CSC 317: Project 1

Polynomial Calculator

Total: 100 points

In this project you will implement a Polynomial Calculator. The calculator will allow the user to compute polynomial addition, polynomial evaluation, and other services. The design should be general and allow easy modification for any type of scalar. You need to implement the following classes (may be abstract or interface). You can provide additional methods, if you wish.

1. **Scalar**: This class should support the following services:

   a) Scalar add(Scalar s): accepts a scalar argument and returns a scalar which is the sum of the current scalar and the argument.

   b) Scalar mult(Scalar s): accepts a scalar argument and returns a scalar which is the multiplication of the current scalar and the argument.

   c) Scalar pow(int exponent): accepts an integer argument and returns a scalar which is the power of the current scalar by the exponent.

   d) Scalar neg(): returns a scalar which is the result of multiplying the current scalar by (-1).

   e) boolean equals(Scalar s): returns true if the argument Scalar and the current Scalar have the same numeric value.

   You can implement/extend this interface/abstract class with reasonable methods that you find necessary to complete the task (Think about the derivative of a poly term – what is missing?). Two classes that extend/implement Scalar should be:

   - RationalScalar: For Rational numbers. A rational number is a number a/b, where a and b are integers, $b \neq 0$. It should be represented using two integer fields.
   - RealScalar: For Real numbers.

2. **PolyTerm**: This class represents a polynomial term. A term is represented by a coefficient (Scalar) and an exponent (nonnegative integer). The polynomial $4x^2 + 3x - 7$, has 3 PolyTerms: $4x^2$ (coeff=4, exp=2), 3x (coeff=3, exp=1), and -7 (coeff=-7, exp=0). This class should support the following services:

   a) boolean canAdd(PolyTerm pt): receives a PolyTerm and returns true if the argument PolyTerm can be added to the current PolyTerm (same power). Otherwise, returns false.

b) PolyTerm add(PolyTerm pt): receives a PolyTerm and returns a new PolyTerm which is the result of adding the current PolyTerm and the argument.

c) PolyTerm mult(PolyTerm pt): receives a PolyTerm and returns a new PolyTerm which is the result of multiplying the current PolyTerm and the argument.

d) Scalar evaluate(Scalar scalar): evaluates the current term using the scalar. For example, $2x^2$ with scalar 3/2 should return 18/4 (i.e., $2 * (3/2)^2$), or even better, 9/2.

e) PolyTerm derivative(): returns the PolyTerm which is the result of the derivation on the current PolyTerm ($2x^2$ will return 4x).

f) boolean equals(PolyTerm pt): returns true if the argument PolyTerm is equal to the current PolyTerm (same coefficient and power)

3. **Polynomial**: This class represents a polynomial. A polynomial is represented by its terms (i.e. a sequence of PolyTerms). *Note, a polynomial does not include two terms with the same exponent.* So if an operation results in multiple terms with the same exponent, they should be combined together. This class should support the following services:

   a) Polynomial add(Polynomial poly): receives a Polynomial and returns a Polynomial which is the sum of the current Polynomial with the argument.

   b) Polynomial mult(Polynomial poly): receives a Polynomial and returns a Polynomial that is the multiplication of the current Polynomial with the argument.

   c) Scalar evaluate(Scalar scalar): evaluates the polynomial using the argument scalar.

   d) Polynomial derivative(): return the Polynomial which is the result of applying first order derivation (P′(x), or dP(x)/dx) on the current Polynomial.

   e) String toString(): returns a friendly representation sorted by increasing power of PolyTerms for example: $9 + 2x - 3x^2 + \cdots$. *Hint: The Collections class (interface in java.util) has a sort function for objects that implement Comparable (interface in java.lang). You can make the PolyTerm class Comparable and sort them.*

   f) boolean equals(Polynomial poly): returns true if the argument polynomial is equal to the current polynomial.

4. **Calculator**: This class will hold the main method. Inside the main method the user will be asked for input.

In addition, every class should include getters and setters as needed, and toString. The toString method should return a friendly representation of the object. You are also free to add more methods or classes.

Input/Output

You can assume that polynomials will be entered in the format (coefficient)+(coefficient)x^1+(coefficient)x^2+..., without any spaces. Some terms may be subtracted, which can also be represented

as negated coefficient with no subtraction. Rational numbers will always appear in the form A/B. Real numbers will be printed *up to* 3 digits after the decimal point. You can assume valid input (without two terms with the same exponent, valid scalars). There is no need to reduce the fractions when displaying the output (e.g., 4/6 does not have to be printed as 2/3), but if you can then this will earn you **5 extra credit points**. You can use the String.split(<operator>) to get the string representations of Polynomial and parse each of its component separately.

Running the program

The program starts with showing the menu:

```
Please select an operation:
     1. Addition
     2. Multiplication
     3. Evaluation
     4. Derivative
     5. Exit
```

The user selects the requested operation and then the following message will show up:

```
Please select the scalar field
```

The user will either enter "R" for Reals or "Q" for Rationals. The next step, the user will be asked to insert polynomials (two or one) in separate lines according to the above format and returns a result shown in a valid form (no exponent will be shown more than once, and terms with coefficient zero do not appear).

If option 5 is selected, the program exits. Otherwise, the menu is shown again, and the user can continue.

Sample Runs

```
Please select an operation:
     1.Addition
     2.Multiplication
     3.Evaluation
     4.Derivative
     5.Exit
> 1
Please select the scalar field Rational (Q) or Real (R)
> Q
```

```
Please insert the first polynomial
> 4+3x^1
Please insert the second polynomial
> 5+2x^1+4x^2
The solution is:
9+5x^1+4x^2

Please select an operation:
     1.Addition
     2.Multiplication
     3.Evaluation
     4.Derivative
     5.Exit
> 2
Please select the scalar field Rational (Q) or Real (R)
> Q
Please insert the first polynomial
> 1+3x^1
Please insert the second polynomial
> 5+x^2
The solution is:
5+15x^1+x^2+3x^3

Please select an operation:
     1.Addition
     2.Multiplication
     3.Evaluation
     4.Derivative
     5.Exit
> 4
Please select the scalar field Rational (Q) or Real (R)
```

```
> Q
Please insert the polynomial
> 5+15x^1+1/3x^2
The derivative polynomial is:
15+2/3x^1


Please select an operation:
     1.Addition
     2.Multiplication
     3.Evaluation
     4.Derivative
     5.Exit
> 4
Please select the scalar field Rational (Q) or Real (R)
> R
Please insert the polynomial
> 5+15x^1+0.75x^2
The derivative polynomial is: 15+1.5x^1
...
```

What to submit

Submit the following files (**all required**) on Canvas:

1. A .pdf file showing the UML class diagram for the entire project.
2. A .zip file containing the source code (.java files organized possibly in (sub-)directory structures)
3. A runnable .jar file. I should be able to run the .jar from command line like so: `java -jar P1.jar`

Grading Breakdown

20 pts	UML diagram
5 X 10 pts	5 test cases
30 pts	Code organization and readability