Problem Set 2 Exercise #18: Maclaurin Series

Reference: Lecture 4 notes

Learning objective: Modular design

Estimated completion time: 40 minutes

Problem statement:

In mathematics, the Taylor series is a representation of a function as an infinite sum of terms calculated from the values of its derivatives at a single point. It may be regarded as the limit of the Taylor polynomials. Taylor series are named after the English mathematician Brook Taylor.

If centred at zero, Taylor series is also called **Maclaurin series**, named after the Scottish mathematician Colin Maclaurin.

As an example, an approximation for e^x is:

$$e^x = \sum_{n=0}^{\infty} \frac{x^n}{n!} = 1 + x + \frac{x^2}{2!} + \frac{x^3}{3!} + \cdots \; for\ all\ x$$

Write a program **PS2_Ex18_Maclaurin.java** that reads in two positive integers *x* and *n*, calculates e^x up to the *n*th term (i.e. ... + $x^{n-1}/(n-1)!$) in the Maclaurin series.

You should adopt a modular design and define a method

```
double compute_maclaurin(int x, int n)
```

to compute the Maclaurin series up to *n*th term.

You may define additional methods as necessary.

Correct your output of real numbers to 3 decimal places.

Sample run #1:

```
Enter x and n: 2 2
3.000
```

Sample run #2:

```
Enter x and n: 2 10
7.389
```

Challenge:

The challenge is NOT to re-compute $n!$ for each term, but make use of the $(n-1)!$ value that is computed in the previous iteration.

If you cannot think in this way, then take a simpler approach to define $n!$ as a method **`factorial()`** in your program.