Laboratory Assignment № 2

Dr. Duncan, CSC 1351, Louisiana State University 09/03/2015

Implementing Static Methods

In the previous laboratory exercise, we defined a class that consisted of constructors and instance methods. A class consisting of those methods describes an object. There are times when we need a method which is not invoked by an object. In Java, the *static* modifier denotes class methods and class variables. These methods and variables are accessed without an object of a class. For example, the standard Java Math class consists of only static methods; *abs*, *pow*, *log*, *exp*, *sin*, *cos* and *tan* are all examples of static methods defined in the Math class. See the online Math API documentation and convince yourself that all its methods are static. It also contains static variables such as *PI* and *E*, constants representing π and Euler's number (the inverse of the natural log).

Some Mathematical Functions

In this lab, you will implement four mathematical functions as iterative static methods.

Definition 1. *The **factorial** of an integer n, denoted $n!$, is the number of ways n distinct objects can be arranged.*

The factorial of a negative number is indeterminate. To compute the factorial of a non-negative number, the formula below is used:

$$n! = \begin{cases} 1 & , n = 0 \\ 1 \times 2 \times \cdots \times n & , n > 0 \end{cases} \tag{1}$$

Definition 2. *The **Fibonacci numbers** are the numbers in the following integer sequence: 0,1,1,2,3,5,8,13,21,34,55,....*

By definition, the first two Fibonacci numbers are 0 and 1, and each subsequent number is the sum of the previous two.

Definition 3. *The **greatest common denominator**, denoted GCD, of two of more integers is the largest positive integer that can evenly divide the numbers.*

For example, the GCD of 12, 15 and 30 is 3. When computing the GCD of two integers, if exactly one of the number is 0, their GCD is the non-zero number. When computing the GCD of two numbers, if both numbers are 0, their GCD is indeterminate. To compute the GCD of two integers, use the algorithm below:

input : a, b : integer
output: gcd(a,b)

intialization;
if *a = 0 AND b = 0* **then**
    $\gcd \leftarrow NaN$;
else if *a = 0 OR b = 0* **then**
    $\gcd \leftarrow |a + b|$;
else
    $a \leftarrow |a|$;
    $b \leftarrow |b|$;
    while $b \neq 0$ **do**
        r $\leftarrow$ remainder(a,b);
        a $\leftarrow$ b;
        b $\leftarrow$ r;
    end
    $\gcd \leftarrow a$;
end

Algorithm 1: An Iterative Euclidean GCD Algorithm

Definition 4. *The **least common multiple**, denoted LCM, of two of more integers is the smallest positive integer that the numbers can evenly divide.*

For example, the LCM of 12, 15 and 8 is 120. When computing the LCM of two integers, if at least one of the number is 0, their LCM is indeterminate. When computing the LCM of two non-zero numbers, their LCM can be found using this formula:

$$LCM(a, b) = \frac{a \times b}{GCD(a, b)} \tag{2}$$

The *AuxiliaryMath* Class

Define a class called *AuxialiaryMath* in which you implement four static methods representing the mathematical function defined above. Each method should be implemented as an iterative method.

1. public static int factorial(int n), where this method returns $n!$.
2. public static int fibonacci(int n), where this method returns the n^{th} Fibonacci number.
3. public static gCD(int a, int b), where this method returns the GCD of a and b.
4. public static lCM(int a, int b), where this method returns the LCM of a and b.

The *AuxiliaryMathDemo* Class

Write a program, *AuxiliaryMathDemo*, that does the following:

1. Prompts the user for three integers, reads the inputs from the keyboard, and computes and displays the GCD of the three integers. (Hint: $gcd(a, b, c) = gcd(gcd(a, b), c)$)
2. Prompts the user for two integers, reads the inputs from the keyboard, and computes and displays the LCM of the integers.
3. Prompts the user for an integer n, reads the integer, and computes and displays $n!$.
4. Finally, prompts the user for two positive integers i and j, where $i < j$, reads the integers and computes and displays the series consisting of the i^{th} through j^{th} Fibonacci numbers.

Since we have not studied exceptions and exception-handling, when the return value of a method is indeterminate/undefined/*NaN*, return -1, an illegal return value for all of the methods.

Additional Requirements

Exhaustively, test your program to ensure that it works. Do not use a negative number as an argument to the *factorial* method. Also, use only positive integers as arguments to the *fibonacci* method. Write header comments for each class using the following Javadoc documentation:

```
/**
 * Explain the purpose of this class; what it does <br>
 * CSC 1351 Lab # 2
 * @author YOUR NAME
 * @since DATE THE CLASS WAS WRITTEN
 */
```

For the *AuxiliaryMathDemo* class, after the @since line, add the following Javadoc:

```
 * @see AuxiliaryMath
```

Add Javadoc style documentation for each method in the *AuxiliaryMath* class. Run the Javadoc utility to make sure that it generates documentation for the *AuxiliaryMath* class. Locate your source files, *AuxiliaryMath.java* and *AuxiliaryMathDemo.java*, and enclose them in a zip file, *YOURPAWSID_lab02.zip*, and submit your lab assignment for grading using the digital dropbox set up for this purpose. Here is a sample program interaction:

Listing 1: Sample Run.

```
1 Enter three integers whose GCD is to be found -> 75 90 100
2 Enter two integers whose LCM is to be found -> 25 40
3 Enter a non-negative integer n to find n! -> 6
4 Enter two  positive integers i and j, where i < j -> 1 5
5
6 gcd(75,90,100) = 5
7 lcm(25,40) = 200
8 6! = 720
9 fib(1) + ... + fib(5) = 7
```