Universitatea Tehnica din Cluj-Napoca

Facultatea de Automatica si Calculatoare

Sistem de procesare a polinoamelor

Documentatie – tema 1 –

Czako Zoltan

Grupa: 30225

Contents

1.Obiectivul temei 3

2. Analiza problemei, modelare, scenarii, cazuri de utilizare.....3

3. Proiectare 3

3.1 Diagrama UML 4

3.2 Proiectare clase 6

3.5 Interfața grafică 7

4. Testare 14

5. Rezultate.....14

7. Bibiliografie 15

1. Obiectivul temei:

Enuntul temei:

„Propuneti, proiectati si implementati un sistem de procesare a polinoamelor de o singura variabila cu coeficienti intregi.”

2. Analiza problemei, modelare, scenarii, cazuri de utilizare

Avem de facut o aplicatie care realizeaza urmatoarele operatii pe polinoamele de o singura variabila, grad n (pentru orice polinom, in general) cu coeficienti intregi:

- Adunarea a doua polinoame
- Scaderea a doua polinoame
- Inmultirea polinoamelor
- Impartirea polinoamelor
- Derivarea unui polinom
- Calcularea valorii polinomului pentru un x particular, o valoare introdusa (exemplu x^2+x = 6, pt x=2)
- Gasirea valorii proprii a polinoamelor
- Reprezentarea grafica a polinoamelor

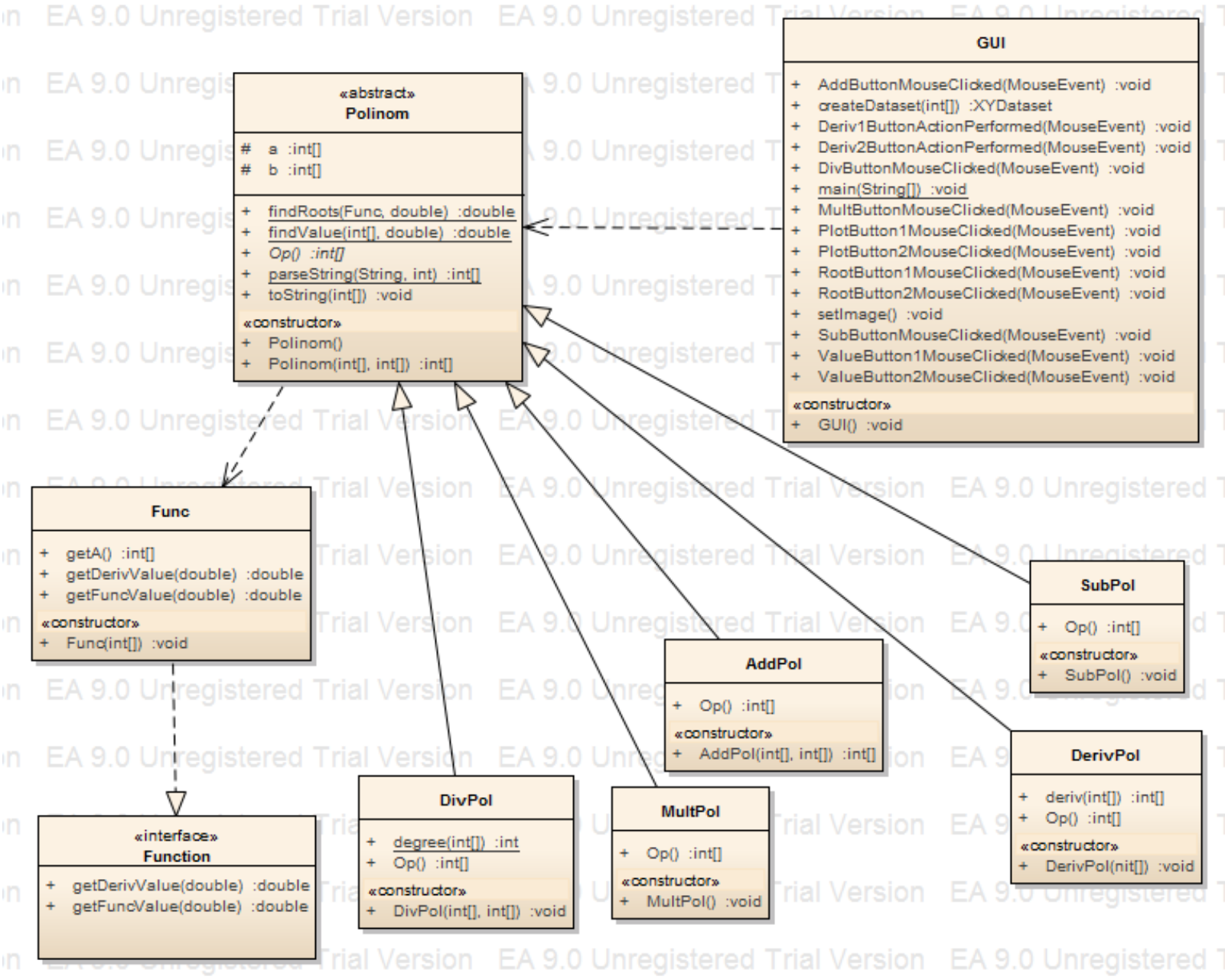
Enuntul problemei lasa loc pentru creativitate si pentru imaginatie, nu impune nicio restrictie, oferand libertate pentru programator.

