Assignment 3

The Greatest and the Least of These	30 pts.
e^x Approximations	30 pts.
File with Random Numbers	40 pts.

TOTAL: 100 pts.

General Requirements

- *All source files you are turning in must be signed. Please add your name to the source code in comments. Unsigned work automatically loses **5 pts**.*
- *Add comments to the source code you are writing:*
  - *Describe the purpose of every variable*
  - *Explain the algorithm you are using for solution*
- *Uncommented work loses up to 25% of the problem points.*
- *Please turn in the entire projects.*
  Put all project folders into one folder and archive it using ZIP or RAR utility.
  Turn it in into the digital drop box.

The Greatest and the Least of These

Write a program with a loop that lets the user enter a series of positive integers. The user should enter -99 to signal the end of the series. After all the numbers have been entered, the program should display the largest and the smallest numbers entered.
Use "The greatest of a set of numbers" algorithm (see below) to solve the problem.

"The greatest of a set of numbers" algorithm:

1. *Get (input) the first number, set it to be the greatest for now.*
2. *Get the next number.*
3. *If the new number is bigger than the "greatest" one – set the new one to be the "greatest".*
4. *Repeat steps 2 - 4 until all the numbers are consumed.*

Requirements:

1. Your code must run correctly on the test cases below.
2. DO NOT use arrays or ArrayLists to store the numbers in the series. Numbers must be analyzed on the spot and "discarded".

Test cases

1. User input (1 number): *-99*
   - Produces output "*No numbers were entered*"

2. Input 2 numbers: *1 , -99*
    - Output: *the largest number: 1 ; the smallest number: 1*
3. Input 5 numbers :*1 , 2 ,55, 0, -99*

Output: *the largest number: 55 ; the smallest number: 0*

Name your solution project GreatestAndLeast.java.

e^x Approximations

The value e^x can be approximated by the following sum:
$1 + x + x^2/2! + x^3/3! + \ldots + x^n/n!$

The expression n! is called the factorial of n and is defined as: n! = 1*2*3* …*n.

Write a program that takes a value of x as input and outputs four approximations of e^x done using four different values of n: 5, 10, 50, and 100. Output the value of x the user entered and the set of all four approximations into the screen.
Sample formula use: calculating e^7 using approximation with n = 5

$1 + 7 + 7^2/2! + 7^3/3! + 7^4/4! + 7^5/5!$

Requirements

1. Input validation is needed - do not allow the user to enter anything but integer. Use exception-handling technique discussed in class and implemented in InputValidation.java example to complete the feature.
2. Please do not do any calculations by writing out the entire sums by hand, use loops and running total calculations instead.
3. Factorial function produces large numbers and it grows very quickly. If you start using regular integers to store factorials, you'll get integer overflows easily. I would suggest using long to store factorials. On top of that, I would recommend to adjust your calculations in such a way that you never get to calculate the actual factorial values. Look at the pattern in the sum: each next term of the sum can be derived from the previous one by means of a very simple calculation.
4. Test your code. Make sure the calculations are correct by checking the actual values of e^x online. The bigger the value of n – the closer the resulting value must be to the actual value of e^x.

Name your solution project Approximations.java.

File with Random Numbers

Write a program that generates 100 random integers in a given range and stores the integers in a file with a given name. When numbers are written into the file each integer

is placed on a separate line.

Requirements

The program must:

1. Ask user for a file name where the numbers will be written to. If the file does not exist it must be created. If the file does exist its content is being replaced.
2. Ask user to provide the range of random integers to be generated.
    - Use exception mechanism to validate user input to be of correct type. See InputValidation.java to get example of how to do that. Use two validation loops to make the user provide
    - Both numbers of the range provided by the user must be positive integers. Continue asking user for input until the numbers fit the requirements.
3. The user may give the numbers of the range in incorrect order. Make sure that lower limit is smaller than upper limit. If it is not the case – swap the numbers.
4. Open the file with the given name, generate random numbers and write them in the file, placing each new number on a new line.
5. Use try/catch block to handle possible IOExceptions.
6. Your main() must not throw any exceptions.

Name your solution project RandomsInFile.java.