CMSC 203

Spring 2018

Assignment #1

Concepts tested by this program:

Set up and run java, javac, and an IDE (Eclipse) on your computer

Create a class

main method

variables

constants

mathematical operators

assignment operator

conditional statements

logical operators

input/output

Eclipse IDE

## Specifications

A useful approximation to the value of n! for large values of n is given by Sn = e-nnn+(1/2)

Develop a Java application in a class named FactorialApproximation that prompts for and reads one integer numeric amount that represents the value of n for n!. Use the approximation formula to calculate n!.

***Requirements***

* Use at least 2 named constants in your code
* Display a header for your application.
* Ask the user for the value of n.
* Restrict the values of n to whole numbers that are greater than 0 and less than 100. Any value that is outside this range should cause the program to end after an error message is printed.
* If the value of n is in the proper range, calculate the estimated value of n!
* Display the approximation of n! that results, with an appropriate label.
* If the value of n is equal to 5 or 7, compare the actual value of 5!, or 7!, to the value calculated by the formula. Then print the difference (actual value - calculated value) with an appropriate label.
* Display your name as programmer at the end.
* Use the Scanner class for input, and use System.out.print or System.out.println for output.
* Your program must use the single class FactorialApproximation and only one static void main method.
* Your code must **not** use loops, return statements, System.exit(), arrays or collections.

