

# **DOCUMENTATIE**

## **TEMA 1**

NUME STUDENT: GRASU LORENA-ELENA  
GRUPA: 30222

# CUPRINS

1.	Obiectivul temei.....	3
2.	Analiza problemei, modelare, scenarii, cazuri de utilizare .....	3
3.	Proiectare .....	3
4.	Implementare .....	3
5.	Rezultate .....	6
6.	Concluzii.....	6
7.	Bibliografie .....	6

## 1. Obiectivul temei

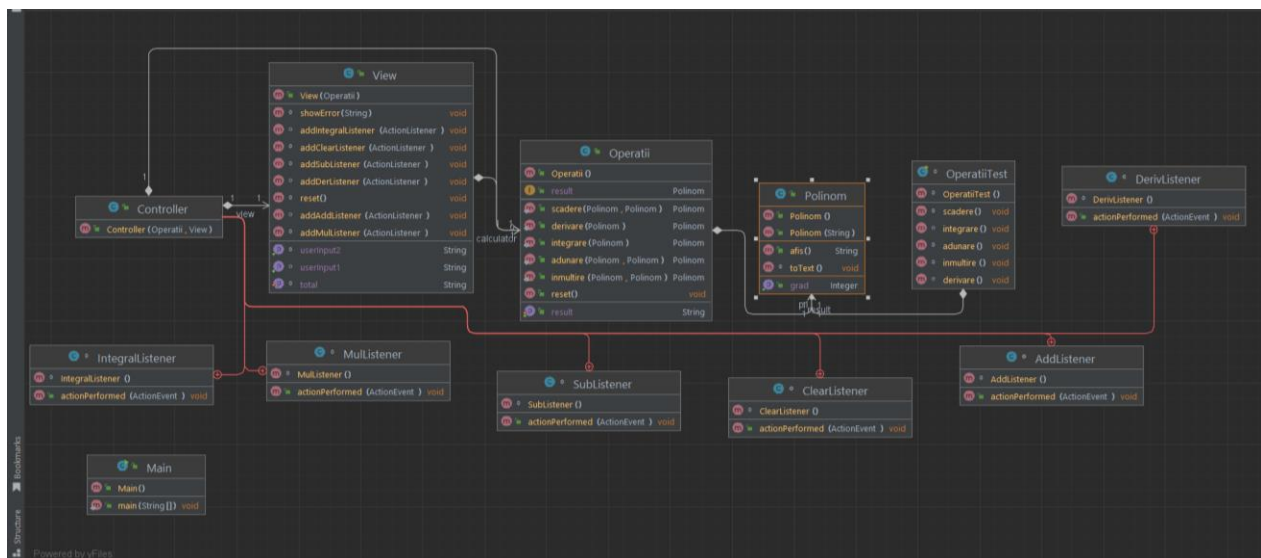
Obiectivul principal al acestei teme este realizarea unui calculator de polinoame functional, care poate realiza operatiile de: adunare, scadere, inmultire, integrare si derivare. Obiectivele secundare sunt dezvoltarea abilitatilor in lucrul cu interfata, testarea si organizarea cat mai eficienta a timpului de rezolvare a problemelor.

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

Problema calculatorului de polinoame in cazul proiectului meu a fost impartirea in monoame, nu am stiut daca sa fac clasa separata sau map, daca e mai usor sa impart cu token sau matcher, insa incercand acestea am ajuns la concluzia ca desi am scris destul de mult la impartirea cu expresie Regex, am incercat sa acopar destul de multe cazuri de eroare de scriere a expresiilor si am simplificat foarte mult functiile pentru operatii folosind map-ul.

## 3. Proiectare

Diagrama UML



## 4. Implementare

Clasa Polinom contine un String si un HashMap pentru salvarea monoamelor

```
11 public class Polinom {
12     public String s;
13     public HashMap<Integer, Double> p = new HashMap<Integer, Double>();
14 }
```

Clasa Operatii are implementarile operatiilor

```

12  public static Polinom adunare(Polinom p1, Polinom p2) {
13      Polinom result = new Polinom();
14      for(Integer i: p1.p.keySet()) {
15          result.p.put(i, p1.p.get(i));
16      }
17      for(Integer i: p2.p.keySet()) {
18          if(result.p.get(i) == null)
19          {
20              result.p.put(i, p2.p.get(i));
21          }
22          else {
23              result.p.put(i, result.p.get(i) + p2.p.get(i));
24          }
25      }
26      return result;
27  }

