From: Java How to Program (9th Edition), Paul Deitel and Harvey Deitel, Prentice Hall, 2012.

4.37) The factorial of a nonnegative integer \( n \) is written as \( n! \) (pronounced as “\( n \) factorial”) and is defined as follows:

\[
n! = n.(n-1).(n-2). \cdots .1 \text{ (for values of } n \text{ greater than or equal to } 1)\]

and

\[
n! = 1 \text{ (for } n=0)\]

For example \( 5! = 5.4.3.2.1 \), which is 120.

a) Write an application that reads a nonnegative integer and computes and prints its factorial.

b) Write an application that estimates the value of the mathematical constant \( e \) by using the following formula. Allow the user to enter the number of terms to calculate.

\[
e = 1 + \frac{1}{1!} + \frac{1}{2!} + \frac{1}{3!} + \cdots
\]

e) Write an application that computes the value of \( e^x \) by using the formula. Allow the user to enter the number of terms to calculate.

\[
e^x = 1 + \frac{x}{1!} + \frac{x^2}{2!} + \frac{x^3}{3!} + \cdots
\]

(60 points)


5.19) (Slightly modified) Calculate the value of \( \pi \) from the infinite series

\[
\pi - 4 - \frac{4}{3} - \frac{4}{5} - \frac{4}{7} - \frac{4}{9} - \frac{4}{11} + \cdots
\]

Print a table that shows the value of \( \pi \) approximated by computing one term of this series, by two terms, by three terms, and so on.

Write a Java application that takes an integer argument indicating the precision of the \( \pi \) calculation and prints both the \( \pi \) value at that precision and the number of terms to reach
that precision level. For instance, if user enters 4, \( p = 3.1415 \) and number of terms to be used may not be true or if user enters 5, \( p = 3.14159 \) and number of terms used in the calculation maybe 154. 

**Important Notes:**

1. All source codes and related homework reports should only be submitted to moodle system: [http://moodle.ube.ege.edu.tr](http://moodle.ube.ege.edu.tr). For further information, you can contact with the grader from sercanodemirci85@gmail.com.
2. Do not forget to include appropriate comments in the source codes. Hence the grader can easily understand the program during his/her assessment.
3. Write the programs in a simple and straightforward manner by considering object-oriented analysis and design principles.
4. Each report should include the printout of the related source code, two or more screenshots (depending on the illustration requirements) which exemplify execution of the programs and proper UML diagrams.