Our Own Hexadecimal

Purpose: To become familiar with Abstract Data Types by creating a HexNumber class to represent and allow mathematical manipulation of hexadecimal numbers.

Learning Objectives:

- To understand and leverage numbers in bases other than 10
- To implement an abstract data type

Background:

Sometimes the built in primitive data types (e.g., int) and corresponding wrapper classes (e.g., Integer) in Java are not enough and we need to build our own representations.

Let's begin by looking at Java's own "BigInteger" class.

For handling numbers, Java provides built-in types such as int and long. However these types are not useful for very large numbers. When you do mathematical operations with int or long types and if the resulting values are very large, the type variables will overflow causing errors in the program.

*Java provides a custom class named BigInteger for handling very large integers (which is bigger than 64 bit values). This class can handle very large integers and the size of the integer is only limited by the available memory of the JVM. However BigInteger should only be used if it is absolutely necessary, as using BigInteger is less intuitive compared to built-in types (since Java doesn't support **operator overloading**) and there is always a performance hit associated with its use. BigInteger operations are substantially slower than built-in integer types. Also the memory space taken per BigInteger is substantially high (averages about 80 bytes on a 64-bit JVM) compared to built-in types.*

Another important aspect of BigInteger class is that it is immutable. This means that you cannot change the stored value of a BigInteger object, instead you need to assign the changed value to a new variable.[1]

What is "operator overloading"?

Consider the "+" operator. We can use this operator with two different addends such as ints, e.g. 3 + 4. We can also use the same operator with doubles (3.5 + 4.5) or Strings ("Hello" + "World"). However, we are not permitted to use the + operator for our BigIntegers. Therefore, we must add BigIntegers like this:

```
BigInteger b1 = new BigInteger("67829302198766745800000000");
BigInteger b2 = new BigInteger("29394820193782839200000000");
BigInteger sum = b1.add(b2);
```

Hexadecimal is base 16

We need 16 digits for hex, but we only have 10 (0 through 9). So we must borrow letters to act as digits: A for 10, B for 11, and so on – up to F for 15. How do we convert a number in a different base to our decimal system? Well let's first consider the decimal system, or base 10.

Decimal number: 4352

We need to use places according to the formula B^n, where B is the base and *n* is the place from right to left.

- The digit '2' is in the one's place (B = 10; n = 0. Hence $10^0 = 1$).
- The digit '5' is in the ten's place (B = 10; n = 1. Hence $10^1 = 10$).
- The digit '3' is in the hundred's place (B = 10; n = 2. Hence $10^2 = 100$).
- The digit '5' is in the thousand's place (B = 10; n = 3. Hence $10^3 = 1000$).

[1] http://www.quickprogrammingtips.com/java/using-biginteger-in-java.html

Hexadecimal number: 1A4 (which equals 420 in the decimal system. Do you see why?)

We apply the formula B^n similarly.

- The digit '4' is in the one's place.
- The digit 'A' is in the 16's place.
- The digit '1' is in the 256's place.

Instructions

1. Create a class called HexNumber. The public members of this class (the "interfaces") are shown in the Javadoc printout below. Consider this the API for the class. **You will need to implement the methods below.**

Method Summary

All Methods | Static Methods | Instance Methods | Concrete Methods

Modifier and Type	Method and Description
HexNumber	add(HexNumber hex) Adds one hexadecimal number to another
static long	decimalValue(java.lang.String hexValue) Analyzes a hexadecimal string and returns its corresponding decimal value
HexNumber	divide(HexNumber hex) Divides one hexadecimal number by another
long	getDecimalValue() Standard getter method; returns the decimal value (base 10) representation of an instance of this class.
java.lang.String	getHexValue() Standard getter method; returns the hexadecimal value (base 16) representation of an instance of this class.
static java.lang.String	hexValue(long decimalValue) Converts a decimal value into a hexadecimal string
static boolean	isHex(java.lang.String potentialHexString) Returns true for valid hexadecimal strings
HexNumber	multiply(HexNumber hex) Multiplies one hexadecimal number to another
HexNumber	subtract(HexNumber hex) Subtracts one hexadecimal number from another
java.lang.String	toString() Returns the hexString for a hexadecimal number

2. **Constructors and Getters**. (10 pts.) You should be able to construct a HexNumber by passing either a decimal value or a String. For example, the hexadecimal number 1A can be instantiated by passing either hex string "1A", "1a", or decimal value 26. Both the hexValue and the decimalValue will be stored as instance variables. Hence, you must be able to calculate one from the other.

Constructors

Constructor and Description
HexNumber(long decimalValue) Constructor to make a HexNumber from a long
HexNumber(java.lang.String hexValue) Constructor to make a HexNumber from a valid hexadecimal string

Like BigInteger, your class will be immutable (i.e., you will not code any setters).

3. **Conversion Methods.** (40 pts.) You should create the class methods `hexvalue()` and `decimalValue()` as described above. Each conversion method is worth 20 points. The conversions should be done algorithmically. If the algorithm only works on positive values, they will be worth only 10 points each. If you utilize built-in methods from an appropriate wrapper class, you will only earn 5 points for each method.

4. **Arithmetic methods**. (40 pts.) Create `add()`, `subtract()`, `multiply()` and `divide()` for 8 pts. each. Consider these examples:

```
Hex value -A = -10      -A + 1B = 11          or, in base 10, 17
Hex value 1B = 27       -A - 1B = -25         or, in base 10, -37
                        -A * 1B = -10E        or, in base 10, -270
                        -A / 1B = 0           or, in base 10, 0
```

Use the methods to calculate (A + (40 / 4) - 14) * 2 in one line of code. All numbers are hexadecimal. (8 pts.)

5. **Encapsulation.** Will you need/want private methods?

6. (5 pts.) Create static method isHex().
   Why make this validation method static?

7. (5 pts.) Build and implement a custom exception to prevent clients from instantiating invalid hex numbers (like "G2").
   What would a good superclass be?

8. If you are building a cohesive abstract data type, is there anything else worth including?

NOTE: I have a Driver file that will exercise your class, so it is important that you adhere to the API.