Practical 02: Conditions and Repetitions

Due 18 Aug by 22:00 **Points** 100 **Available** after 30 Jul at 10:00

Analyzing data and using repetition and choice

This assignment will focus on analysing data and using repetition and choice. You should use linux, svn and java to complete this practical.

You must come to practicals in week 03 to get your marks and feedback from our tutors.

Submission

Note: all class work must be submitted to your **SVN repository AND** **web submission** (https://cs.adelaide.edu.au/services/websubmission/index.php?menu=Classlist) . For this workshop, you shall use:

```
https://version-control.adelaide.edu.au/svn/aXXXXXXX/2019/s2/fcs/week-02/practical-02/
```

Important!!!

Most of your practical activities have components with a wide range of interpretation. We use these components to assess your creativity to propose solutions, and capability to implementing these solutions. We highlight that are no wrong solutions. Moreover, we are highly interest in understanding your decisions and this process. Therefore, the explanation for your approach for solving these problems will have a high impact on your marks.

For more information about this, please see **Discussions/How does work Practical Marking? (click here)**

Part 1 - Analysis

1) We want you to generate 10 different positive integer variables, using **for loop** and **random generator** (https://docs.oracle.com/javase/8/docs/api/java/util/Random.html) (see example **random number from geek4geeks** (https://www.geeksforgeeks.org/java-util-random-nextint-java/)), and calculate their average. In mathematics average can mean several different things but, in this case, we want you to calculate the mean of the values.

Firstly, you should calculate the mean of these values. The mean is given by the sum of all the variables, divided by the number of variables you're summing.

If we had the values 1, 2, 3, 3, 4, 5, 6, 8, 4, 9, then the mean would be 4.5. The result of dividing the sum, 45, by the number of values, 10. **Keep in mind that while your variables are integer values the mean may not be! Make sure you store the mean in the correct data type.**

```
Save this project in the following folder week-02/practical-02/problem-01/RandomNumbers.java
add to your svn repository
```

Reference Reading: Integer division

In Java when you divide an integer by another integer what do you think happen? For example what result would you expect if you had this command in your program:

```
float result = 2/10;
```

You would probably expect a result of 0.2 correct? If you try this yourself you will find that Processing actually stores 0.0 in the variable result. Why is this?

When performing division Processing differentiates between integer division and floating point division. If both of the numbers in the division are integers then Processing will return an integer, so it will round the result of the division down to the nearest integer, in our example this was 0.

So to correctly calculate the mean you will need to indicate that we are doing floating point division. The easiest way to do this is by making the number we divide by a float like this:

```
float mean = sum/10.0;
```

Another method is to cast the result to a float. This tells processing that we are expecting a floating point result, so it won't round the result of the division.

```
float mean = (float)sum/10;
```

Part 2 - Repetitions

Problem-02 - In this problem we want you to produce a horizontal star line that shows your 10 variables from top to bottom as individual, distinct stars varying based on its value. Hence, if the random number is `4`, you will have `****`.

```
Number (4): ****
Number (4): ****
Number (2): **
Number (7): *******
Number (5): *****
Number (0):
Number (3): ***
Number (4): ****
Number (5): *****
Number (2): **
```

Constraints:

1. The random numbers should not be bigger than 20;
2. The code should generate only 10 random numbers;

```
Save this project as ..problem-02/*.java
add to your svn repository
make a submission on websubmission
Note that * represents the filename. You are required to define a filename following the Java Conventions
```

Problem 03 - Having plotted your starts on the console. We want you to calculate the average of stars given all the random numbers. The average should then be printed on the console with it's respective number of stars. The average value should be rounded (`it will require some research online, take it as a challenge`) in order to avoid breaks on your code.

```
Number (7): *******
Number (3): ***
Number (6): ******
Number (4): ****
Number (8): ********
Number (4): ****
Number (0):
Number (7): *******
Number (3): ***
Number (9): *********
Average (5): *****
```

```
Save this project as ..problem-03/*.java
add to your svn repository
```

Problem 04 - Now, please identify **even** and **odd** random numbers; Even numbers should be identified by `+` and odd numbers should be identified as `-`;

```
Number (5): -----
Number (3): ---
Number (0):
Number (8): ++++++++
Number (2): ++
Number (5): -----
Number (8): ++++++++
Number (1): -
Number (1): -
Number (8): ++++++++
Average (4): ****
```

```
Save this project as ..problem-04/*.java
add to your svn repository
make a submission on websubmission
Note that * represents the filename. You are required to define a filename following the Java Conventions
```

