws ***horizontal bars*** corresponding to the repartition of the values provided in argument.



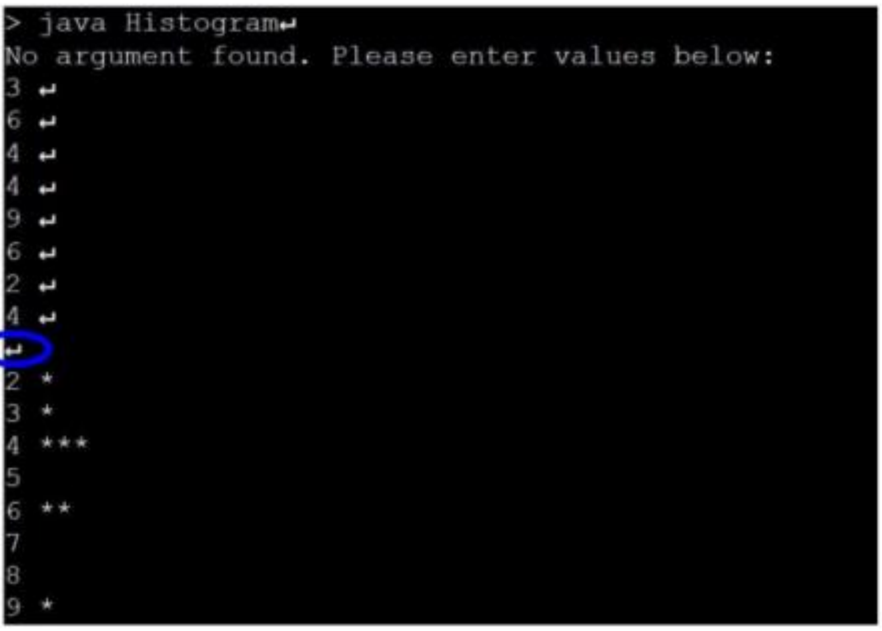
```
> java Histogram 7 5 6 9 8 5 5 3 3 8 5 6↵
3 **
4
5 ****
6 **
7 *
8 **
9 *
```

Requirements

- Values are expected to be ***positive integers between 0 and 9***..
- If a value is not an integer (decimal or negative number, character, etc.), it must be ignored.
- Think generic. Avoid using “hard coded” values in functions.
- Think modular. Split your code into re-usable functions. In particular, the acquisition of values and their processing (checking and display) should be clearly separated. The processing methods should not know the origin of values.

3- Standard input

Modify your program to add the following feature: the program asks the user for values if no argument is found.
By convention, an empty line validates the sequence of values.



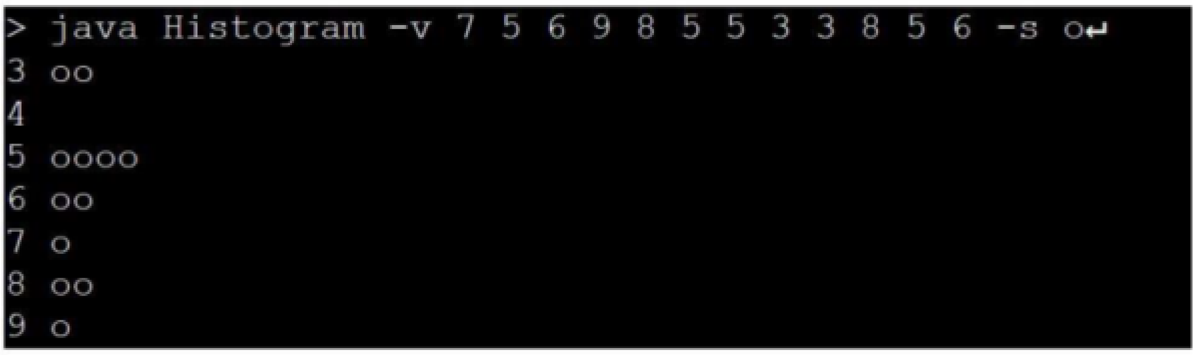
```
> java Histogram↵
No argument found. Please enter values below:
3 ↵
6 ↵
4 ↵
4 ↵
9 ↵
6 ↵
2 ↵
4 ↵
↵
2 *
3 *
4 ***
5
6 **
7
8
9 *
```

Guidelines

• Think incremental. Adding a feature to your program should mainly only consist in adding functions. Any or minor changes should be required on previously coded functions. Re-use the existing code as much as possible.

4- Symbol tuning

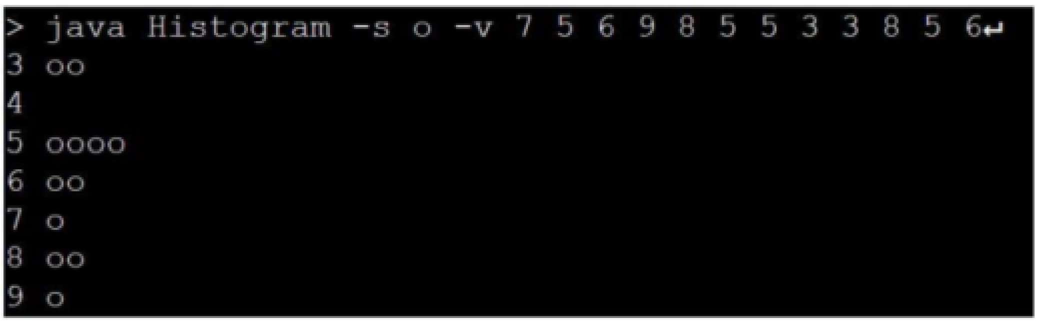
Modify your program to add the following feature: the symbol used to print histograms is customizable.
The arguments of the program becomes:
-v <integers>: to communicate the data to be shown in the histogram
-s <symbol>: to customize the symbol used in histograms



```
> java Histogram -v 7 5 6 9 8 5 5 3 3 8 5 6 -s o↵
3 oo
4
5 oooo
6 oo
7 o
8 oo
9 o
```

Requirements

- Arguments can be provided in ***any order***.



```
> java Histogram -s o -v 7 5 6 9 8 5 5 3 3 8 5 6↵
3 oo
4
5 oooo
6 oo
7 o
8 oo
9 o
```

- Some arguments can be missed:

◦ If the argument “-v” is not provided, the program asks the user for values (entered using the keyboard).