Project 3: Fun with Curve Math

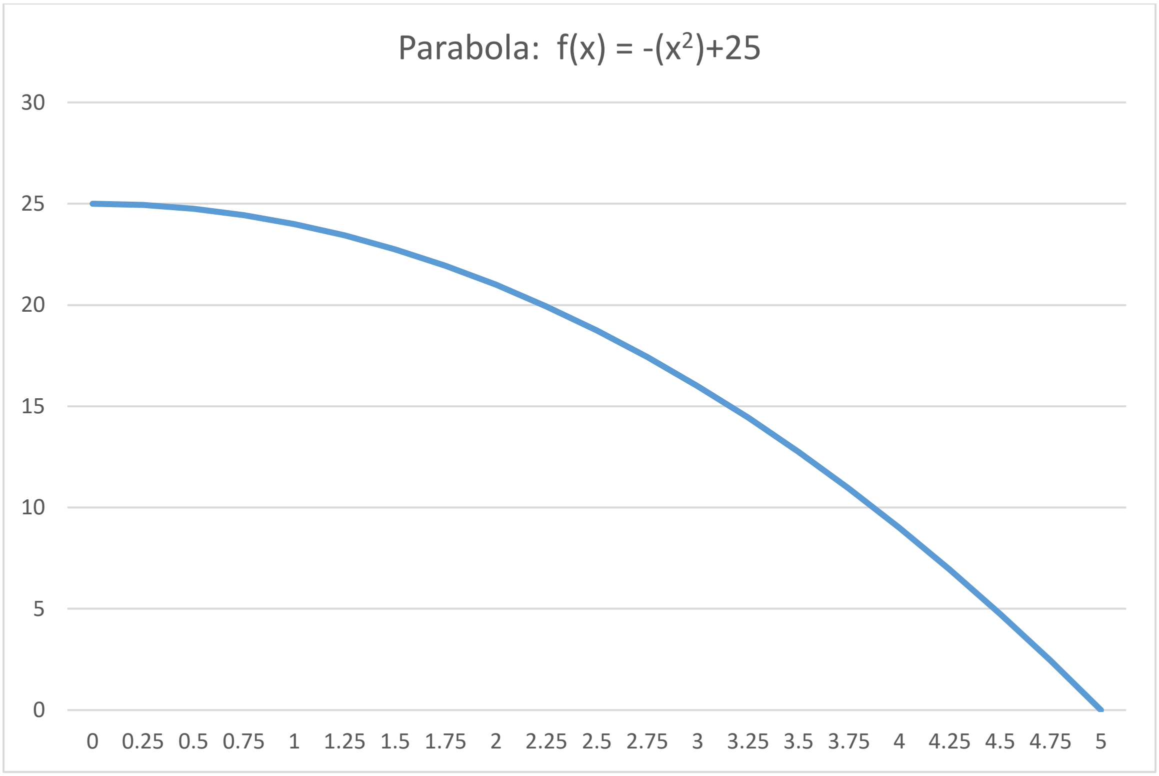
Please carefully read this *entire document* before beginning your work.

1 Objective

This project will require indefinite loops to do its work. You will also get a chance to display information to, and gather input from, the user. The project's subject matter ties in nicely with the math that you have studied. It will also test your ability to design your approach, develop incrementally, and test your code.

2 Background

We are working with a parabola specified by this equation: $f(x) = -(x^2) + 25$ with *x*'s domain in the range 0 to 5 (inclusive). The partial parabola looks like this:



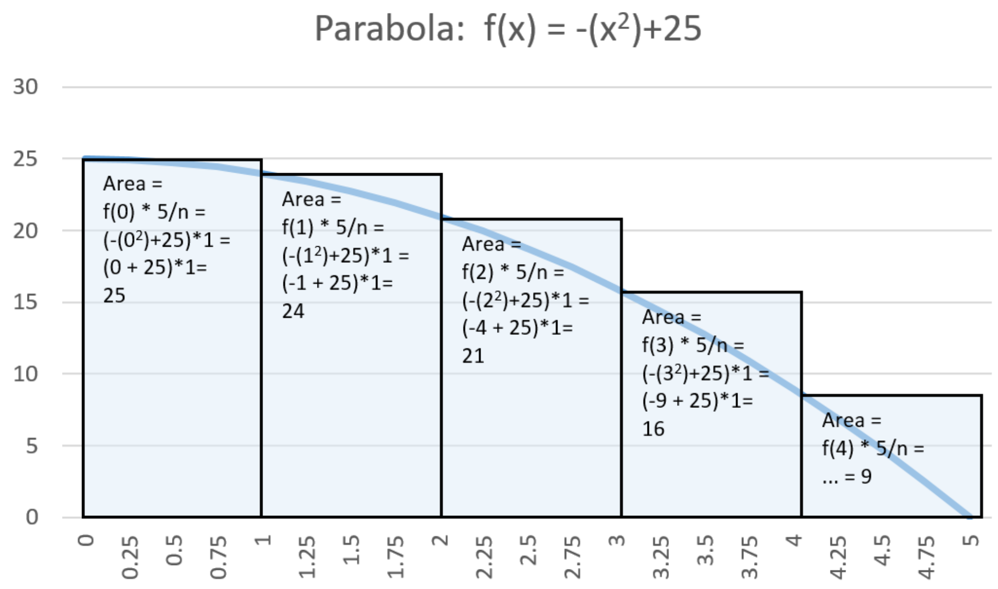
For example, when the *x* coordinate is 4, the y coordinate is $-(4^2) + 25 = -16 + 25 = 9$

The slope of the line at a given point is given by: $f'(x) = -2\text{x}$ (determined using limits or calculus), so for example when the *x* coordinate is 2.5, the slope of the line is $-2 \cdot 2.5 = -5$.

