

# LABORATORY ASSIGNMENT № 6

Dr. Duncan, CSC 1350, Louisiana State University

10/22/2019

## Computing a Series

### Learning Objectives

1. More on Using Loops and
2. Modular Programming, and
3. Implementing and Using Static Methods

In last week's lab session, you implemented a modular design using static methods in the main class. Those methods were used to disaggregate the overall task of the program into more manageable subtasks. The maintenance, implementation and design of the program can be further enhanced by separating those methods from the main method. In that way, we separate the implementations of those methods from the program that uses them. In today's lab session, you will implement the same methods but this time in a separate class and file, *AuxMath*. The class *AuxMathTester*, the program that uses these methods, will consist of only the *main* method. To configure the Netbeans project correctly, you must add *AuxMathTester* to the project before *AuxMath*. Again, we discuss the functions and their series approximations below:

### Series Approximations of Two Functions

The natural logarithm of a number  $x$ , denoted  $\ln(x)$  where  $x > 0$ , is given by the series below.

$$\ln(x) = 2 \left[ \left( \frac{x-1}{x+1} \right) + \frac{1}{3} \left( \frac{x-1}{x+1} \right)^3 + \frac{1}{5} \left( \frac{x-1}{x+1} \right)^5 + \frac{1}{7} \left( \frac{x-1}{x+1} \right)^7 + \frac{1}{9} \left( \frac{x-1}{x+1} \right)^9 + \dots \right] \quad (\text{eq1})$$

The inverse of the natural log function is the exponential function  $e^x$  since the base of the natural log is  $e = 2.7182818284590452353602874713527$ . The number  $e$  is called the Euler's number. An approximation of the exponential function is given by the series below.

$$e^x = 1 + x + \frac{x^2}{2} + \frac{x^3}{6} + \frac{x^4}{24} + \frac{x^5}{120} + \frac{x^6}{720} + \dots \quad (\text{eq2})$$

### Some Mathematical Identities

There is an equivalence between  $\ln(x)$  and  $\log_b(x)$ , where  $x > 0$ .  $\log_b(x)$  is defined only for  $b > 0$  and  $b \neq 1$ .

$$\log_b(x) = \frac{\ln(x)}{\ln(b)} \quad (\text{eq3})$$

Table 1 gives the equivalence between the power, exponential and natural log functions. Observe that  $x^n$  is not always defined.

x	n	$x^n$ Equivalence
0	$n \leq 0$	undefined
0	$n > 0$	0
$x \neq 0$	0	1
1		1
	1	x
$x \neq 0$	-1	1/x
$x > 0$		$e^{n \ln( x )}$
$x < 0$	n not an integer	undefined
$x < 0$	n is even	$e^{n \ln( x )}$
$x < 0$	n is odd	$-e^{n \ln( x )}$

Table 1: Calculating Powers

Use Double.NaN, a Java constant that denotes Not-a-Number, when a method returns an undefined value.

## The Auxiliary Math Class

Be sure to set up the starter code for the *AuxMathTester* class, the main class, in your project before adding the *AuxMath* class. You will incrementally implement the methods below in the *AuxMath* class as you code the various versions of the program. Each method in the *AuxMath* class is a *public static* double-returning method. Do not implement any additional methods in *AuxMath*. Do not use any standard Java library Math class method in this program. When implementing the *nLog* method, approximate the natural log using the first 10,000 terms of the series in equation (eq1).

```
/**
 * Computes the natural log of the specified number using a series approximation.
 * @param x a positive number
 * @return the series approximation of ln(x) using the first 10000 terms or NaN
 * when x is not a positive number
 */
public static double nLog(double x)
```

When implementing the *xpon* method, approximate the exponential function using the first 400 terms of the series in equation (eq2) but handle the trivial cases  $e^{-1}$ ,  $e^0$  and  $e^1$  before using the series approximation.

```
/**
 * Computes the exponential of the specified number using a series approximation.
 * @param x a number
 * @return the series approximation of e^x
 */
public static double xpon(double x)
```

Implement the *bLog* method as a derived method of the *nLog* method using the identity in equation (eq3).

```
/**
 * Computes the logarithm of the specified to the specified base
 * @param x a positive number
 * @param b the base of the logarithm
 * @return log_b(x) or NaN if log_b(x) is undefined
 */
public static double bLog(double x)
```

Implement the *pwr* method as a derived method of the *nLog* and *xpon* methods using the identities in Table 1.

```
/**
 * Computes the specified power of the given base
 * @param x the base of the power
 * @param n the exponent of the power
 * @return x^n or NaN if x^n is undefined
 */
public static double pwr(double x, double n)
```

### The *AuxMathTester* Program

Create a project *AuxMathTester* containing the *AuxMathTester* class. As indicated earlier, neither the *AuxMath* nor the *AuxMathTester* class should use standard Java Math class methods.

