Assignment: Inheritance and Exceptions
Binary, Octal, Decimal and Hexadecimal
60 points

A Desktop Calculator Emulator

Create a base class LongInteger that represents a single long integer value in decimal (base 10). The class should have methods that allow you to add, subtract, multiply, and divide two LongInteger objects. Each of these methods should return a new LongInteger object or modify an existing LongInteger. There should also be get and set methods to allow the value of the integer to be changed. Implement a toString() which returns the string representation of the integer in the appropriate base.

Derive a class from LongInteger called BinaryInteger, which is a binary version of LongInteger. It should be able to do all the things LongInteger does (add, subtract, multiply, divide) by means of inheritance. Of course, the toString method needs to return a binary number in string format.

Derive a class from LongInteger called OctalInteger, which is a base 8 version of LongInteger. As with BinaryInteger, OctalInteger should have add, subtract, multiply, divide methods through inheritance. It should also have its own toString.

Derive a class from LongInteger called HexInteger, which is a hexadecimal (base 16) version of LongInteger with the same attributes and methods as the others. It should also have its own toString.

Create a class IntDriver which will contain a main method. The class should do the following:

1. Present the user with a display that allows them to select one of the above four modes. The user is then supposed to enter a number of that type. If the number they enter is not of the specified type, throw an exception to handle it. The type of exception you throw is up to you, but you should include the type of number expected and the input the user entered. This information should be printed to the screen (ex: BinaryInteger

expected, user entered 1010102). The user should then be re-prompted for an integer of the chosen type.

2. Enter an operator (+, -, * or /)
3. Enter a second number.
4. Enter "=" to perform the calculation and display the result.

```
Decimal mode


Bin - Binary            +
Oct - Octal             -
Dcm - Decimal           *
Hex - Hexadecimal       /
Q   - Quit              =
Option or value -->  oct

Octal mode


Bin - Binary            +
Oct - Octal             -
Dcm - Decimal           *
Hex - Hexadecimal       /
Q   - Quit              =
Option or value -->  675

Octal mode
675 (octal)

Bin - Binary            +
Oct - Octal             -
Dcm - Decimal           *
Hex - Hexadecimal       /
Q   - Quit              =
Option or value -->  +
```

```
Octal mode
675 (octal)      +

Bin - Binary                    +
Oct - Octal                     -
Dcm - Decimal                   *
Hex - Hexadecimal               /
Q   - Quit                      =
Option or value -->  765

Octal mode
675 (octal)      +         765 (octal)

Bin - Binary                    +
Oct - Octal                     -
Dcm - Decimal                   *
Hex - Hexadecimal               /
Q   - Quit                      =
Option or value -->  =

Octal mode
1662 (octal)

Bin - Binary                    +
Oct - Octal                     -
Dcm - Decimal                   *
Hex - Hexadecimal               /
Q   - Quit                      =

Option or value -->  Q
```

Extra Credit

(15 points possible)
Be sure to state that you're attempting extra credit in the comments at the top of your IntDriver.java file.

- Enhance your calculator to handle negative values (5 points.)
- Write your own methods to convert to and from decimal to binary, octal and hex (10 points.)

NOTE

If you properly utilize inheritance on this assignment, your derived classes will be quite small in terms of the code they contain. Try to design your base class such that everything can be re-used or built on in your derived classes.

To Turn In

Turn in all source code in a single zip file. Name your file as usual.