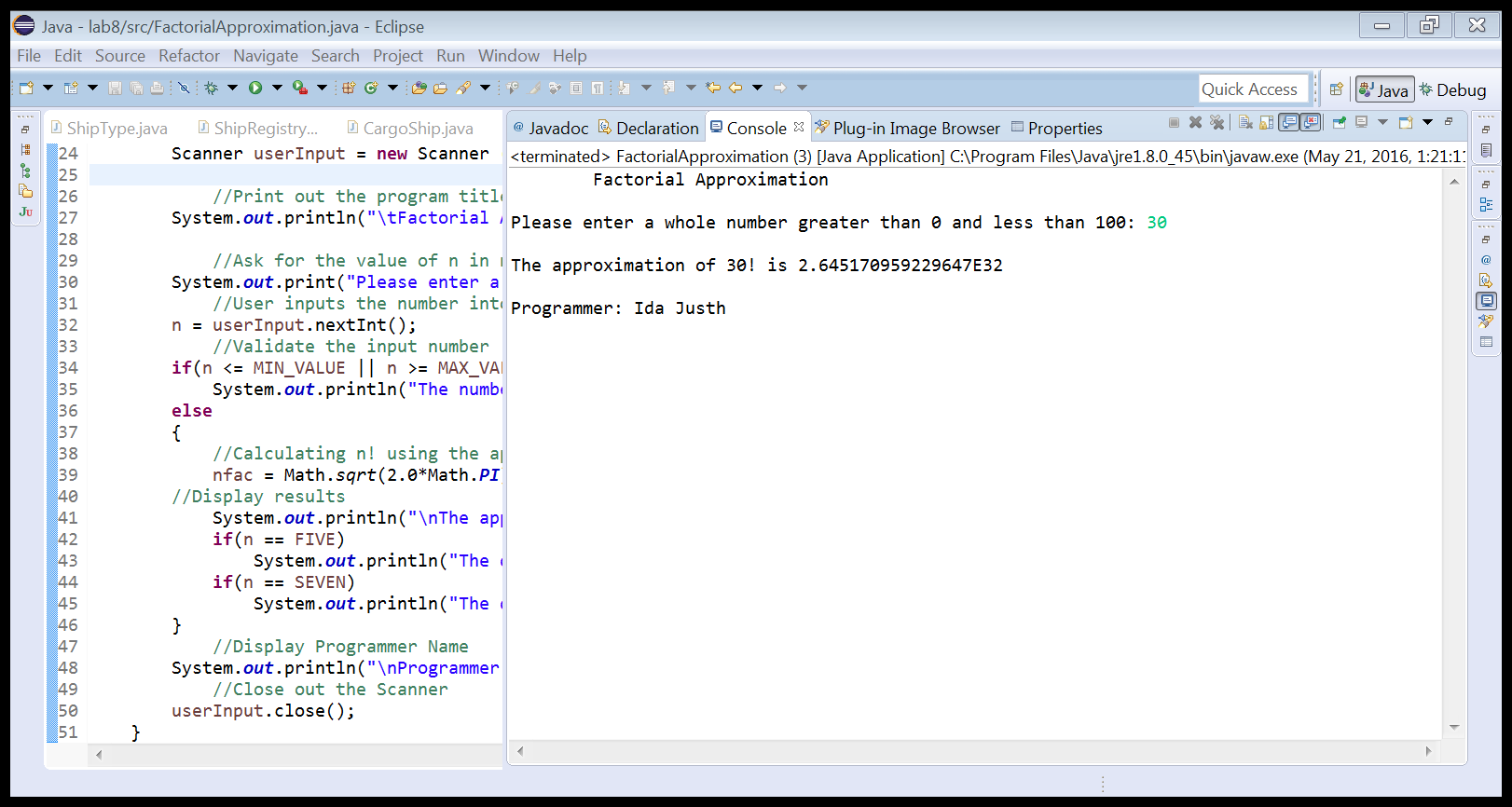
***Design***

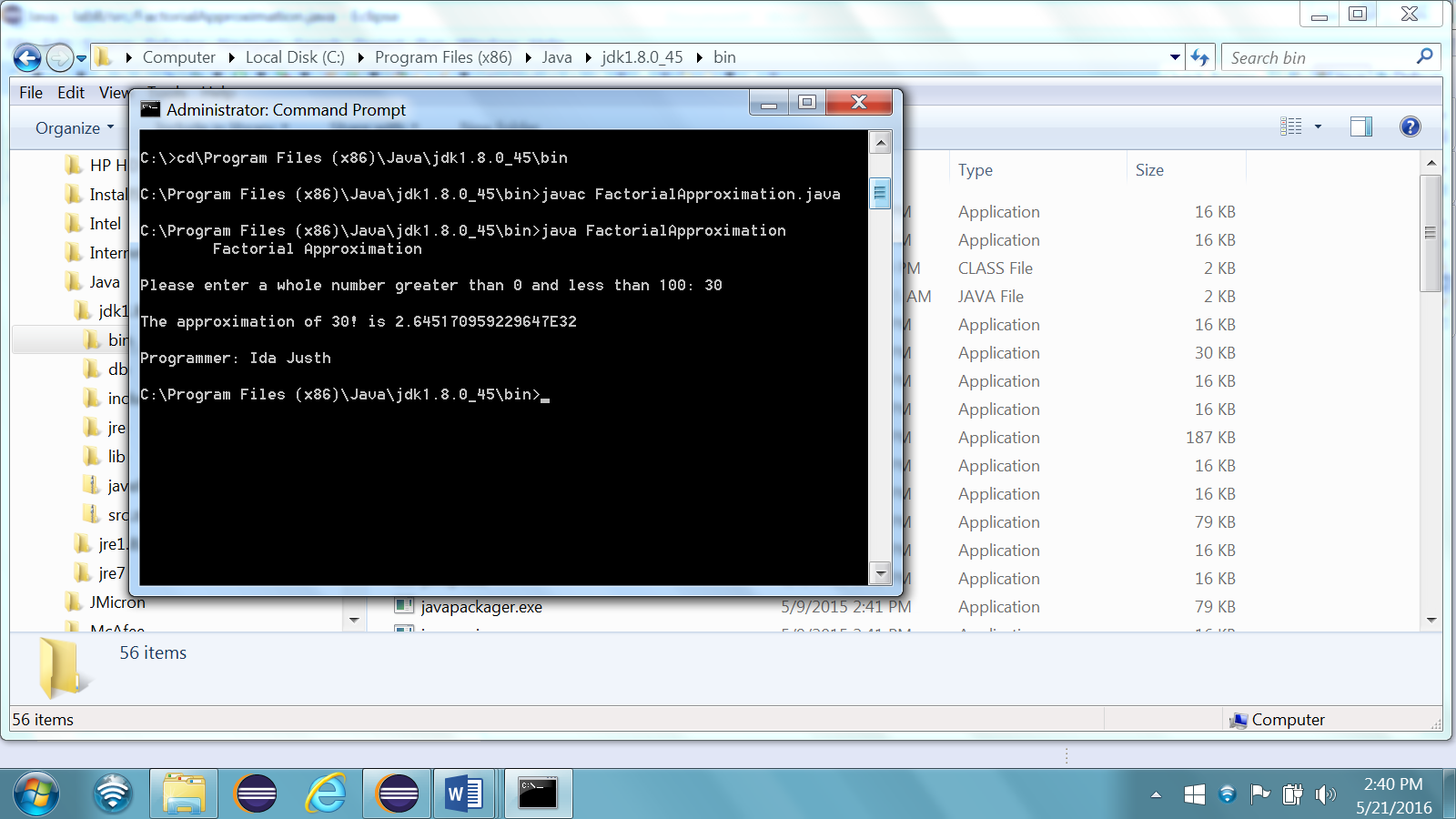
* “Think before you code”
* Design your application using pseudo-code
* Create a set of test cases that show the expected output of at least three normal cases, one that should produce an error message, and a “boundary case”.
* Record your pseudo-code and test cases in a Word or PDF document.

