CST8116 Assignment 03

Software Development Process, Object Oriented program with decisions and loops.

Instructions

- The Software Development Process as presented by Cay Horstmann [1] will be used as the basis for this lab assignment.
  1) Understand the problem
  2) Develop and Describe an Algorithm
  3) Test Algorithm with Simple Inputs
  4) Translate the Algorithm into Java
  5) Compile and Test Your Program

1) Understand the problem

- A client contracted your company to create a program that has the following requirements. The program needs to permit entry of two numbers as operands for use in simple arithmetic binary operations with one number used on the left-side of the operator, and the other on the right-side of the operator. The program should display a menu that permits a user to edit the two operands (the numbers), to display the two operands currently in memory, and perform addition, subtraction, multiplication, division, and remainder. The program must continue interacting with the user until an exit program option is selected. Additionally, the program must not crash if an invalid menu option is input, or an invalid value for either operand. The minimum and maximum numbers the program should accept for operands are -10000.0 to 10000.0

- Briefly outline how your program would validate the user input for a single number, as well as how your program would interact with a user based on menu choices.

- A junior programmer started the project and has completed some of the work with some tests in method main. You are provided some starter code, UML class diagrams and will need to focus on completing the program. Chiefly, handling bad input for the number entries, and creating an interactive program loop in method main to run the program. An example of how the program should behave, worked out with the client previously is included. Reviewing the UML class diagrams you realize they have some mistakes that need to be corrected, and a review of the source code shows that there are no programmer comments which you will need to add, you can use author: junior programmer, with a line - modified by: *Your Full Name* within your comment header block.

2) Develop and Describe an Algorithm: using objects, loops and decisions

- Create and document pseudocode and flowcharts for: handling user input for an integer number, user input for a double number, and the main-method logic of the program. Your coding for the double user-input will be similar for the int input, however use EPSILON to check for equality of edge-cases for the double input.
- Tip: Solve the integer input problem first and test it, then adapt it for double, then create the program loop in method main. Or work on method main first, then work on validating the inputs within the class UserInput methods afterwards (in other words do not work on method main, and validating input at the same time, focus in on one method at a time).
- Update the UML class diagrams to correct them and fill in any missing pieces. (UMLet file is provided).
- The pseudocode and flowchart should document the exact same algorithm.
- If working on paper either scan the diagram into your computer, or use a cell phone to take a picture and email it to yourself Both your pseudocode and flowchart should be placed into your MS Word document, the pseudocode as text while the flowchart should be an image. **Ensure your name is visible within your flowchart images, either entered into a box shape to the side, or printed legibly by hand.**

3) Test Algorithm with Simple Inputs

- Create three testing tables with some sample data to check your three methods, the two data entry methods, and the main method. Use valid inputs, edge cases (min/max and outside the range), and invalid inputs like text, numbers too large, or wrong data type. **See the appendix for an updated test plan format.**
- If there is a problem with the algorithm based on this desk-check correct the pseudocode and flowchart, UML class diagram(s) and repeat this step again. (You only need to turn in the final table, no need to document all of your troubleshooting here.

4) Translate the Algorithm into Java

- You are to use the Eclipse IDE to create your Java program, use a project name like Assignment 03.
- You may use any decision structures (if, switch) and any loops (while, do-while, for) in implementing your program.
- Don't forget to comment your code files, with the expected code header, as well as comment the class, constructor(s), and all method headers.

Note: You are not required to copy and paste code into the MS Word document as required previously, however you must upload your .java source code files along side of your MS Word document.

5a) Compile and Run Your Program

- Compile and run your program, using your documented test values.
- Take screen shot(s) of the eclipse console ensure your name as output by the program is captured in the image. Add brief statements below each image to describe what was captured "program start up" "program validating against text entry" "program validating numbers outside of range" "program performing add operation" etc. Review the appendix at the end of this document and envision making small screen shots of each part.

