CS 1050, Summer, 2021 60 points

Project #2: A Scientific Calculator

Overview

This program implements a simple scientific calculator.

Major topics

* writing a program from scratch
* multiple methods
* testing
* work in teams

In this document

* program logic & program structure (class, methods, variables)
* input / output
* assumptions and limitations
* documentation & style(See Foundation file #56
* suggested schedule
* cover letter discussion questions
* grading, submission, and due date
* additional notes
* challenges
* sample output
* test plan

Program Logic

* The program repeatedly prompts the user to select a type of math problem then calculates and outputs the answer.
* The program runs until the user does not want to do any more problems.
* A final tally of problems selected is printed at the end.

Program Structure and Data

*Calculator class*

|  |
| --- |
| main method  printReport method  calculate method  printIntro method  constructor |

* one class: CS1Calculator (name of the class and Java file)
* methods: use the method descriptions as English comments in the code, laying out each logic step to be written in Java
  + main: instantiates the calculator, calls printIntro, calculate, and print report
  + constructor: initializes counters to zero, instantiates and initializes a Scanner to read from the keyboard
  + printIntro method: explains program to user
  + calculate method: primary method with a loop until the user is done
    - run a *while* loop until the user doesn't want to do another problem
      * display calculation choices
      * get the user’s choice, echo their choice (display it back on the screen)
      * get one or two operands (numbers to use in the calculation)
        + pick the calculation, then calculate and print answer
      * count operation type (find the minimum, etc.)
      * display the calculation choices again
    - calculate total problems done (find the minimum count, etc.)
  + printReport method
    - outputs total of each problem done and overall total problems (see sample output)
* instance (class level) data
  + a Scanner (needs an import statement)
  + 4 ints to count the number of each problem type (find the minimum, etc.)
  + 1 int to hold total problem count
* local variables: as needed in each method (perhaps none in some methods)

I/O

* interactive only (keyboard & screen), no data files
* input
  + user
    - enters the type of problem (find the minimum, etc.)
      * User enters “T” to display operation counts and exit
    - enters one or two numbers for each calculation
  + the user may enter either uppercase or lowercase for the problem type selection
  + reading numbers: all input is String type, so it must be converted to int using a parser method from the Integer class:

firstOperand = scan.nextLine();

firstNumber = Integer.parseDouble(firstOperand);

* output (see sample output later in this document)
  + directions to the user about the problem
  + prompts to choose problem type and to enter numbers, or to stop entering problem types
  + calculated answer to the math problem
  + summary of problem types completed, with a total count of all problems

Allowable Calculations

1. Find the minimum of two numbers Math.min(x,y)

2. Find the maximum of two numbers Math.max(x,y)

3. Raise a number to a power Math.pow(x,y)

4. Find the absolute value of a number Math.abs(x)

Assumptions & Limitations

* input can be double or int. Use nextDouble()
* input numbers and answers will fit in the range of the double data type
* binary (two operands) or unary (one operand) input
* the only operations supported are the ones listed under Allowable Calculations
* perfect user: all input will be correct (correct choice selected, no division by 0)
* the user wants to do at least one problem

Program documentation

Include the following prologue comment, filled out, at the top of the class.

---

Program:

Description: << one or two sentences

Author: << you

Date:

Vocabulary word (non-computer)

Inspirational quote (non-political, non-religious)

---

Program style

Follow the style shown in the textbook and in the Java style guide, file #56 under Foundation. This includes the use of uppercase or lowercase, indentation of blocks (method bodies, loop bodies), alignment of braces, and general white space for readability.

Testing

See the test plan at the end of this document.

Discussion Questions ( for the cover letter)

1. Overflow. Find out what happens when you try to raise a number to a large exponent whose result exceeds the maximum double value (Double.MAX\_VALUE). What result did you get? Explain why overflow happens on a computer.
2. What syntax (Java grammar rules) did you have the most trouble getting right?
3. What semantics (meaning of the code) did you have the most trouble understanding?

Grading

I will run your program with various numeric inputs and check the results. You can contact me to go do a high‑level check of the program and style. This can be done once before the due date.