Aceasta aplicatie poate fi utilizata cu scop educativ, pentru realizarea rapida a operatiilor pe polinoame, si pentru autoverificare in procesul de invatare.

3. Proiectare (diagrama UML, structuri de date, proiectare clase , interfete , relații, packages, algoritmi, interfața utilizator)

Pentru realizarea acestui proiect, am folosit conceptul de mostenire, deoarece in cazul nostru e mai eficient decat o interfata. Eficienta consta in faptul ca exista mai multe metode care sunt utilizate de aproape toate clasele fiu, dar exista si clase care trebuie suprascrise, trebuie particularizate pentru clasa respectiva. Metoda folosita de clasele fiu, fara suprascriere este metoda parseString. Exista si o metoda abstracta, metoda Op, care realizeaza diferite operatii, in functie de cerinte. Un alt avantaj al folosirii mostenirii este faptul ca fiecare clasa foloseste o memorie comuna.

Diagrama UML:



Numele packageului folosit, care contine toate clasele e tema1. Pe langa mostenire am folosit si o interfata, pentru ca proiectul sa fie deschis pentru dezvoltare, adica daca vrem sa adaugam operatii mai complexe, de exemplu operatii logaritmice, exponentiale etc. , atunci nu trebuie facat altceva decat creeam o clasa noua care implementeaza interfata Function, astfel putem calcula radacina cu aceeasi metoda findRoots, iar pentru afisarea frumoasa a ecuatiilor suprascriem metoda toString.

Clasa polinom contine metoda statica findRoots, o metoda care foloseste algoritmul lui Newton pentru gasirea radacinii unui polinom, daca polinomul e de grad mai mare decat 3, altfel pentru polinoame de grad doi calculeaza delta, iar pentru polinoame de grad 1 inverseaza semnul coeficientului liber.

Moteda statica parseString folosim pentru recunoasterea expresiei introduse de utilizator. Utilizatorul introduce polinomul in format String (de exemplu x^2+2x+1) iar aceasta metoda parseaza Stringul, gaseste puterile, cauta ceoficientile, si in functie de putere salveaza in memorie, in ordinea crescatoare a puterilor.

Pentru reprezentarea grafica a polinoamelor am folosit biblioteca externa JFreeChart. Modul de utilizare a acestei biblioteci: intr-o metoda numita createDataset pentru un sir de numere calculez valoarea polinomului in punctele respective, iar din perechile (numar, valoare) realizez un XYDataset. Folosind acest dataset cu ajutorul metodei statice createXYLineChart din biblioteca JfreeChart desenez graficul polinomului.

