Integration Project

Using a console application the user should be able to compute the numerical integral of a polynomial function using three different methods. The first method is called the Rectangle method (midpoint or mid-ordinate rule) for approximating the integral. The second one is called the Trapezoidal method (trapezoid rule or trapezium rule) for approximating the integral. The third and final method the user can use is the exact numerical solution based on the symbolic integration of the polynomial (basic rules for integrating a polynomial).

Three Methods of Integration

Contents

Initial Scope

Project (Customer Problem/Needs)

Using a console application the user should be able to compute the numerical integral of a polynomial function using three different methods. The first method is called the Rectangle method (midpoint or mid-ordinate rule) for approximating the integral. The second one is called the Trapezoidal method (trapezoid rule or trapezium rule). The third and final method a user can use is the exact numerical solution based on the symbolic integration of the polynomial (basic rules for integrating a polynomial).

Software Used

1. Language: Most recent Java 8
2. Console Application (no GUI)
3. IDE: Eclipse
4. Version Control: GitHub (Source code and other files)
5. Multiplatform: Windows, Linux, etc.

Program Initial Description (Provided by Customer)

This will run as a console application. It will accept input from the user (should not crash/break and handle the errors).

The steps for the program are

1. Prompt the user for the degree of the polynomial.
2. Depending on the degree prompt the user for the right amount of coefficients.
3. Prompt the user for the bounds of the integral (integrate from a to b).
4. Prompt user to select integration method as described in depth below.
    a. Rectangle method (Prompt user how many columns to subdivide)
    b. Trapezoidal method (Prompt user how many columns to subdivide)
    c. Symbolic Integration

This program will continuously loop as described above, asking the user if they would like to run another polynomial integration calculation.

Rectangle Method

```
RectangleMethod(a, b, numberOfColumns, function)
        range = b – a
        midpoint = range/2
        sum = 0.0
        for index = 0 to numberOfColumns – 1
                x = a + range * (index + midpoint) / numberOfColumns
                sum = sum + function.evaluate(x)       /**evaluate - Horner’s Rule Evaluate
Polynomial**/
        return sum * range / numberOfColumns
endFor
endRectangleMethod
```

Trapezoidal Method

```
TrapezoidalMethod(a, b, numberOfColumns, function)
        Range = b – a
        Sum = 0
        For index = 1 to n-1
                X = a + range * index / numberOfColumns
                Sum = sum + f.evaluate(x)
        endFor
        sum = sum + (function.evaluate(a) + function.evaluate(b)) / 2
        return sum * range / numberOfColumns
endTrapezoidalMethod
```

Symbolic Method

Using the symbolic method to solve the exact solution of the integration of a polynomial from a to b we must we the Constant Rule, Sum Rule, and the Exponent Rule.

The Constant Rule: $\int cf(x)dx = c\int f(x)dx$.

The Sum Rule: $\int [f(x) \pm g(x)]dx = \int f(x)dx \pm \int g(x)dx$.

Exponent Rule: $\int x^n dx = \frac{x^{n+1}}{n+1} + C, n \neq 1.$

```
symbolicMethod(a, b, numberOfColumns, function)
        for function.coefficient from index 0 to length of coefficient array
                function.coefficient[index] = function.ceofficient / (function.exponent[index]
+ 1)
        return function.evaluate(b) – function.evaluate(a) //evaluate uses Horner's Rule
Evaluate Polynomial
endSymbolicMethod
```

Evaluate Polynomial

Using Horner's Rule with modifications to help speed up the calculations and reduce the computing power necessary for evaluating a polynomial at point x.

```
HornersRule(x)
        Reverse coefficients from making sure from lowest degree to highest degree
        Sum = 1st coefficient with lowest degree
        For 2nd coefficient to last coefficient of array
                Sum = sum * x + next coefficient
        endFor
endHornersRule
```

User Stories (1 point about 4 hours)

This section describes the different user stories that have been thought out before working on the implementation of the project. This would usually be done in a program that would give you burn down charts such as Rally.

Estimated number of weeks before needing to complete: 3 weeks.

No.	Title	Description	Plan Estimate (Points)
1	Prompt User for Max Degree	Ask the user to enter a maximum degree of the polynomial. The degree of the polynomial should be a positive integer.	
2	Prompt User for Coefficients	Ask the user to enter the coefficients for each term of the polynomial. The coefficients could be positive or negative fractions, decimals, or whole numbers.	
3	Prompt User to Select Integration Method	Ask the user which integration method the user would like to use. Three different methods. First method is the Rectangle method. Second method is the Trapezoidal method. Third and final method symbolically solving the integration.	
4	Prompt User for range	If user selected the integration method of Rectangle method or Trapezoidal Method then ask the user to specify a range to approximate the integral.	
5	Prompt User to Continue	Ask the user if they would like solve another polynomial in the same process as above. Should loop until user decides to exit the program.	