***To Deliver***

* Week 1: Submit your initial design (pseudo-code and test cases) in Blackboard.
* Week 2: Run the application twice, and turn in screen shots of both runs in Blackboard.
* Run it once using command-line Java (javac to compile, and java to run). Your screen shot should show both compilation and execution. If your computer cannot find the javac or java commands, see the lecture slides explaining how to set the path for Java.
* Run it again using an Integrated Development Environment (IDE). The one used in class will be Eclipse. Provide a screen shot of the console output. See the lecture slides explaining how to download and install Eclipse.
* Turn in your final design (pseudo-code and test cases) based on your study of the “approved solution” design.
* Turn in your Design Reflection, a text document how you modified your initial design based on the approved solution, or why you did not modify it.
* Turn in the working java file FactorialApproximation.java (not the .class file).

Example screen shots:





* Upload two .zip files for each assignment. The first .zip file will contain **all** the files required for the assignment (final design with pseudocode and table of test cases, screen shot of runs, design reflection document and the .java file, and will be named: LastNameFirstName\_AssignmentX.zip.
* The second .zip file will only contain the .java files and will be named: LastNameFirstName\_AssignmentX\_Moss.zip. This .zip will not have any folders in it – only .java files.

Here’s an example for Assignment 1:

AlexanderRobert\_Assignment1.zip [a compressed file containing the following]

PseudoCode\_and\_TestTable.docx

CmdLine\_output.jpg

Eclipse\_console\_output.jpg

Design Reflection.docx

FactorialApproximation.java

AlexanderRobert\_Assignment1\_Moss.zip [a compressed file containing only the following]

FactorialApproximation.java

**Grading Rubric**

**CMSC 203 Project #1**

Name \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

**PROGRAMMING**

Compiles 40 pts \_\_\_\_\_

Accuracy

Passes test cases 15 pts \_\_\_\_\_

Passes private instructor tests 15 pts \_\_\_\_\_

Execution: runs without errors (either run-time or logic errors) 30 pts \_\_\_\_\_

Possible Sub-total 100 pts \_\_\_\_\_

**REQUIREMENTS** (Subtracts from Programming total)

**Documentation:**

Screen shot of console run was not provided - 5 pts \_\_\_\_\_

Screen shot of IDE run was not provided - 5 pts \_\_\_\_\_

Documentation within source code was missing or incorrect - 5 pts \_\_\_\_\_

**Programming Style:**

Incorrect use of indentation, statements, structures - 5 pts \_\_\_\_\_

User interface

Not clear to user how data is to be entered; UI does not follow requirements - 10 pts \_\_\_\_\_

Output is easy to understand - 5 pts \_\_\_\_\_

**Design:**

Implementation does not follow design - 5 pts \_\_\_\_\_

Does not define a class named FactorialApproximation - 5 pts \_\_\_\_\_

Does not print header - 5 pts \_\_\_\_\_

Does not print programmer name - 5 pts \_\_\_\_\_

Does not use at least 2 named constants - 5 pts \_\_\_\_\_

Possible decrements: -60 pts \_\_\_\_\_

Possible total grade: 100 pts \_\_\_\_\_