Pentru impartirea corecta a polinoamelor am implementat urmatorul algoritm:

```
        degree(P):
  return the index of the last non-zero element of P;
         if all elements are 0, return -∞

polynomial_long_division(N, D) returns (q, r):
  // N, D, q, r are vectors
  if degree(D) < 0 then error
  if degree(N) ≥ degree(D) then
    q ← 0
    while degree(N) ≥ degree(D)
      d ← D shifted right by (degree(N) - degree(D))
      q(degree(N) - degree(D)) ← N(degree(N)) / d(degree(d))
      // by construction, degree(d) = degree(N) of course
      d ← d * q(degree(N) - degree(D))
      N ← N - d
    endwhile
    r ← N
  else
    q ← 0
    r ← N
  endif
  return (q, r)
```

Un exemplu sugestiv pentru intelegere algoritmului:

```
0    1    2    3
   ----------------------
N:  -42    0  -12    1        degree = 3
D:   -3    1    0    0        degree = 1
```

d(N) - d(D) = 2, so let's shift D towards right by 2:

```
N:  -42    0  -12    1
d:    0    0   -3    1
```

N(3)/d(3) = 1, so d is unchanged. Now remember that "shifting by 2" is like multiplying by x^2, and the final multiplication (here by 1) is the coefficient of this monomial. Let's store this into q:

```
                            0     1     2
                           ---------------
                      q:   0     0     1
```

now compute N - d, and let it be the "new" N, and let's loop

```
N:  -42    0   -9    0        degree = 2
D:   -3    1    0    0        degree = 1
```

d(N) - d(D) = 1, right shift D by 1 and let it be d

```
N:  -42    0   -9    0
d:    0   -3    1    0        * -9/1 = -9

                           q:   0    -9     1

d:    0   27   -9    0
```

N ← N - d

```
N:  -42  -27    0    0        degree = 1
D:   -3    1    0    0        degree = 1
```

looping again... d(N)-d(D)=0, so no shift is needed; we multiply D by -27 (= -27/1) storing the result in d, then

```
                          q:  -27   -9     1

   and

N:  -42  -27    0    0        -
d:   81  -27    0    0        =
N: -123    0    0    0        (last N)

    d(N) < d(D), so now r ← N, and the result is:

       0   1  2
   -------------
q:   -27  -9  1   →  x² - 9x - 27
r:  -123   0  0   →          -123
```

Descrierea claselor:

Clasa Polinom: o clasa abstracta, care e parinte pentru urmatoarele clase: AddPol, SubPol, MultPol, DivPol, DerivPol. Aceasta clasa cuprinde o metoda abstracta Op, care e suprascrisa in clasele fiu. Are ca memorie doua siruri de int, pentru pastrarea coeficientilor iar pentru parsarea Stringului introdus de utilizator (adica polinomul) are o metoda statica parseString. Pentru gasirea derivatei si a valorii intr-un punct avem metodele statice findDerivValue si findValue. Aceste metode se mai folosesc si in metoda statica findRoots pentru implementare algoritmului lui Newton pentru gasirea radacinii polinoamelor de grad mare.

Clasele AddPol, SubPol, MultPol: Aceste clase extind clasa abstracta Polinom si fiecare are un cod particular pentru implementarea metodei abstracte Op din clasa Polinom. Aceste clase realizeaza adunare, scaderea, respectiv inmultirea polinoamelor.

Clasa DivPol: Aceasta clasa extinde clasa Polinom si implementeaza in metoda Op un algoritm cunoscut pentru obinerea rezultatului impartirii a doua polinoame si anume algoritmul Long Division.

Clasa Func: Are doua metode getFuncValue si getDerivValue si implementeaza interfata **Function,** folosit pentru generalizarea operatiilor si pentru posibilitatea extinderii a proiectului.

Pentru realizarea interfetei grafice, am folosit componente ale Swingului. Aceste componente sunt pastrate intr-o clasa separata, numita **GUI** (de la grafical user interface).

Descrierea clase GUI:

Aceaste clasa are o metoda initComponent, care se apeleaza in momentul de pornire al aplicatiei si realizarea initializarea componentolor, crearea butoanelor, textfieldurilor, incarcarea backgroundului, pozitionarea componentelor, crearea JFramelui in care lucram, setarea titlului, layout-ului, dimensiunea, stilul etc. Acestuia, adaugarea Event Listenerurilor, incarcarea imaginilor etc.