5b) Test Your Program

- **Copy your testing table from step 3 into this section**, but perform all of the tests again within your Java program to document the actual outputs with the inputs.
- Your program should not crash due to invalid input from a user.
- **Note: Your MS Word document must have two separate test tables, one for the algorithm and one for the program testing. If you only include one table in your document you will loose marks.**

Microsoft Word Document Format

See the template example, suggested headings below:

1) Understand the problem
    - Brief paragraph
2) Develop and Describe an Algorithm
    - UML class diagrams, Pseudocode, Flowchart
3) Test Algorithm with Simple Inputs
    - Test plan as table
4) Translate the Algorithm into Java
    - Program source code, with programmer comments.
5a) Compile and Run Your Program
    - Screen shot of running program with student name as output
5b) Test Your Program
    - Second test plan as a table
6) References:
    - Document any sources you used that are either from your textbooks, or external from the lecture handouts.

Grading (18 points)

Criteria	Missing / Poor (0)	Below Expectations (1)	Meets Expectations (2)
Understand the problem	Missing or poorly done.	Not all requirements are documented, steps are not in a sequential order, missing some decisions or repetitions.	Problem solution statement is correct, with logical sequence of steps including decision structure(s) and repetition structure(s)
Algorithm: UML class diagram	Missing or poorly done, or program does not use objects designed by the student.	Class diagram(s) are not in correct format, properties and methods may not be assigned correctly to the classes and / or there is only one class that contains all of the program functionality.	Class diagram(s) are correct format, properties and methods are assigned to appropriate classes, based on the word problem.
Algorithm: pseudocode	Missing or poorly done, or program does not use objects designed by the student.	Does not meet all requirements: Correct format, steps are in an order that produces correct results, uses decisions and loops, plans logic for requested methods.	Meets requirements: Correct format, steps are in an order that produces correct results, uses decisions and loops plans logic for requested methods.
Algorithm: flowchart(s)	Missing or poorly done, or program does not use objects designed by the student.	Does not meet all requirements: Flowchart has start, stop, input/output, processing, and sequential flow, decisions, loops, plans logic for requested methods and / or flowchart algorithm matches pseudocode algorithm and flowchart parts are put together correctly.	Meets all requirements: Flowchart has start, stop, input/output, processing, and sequential flow, decisions, loops, plans logic for requested methods and / or flowchart algorithm matches pseudocode algorithm and flowchart parts are put together correctly.
Test Plan: Algorithm	Missing or poorly done.	Does not meet all requirements: Uses specified format to test the specified algorithms. Has test cases for valid input, invalid inputs as requested.	Meets all requirements: Uses specified format to test the specified algorithms. Has test cases for valid input, invalid inputs as requested.
Source Code	Missing or poorly done, missing student full name in programmer comment at top of file and / or program does not use classes created by the student.	Does not meet all requirements: Comment header at top of each source code file, each class and class-member has brief comment before the class, constructor, or method header. Program logic produces correct results. Java programming conventions followed.	Meets all requirements: Comment header at top of each source code file, each class and class-member has brief comment before the class, constructor, or method header. Program logic produces correct results. Java programming conventions followed.
Screen Shot(s) Running Program	Missing or poorly done or students name is not in the output of the program within the screen shot.	Does not meet all requirements: Screen shot(s) must show inputs, outputs, including student's full name and demonstration of all program features. Brief description provided under each image indicating what is depicted.	Meets all requirements: Screen shot(s) must show inputs, outputs, including student's full name and demonstration of all program features. Brief description provided under each image indicating what is depicted.
Test Plan: Program	Missing or poorly done.	Does not meet all requirements: Uses specified format to test the specified methods. Has test cases for	Meets all requirements: Uses specified format to test the specified methods. Has test cases

		valid input, invalid inputs as requested.	for valid input, invalid inputs as requested.
Submission	Missing	Student does not upload both the MS Word document and .java files, and / or does not follow lab professor's directions for submissions.	Student does upload both the MS Word document and .java files, and follows lab professor's directions for submissions.

Submission Requirements

- Upload your MS Word document as well as your Java file(s) to the Brightspace submission area by the due date. (See Brightspace for due date).
- Follow your lab professor's instructions regarding lab submissions for their lab section.

References

[1] Cay Horstmann. (2019). Big Java Early Objects. 7th Ed. Wiley.
[2] Joyce Farrell. (2018). Programming Logic & Design Comprehensive. 9th Ed. Cengage Learning.