Problem 05 - Please identify the numbers bigger or smaller than 10. If the random number is bigger or equal to `10`, it should be assigned to `>=`, otherwise `<`.

```
Number (0):
Number (3): <<<
Number (7): <<<<<<<
Number (8): <<<<<<<<
Number (8): <<<<<<<<
Number (8): <<<<<<<<
Number (10): >=>=>=>=>=>=>=>=>=>=
Number (13): >=>=>=>=>=>=>=>=>=>=>=>=>=
Number (12): >=>=>=>=>=>=>=>=>=>=>=>=
Number (10): >=>=>=>=>=>=>=>=>=>=
Average (7): *******
```

```
Save this project as ..problem-05/*.java
add to your svn repository
make a submission on websubmission
Note that * represents the filename. You are required to define a filename following the Java Conventions
```

Problem 06 - Now, using `switch` statements, please identify the different ranges. Hence, follow the suggested markers:

- `o`, for numbers between (0,5];
- `x`, for numbers between (5, 10];
- `s`, for numbers between (10, 15];
- `*`, for numbers bigger than 15;

```
Save this project as ..problem-06/*.java
add to your svn repository
make a submission on websubmission
Note that * represents the filename. You are required to define a filename following the Java Conventions
```

Part 3 - User interface and operations

Problem 07 - In this exercise you are asked to create a user interface that perform addition operations. This user interface should run over a `while` loop. For every operation performed, the user should decided between `a) sum again` or `b) exit`.

```
Welcome dear user!
Would you like to:
a) sum again
b) exit
Option:
```

If the option `a` is selected, then:

```
Please, insert the first number: 55
Please, insert the second number: 10
----
Thank you for your enquiry, the sum between 55 and 10 is 65.
```

Otherwise, `b`:

```
Thank you! Have a good day.
```

```
Save your project as week-02/practical-02/user-interface/*.java
add to your svn repository
```

```
Save this project as ..problem-07/*.java
add to your svn repository
make a submission on websubmission
Note that * represents the filename. You are required to define a filename following the Java Conventions
```

Withdraw Money from ATM

Problem 08 - We wish to develop a ATM class that will test whether or not a particular number, representing an amount of cash, can be represented as a combination of $20 and/or $50 notes.

In your ATM class, you should have one function named `run` *(week 3 content).* This function will take a number which represent an amount of cash as input, and a string as output. The string should either be "Here is **X** $20 notes and **Y** $50 notes." when the amount can be dispensed using **X** $20 notes and **Y** $50 notes or "Sorry, the value you input cannot be withdrew". Your program should be able to calculate the value for **X**and **Y**.

If there are multiple ways to dispense an amount, please use the composition with the fewest number of bills.

Note: it is optional whether you will use ***user input*** or ***hard values*** on your code; Please, to use user input please check on **Java Tutorial website (https://www.tutorialspoint.com/java/java_files_io.htm)** .

```
For example,

If the input is 60, then the output should be "Here is 3 $20 notes and 0 $50 notes."

If the input is 100, then the output should be "Here is 0 $20 notes and 2 $50 notes."

If the input is 120, then the output should be "Here is 1 $20 notes and 2 $50 notes."
```

```
Save this project as ..problem-08/*.java
add to your svn repository
make a submission on websubmission
Note that * represents the filename. You are required to define a filename following the Java Conventions
```

Basic Marking Scheme

Criteria	Ratings	Pts
Compilation In order to achieve full marks - your code must compile and run;		20.0 pts
Basic Functionality Your code (1) perform all the functions correctly, (2) use latest concepts learned in class, (3) has a clear, creative and sophisticated way of solving the problem.		40.0 pts
Functionality Extension Your code (1) perform all the functions correctly, (2) use latest concepts learned in class, (3) has a clear, creative and sophisticated way of solving the problem, and (4) you propose novel ways to solve the problems - or extra functionalities.		10.0 pts
Code Formatting Your code (1) has the right proportion of comments and place line and block comments correctly, (2) follow correct indentation every new indentation level, (3) has good variable naming, (4) has clear organization between tabs and is easy to read.		30.0 pts
		Total points: 100.0