Butoanele trateaza Event-urile (MouseClicked) asemanator, de aceea voi descriu numai un singur mod de tratare

```
SubButton.addMouseListener(new java.awt.event.MouseAdapter() {
        public void mouseClicked(java.awt.event.MouseEvent evt) {
           SubButtonMouseClicked(evt);
        }

private void SubButtonMouseClicked(java.awt.event.MouseEvent evt) {

        TxRes.setText(" ");

        String p1,p2 = new String();
        p1 = TxPol1.getText();
        p2 = TxPol2.getText();
        int g1,g2;

        g1 = Integer.parseInt(TxGrad1.getText());
        g2 = Integer.parseInt(TxGrad2.getText());

        try
        {
           Polinom pol = new SubPol(PolinomOperations.parseString(p1,
g1),PolinomOperations.parseString(p2, g2));
           int[] rez  = pol.Op();
           TxRes.setText(pol.toString(rez));
        }
        catch(NumberFormatException e)
        {
           JOptionPane.showMessageDialog(this,"Wrong polinom format","Polinom
Error",JOptionPane.ERROR_MESSAGE);
        }
     }
```

Daca apare un mouse click pe butonul respectiv, mouse listenerul apeleaza metoda descrisa mai sus, care citeste din JTextFilduri Stringul introdus, parseaza cu ajutorul ajutorul metodei parseString, iar dun coef1 si coef2 creeaza un obiect. Pentru acest obiect apeleaza operatia respectiva cu ajutorul metode Op iar rezultatul afiseaza formatat cu ajutorul metodei toString.

Asemanator :

```
MultButton.setText("Mult");
        MultButton.addMouseListener(new java.awt.event.MouseAdapter() {
            public void mouseClicked(java.awt.event.MouseEvent evt) {
                MultButtonMouseClicked(evt);
            }
        });
        MultButton.addActionListener(new java.awt.event.ActionListener() {
            public void actionPerformed(java.awt.event.ActionEvent evt) {
                MultButtonActionPerformed(evt);
            }
        });


    private void MultButtonMouseClicked(java.awt.event.MouseEvent evt) {

        TxRes.setText(" ");

        String p1,p2 = new String();
        p1 = TxPol1.getText();
        p2 = TxPol2.getText();
        int g1,g2;
        int[] coef1, coef2;

        g1 = Integer.parseInt(TxGrad1.getText());
        g2 = Integer.parseInt(TxGrad2.getText());

        try
        {
            coef1 = PolinomOperations.parseString(p1, g1);
            coef2 = PolinomOperations.parseString(p2, g2);
            Polinom pol = new MultPol(coef1,coef2);
            int[] rez  = pol.Op();
            TxRes.setText(pol.toString(rez));
        }
        catch(Exception e)
        {
            JOptionPane.showMessageDialog(this,"Wrong polinom format","Polinom
Error",JOptionPane.ERROR_MESSAGE);
        }
```

```
        }
```

O metoda care difera mult de celelalte metode da tratare a eventului este metoda PlotButton2MouseClicked.

```
PlotButton2.addMouseListener(new java.awt.event.MouseAdapter() {
        public void mouseClicked(java.awt.event.MouseEvent evt) {
           PlotButton2MouseClicked(evt);
        }
      });

        private void PlotButton2MouseClicked(java.awt.event.MouseEvent evt) {


            TxRes.setText(" ");

            String p1,p2 = new String();
            p1 = TxPol2.getText();
            //p2 = TxPol2.getText();
            int g1;
            int[] coef1;

            g1 = Integer.parseInt(TxGrad2.getText());
           // g2 = Integer.parseInt(TxGrad2.getText());

            try
            {
               coef1 = PolinomOperations.parseString(p1, g1);
               //coef2 = PolinomOperations.parseString(p1, g1);
               String chartTitle = "Object Movement Chart";
               String xAxisLabel = "X";
               String yAxisLabel = "Y";

               XYDataset dataset = createDataset(coef1);

               JFreeChart chart = ChartFactory.createXYLineChart(chartTitle,
               xAxisLabel, yAxisLabel, dataset);

               BarRenderer renderer = null;
               CategoryPlot plot = null;
               renderer = new BarRenderer();

               ChartFrame myFrame = new ChartFrame("Polinom chart",chart);
               myFrame.setLocation(200, 200);
```

