Ex 0

Monom section:

consists of the functions:

`public Monom(double a, int b)`-
constructor for Monom, gets coefficient and power.

`public Monom(Monom ot)`-
copy constructor.

`public double f(double x)`-
gets a double for the value that we want to use in the function. Than it compute it while referring to the power and coefficient.

`public Monom addM (Monom m1)`-
adding 2 Monoms, only if the power is the same in both of them, By adding Coefficient.

`public void Negative (Monom m)`-
multiply the coefficient with -1.

`public void derivative(Monom m)` –
change the coefficient to (a*power), and the power to power -1 while power is not 0.
if it's 0 the function change it to 0 and ultimately in Polynom section it gets erased.

`public Monom Mult(Monom m)`-
returns a Monom consist of the multiplication of two Monoms.

Polynom section:

`public void add(Monom m1)`-
adding Monom to the Polynom, if the power is the same as one of the Monoms in the Polynom, combines them.

`public void add(Polynom_able p1)`-
adds two polynoms , if the power is the same as one of the Monoms in the other Polynom, combines them.

`public Polynom(Monom m1)`-
Consructor, gets a monom. Uses add function to add the monom.

public Polynom()-
Empty Constructor.

public Polynom(String s)-
Consructor, gets a string of a polynom.
Acknowledge only string in the form of "a*x^b+c*x^d…" (and x/x^a or a lone number)

public void substract(Polynom_able p1)-
substract two polynoms. Uses function Negative to change all the monoms in the polynom we get to negative. And uses add (polynom_able) function to add both polynom to one.

public void Negative (Polynom_able p)-
Changes all the Monoms to negative.

public void multiply(Polynom_able p1)-
Multiply two Polynoms.

public boolean equals (Polynom_able p1)-
Checks if two polynoms are equal. Returns Boolean.

public double root(**double** x0, **double** x1, **double** eps)-
uses the bisection methods
checks where the function cuts the x-is section by parameters
and return the value of x

public Polynom_able copy()-
create a deep copy of this Polynom

public Polynom_able derivative()-
compute a new polynom which is the derivative of this polynom

public double f(**double** x)-
computes the value of the polynom by the parameter x

public double area(**double** x0,**double** x1, **double** eps)-
Compute Riemann's Integral over this Polynom starting from x0, till x1 using eps size steps

public double area_positive(**double** x0,**double** x1, **double** eps)-

Computes the area above the x-is section.

public double area_negative(**double** x0,**double** x1)-

Computes the area under the x-is section.

public Iterator<Monom> iteretor()-
an Iterator (of Monoms) over this Polynom.

boolean is_A_double(String s)-
Checks if the string can be represented as a double number.

public boolean isZero()-
Check if the coefficient of the monom is zero('zero monom'), returns true if it is.

public void removeZero()-
Remove 'zero monom' from polynom.

public void nullify()-
Empties the polynom and set size to 0.

public void sortCmpare()-

Uses the values of monom comparator to sort the monoms by the power from the biggest to the lowest.

private int size-
Size of the polynom.

private Monom m1-
New monom created. Exist inside Molist.

private Object getMoList;-
List of Monoms.

public void gui()-

Prints the graph representing the polynom.

Monom_Comperator:
public int compare (Monom m1, Monom m2)-
Compares the size of the power of two monoms, returns 1,0 or -1.
A necessary way to know how to sort the Polynom.