Criteria

* Program complies without errors, runs without crashing, and gets correct answers.
* All deliverables submitted (below).
* All specifications followed (this document).

Submission

* Your on-time submission is your claim that the program compiles, runs without crashing, and gets correct answers on all test data. Also, this is your claim that you are submitting your own or your team’s work, not someone else's and not taken from a source such as a web site.
* There is ONE set of deliverables (submissions) per team. The first line of the project cover letter should have your name if you worked alone, or the names of all the team members.
* Deliverables (what to submit)
  + completed project cover letter; docx, doc or pdf format only
  + source code (one .java file, no .class file)
  + text file with cut-and-paste copy of testing session #5 (all screen output and user input starting with the welcome message and ending with the report totals)
  + pseudocode (written ***before*** you write Java code and from which you write Java code!)

Dates

Target completion date: Monday, June 28. This gives you a few days to fix the inevitable surprises and errors that pop up

Due date: Sunday, July 4, 11:59 pm

* all four deliverables uploaded to Canvas in this order: cover letter, .java file, test text file and pseudocode

Late date: Wednesday, July 7, 11:59 pm (last date to submit the project, marked late)

-----------------------------------------------------------------------------------------------------------------------------------------

Additional notes

Program development

* Write pseudocode to get the flow and features of the program. Write code from the pseudocode.
* Write teeny programs to learn a single Java capability (for example, reading two numbers).
* Work on one small part of the program at a time, until it compiles and runs correctly.
* Do easy pieces first, such as printing out messages.
* Do not spend much time being stuck: use the 30-minute guideline to call or email for help.
* When you have several compile errors, fix them starting with the first. Some of the other errors may be "cascading errors" – not true errors, just the result of the first error, so they will disappear when the prior error is corrected.
* Keep track of questions as you go; don't assume or guess.

Other notes

* About the due dates
  + Projects will be accepted up to one week late, with a deduction of 25%. If you missed the project due date, submit whatever you have done by the late date; list missing and erroneous pieces in the cover letter. Some part of the project must be running correctly to receive credit.
* If you do any of the challenges, you may include them in your submission or not; you could also work on them after the project is submitted.

Further work / challenges (optional, no extra credit). For any challenges you do, run additional test cases to check the new functionality.

* Experiment with the NumberFormat class to learn how to place commas in results larger than 999, and round to a certain number of decimal places; include that for answers before they are printed.
* Print totals for each type only if the user did any of that type. For example, if they did not do any max problems, don't print that report line which would show zero max problems.

-----------------------------------------------------------------------------------------------------------------------------------------

Sample interactive part of a session (bold shows user answers); match the following format:

---

Select the problem type you would like to calculate by entering the first letter of the operation.

(N) Minimum of two numbers

(X) Maximum of two numbers

(P) First operand raised to the power of the second operand

(A) Find the absolute value of a number

(T) Display total counts, then exit the program

Operation? **x**

You selected maximum.

Enter your first number: **55.44**

Enter your second number: -**1.6**

The answer is 55.44

[Display the menu again, say user selects “T”]

---

Sample results from one session; match the following format.

Calculator Report

Minimum problems: 0

Maximum problems: 1

Power problems: 0

Absolute value problems: 0

Total problems: 1

Test Plan

* Run the following test data using the exact numbers shown. If you use different numbers, you will lose points. Be sure to check that each result is correct.
* All results display commas when appropriate, result rounded to 2 places, like 1,234,567.89
  + test 1: minimum of (42, 84) = 42.00
  + test 2: maximum of (42,84) = 84.00
  + test 3: 3 to the 3.5 power = 46.77
  + test 4: Absolute value of -42 = 42.00
  + test 5: minimum of (3.14159,2.71828) = 2.72
  + test 6: 3.14159 to the 15th power = 28,657,782.87
  + test 7: maximum of (-42.0,42) = 42.00
  + totals
    - Minimum problems: 2
    - Maximum problems: 2
    - Power problems: 2
    - Absolute value problems: 1
    - Total problems: 7

Note: no tests for bad data are needed since this program is guaranteed a perfect user (i.e., no bad input).