```

```

28  public static Polinom scadere(Polinom p1, Polinom p2) {
29      Polinom result = new Polinom();
30      for(Integer i: p1.p.keySet()) {
31          result.p.put(i, p1.p.get(i));
32      }
33      for(Integer i: p2.p.keySet()) {
34          if(result.p.get(i) == null)
35          {
36              result.p.put(i, -1 * p2.p.get(i));
37          }
38          else {
39              result.p.put(i, result.p.get(i) - p2.p.get(i));
40          }
41      }
42      return result;
43  }
44  }
45  public static Polinom inmultire(Polinom p1, Polinom p2) {
46      Polinom result = new Polinom();
47      for(Integer i: p1.p.keySet()) {
48          for(Integer j: p2.p.keySet()) {
49              if(result.p.get(i+j) == null) {
50                  result.p.put(i+j, result.p.get(i+j) + p1.p.get(i) * p2.p.get(j));
51              }
52              else {
53                  result.p.put(i+j, result.p.get(i+j) + p1.p.get(i) * p2.p.get(j));
54              }
55          }
56      }
57      return result;
58  }

```

```

59  public static Polinom derivare(Polinom p1) {
60      Polinom result = new Polinom();
61      for(Integer i: p1.p.keySet()) {
62          if(i != 0) {
63              result.p.put(i-1, p1.p.get(i)*i);
64          }
65          else result.p.put(0, p1.p.get(i)*i);
66      }
67      return result;
68  }
69  }
70  public static Polinom integrare(Polinom p1) {
71      Polinom result = new Polinom();
72      for(Integer i: p1.p.keySet()) {
73          result.p.put(i+1, p1.p.get(i) / (i+1));
74      }
75      return result;
76  }

```

Clasa View continue tot ce tine de partea vizuala, butoane, interfata, are ca si componenta si un obiect de tip operatii

```

5 usages
18 public class View extends JFrame{
11
2 usages
12 private JTextField userInput1 = new JTextField( columns: 30);
2 usages
13 private JTextField userInput2 = new JTextField( columns: 30);
5 usages
14 private JTextField totalTf = new JTextField( columns: 30);
2 usages
15 private JButton addBtn = new JButton( text: "+");
2 usages
16 private JButton subBtn = new JButton( text: "-");
2 usages
17 private JButton multBtn = new JButton( text: "*");
2 usages
18 private JButton integBtn = new JButton( text: "Integrate");
2 usages
19 private JButton derivBtn = new JButton( text: "Derivate");
2 usages
20 private JButton clearBtn=new JButton( text: "Clear");
21

```

Clasa Controller este folosit pentru ActionListener, continue un obiect de tip operatii si unul de tip view

```

19 view.addSubListener(new SubListener());
20 view.addClearListener(new ClearListener());
21 view.addDerivListener(new DerivListener());
22 view.addIntegralListener(new IntegralListener());
23 }
24
25 ////////////////////////////////////////////////// inner class MultiplyListener
26 /**
27  * When a multiplication is requested. 1. Get the user input number from the
28  * View. 2. Call the model to multiply by this number. 3. Get the result from
29  * the Model. 4. Tell the View to display the result. If there was an error,
30  * tell the View to display it.
31  */
32 1 usage
33 class MultiplyListener implements ActionListener {
34 of public void actionPerformed(ActionEvent e) {
35 String userInput1 = "";
36 String userInput2 = "";
37 try {
38 userInput1 = view.getUserInput1();
39 userInput2 = view.getUserInput2();
40 Polinom p1=new Polinom(userInput1);
41 Polinom p2=new Polinom(userInput2);
42 Polinom result=p1.multiplying(p1,p2);
43 view.setTotal(result.afis());
44 } catch (NumberFormatException nfx) {
45 view.showError( errorMessage: "Bad input");
46 }
47 }
48 } // end inner class MultiplyListener