Appendix: Test Plan Examples

Sample Test Plan Tables Algorithm / Java Program

- There are more tests than shown here for you to create, so use this as a starter and add more.
- Internal input to a method means the arguments passed into the method parameters
- External input comes from outside of the program, e.g. user input
- Internal output is the return value from the method
- External output would be messages sent to the user, e.g. error messages

Test for method input(int,int) of class UserInput

Internal input (arguments)	External input (user)	Expected internal and or external output	Actual internal and or external output	Description
minValue = 1 maxValue = 8	8	return 8 No external output	return 8 No external output	Matches
minValue = 1 maxValue = 8	"Tuna Fishy Fish"	Output: Please enter an integer number between 1 to 8 (repeat input loop)	Output: Please enter an integer number between 1 to 8 (input loop repeats)	Matches

Test for method input(double, double) of class UserInput

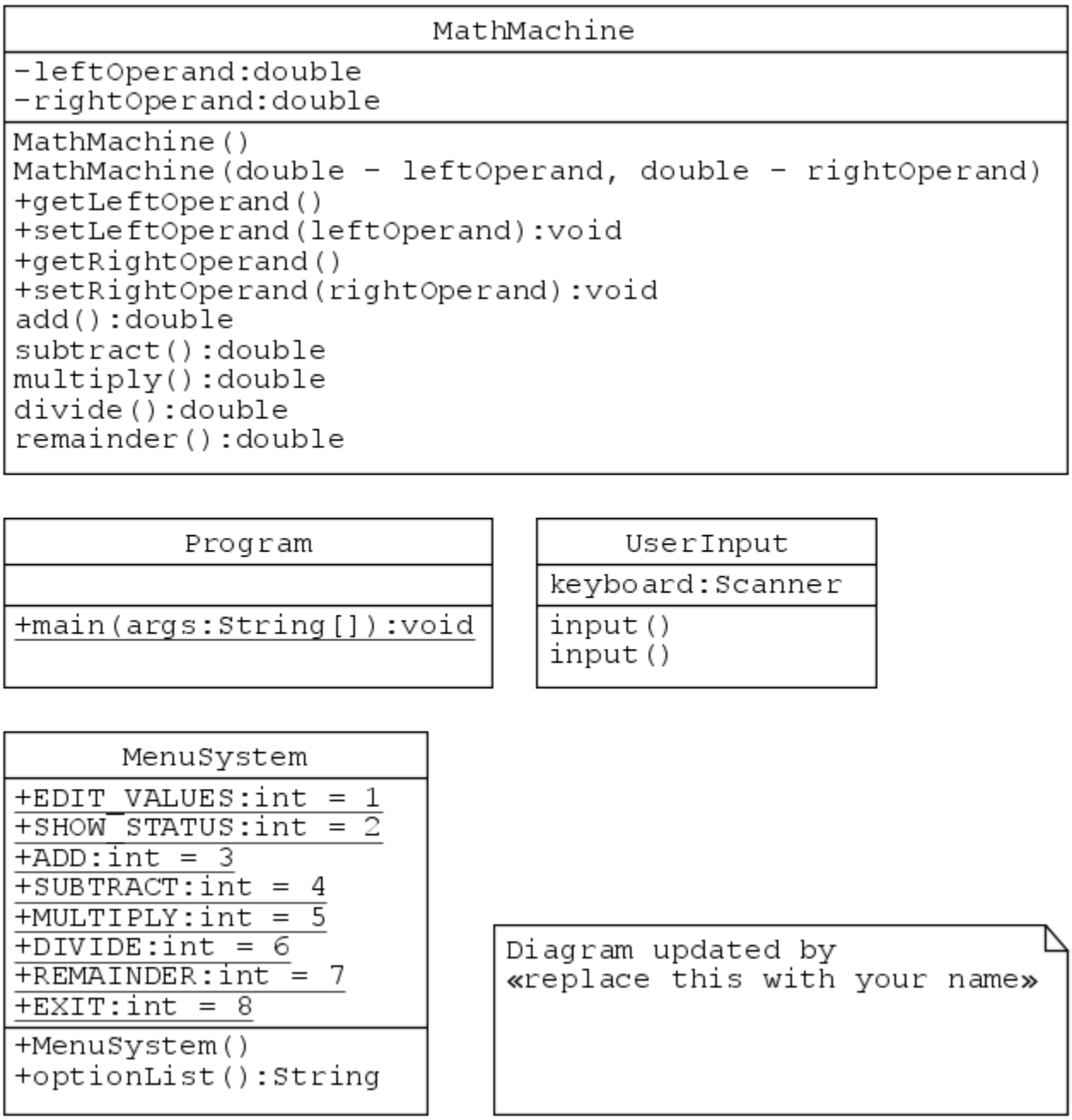
Internal input (arguments)	External input (user)	Expected internal and or external output	Actual internal and or external output	Description
minValue = -1000.0 maxValue = +1000.0	42	return 42.0	return 42.0	Matches

