This project will require you to create a program that will take a mathematical expression in a single variable (x) and either

- evaluate the expression for a given value of x, or
- calculate the derivative (with respect to x) of the expression

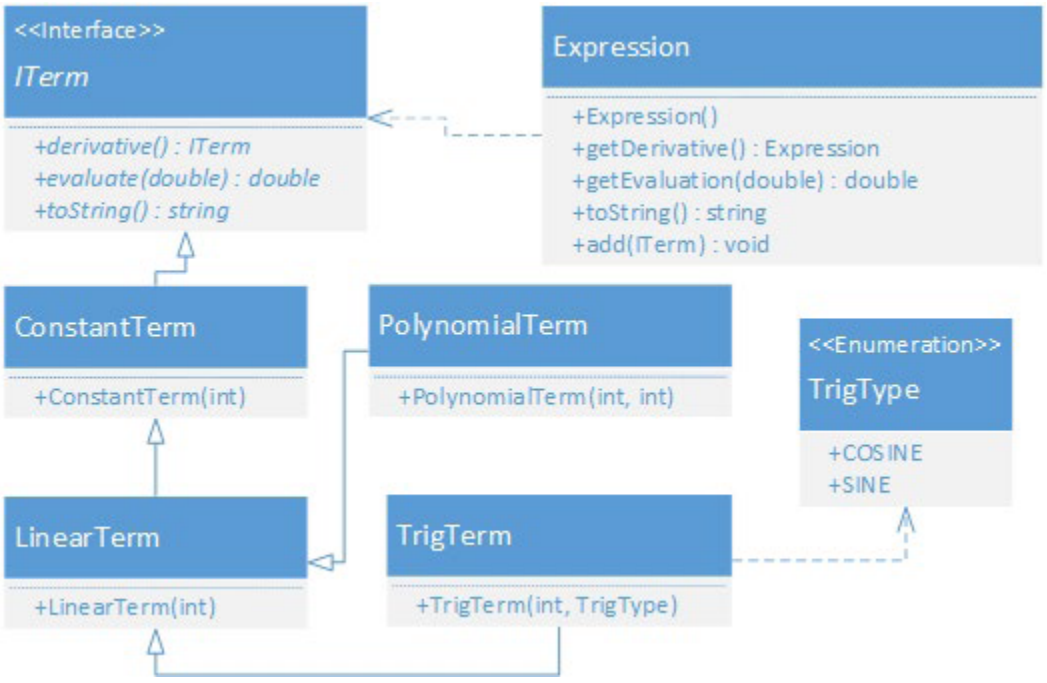
The purpose of this project is to demonstrate a working knowledge of

- Class creation
- Inheritance
- Polymorphism
- Linked Lists
- Generics

This project is designed to be a take-home part of exam #2. You are not to discuss or share code with anyone, including CSMC or tutors. You may use the book, but no other outside materials.

Diagram

The diagram only shows the public methods necessary for objects to function. You may add any additional private/protected/public members you deem necessary since they will not be tested.



`ITerm` and Implementers

You will need the following classes implementing `ITerm`:

- A `ConstantTerm` object will represent a term of the form

`±a`

where `a` is of type `int`. The derivative of a constant term is always

`+0`

- A `LinearTerm` object will represent a term of the form

`±ax`

where `a` is of type `int` and `x` is the independent variable. The derivative of a linear term is a constant term of the form

`±a`

- A `PolynomialTerm` object will represent a term of the form

`±ax^b`

where `a` is of type `int`, `b` is a positive `int` greater than one, and `x` is the independent variable. If `b>2`, the derivative of the polynomial term is a polynomial term of the form

`±(ab)x^(b-1)`

If `b=2`, the derivative of the polynomial term is a linear term of the form

`±2ax`

- A `TrigTerm` object will represent a term of the form

`±a cos(x)`

or

`±a sin(x)`

where `a` is of type `int` and `x` is the independent variable. The derivative of the sine term is a trigonometric term of the form

`±a cos(x)`

The derivative of the cosine term is a sine term of the form

`∓a sin(x)`

Note that the sign flips when taking the derivative of cos(). Evaluation of trigonometric functions should be done in *degrees*.

Each class implementing `ITerm` will need to override the following functions:

- `derivative()` - returns a new `ITerm` that represents the derivative of the current term
- `evaluate(double)` - returns the evaluation of the term with the double value substituted for `x`
- `toString()` - returns a string representation of the term (see below for examples)

`TrigType` Enumeration

The `TrigType` enumeration should have two values used to distinguish between the trigonometric functions:

- `COSINE`
- `SINE`

`ProjNode` and `ProjLinkedList`

You will need to create a `ProjNode` class that holds data and points to the next node. You will also need a `ProjLinkedList` class that contains a 'head' object, and the necessary functions. Both classes will need to be generic to contain any type of data object, and you will need to use them in your `Expression` class to store `ITerm` objects. At minimum I expect the `ProjLinkedList` to contain:

- a default constructor
- `add(T)` - a function that adds to the beginning of the list
- `getAt(int)` - a function that returns the value at the `int`-th node (zero-based)

`Expression` class

The `Expression` class will need to contain a `ProjLinkedList` of `ITerm`s. It will also need the following functions:

- `getDerivative()` - returns a new `Expression` object containing the derivative of each term of the original object. Zero-valued constant terms should not be included.
- `getEvaluation(double)` - returns the sum of the evaluations of the individual terms.
- `toString()` - returns a string version of the expression. The polynomial terms should be displayed in descending exponential order, followed by linear, constant, sine, and cosine
- `add(ITerm)` - function that will add an `ITerm` to the expression

`main(String[] args)`

The `main()` function will not be tested. Use it for your own tests.

EXAMPLE

```
ITerm t1 = new LinearTerm(5);
ITerm t2 = new PolynomialTerm(-4,3);
ITerm t3 = new TrigTerm(-6,COSINE);
System.out.println(t1); // + 5x
System.out.println(t1.evaluate(5)); // 25
System.out.println(t2); // - 4x^3
System.out.println(t2.evaluate(2)); // -32
System.out.println(t3); // - 6cos(x)
System.out.println(t3.evaluate(45)); // -4.24
Expression e1 = new Expression();
e1.add(t1);
e1.add(t2);
e1.add(t3);
Expression e2 = e1.getDerivative();
System.out.println(e1); // - 4x^3 + 5x - 6cos(x)
System.out.println(e2); // - 12x^2 + 5 + 6sin(x)
System.out.println(e1.getEvaluation(0));// -6

System.out.println(e2.getEvaluation(0));// 5
```