- **Preliminary Version** : The *AuxMathTester* class will contain only the *main* method. Add the *AuxMath* class to the project. Then implement the *nLog* method as described above in the *AuxMath* class. Write code in the main method to perform unit testing on this method. Use several arguments including cases in which the method returns NaN. Verify that you are getting very close, if not exact, approximations of the natural log function.
- **Second Version**: Add the implementation of the *xpon* method as described above to the *AuxMath* class. Then add code in the main method to perform unit testing on the method. Use several arguments including 0 and positive and negative numbers.
- **Third Version**: Add the implementation of the *bLog* method as described above to the *AuxMath* class. Then add code in the main method to perform unit testing on the method. Use several arguments including those for which the method returns NaN. Verify that the method works as expected.
- **Fourth Version**: Add the implementation of the *pwr* method as described above to the *AuxMath* class. Then add code in the main method to perform unit testing on the method. Use several arguments including those for which the method returns NaN. Verify that the method works as expected.
- **Final Version**: Modify the main so that the program interactivity and output is exactly as shown in the sample runs, barring truncation errors in the approximated results. Display all approximated results to ten decimal places.

Note: In order to determine whether a double has an integer equivalent, type-cast the double to an integer and compare the integer to the original double. If the integer and double are equal, then the double can be expressed as an integer (without losing precision). For example, suppose *dbl* is a *double*. If the Boolean expression **(int)(dbl) == dbl** is true, then *dbl* can be expressed as an integer without the loss of precision. If not, it cannot be expressed as an integer without the losing precision. You will need to use this trick in the implementation of the *pwr* method.

## Other Requirements

Remove all Netbeans auto-generated comments. Include the Javadoc documentation for each method *AuxMath*, as shown on this handout. Write header comments for both *AuxMath* and *AuxMathTester* using this Javadoc documentation template:

```
/**
 * Explain the purpose of this class; what it does <br>
 * CSC 1350 Lab # 6
 * @author YOUR NAME
 * @since DATE THE CLASS WAS WRITTEN
 * @see AuxMath (ADD THIS TAG ONLY IN THE AuxMathTester Javadoc HEADER COMMENTS)
 */
```

Here are sample program interactions:

Listing 1: Sample Run

---

```
Enter x to compute ln(x) -> 5
ln(5.0) = 1.6094379124
Enter x and b to compute log_b(x) -> 8 2
log_2.0(8.0) = 3.0000000000
Enter x to compute e^x -> 1.6094379124
e^1.6094379124 = 4.9999999998
Enter x and n to compute x^n -> 2 3
2.0^3.0 = 8.0000000000
```

---

Listing 2: Sample Run

---

```
Enter x to compute ln(x) -> -12
ln(-12.0) = NaN
Enter x and b to compute log_b(x) -> 100 2
log_2.0(100.0) = 6.6438561898
Enter x to compute e^x -> -3.14
e^-3.14 = 0.0432827979
Enter x and n to compute x^n -> 17 1.35
17.0^1.35 = 45.8253767174
```

---

---

#### Listing 3: Sample Run

---

```
Enter x to compute ln(x) -> 25
ln(25.0) = 3.2188758249
Enter x and b to compute log_b(x) -> 54 1
log_1.0(54.0) = NaN
Enter x to compute e^x -> 3.5
e^3.5 = 33.1154519587
Enter x and n to compute x^n -> 2.3 -1.5
2.3^-1.5 = 0.2866871623
```

---

---

#### Listing 4: Sample Run

---

```
Enter x to compute ln(x) -> 23
ln(23.0) = 3.1354942159
Enter x and b to compute log_b(x) -> 0 2.3
log_2.3(0.0) = NaN
Enter x to compute e^x -> 1
e^1.0 = 2.7182818285
Enter x and n to compute x^n -> -4 -3
-4.0^-3.0 = -0.0156250000
```

---

---

#### Listing 5: Sample Run

---

```
Enter x to compute ln(x) -> 36
ln(36.0) = 3.5835189385
Enter x and b to compute log_b(x) -> 59 -3
log_-3.0(59.0) = NaN
Enter x to compute e^x -> 0
e^0.0 = 1.0000000000
Enter x and n to compute x^n -> 2 -5
2.0^-5.0 = 0.0312500000
```

---

---

#### Listing 6: Sample Run

---

```
Enter x to compute ln(x) -> 700
ln(700.0) = 6.5510803350
Enter x and b to compute log_b(x) -> 64 4
log_4.0(64.0) = 3.0000000000
Enter x to compute e^x -> -1.23
e^-1.23 = 0.2922925777
Enter x and n to compute x^n -> -7.25 -2.3
-7.25^-2.3 = NaN
```

---

#### Listing 7: Sample Run

---

```
Enter x to compute ln(x) -> 0
ln(0.0) = NaN
Enter x and b to compute log_b(x) -> -32 -2
log_-2.0(-32.0) = NaN
Enter x to compute e^x -> 2.2
e^2.2 = 9.0250134994
Enter x and n to compute x^n -> -2.34 -1.1
-2.34^-1.1 = NaN
```

---

### Submitting Your Work for Grading

Navigate your way to the ...\\NetBeansProjects\\AuxMathTester\\src\\auxmathtester folder using Windows file explorer. You should find *AuxMath.java* and *AuxMathTester.java*, files containing your source code for the program. Click one of the files and then hold down the shift key and click the other so that both files are selected. Right-click the selected files and create a compressed (zipped) folder containing a copy of each file. Rename the zip file *PAWSID\_lab06.zip*, where *PAWSID* is the prefix of your LSU/Tiger email address - the characters left of the @ sign. Double-click the zip file to verify that both *AuxMath.java* and *AuxMathTester.java* are included the zip file. If the zip file does not contain both files, delete the zip file and repeat the steps. Upload the zip file to the digital drop box on Moodle.