Polynom & Monom classes

Our class represents Monoms and Polynoms as defined in Wikipedia.

Monom is an expression of the form a*x^b ($a*x^b$) or ax^b (ax^b) when ***a*** is a real number and ***b*** is a non-negative integer.

Polynom is a mathematical expression of one or more Monoms summed up together.

Our program ignores spaces, for example “3* x^2 “=”3*x^2”.

In our program, we have many methods which can be applied on Monoms and Polynoms.

Monom class

Every Monom object has a coefficient field represents ***a,*** and a power field represents ***b.***

The main methods in this class are as follows:

- derivative
- add
- multiply
- toString - prints the Monom's logical value
- f(x) - returns the Monom's value within the given x
- isZero - checks whether the Monom's value is 0
- a constructor that builds a new Monom according to a string

Polynom class

Polynom class implements the Polynom_able class.

Every Polynom object has one field of arraylist<Monom> type represent the list of Monoms the Polynom has. The list is sorted by power values up to down. In case of tow Monoms with the same power value they combined to one Monom. The main methods in this class are as follows:

- add- Polynom to a Polynom and Monom to a Polynom
- subtract
- multiply- by a Monom or a Polynom

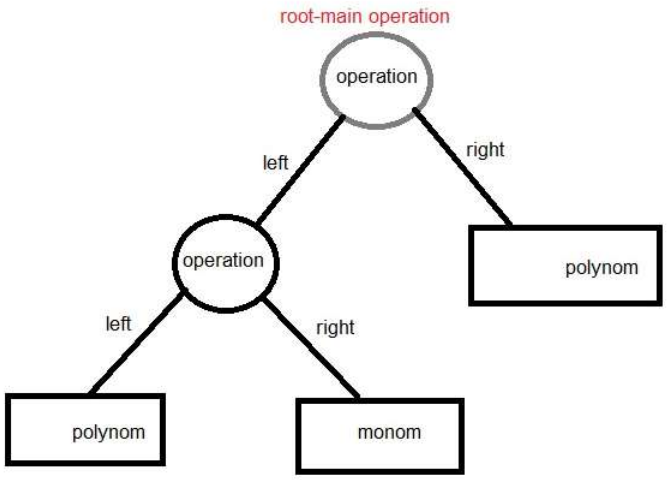
- f(x) - returns the Polynom's value within the given x
- equals
- copy constructor
- derivative
- isZero - checks whether the Polynom's value is 0
- area – returns the area value between the Polynom graph and the x axis (only the areas which are above the X axis)
- root – returns the Polynom root between tow given Xs values.
- toString - prints the Polynom's logical value sorted by power values
- a constructor that builds a new Polynom according to a string

Complex function class

Our class represents an object called complex function which defined as follows:
Left field- type function which means it could be a Polynom, a Monom or a complex function as well.
Right field- type function which means it could be a Polynom, a Monom or a complex function as well.
Op- type operation -Enum that could be one of the following only-

*__Plus__- addition
*__Times__- multiplication
*__Divid__- division
*__Max__- gets the maximum between left value and right value.
*__Min__ -gets the minimum between left value and right value.
*__Comp__- Function composition
*__None__- gets the left side value only
*__Error__- technically is a valid operation but cannot but be applied on a complex function -throws exception.

a complex function for example looks like:



Methods that can be applied on complex function:

cf is a complex function object, f is a function, x is a double, s is a String and obj is an object in this example

- cf.f(x)- returns the complex function's value within the given x according to the operations as defined.
- cf.mul/comp/div/etc.(f)- builds a new complex function which it's left side is cf's old pointer, it's right side is f copy and it's operation is the operation specified. cf now points this new Complex function.
- cf.toString – returns a string represents the complex function structure.
- cf.initFromString(s)-returns a pointer to the function the string represents (could be a complex function or a Polynom (every Monom is a Polynom.
  cf is not affected by this action.
- cf. equals(obj)- in case obj is a function, cf is logically compered to obj (suppose to return true if equal for every value in the X axis, false otherwise)
  this method is very tricky since we cannot check every value in the scale.
  in our program we decided to check 10 random ranges in addition to the range (-2,2).
  our method allows 2 unequal values in order to cover cases of holes in one function which is being compared to an equal but

continues function. This compare can of course fail since we don't check every value. Statistically it has high odds to succeed but not 100% of the time.