Lab 8 – Parameter Passing

We have seen in the last lab the flexibility that methods provide us when programming. We will expand this through the use of parameters.

Part 1

Find the *BlueJ* project folder named `Lab8_Start` in the `Resource` folder (or download and decompress `Lab7_Start.zip` on Blackboard). Copy this to your home directory. Rename your copy to `Lab8_lastname_firstname`. This project contains a program which you will use for the first part of the lab.

Open the class `Part1`. This program simply calls a void method that prints information passed to it via parameters.

1. Take a close look at the program, paying particular attention on:
    a. How each method is defined particularly how parameters are included
    b. How each method performs its function
    c. How the methods are called from the main
    d. Where each method is located
    e. How the whole program and each method is documented

2. Run the program and observe the output.

We will look at some common errors when dealing with parameters. It is important that you pay attention to the error messages so that if these messages appear when you program, you know how to properly correct the error.

1. Incorrect number of parameters
    a. Number of parameters passed not enough.

        Change the method call in the main to read: `printInfo("Ruben", 3);`
        Compile the program. What happens? Note any error messages.

    b. Number of parameters passed too many.

        Change the method call to read: `printInfo("Ruben", 3, 12, 8, 3.5);`
        Compile the program. What happens? Note any error messages.

2. Incorrect data types passed

    Change the method call to read: `printInfo("Ruben", 3.0, 12, 2.6);`
    Compile the program. What happens? Note any error messages.

    Change the method call to read: `printInfo("Ruben", 3, 12, "2.6");`

Compile the program. What happens? Note any error messages.

3. Order of parameters is important

   Change the method call to read: `printInfo("Ruben", 12, 3, 2.6);`
   Compile the program. The program compiles. Run the program. What happened?

Part 2

In the last lab, you wrote a program that prints a message depending on the calculated average. In that exercise, three methods were used to print the appropriate message.

It is possible that the same result can be accomplished using a single method (instead of 3) by passing the calculated average as a parameter to this method and using the value of this passed parameter to print the appropriate message.

To do this:

1. Create a new Driver class and name it `Part2`.
2. Copy the `main` and the `calculateAverage` methods from the previous lab into this class.
3. Change the call to print the message in the main. Instead of an `if-else if-else`, it should be only one line: a call to the a method `printMessage` and pass the calculated average to this method.
4. Create the `printMessage` method. This method will receive the average as a parameter and use this average to print the proper message using a selection structure. Use the program in Part 1 as a guide.
5. Run the program as you did in the previous lab and check your results.

Part 3

In the above program, when you analyzed the data you were given, you had to run the program 3 times. Now change the program so that it can take any number of sets, calculate the average of each set and print the average and the appropriate message in a nice tabular form. Use a sentinel loop which terminates if the user answer to the question "Are here more sets to process?" is "no" which is coded with a value of 2.

Compile and run the program and check your results.

Submission

Submit your work in the usual way.