Homework #7

(Java Programming for Beginners - OnLine)

Note: *First complete your midterm and then try this homework as you can use part of the code from your midterm to complete this homework*

Description: Convert your Midterm –A Simple Calculator Program-- into a class based program, using your knowledge of Object Oriented Programming techniques.

Write a OOPCalculator class, by giving it proper states and behaviors. For example, you can have menuChoice, firstFloat, secondFloat as member variables. You can give it behaviors like, askCalcChoice, askTwoValues, displayResults to get the choice from users, to get the numbers and display the calculation results respectively. All of the code written for calculation needs to be properly placed within these methods of the OOPCalculator class. For example: askCalcChoice first displays the menu, prompts the user to input the choice, and then waits until user presses enter. Once a valid input is entered, it returns the choice (between 1 to 5 or A for addition, S for subtraction, M for multiplication, D for division or X for exit). Similarly, other methods will implement rest of the behaviors. The users of your OOPCaluclator class do not need to know internal details of how you have implemented your behaviors. These are the only public interface exposed to the users, rest of the details stays with OOPCalculator class.

So, a driver program (written by user of your OOPCalculator or even from within main of your own class) can use your class by writing a code like this:

```
public class TestCalculator {
      public static void main(String[] args) {
         OOPCalculator calc = new OOPCalculator();
         while (calc. askCalcChoice () != 5){ //it will set choice
                  calc. askTwoValues ();              //it will set two values
                  calc.displayResults();       //do calc, display result
                                    //and wait on press enter key
         }
         calc.displayBye();             //thanks the user for using and waits for press enter key
     } //end main
} //end class
```

All the error checking and exception handling for a wrong choice, or wrong value input are handled by your OOPCalculator class, the user should not need to handle it.

Sample Run of the program: When the driver program is run, the behavior is exactly same as Midterm I, except, provide one more functionality. Allow user to select 'A' or 'a' for addition,' S' or 's' for

subtraction,' M' or 'm' for multiplication, 'D' or 'd' for division, and 'X' or 'x' for exit, along with integer choice between 1 to 5.

Welcome to <*John Doe*'s> Handy Calculator

1. Addition
2. Subtraction
3. Multiplication
4. Division
5. Exit

What would you like to do? *M*

Please enter two numbers to multiply separate by a space: *24.0* *4.0*
Result of multiplying 24.00 and 4.00 is 96.00.

Press enter key to continue ….

Welcome to <*John Doe*'>s Handy Calculator

1. Addition
2. Subtraction
3. Multiplication
4. Division
5. Exit

What would you like to do?

Continue until user selects 5, or 'x', or 'X' to exit

Note:

1) Replace the <*John Doe*'s> with your name
2) Make sure to properly word the output for the choices and result. Meaning, if user selects 'M' from menu, the prompt should be for 'Multiplication' not Addition.
3) *3, 24.0 4.0* are shown in the example as underline to emphasize that it is entered by the user and not part of the program. Underline and italics are not a requirement
4) Your program should allow input of integer or decimal numbers. The output should always be in decimals with two decimal digits as precision.
5) Make sure your program will continue displaying menu after result is shown and user has pressed enter key. Your program will exit only when user selects 5 or 'x' or 'X'.
6) If user selects other than 1-5, show a message that they must select between 1 and 5 (or their character equivalents). Give them a chance to re-enter. Continue until a valid number is entered.

7) If they enter invalid values instead of numbers (e.g. strings), provide an error message and give them chances to re-enter. Continue until valid numbers are entered.
8) Make sure to catch divide by zero issue. In case of division choice, the second number should not be zero – so let the user know that the second number cannot be zero.