```

Clasa Main

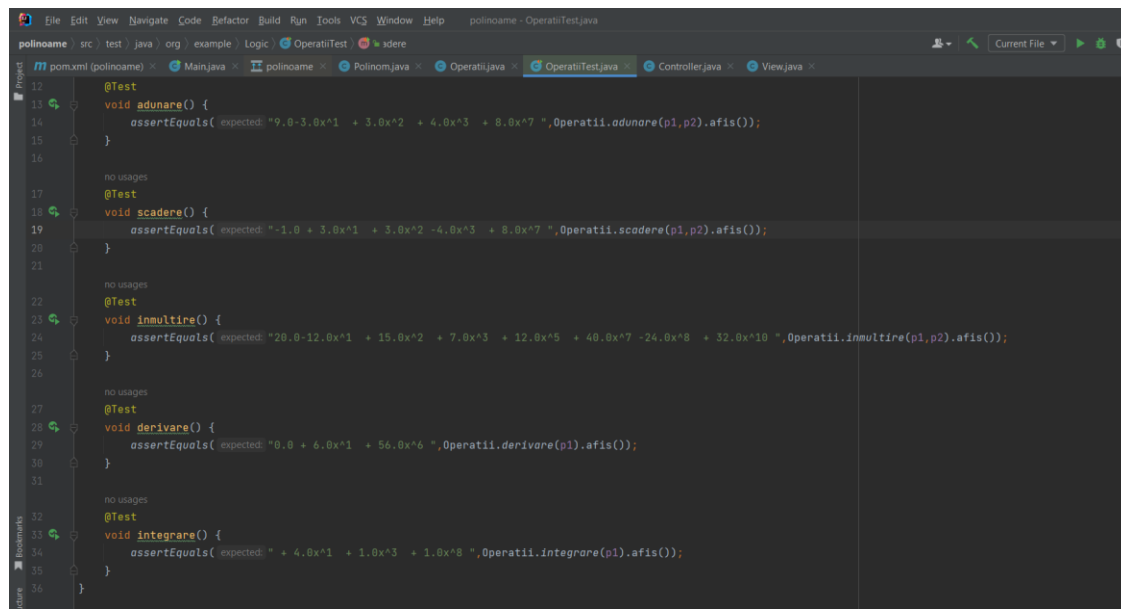
In ea unim MVC

```

no usages
7 public class Main {
no usages
8 public static void main(String[] args) {
9 Operatii o =new Operatii();
10 View view =new View(o);
11 Controller c= new Controller(o,view);
12
13 view.setVisible(true);
14 }
15 }

```

Clasa OperatiiTest – testeaza cele 5 operatii



```
12  @Test
13  void adunare() {
14      assertEquals( expected: "9.0+3.0x^1 + 3.0x^2 + 4.0x^3 + 8.0x^7 ",Operatii.adunare(p1,p2).afis());
15  }
16
17  no usages
18  @Test
19  void scadere() {
20      assertEquals( expected: "-1.0 + 3.0x^1 + 3.0x^2 -4.0x^3 + 8.0x^7 ",Operatii.scadere(p1,p2).afis());
21  }
22
23  no usages
24  @Test
25  void inmultire() {
26      assertEquals( expected: "20.0-12.0x^1 + 15.0x^2 + 7.0x^3 + 12.0x^5 + 40.0x^7 -24.0x^8 + 32.0x^10 ",Operatii.inmultire(p1,p2).afis());
27  }
28
29  no usages
30  @Test
31  void derivare() {
32      assertEquals( expected: "0.0 + 6.0x^1 + 56.0x^6 ",Operatii.derivare(p1).afis());
33  }
34
35  no usages
36  @Test
37  void integrare() {
38      assertEquals( expected: " + 4.0x^1 + 1.0x^3 + 1.0x^8 ",Operatii.integrare(p1).afis());
39  }
40
41  no usages
```

## 5. Rezultate

Testele de mai sus sunt realizate cu ajutorul unelei Junit, avand grija la formatul rezultatului pe care l- am scris

## 6. Concluzii

Calculatorul este unul functional, insa fara operatia de impartire, astfel incat aceasta ar putea fi una din imbunatatire. Un alt aspect care necesita reparatii este interafata care ar putea fi mai frumoasa/colorata

## 7. Bibliografie

1. Fisierle puse la dispozitie la laborator
2. [www.tutorialspoint.com](http://www.tutorialspoint.com)
3. [www.geeksforgeeks.com](http://www.geeksforgeeks.com)
4. Redit
5. [www.stackoverflow.com](http://www.stackoverflow.com)
6. [www.regexr.com](http://www.regexr.com)
7. [www.w3schools.com](http://www.w3schools.com)
- 8.