Computer Science 111L Intro to Algorithms and Programming: Java Lab

Programming Project #4 MenuCalculator (30 points)

Due: 4/27/20

The goal of Programming Project #4 will be to write a menu-driven calculator program which can perform addition, subtraction, multiplication, division and other arithmetic operations on integer and double-type numbers.

Start by creating a NetBeans Project named **Project4** containing a package named **menucalculator** and a main class named **MenuCalculator**.

In the body of the **MenuCalculator** `main()` method, write the single statement which invokes a method named `showMenu()`. The concept of invoking a method is fundamental to the subject of programming with methods, so **do not ask your classmates or even your instructor how this statement should be written**. As the name suggests, the `showMenu()` method should show the user an ordered menu of options and a prompt, like:

```
Calculator Menu
---------------
1. Add two integers
2. Add two doubles
3. Subtract two integers
4. Subtract two doubles
5. Multiply two integers
6. Multiply two doubles
7. Divide two integers
8. Divide two doubles
9. Compute a factorial product (n!)
10.   Exit

Your choice?>
```

The user's response to the "Your choice?>" prompt should be stored in an int-type variable named **`userChoice`**. A selection structure (either an if…else if or a switch) should be used to "decide" which of the nine calculator methods to invoke, depending on the option the user has chosen. For example, if it is the case where the user has chosen 1, then the statements in case 1 should prompt the user to enter the two integers to be added together, then pass those two numbers to the `add()` method which takes two integer inputs. After the `add()` method is done with its work, it should pass the sum of the two integers back to the case 1 statement which invoked it, and then a statement in case 1 must display the returned value in a message. Here is an example of what the `showMenu()` method's output should look like when the user chooses option 1:

```
Calculator Menu
---------------
1. Add two integers
2. Add two doubles
3. Subtract two integers
4. Subtract two doubles
5. Multiply two integers
6. Multiply two doubles
7. Divide two integers
8. Divide two doubles
9. Compute a factorial product (n!)
10.   Exit

Your choice?> 1
```


```
Enter the first number: 8
```


```
Enter the second number: 2
```


```
8 + 2 = 10

Calculator Menu
---------------
1.    Add two integers
2.    Add two doubles
3.    Subtract two integers
4.    Subtract two doubles
5.    Multiply two integers
6.    Multiply two doubles
7.    Divide two integers
8.    Divide two doubles
9.    Compute a factorial product (n!)
10.   Exit

Your choice?>
```

Each of the first nine cases in the `showMenu()` method's switch should correspond to one of the calculator methods you will write. The first eight calculator methods will take two inputs and each will return only one value. The ninth calculator method will take only one input. You should also notice that after the add operation result is displayed, the menu is re-displayed. This means that it will be necessary to "group" the majority of the `showMenu()` method's statements into a loop structure. The loop will continue to repeat itself while `userChoice` is less than 10.

Here is a list of method headers which must be implemented in this project. Below each method header is a pseudocoded description of the "task" that the method must perform:

```
public static void main(String[] args)
```

The `main()` method simply invokes the `showMenu()` method.

`public static void showMenu()`

This method should display the menu of calculator methods, prompt the user for his or her choice, "decide" which calculator method to invoke, prompt the user for additional values, invoke the appropriate calculator method, then display the output of that calculator method. The `showMenu()` method should repeat itself until the user chooses option 10, to Exit from the `showMenu()` method.

`public static int add(int n1, int n2)`

This method returns the result of the integer addition of the parameters n1 + n2.

`public static double add(double n1, double n2)`

This method override returns the result of the double addition of the parameters n1 + n2.

`public static int subtract(int n1, int n2)`

This method returns the result of the integer subtraction of the parameters n1 – n2.

`public static double subtract(double n1, double n2)`

This method override (you guessed it) returns the result of the double subtraction of the parameters n1 – n2.

`public static int multiply(int n1, int n2)`

This method returns the result of the integer multiplication of the parameters n1 * n2.

`public static double multiply(double n1, double n2)`

This method override returns the result of the double multiplication of the parameters n1 * n2.

`public static int divide(int n1, int n2)`

This method returns the result of the integer division of the parameters n1 / n2.

`public static double divide(double n1, double n2)`

This method returns the result of the double division of the parameters n1 / n2.

`public static int factorialProduct(int n)`

This method returns the factorial product (n!) of the parameter n. The factorial product is computed with the following rules: if n is 0, then 0! is 1. For all values of n > 0, n! = n x (n-1)!. What this means is that n! is the product of all of the factors of n, multiplied together. For example, if n is 5, then n! = 1 * 2 * 3 * 4 * 5 = 120. Your `factorialProduct()` method should use a for loop to accumulate and return this factorial product. See pages 720 to 721 in the textbook for a further discussion on computing factorials.

Your program should use NO GLOBAL VARIABLES and all variables should be declared LOCALLY, within the body or header of the method which uses those variables.

Remember to write a Universal comment header at the top of your source code file.

When you have completed the exercise, ZIP the entire project folder and upload this ZIP file to Canvas. In NetBeans, click **File**, select **Export Project**, click **To ZIP…**, select the location where you want the ZIP file to be saved, type the name of the ZIP file, **Project4.zip** in the textbox, click the **Save** button, then click the **Export** button.

If the IDE you are using does not have an export-project-to-zip option, you may have to manually ZIP the file by navigating to the project folder in a file browser and selecting your OS's "compress folder" option. Under Mac OS, the file browser is called **Finder** which offers a **Compress Folder** option. Under Windows, the file browser is called **File Explorer** and it offers a **Send to => Compressed (zipped) folder** option.