3 Area Estimation

The area under the curve can be estimated by using a series of *n* rectangles, each of which starts at a calculated value of *x*, with a width specified by 5/*n* (x's maximum domain divided by how many rectangles we are using) and the rectangle's left-side *y* determined by the parabola's defining equation. This is called a Left Riemann Sum. Note the number of rectangles might be greater than the x domain, e.g., 15 when x the graph crosses the x axis at 5 as shown below.

Were we to visualize the estimation for five rectangles, it would look like this:



The total area is the sum of the areas of all the rectangles, in this case 25 + 24 + 21 + 16 + 9 = 95. The more rectangles one uses, the closer the resulting estimation to the actual area. *Note that in this example there are a lot of integers; that will not always be the case, e.g., for n > r. Be careful with this.*

The actual area as determined by limits or calculus is: $\left(125 - \frac{125}{3}\right) = 83.\overline{33}$. So, after we calculate the area using rectangles, we can tell how far off the estimate is. In this case it's 14% too big; the overestimation is apparent from the graph, in fact.

4 User Interface

Display this menu:

```
Parabola Calculations Menu

1. Find y for a given x
2. Find the slope of the tangent line at a given x
3. Calculate area estimate
4. Exit the program

Enter your choice:
```

Keep the user within the menu system until they choose to exit, i.e., they can do more than activity while they are there. If the user enters an integer outside the range of valid menu choices or an invalid data type, tell them they have made an error. More details about the menu choices:

1. Ask the user for an x coordinate, then report to them the corresponding y value.
2. Ask the user for an x coordinate, then report to them the slope of the tangent line.
3. Ask the user how many rectangles to use, then report to them the calculated area and how far off the calculation is from the actual.
4. Exits gracefully (not artificially) from the program (i.e., a loop should end, you should not forcibly terminate the program or break out of the loop).

4.1 Gathering User Input

Use a Scanner object to get keyboard input from the user.

5 Code Implementation

Create *two* separate classes, one that deals with the user, and one that does the math computations. Examples of this type of code division can be found in MyMath.java and TestMyMath.java. Follow the Course Style Guide.

5.1 ParabolaCalc Class

- This class should have *no user communication*[1]; it is purely a *supplier* of calculation functions.
- There should be no main method here.
- Write static methods for calculations that accept parameters and return results. Provide three public methods, one for each of the specified calculations. Helper functions may be created.
- Functions *must* implement precondition tests on parameter values and throw exceptions when appropriate (see section 4.4 in your text).

5.2 ParabolaApp Class

- This "main" class should contain *all* user communication, consisting of console display and user input via the Scanner class. This class is a *client* of calculation functions from the ParabolaCalc class.
- This class should have a main method.
- You may define class constants, but no other class-level variables.
- Implement generic (not problem-specific) functions to get in-range integers and floating-point values from the user. Pass in the expected minimum and maximum; the functions should ensure the user makes an entry in the specified range *and* using the right data type (using Scanner look-ahead) and then passes back the user's choice. You'll make of use of such functions in later projects.
- Create *only* one Scanner object instance. If you don't understand why, think and ask! It's a common misconception that more are needed.

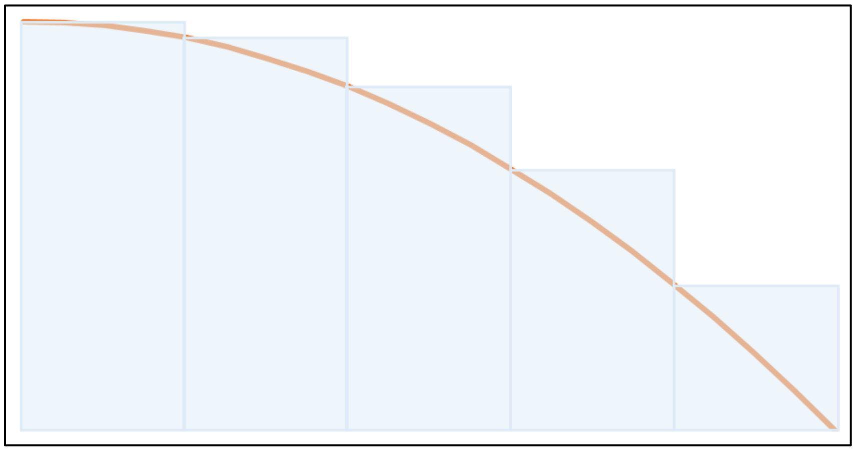
5.3 General

- Use procedural decomposition. Further decomposition is allowed if you think it helpful.

[1] That means no questions to the user, and no output intended for the user. Supplier code should have no printing in it, e.g., print, println, printf, nor printing stack traces.

6 Extra Credit

For extra credit, draw the resulting rectangles on a DrawingPanel (when the user chooses area estimation from the menu). Make sure the aspect ratio looks good (i.e., don't stretch or squish the parabola; it should look correct).



7 Testing

- Test each function individually (unit testing), then the entire program (integration testing). You don't need this to be in code for *this* project; but be aware that you must code it in future projects.
- Don't test only "happy path" inputs; you want to know what happens when bad inputs enter the system, then modify your code to handle them if you know enough to do so.

8 Submitting Your Work

Use BlueJ to create a .jar file; ask, if we haven't yet covered this in class. Be sure and specify "include source" so your code will be included. Submit the .jar file.

9 Grading Matrix

Area	**Percent**
Basic functionality and menu	
Menu display	10%
Choice processing	10%
Looping	15%
General coding	10%
Validation	
Range checking/prompting	10%
Data type checking/prompting	15%
Combined function doing both	10%
Preconditions in calculation class	10%
Documentation and style	10%
Extra Credit: rectangles drawn	3%
Total	**103%**