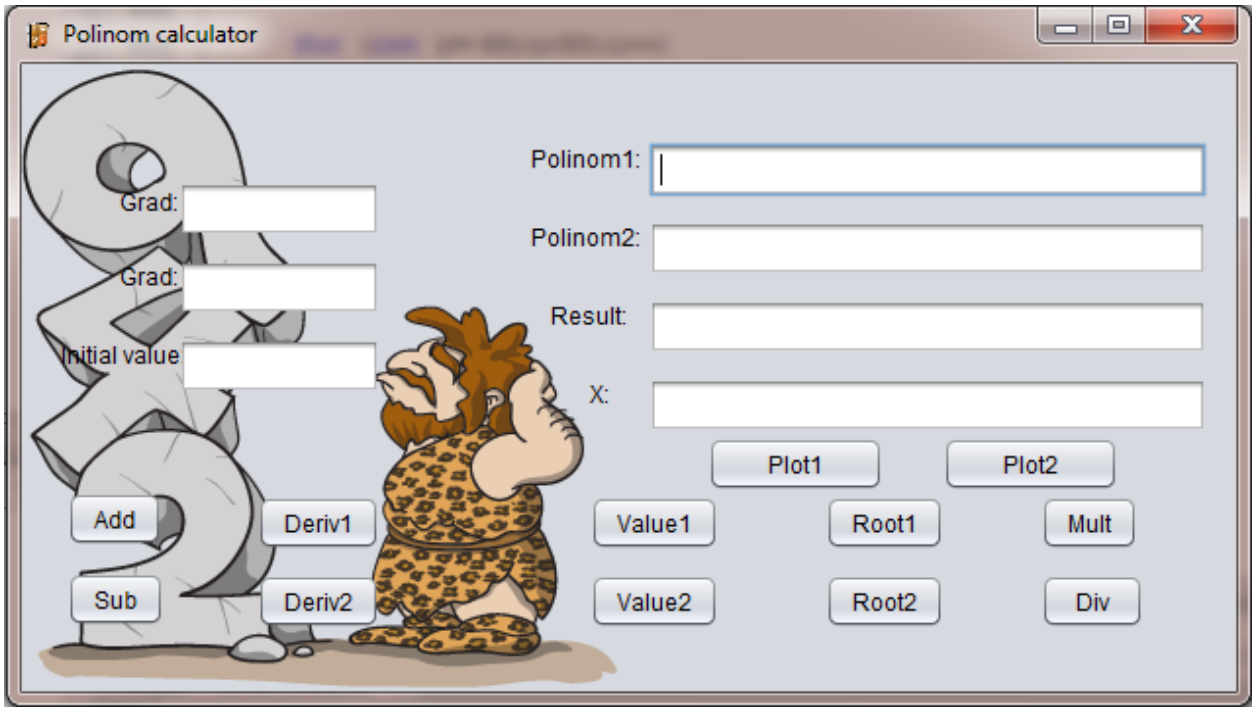
```
                myFrame.setSize(400, 400);
                myFrame.setVisible(true);
            }
            catch(Exception e)
            {
                JOptionPane.showMessageDialog(this,"Wrong polinom format","Polinom
Error",JOptionPane.ERROR_MESSAGE);
            }

        }

    private XYDataset createDataset(int[] a) {
        XYSeriesCollection dataset = new XYSeriesCollection();
        XYSeries series1 = new XYSeries("Polinom de grad " + (a.length-1));
       // XYSeries series2 = new XYSeries("Polinom 2");

        for (int i=-50;i<50;i++)
        {
            double y = PolinomOperations.findValue(a, i);
            series1.add(i, y);
        }

       /* for (int i=-50;i<50;i++)
        {
            double y = PolinomOperations.findValue(b, i);
            series2.add(i, y);
        }*/

        dataset.addSeries(series1);
       // dataset.addSeries(series2);

        return dataset;
    }
```

Metoda createDataset creeaza pe baza unui polinom un sir de puncte, adica calculeaza pentru polinomul respectiv valoarea lui in punctele de la -50 la 50 si din punctele (i, valoare in punctul i) creeaza un XYDataset.

In metoda Plot Button 2 Mouse Clicked folosind acest set de date, adica XYDataset, cu ajutorul metodei createXYLineChart(chartTitle,xAxisLabel, yAxisLabel, dataset); din clasa ChartFactory din biblioteca JFreeChart desenam un plot pe un ChartFrame nou numit Polinom chart.

