CS161 Final Project

Eric Rouse

Integration1.java

Source Code

Readability

There are not very many comments in the variables section. I twould be nice to know what each varaiable is upon creation. Especially this line:

double shape,n,x,x2,a,b,b1,b2,width,height,area,hold; //initialization of variables

It isn't readily apparent what all these are for. It is efficient use of space, but not very easy to read. Also, the variable names are very basic. It would be better to use variable names that explain what the variable is.

In fact, there are very few comments throughout. The code layout (tabs divisions, etc) is very good, but it is hard to follow without comments.

Overall I would say the readability could use quite a bit of work.

Structure

The code does not make use of method calls, but that was not required in this assignment. It could be broken up a little better especially the "How many?", "Starting X Value", "Ending X Value", which is repeated several times.

The loops are well laid out, they perform the required input gathering. Unfortunately they don't catch the ever present InputMismatchException. So any non-integer the user enters crashes the program.

It does make sure that the second bound is higher than the first. It does this by reversing them if the second bound is lower, which is a really good idea. It doesn't require the user to enter any new numbers.

User Interface/Running the Code

Effectiveness

The program does not output the result of 2x^2. It ouputs 0.0 for every input. It correctly collects inputs (if they are integers) and continues until the user selects quit. It appears that the if statement used to make sure a is less than be is incorrectly applied. It reverses a and b if a is less than b. Should be done only if a is greater than b.

Robustness
The program does not handle invalid menu choices, it continues on as though you selected 1 two or three. It also doesn't handler input mismatches. It just crashes.

Usability
It is user friendly when it gathers information from you. It doesn't tell you why you are giving it information right off the bat, by which I mean you don't know the point of the program, it never explains it to you. Which would be nice. It does not print any error messages at all

Integration2.java:

Source Code

Readability
Code readability is really great, everything is indented nicely and there are plenty of descriptive comments. The variables are named intelligently and there are descriptive comments for variables that are unclear. I really appreciate the use of the DecimalFormat object to make intelligible decimal output.

Structure
The structure could use some work I think, as there are many, many nested while loops. I wonder if they are all completely necessary. The first while loop under the primary while loop isn't needed if a try/catch is used to get the input mismatch error. Really this is a stylistic choice, probably. I personally don't like so many nested while loops, preferring try/catch, but errors are caught, which is the main thing.

Other than that minor note everything has a very nice structure, nothing is repeated that isn't absolutely necessary.

User Interface/Running the Code

Effectiveness
The program collects the appropriate input and doesn't crash if the user inputs bad data. It continues until the user decides to stop. The final area calculation appears to be right on.

Robustness
The program graciously handles all erroneous input. It tells you that your input was bad and gives you as many chances as you need to input good data. It handles errors in all input areas of all error types.

For instance, if the bounds are incorrect it tells you the lower bound must be smaller than the upper bound and lets you pick a different input.

Usability
The program is incredibly user friendly helping you along on your integration journey. The only thing it doesn't do that it probably should is let you pick a different amount of trapezoids than rectangles.

Integration4.java

Source Code

Readability

This code is pretty readable. It has lots of good spacing and indentation. The variable names are very clear and descriptive in and of themselves and the comments increase the awareness of the authors intent.

Structure

I see a lot of good structure here, very modularized/compartmentalized. Appropriate variables are used and not there are not too many.

This author used bits to make pseudo methods (doRect, doTraps). If the user chooses to integrate by rectangles doRect is set to true and later an if statement operates based on this. This divides and conquers very well.

User Interface/Running the Code

Effectiveness

The program is quite effective in that it calculates the function as it is supposed to. It gathers input as it is supposed to and it allows the user to keep going until they quit. They prompt the user along with some very clear prompts.

Robustness

This program handles all possible errors in all possible input states . It tells you that your input was bad and gives you as many chances as you need to input good data. It handles errors in all input areas of all error types.

For instance if you don't select 1, 2 or 3 during the "Which method" input time it lets you know it needs an input of 1, 2 or 3.

Usability

Of the three programs I looked at this is the most user friendly. It uses a lot of whitespace and doesn't have any unnecessary or overly repeated prompts.

Self Reflection

For me, it really struck me how alien it is to view someone elses code. Like landing on someone's homemade planet and trying to find your way around. Even though you made the same kind of planet yourself.

Comments and well named variables are critical. It is a bad idea to just assume that they are obvious based on context or name. Everything should be commented. It doesn't take more than a few words on each variable to let someone know what you are doing with it.

Also an introductory sentence or two at the start of each loop seems like a good idea. This way a person reading your code doesn’t have to delve into the loop operation to figure out why it is there.

Finally, there are so many ways of accomplishing the same things. I looked at all five versions of the code and my methods lined up with some but not others. It is quite interesting to see how many different ways people solved this problem.