CSD 3464 – Assignment 08

Overview:

The following question is an **adaption** to *Absolute Java* (6th Ed.)'s Chapter 09 Programming Project Q6 (pg. 578). Please follow the instructions included in **this** document and implement the following Java files:

⇒ CalculatorTester.java *(Contains a `main()` method)*
⇒ Calculator.java
⇒ UnknownOperatorException.java

The above classes/files should both be inside a package called *q6*.

Important Note: Starter files are provided for you with helpful hints inside.

Helpful Information:

In this assignment you will complete the Programming Project 6 from pg. 578 of your textbook with the below modifications:

- Exclusively use the `BigDecimal` class to do the arithmetic operations in this question. The `BigDecimal` class is often preferred over primitive types as you do not have to concern yourself with overflow errors (i.e. using numbers larger than allowed for the specific data type) and it offers a higher level of precision in calculations.

- To create a `BigDecimal` object use the following constructor: `BigDecimal(char[] in)`
  - For example, `BigDecimal bd1 = new BigDecimal("1234.678")` will create a `BigDecimal` object with the value `1234.678`
  - **IMPORTANT:** When creating a `BigDecimal` object an `NumberFormatException` may be thrown if the String provided is not a valid numeric representation.
    - In the `main` program **must** `catch` this error and re-prompt the user for a valid input before proceeding.

- Big Decimal has a number of methods but in this question you are required to use:
  - `public BigDecimal` **`add`**`(BigDecimal augend)`
    - Which adds two `BigDecimal` objects and returns their result as a new `BigDecimal` object
  - `public BigDecimal` **`subtract`**`(BigDecimal subtrahend)`
    - Which subtracts two `BigDecimal` objects and returns their result as a new `BigDecimal` object

- `public BigDecimal multiply(BigDecimal multiplicand)`
  - Which multiplies two `BigDecimal` objects and returns their result as a new `BigDecimal` object
  -
- `public BigDecimal divide(BigDecimal divisor)`
  - Which divides two `BigDecimal` objects and returns their result as a new `BigDecimal` object
  - **IMPORTANT:** When performing the division if the `divisor` is zero an `ArthimeticException` will be thrown. When this happens you must `catch` the error and Display the message "`Divide by zero not permitted`" and do not update the running total of your calculator.

- Finally, when printing a `BigDecimal` to the console use the below code to prevent displaying any trialing zeros:
  - `myBigDecimalObj.stripTrailingZeros().toPlainString()`

Instructions:

Step 01 – Implement UnknownOperatorException.java

Open UnknownException.java file.

The `UnknownOperatorException` is a class that extends `Exception`. Like all exceptions it should implement:

1. A no-argument constructor that sets a default error message reading "`UnknownOperatorException: Values are limited to +, -, *, /`"
2. A constructor that takes a single String argument and sets the error message to that provided string provided.

Step 02 – Implement Calculator.java

Open Calculator.java file.

The `Calculator` is a class that has been partially implemented for you. This class will be used by your `main` program, in step 03, to perform all calculations and display results of the various calculations required.

What you will notice is the `Calculator` class has a single instance variable called `result` that is of type `BigDecimal`. Additionally, the class has two constructors completed for you:

1) A no argument constructor that sets the starting calculator balance to 0
2) A single argument that sets the starting value of the calculator based on the provided String value; for example, "123.123456"

A completed toString() method has also been provided to you which returns the current value of the calculator as a String value.

Your job in the calculator class is to implement the four private methods based on the instructions provided in the file.

```
1) private void add(BigDecimal valueToAdd)
2) private void subtract(BigDecimal valueToSubtract)
3) private void multiply(BigDecimal valueToMultiplyBy)
4) private void divide(BigDecimal valueToDivideBy)
   throws ArithmeticException
```

Note: Please see review the Helpful Information of this document for further guidance on how to work with `BigDecimal` objects

Lastly, you must implement `the public void performOperation` method. Instructions to implement the method are provided in the file.

- **Note:** In the next step you will use this method to perform arithmetic operations within the `main`

Step 03 – Implement CalculatorTester.java

Open CalculatorTester.java file.

The third and final step is to implement the `CalculatorTester` class with a `main` method that reads user input and performs operations as shown in sample input/output below.

- Text in **bold** is provided by user via keyboard
- Calculator should execute at least once and after that it will continue as long as a "`y`" value is provided
  - **Hint:** Use a do-while loop
  - Only consider the first character in a case-insensitive manner
    1. `yes` should be interpreted as "`y`"
    2. `N` should be interpreted as "`n`"

  If neither, "`y`" or "`n`" is provided re-prompt user until appropriate value is provide.
- During each iteration of the do-while loop keep track of `result` as shown below.
  - You must use your `Calculator` class and specifically its `performOperation` method you implemented
  - On each user input one of three types of exceptions could occur:
    1. **NumberFormatException** - value provided to add, subtract, divide, multiply is not numeric
       - Will be thrown by BigDecimal constructor
    2. **UnknownOperatorException** – Operator is not "*", "/". "+", or "-"
       - Will be thrown by `performOperation` method
    3. **ArthimeticOperation** – A division by zero is attempted
       - Will be thrown by `performOperation` method

    You must catch and handle all three exceptions as shown below.

Sample Input/Output:

```
Calculator is on.
result is = 0.0
+ 5
result  +  5.0  = 5.0
new result  = 5.0
* 2.2
result  *  2.2  = 11.0
new result  = 11.0
^ 2
UnknownOperatorException: Values are limited to +, -, *, /
result =  11.0
/ 0
ArithmeticException: Cannot divide by zero!
result =  11.0
* 0.1
result  *  0.1  = 1.1
new result  = 1.1
r
Final result = 1.1
Again? (y/n)
yes
result  = 0.0
+ 10
result  + 10  = 10.0
new result  = 10.0
/ 2
result  / 2  = 5.0
new result  = 5.0
+ abc
NumberFomratException: Value abc is not numeric!
result =  5.0
r
Final result = 5.0
Again? (y/n)
K
Again? (y/n)
N
End of Program
```