Minor Assignment 3

Exponential Calculator

Instructions: Create a new project in Eclipse, download the outline of the ExponentialCalculator.java file from Canvas, and import it into your project. Put a comment with your name at the top. This indicates that the work is yours alone, as stated in the Collaboration and Academic Dishonesty page.

Fill in the methods appropriately, test your code, and submit the completed .java file to Canvas by the due date.

Please refer to the "Getting Help" and "Academic Dishonesty" portions of the syllabus for direction on how to get help. You are always welcome to message your instructor. Make use of this if you are stuck!

Background: In Math 141 you learn about the exponential function, $f(x) = e^x$ where $e = 2.71828\ldots$ However, don't learn how to calculate it other than using your calculator. Someone had to tell the calculator how to do this.

In this assignment you'll develop the computer code to calculate it. You'll write two helper methods (factorial and power) and use them in conjunction with a formula from Math 153 to compute e^x for any x. You'll then compare the result to Java's built in exponential function.

The Problem: Write the following methods to approximate e^x:

Method 1: factorial - The factorial of a positive integer, n, is defined to be $n! = 1 \cdot 2 \cdot 3 \cdot \ldots \cdot n$. For example, $5! = 1 \cdot 2 \cdot 3 \cdot 4 \cdot 5 = 120$. Note that $1! = 1$ and that $0!$ is defined to be 1 as well.

You'll code up a method that accepts an integer variable and uses a **for** loop to compute its factorial. Return the value of the factorial. **Don't use the built in factorial function.**

Method 2: power – For any number a, and a positive integer b, a^b is equal to a multiplied by itself b times: $a^b = a \cdot a \cdot \ldots \cdot a$. For example, $4^3 = 4 \cdot 4 \cdot 4 = 64$.

Write a method that accepts two numbers, a double a and an int b and uses a **for** loop to compute the value of a^b.

Method 3: myExp - To approximate e^x, your calculator uses the formula:

$$e^x = 1 + \frac{x}{1!} + \frac{x^2}{2!} + \frac{x^3}{3!} + \frac{x^4}{4!} + \ldots$$

In Math 153 this is called an *infinite series*. Many functions you know ($\sqrt{x}, \ln(x), \cos(x), \sin(x),\ etc$) are computed with similar formulas. In general, adding more terms following the pattern gives a better estimate for the exact value.

Write a method called myExp that accepts a double x and an int n and returns the approximate value of e^x by using the above formula through the n-th power. For example, myExp(2.5, 5) would compute

$$1 + 2.5 + \frac{2.5^2}{2!} + \frac{2.5^3}{3!} + \frac{2.5^4}{4!} + \frac{2.5^5}{5!}$$

Method 4: main - Lastly, inside of your main method you'll test your calculation by computing the so-called *true relative error.* If X is an approximation to Y, then the true relative error is the percentage difference between the two of them. We want to have only positive numbers, so use the absolute value:

$$true\ relative\ error = \frac{|X - Y|}{Y}$$

In the main method, compute the true relative error between myExp(2.5,5) and Math.exp(2.5). Then compute the true relative error between myExp(2.5,10) and Math.exp(2.5).

In your main method, print out sentences along with the approximations you are computing and the true relative errors in those approximations.

Note: The absolute value function in Java is Math.abs().

Extra credit (1 point): When n is not too big (like $n = 5$ or $n = 10$) your code should work pretty well, and the error should be small. In addition, the error should get smaller as n gets bigger. But try approximating $e^{2.5}$, and computing the true relative error, using $n = 17, n = 18, n = 19, n = 20.$

You should notice that the error isn't always small, and doesn't decrease.

Explain why this happens.

Note: This is a challenging question!