Tasks

User Story No. 1 Task No.	Title	Description	Task Est. (Hours)	To Do (Hours)	Actual (Hours)
1	Prompt User for Max Degree	Ask the user “Please enter a max degree for the polynomial: “ and obtain user input via keyboard.			
2	Handle Invalid User Input	Should be a positive integer. Check to verify this and handle exceptions such as when user enters a character instead of positive integer.			

User Story No. 2	Title	Description	Task Est. (Hours	To Do (Hours)	Actual (Hours)

Task No.			)		
1	Prompt User for Coefficients	Ask the user “Please enter the coefficients of the polynomial separated by spaces up to the exponent plus one (Exponent number entered from above): “ and obtain user input via keyboard.			
2	Handle Invalid User Input	User could enter an integer or double (positive or negative whole number, decimal, or fraction). Check to verify this and handle exceptions such as when user enters a character instead of positive integer.			

User Story No. 3 Task No.	Title	Description	Task Est. (Hours)	To Do (Hours)	Actual (Hours)
1	Prompt User to Select an Integration Method	Ask the user “Please select the integration method you would like to use to solve this polynomial: 1. The Rectangle Method 2. The Trapezoidal Method 3. Solve by Symbolic Integration“ and obtain user input via keyboard.			
2	Handle Invalid User Input	The user selection should be an integer from one of the options above. Check to verify this and handle exceptions such as when user enters a character instead of positive integer.			
3	Implement the Rectangle Method	Implement the Rectangle Method from pseudocode to code.			
4	Implement the Trapezoidal Method	Implement the Trapezoidal Method from pseudocode to code.			
5	Implement the Symbolic Method	Implement the Symbolic Method from pseudocode to code.			
6	Show user results	Once program has an answer should print to the user the			

		polynomial they entered as readable format, for example x^2 + 1/3x + 2 as well as the answer the program has calculated for the polynomial. Example if user chose Rectangle method the output would be something like “The approximation of the polynomial, x^2 + 1 from 1 to 2, using the Rectangle method with 100 columns approximates to 3.33333.“ Need to create methods to handle printing the polynomial as a readable format as shown in the output text example.			

User Story No. 4 Task No.	**Title**	**Description**	**Task Est. (Hours)**	**To Do (Hours)**	**Actual (Hours)**
1	Prompt user for range (from a to b)	For the methods of Rectangle or Trapezoidal, ask the user “Please enter a range to approximate the integral (from a to b): “ and obtain user input via keyboard.			
2	Handle Invalid User Input	Could be a positive or a negative integer for the range. Check to verify this and handle exceptions such as when user enters a character instead of positive or negative integer.			

User Story No. 5 Task No.	**Title**	**Description**	**Task Est. (Hours)**	**To Do (Hours)**	**Actual (Hours)**
1	Prompt user to continue	Ask the user “Would you like to integrate anther polynomial?“ and obtain user input via keyboard.			
2	Handle Invalid User Input	If user entered Yes then loop through the program again. Otherwise if user entered No then exit the program. If			

		anything else is entered then it is invalid input and the question should be asked again.			

Other Tasks Task No.	**Title**	**Description**	**Task Est. (Hours)**	**To Do (Hours)**	**Actual (Hours)**
1	Pseudocode before Implementation	Write out the process first along with figuring out any possible errors that could happen when running these integration methods.			
2	Testing Different Integration Methods on Different polynomials	Test each integration method to make sure everything works as expected. Can solve own problems by hand and even use http://www.wolframalpha.com/ to help check solutions.			
3	Test Overall Program (Menus, User Inputs, etc.)	Test the overall program making sure the menus, user inputs, and exceptions are handled correctly.			
4	Test on Multiple Platforms	Test on multiple platforms. Need to at least test on a Windows and a Linux (Ubuntu) machine. (I do not have a MAC to test on but since the program is written in Java should run on multiple platforms without a problem)			
5	Configuration File for Text	Create a configuration properties file to load in the text for the program. This will make it easier for maintaining the text and changes. Also, this is a good practice for if there were customer customizations or even having it in different languages.			

Runtime Setup

Download and Install the latest Java JRE 8 version onto your machine from https://www.java.com/en/download/.