Question: Integral of a Polynomial (100 points)

Please read important information first.

Description

In this project you are going to calculate the definite integral of a polynomial using **Riemann Sums**

Riemann sum is a numerical method to approximate the integral of a continuous function f(x). The method applies the following steps:

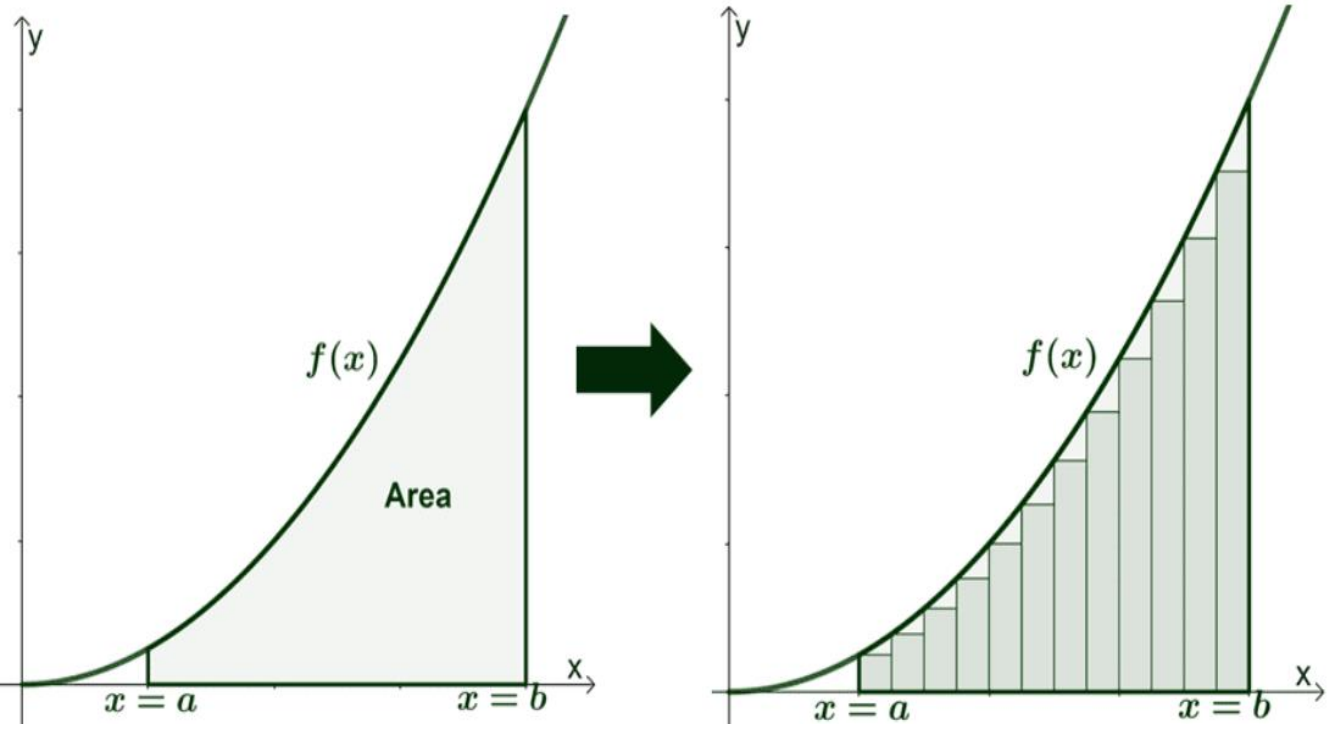
- Divide the integration intervals to subintervals.
- Approximate the area under the curve as a rectangle in each subinterval.
- Sum the area of the approximated rectangles.

Thus, integrating f(x) over integral [a,b] can be mathematically formulated as:

$$\sum_{k=0}^{h-1} f(a + k * h) * h \text{ where } h = \frac{b-a}{n}$$

Where n is an arbitrary number that denotes intervals.

A nice visualization from storyofmathematics.com/riemann-sum can be seen as below, here note that we are approximating the area under f(x) from point a to point b using rectangles. An important point is to realize that width of these rectangles are always same, and equal to a small arbitrary value, **DeltaX**, such as 0.0001, 0.001 so on.



You are required to complete the Polynomial class with the following UML diagram:

Polynomial	
-coefficients: ArrayList<Integer>	Coefficient list, this will store all coefficients in the given string as integers
-deltaX: double	Value of deltaX, width of rectangles
+Polynomial(polynomial: String)	Constructor, it will get the polynomial as a String, will extract the coefficients and store them in the coefficients ArrayList
+valueAt(point: double):double	ValueAt method will get a point and calculate the polynomial value of that point. (will calculate f(x))
+setDeltaX(deltaX: double):void	SetDeltaX method will be used to change the value of deltaX of polynomial class
+computeIntegral(min: int, max: int):int	ComputeIntegral method will get the min and max value, that are showing the interval of the computed integral This method will use coefficients, deltaX and valueAt method to solve the problem with Riemann Sum

Notes

- **Your code will be tested in the following manner.**

At first, a polynomial will be created.

```
Polynomial p1 = new Polynomial("x^3-x^2-3x+5")
```

Given polynomial will be in this form: "ax^4+bx^3+cx^2+dx+e". It will be 4th degree equation at most. The equation may or may not have all coefficients, for example it may not have a term with x^2, x^3 or so on (We can give a polynomial such as "ax+b"). All terms may not be in this order, for example we can give the same polynomial as "bx^3+ax^4+dx+e+cx^2".

We will assume the given polynomial is in valid form. Note that this form ("ax^4+bx^3+cx^2+dx+e") does not have spaces between characters, it has ^ to show degree and it does not have * character. It can have + or - between coefficients.

Then we may change deltaX:

```
p1.setDeltaX(0.0002)
```

At last, we will compute the integral:

```
p1.computeIntegral(0,10)
```

- Note that computeIntegral method returns an integer, you will compute Riemann sum as a double value however you need to return it as an int using type casting. As an example, you can have `double integration` in the method, holding the value of Riemann sum, you should return `(int)integration`. We need to do this to get consistent results on test cases.

Grading

- Your code will be tested in different scenarios to compute the integral. You need to consider edge cases to be able to get full points. You will get points from every testing scenario. Testing will be done in the manner that was explained in the previous section. Therefore if your setDeltaX method does not work, your code may fail from some of the test cases. Make sure that all of your methods work.
- You are expected to use valueAt method in computeIntegral method.
- You are expected to fill the coefficients in coefficients ArrayList of Polynomial class and use it in your solution.

Package classification

This is the package classification described in important information.

- **Changeable**
  - question