Monom class:

This class represents a simple "Monom" of shape a*x^b, where a is a real number and a is an integer (summed a none negative), see: https://en.wikipedia.org/wiki/Monomial The class implements function and support simple operations as: construction, value at x, derivative,
add and multiply.

constructor

public Monom(double a, int b)
this constructor get argument-'power' and ' coefficient'   and implement them in the class Monom
'a' represents a coefficient and 'b' represents power

Monom public(Monom ot)
 This is a "copy constructor" that get his argument from 'ot' Monom

Monom (String s)
this constructor get his argument from a String.
The constructor define what is a normal Monom:
the **coefficient** must be 'double' negative or positive
and the **power** must be 'int' positive
correct input for example: 'coefficient'X^'power', ' coefficient'X, X, -X, -X^'power', X^'power',
' coefficient ' ..

Functions

get_coefficient ()- return coefficient

get_power()- return power

derivative()- The derivative of a function of a real variable measures the sensitivity to change of the function value (output value) with respect to a change in its argument (input value). Derivatives are a fundamental tool of calculus. this function is doing derivative by the fprmula "xn=nxn–1"

f(double x)- return Monom value at point 'X'

isZero()- return **true** if the Monom it 0 else **false**

add(Monom m)- adding the Monom m to Monom
notice: the power of 'm' must be the same power of the Monom!

Muptiply(Monom d)- multiplu the Monom 'd' in the Monom

toString()- print as a String the Monom

equalse(Monom a)- if 'a' is the same Monom return **true** else return **false**

function initFromString(String s)- init the Monom from the String
function copy()- return a new monom

Polynom class:

This class represents a list of 'Monom's that will store in "hashMap"
Whereas the key will be the power of Monom
And the value will be the object Monom

Constructors

Polynom()- as a default "Polynom 0" , (0.0X^0).

Polynom(String s)- this constructor convert string of Polynom to hashMap of Polynom
Notice: the string must be contain a 'normal' Monoms without space whereas between the monoms must be the sing '**+**' or '**-**'

Functions

f(double x)- return Polynom value at point 'X'

add(Polynom_able p1)- adding Polynom p1 to this Polynom

add(Monom m1)- adding Monom m1 to hashMap of Polynom

substract(Polynom_able p1)- subtract Polynom p1 to this Polynom

multiply(Monom m1)- multiply p1 Monom at each this Monom and minimize same power Monom

multiply(Polynom_able p1)- multiply each p1 Monom at each this Monom and minimize same power Monom

equals(object p1)- Test if this Polynom is logically equals to p1

equalsPolynom(Polynom_able p1)- return true if the other polynom equals to this polynom

isZero()- if this is the 0 Polynom return **true** else return **false**

root(double x0, double x1, double eps)
- Compute a value x' (x0<=x'<=x1) for with |f(x')| < epsilon
- assuming (f(x0)*f(x1)<=0, else should throws runtimeException

Polynom_able copy()- return a new hashMap of Polynom

toString()- print as a String the Polynom

Polynom_able derivative()- return polynom that represent the derivative of this polynom

area(double x0, double x1, double eps)- Compute a Riman's integral from x0 to x1 in eps steps.

Iterator<Monom> iteretor()- return object iterator of Polynom

function initFromString(String s)- init the hashMap of polynom from the String

<u>class Functions_GUI</u>

<u>Functions</u>

drawFunctions(int width, int height, Range rx, Range ry, int resolution)
-Draws all the functions in the collection in a GUI window using the
given parameters for the GUI window and the range & resolution
* **width** - the width of the window - in pixels
* **height** - the height of the window - in pixels
* **rx** - the range of the horizontal axis
* **ry** - the range of the vertical axis
* **resolution -** the number of samples with in rx: the X_step = rx/resolution

drawFunctions(String json_file)-
Draws all the functions in the collection in a GUI window using the
given parameters from 'json file' and convert them to arguments

**drawFunctions(**)- draw empty func

function get(int i)- return object at 'i' from the kinkedLsit

add(function arg0)- return true if the arg0 has added to the list else return false

addAll(Collection<? extends function> arg0)- return true if all the collection has added else return false

clear()- clear all the linkedList

contains(Object arg0)- true if this linkedList contain 'arg0' else false

containsAll(Collection<?> arg0)- true if this linkedList contain all the collection else false

isEmpty()- true if this linked.list empty else false

Iterator<function> iterator() – return object iterator of function

remove(Object arg0)- return true if the object removed from the list

removeAll(Collection<?> arg0)- return true if all the objects are removed from the list

retainAll(Collection<?> arg0)-
Retains only the elements in this collection that are contained in the specified collection (optional operation). In other words, removes from this collection all of its elements that are not contained in the specified collection

size()- return the size of the linkdList

Object[] toArray()- Returns an array containing all of the elements in this listin proper sequence (from first to last element).

initFromFile(String file)- Init a new collection of functions from a file

saveToFile(String file)- save a collection of functions on file

class ComplexFunction

this class represents a complex function that base on 'tree database'
the tree can get Polymom and Monom and implement them on the tree by using the operations: **plus, mul, div, max, min, comp** (f1(f2(x)))

constructor

ComplexFunction()- A default constructor build null tree

ComplexFunction(function func)- constructor tree with left node from func
And null in right and in operation

ComplexFunction(Operation op, function left, function right)-
Constructor tree with left and right that can be function
And operation can be one of the list:
Plus, Times, Divid, Max, Min, Comp , None, Error

ComplexFunction(String op, function left, function right)-
Constructor tree with left and right that can be function
And operation can be one of the list:
Plus, Times, Divid, Max, Min, Comp , None, Error

ComplexFunction(String left)- Constructor the left tree from string

ComplexFunction(String string, String left, String right)-
Constructor the operation, left, right tree from string

Functions

initFromString(String s)- Initial a new collection of functions from a file
return a new complexFunction

FromString(String s, ComplexFunction cf1)- Initial a new collection of functions from a file and from complexFunction type
return a new complexFunction

f(double x)- return the value of the function at 'x'

getmed(String s, int indexOf)-

getop(String oper)- get operator from string

copy()- return new complexFunction

plus(function f1)- put this complexFunction in the left node and put 'plus' in the operator and f1 in the right node

mul(function f1)- put this complexFunction in the left node and put 'mul' in the operator and f1 in the right node

div(function f1)- put this complexFunction in the left node and put 'div' in the operator and f1 in the right node

max(function f1)- put this complexFunction in the left node and put 'max' in the operator and f1 in the right node

min(function f1)- put this complexFunction in the left node and put 'min' in the operator and f1 in the right node

comp(function f1)- put this complexFunction in the left node and put 'comp' in the operator and f1 in the right node

left()- return the left node

right()- return the right node

getOP()- get this operator

toString()- convert to string

equals (Object f1)- return true if 'f1' and this object are equals at 'x'
else return false