Test for method main(String[]) of class Program

Internal input (value returned from method)	External input (user)	Expected internal and or external output	Actual internal and or external output	Description
menuOption = 8	(none, the data was returned from method input of class UserInput	Thanks for using the program. (Program shuts down)	Thanks for using the program. (Program shuts down)	Matches

Appendix: Initial UML Class diagrams

- These do have errors and omissions, correct and update.
- The UMLet file used to create these diagrams is included with your starter files.



Appendix: Example of program use

Note: User inputs are in bold font, highlighted, example **user input**

```
Please select from an option below:
1 to edit the operands
2 to view operand values
3 to add the operands
4 to subtract the operands
5 to multiply the operands
6 to divide the operands
7 to calculate the remainder
8 to exit the program
Program by Stanley Pieda
tuna fishy fish
Invalid input. Enter integer numbers from 1 to 8
42
Invalid input. Enter integer numbers from 1 to 8
0
Invalid input. Enter integer numbers from 1 to 8
1
Enter left operand value: tuna
Invalid input. Enter decimal numbers from -1000.000000 to 1000.000000
3333
Invalid input. Enter decimal numbers from -1000.000000 to 1000.000000
-3333
Invalid input. Enter decimal numbers from -1000.000000 to 1000.000000
20
Enter right operand value: fish
Invalid input. Enter decimal numbers from -1000.000000 to 1000.000000
2222
Invalid input. Enter decimal numbers from -1000.000000 to 1000.000000
-2222
Invalid input. Enter decimal numbers from -1000.000000 to 1000.000000
30

Please select from an option below:
1 to edit the operands
2 to view operand values
3 to add the operands
4 to subtract the operands
5 to multiply the operands
6 to divide the operands
7 to calculate the remainder
```

```
8 to exit the program
Program by Stanley Pieda
2
Left Operand: 20.000000, Right Operand: 30.000000

Please select from an option below:
1 to edit the operands
2 to view operand values
3 to add the operands
4 to subtract the operands
5 to multiply the operands
6 to divide the operands
7 to calculate the remainder
8 to exit the program
Program by Stanley Pieda
3
20.000000 + 30.000000 is 50.000000

Please select from an option below:
1 to edit the operands
2 to view operand values
3 to add the operands
4 to subtract the operands
5 to multiply the operands
6 to divide the operands
7 to calculate the remainder
8 to exit the program
Program by Stanley Pieda
4
20.000000 - 30.000000 is -10.000000

Please select from an option below:
1 to edit the operands
2 to view operand values
3 to add the operands
4 to subtract the operands
5 to multiply the operands
6 to divide the operands
7 to calculate the remainder
8 to exit the program
Program by Stanley Pieda
5
20.000000 * 30.000000 is 600.000000
```

```
Please select from an option below:
1 to edit the operands
2 to view operand values
3 to add the operands
4 to subtract the operands
5 to multiply the operands
6 to divide the operands
7 to calculate the remainder
8 to exit the program
Program by Stanley Pieda
6
20.000000 / 30.000000 is 0.666667

Please select from an option below:
1 to edit the operands
2 to view operand values
3 to add the operands
4 to subtract the operands
5 to multiply the operands
6 to divide the operands
7 to calculate the remainder
8 to exit the program
Program by Stanley Pieda
7
20.000000 MOD 30.000000 is 20.000000

Please select from an option below:
1 to edit the operands
2 to view operand values
3 to add the operands
4 to subtract the operands
5 to multiply the operands
6 to divide the operands
7 to calculate the remainder
8 to exit the program
Program by Stanley Pieda
8
Thanks for using the program
Program by Stanley Pieda
```