Folosirea interfetei grafice:

La deschiderea programului apare urmatoarea fereastra:



Pentru folosirea corecta a aplicatiei urmariti instructiunile: Pentru fiecare polinom introduceti prima data gradul maxim al polinomului in casutele numite Grad1 si Grad2.

Dupa introducerea fiecarui grad, introduceti primul polinom in casuta Polinom1 si al doilea polinom in casuta Polinom2. Polinomul trebuie sa fi de forma x^2+3x+x^0, unde ^ reprezinta puterea iar 3x inseamna 3 * x. Coeficientul liber trebuie introdus cu ^0, altfel programul nu va lua in considerare coeficientul liber. Pentru adunarea celor doua polinoame folositi butonul Add iar pentru scadere Sub. Daca doriti sa inmultit cele doua polinoame, atunci dati click stanga pe butonul Mult. Pentru impartirea polinoamelor folositi butonul Div, avand in veder ca polinomul al doilea trebuie sa aiba un grad mai mic decat primul polinom.

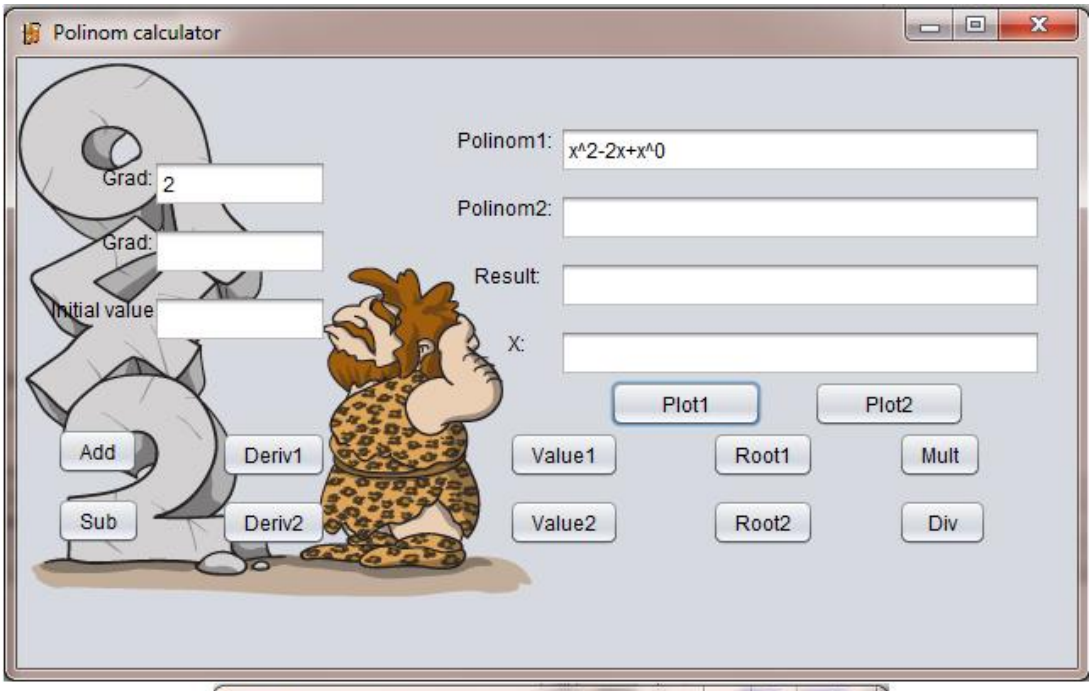
Daca doriti sa calculati valoarea primului polinom in punctul x, atunci introduceti numarul dorit in casuta cu numele X si dati click stanga pe butonul Value1. Daca doriti sa calculati valoarea polinomului 2 in punctul x, atunci introduceti numarul dorit in casuta cu numele X si dati click stanga pe butonul Value2.

