CSE 1321L: Programming and Problem Solving I Lab

Lab 10

Methods

What students will learn:

- How to design methods with regards to return type and parameters
- How to code methods
- How to call methods

Overview: Up until now, the programs you have written have started at the "main" method and executed from top to bottom. Methods are also commonly called "functions" or "procedures". A method is a group of code that has a name and a single function. We can send information to the method to get it started (called parameters), and the method can return information when it is done with a return statement. This information must be the same data type as the return type. These are choices that you have to make when designing a method – and they can be confusing when you first see them. All methods you write in this class have a "template" that looks like:

```
<return type> <method name> ( <parameters> )
```

About each one:

- The return type is a data type (e.g. int, Boolean, char, string, int[ ], char[ ]…). It is the kind of data this method will return. If the method does not return data, the return type is void. If the method does return data, you must include a return statement in your method (this is usually done when the method finishes calculating a result).
- The method name is easy. Valid method names follow the same rules as naming variables.
- Parameters are listed between parentheses ( ) and are just a set of variables used to "catch incoming data". They are a listing of the data that must be passed to the method for it to do its job. For example, if you have a method called addTwoNumbers(), it needs two numbers to do its job, so those appear as two variables. If a method does not require any information to get started, the parentheses are empty. Tying all three things together, the example method would look like:

```
int addTwoNumbers (int x, int y) {
    int result = x + y;
    return result;
}
```

For the Java and C# folks, you would have to put the keywords "public" and "static" before the return type.

Note: the method above won't run until you call it (from main). That code would look like:

```
int temp = addTwoNumbers(6, 14);
```

As always, don't forget to include your name/section as comments for the header, document your code (also with comments) and call the class names and filenames the correct things (e.g. Lab10A .java, .cs, .cpp).

Lab10A: Let's step back a bit…

Let us go back and learn methods with something simple. Please remember to refer to the previous page for a method's structure.

Note: In addition to the above-mentioned method structure, in Java and C#, the methods must be public and static. We will discuss why this is the case and when we would not have methods like so later in the semester.

So, something like this:

```
public static <return type> <method name> (<parameters>) {}
```

Write a program that contains three methods:

Method max (int x, int y) returns the maximum value of two integer values.
Method min (int x, int y) returns the minimum value of two integer values.
Method average (int x, int y) returns the average of two integer values.

You should design the above methods to be in your program first before you proceed to write anything in the main method. Remember, write these methods assuming the variables in them already exist. A method is a function which you feed with input (determine what this input is using the parameters in the method); the method should then proceed in using these parameters in the way it was designed to use them. Therefore, whenever you designate a parameter, it is a variable that can be used in your methods.

Write a main method that asks the user for three numbers. These three numbers should then be passed as actual parameters to the three methods you have which should return values back to the main method to be printed in a manner like shown below in the sample.

Tip: When getting the average, please make sure you use a double or float for the variable. Pay attention as you may get a truncated answer as you are getting the average of integers. Remember casting!

Remember, the class name should be Lab10A.
The user input is indicated in bold.

Sample output #1:

```
Enter number 1: 4
Enter number 2: 9
Min is 4
Max is 9
Average is 6.5
```

Sample output #2:

```
Enter number 1: 45
Enter number 2: 11
Min is 11
Max is 45
Average is 28
```

Lab10B: Rectangled Decision!

Design and implement a program that implements the following 3 methods:

- Method `isValid(...)` returns true if the sum of the width and height is greater than 30

```
boolean isValid(double width, double height){}
```

- Method `area(...)` returns the area of the rectangle if it is a valid rectangle

```
double area(double width, double height){}
```

- Method `perimeter(...)` returns the perimeter of the rectangle if it is a valid rectangle

```
double perimeter(double width, double height){}
```

Always remember that in Java and C#, the methods must be public and static

The main method should prompt the user to enter the width and height of a rectangle (double values) and uses the methods to print out a message followed by the area and perimeter if the rectangle is valid. Otherwise, it prints out only the message " `This is an invalid rectangle` ."

Note: that method `isValid(...)` is used to validate the input before attempting to compute the area and perimeter.

Design the main method in the test program such that it allows the user to re-run the program with different inputs (using a sentinel loop structure).

Remember, the class name should be Lab10B.
The user input is indicated in bold.

Sample output:

```
Enter width: 4.0
Enter height: 5.0
This is an invalid rectangle

Do you want to enter another width and height (Y/N)? : Y

Enter width: 20.0
Enter height: 15.0
This is a valid rectangle
The area is: 300.0
The perimeter is: 70.0

Do you want to enter another width and height (Y/N)? : N

Program Ends
```

Instructions:

- Programs must be working correctly.
- Programs must be saved in files with the correct file name.
- If working in Java or C#, class names must be correct.
- Programs (only .java, .cs or .cpp files) must be uploaded to Gradescope by due date.