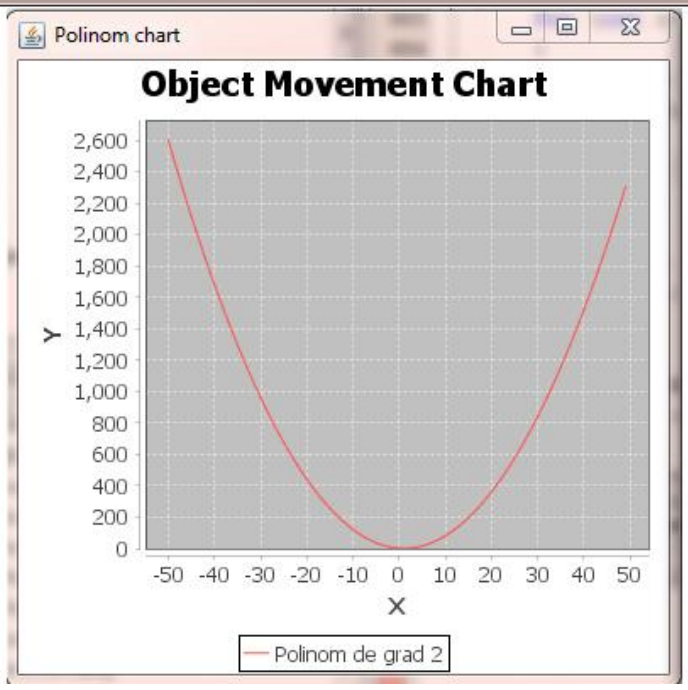
Pentru calcularea valorii proprii al primului polinom introduceti o valoare initiala (cat mai apropiata de redacina) in casuta numita Initial value, apoi dati click stanga pe Root1.

Pentru calcularea valorii proprii polinomului 2 introduceti o valoare initiala (cat mai apropiata de redacina) in casuta numita Initial value, apoi dati click stanga pe Root2.

Butonul Plot1 va reprezenta grafic polinomul 1 iar butonul Plot2 va reprezenta grafic pe polinomul 2.

Exemplu pentru plot:





4. Testare

Aceasta aplicatie a fost testata de mai multe ori. Testarea am facut prin introducerea manuala a datelor, verificand si cazurile particulare (de exemplu la gasirea radacinii am testat si pentru delta < 0 sau delta = 0, pentru cazul in care nu avem coeficient liber, sau in caurile in care lipsesc niste coeficienti) iar rezultatul testarii a fost intotdeauna conform realitatii, fara greseli.

Pentru o testare mai buna am facut aple si la alte persoane, care au incercat utilizarea aplicatiei cautand greseli, iar daca au aparut ceva greseli pe care pana atunci nu am luat in considerare, pe care pana atunci nu am gasit, am reformulat codul pentru corectarea acestor greseli. In prezent aplicatia merge fara greseli.

5. Rezultate

După testarea programului vedem că programul functionează perfect, verifică datele de intrare , afișează rezultatele pe interfața grafică, si efetuează comenzile primite de la utilizatorul programului.

6. Concluzii, ce s-a invatat din tema, dezvoltari ulterioare

Lucrand la acest proiect am invatat flosirea Swingului, al Event Handlerului, am facut recapitulare de mostenire si am incercat prctic conceptul de mostenire, de suprascriere si de interfata.

Pe langa programare am invatat is cateva teoreme matematice, de exemplu metoda lui Newton pentru calcularea radacinii a unui polinom si am invatat o metoda destul de simpla si eficienta pentru impartirea polinoamelor.

La partea de grafica am invata cum sa desenez un plot, si am studiat folosirea bibliotecii JFreeChart, cu ajutorul careia se poate desena orice fel de diagrama, deci e un instrument foarte bun pentru statistici, pentru bussines si management.

Acest proiect poate fi dezvoltat, adaugand operatii noi, de exemplu integrarea sau prin adaugarea ecuatiilor mai complicate, care contin logaritmi, expresii exponentiale etc.

Acest proiect a fost foarte util pentru recapitularea conceptelor de programare orientata pe biect, pentru o mai buna intelegere a mostenirii si a interfetei si pentru folosirea componentelor swingului si am inteles mai bine cum functioneaza Event Handlerul.

Pe langa toate aceste fapte am folosit foarte mult exceptiile, si am gasit exceptii necunoscute inca pentru mine.

7. Bibiliografie

Carte : Thinking in java – fourth edition Bruce Eckel President, MindView, Inc.

Internet :

https:// www. youtube . com / - turoriale

http:// stackoverflow . com /

http:// docs. oracle. com / javase / tutorial /

http:// rosettacode. org / wiki / Polynomial_long_division

http:// www. uml-diagrams . org / use – case – reference . html