MINISTRY OF EDUCATION AND RESEARCH
TECHNICAL UNIVERSITY
OF CLUJ-NAPOCA, ROMANIA

TEHNICI DE PROGRAMARE

TEMA_1

CALCULATOR DE POLINOAME

Student: Tóth Szilveszter Zsolt

Gr.: 30224

1. Obiectivul temei

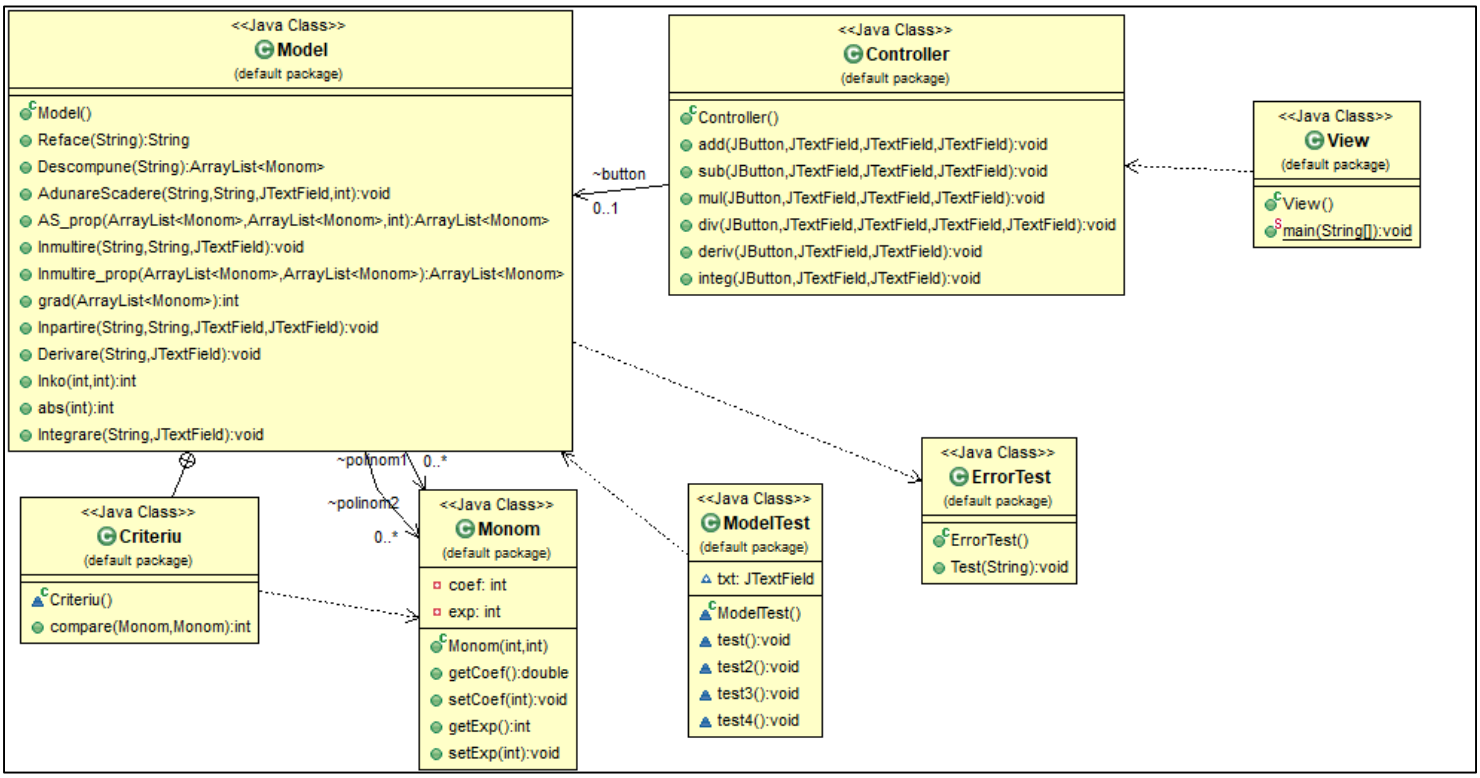
La aceasta tema am avut de creat si implementat in mod corect un calculator pentru polinoame. In aceasta am creat cele sase operatii fundamentale: adunare, scadere, inmultire, inpartire, derivare, integrare.

2. Analiza problemei

Avand in vedere ca sunt cazuri in care aceste operatii pot deveni destul de complexe (de exemplu la divizare), avem nevoie de o solutie cat mai compacta in care sa fie tratate toate cazurile speciale (divizare cu 0, input incorect etc). Vom folosi arhitectura de tip MVC (Model-View-Controller) pentru a aborda o solutie cat mai object oriented si pentru a da programului o mai mare lizibilitate. In mod general vorbind, in Model vom avea operatiile logice, tot ce tine de prelucrarea informatiilor, in View se vor gasi doar elementele care tin de interfata grafica (butoane, text field-uri etc), iar in Controller va fi partea care leaga cele doua parti mentionate mai devreme (in cazul nostru aici se vor gasi ActionListener-urile). Avand in vedere ca incercam sa adaptam un stil cat mai oop, vom evita folosirea in exces a for-urilor si array-urilor simple. In loc, vom folosi colectii, structuri foreach si vom evita scrierea de clase imense (cu putinta sa fie organizate cat mai compact si logic pe cat posibil).

3. Proiectare si Implementare

Cum am mentionat si la punctul anterior, vom aborda in mod oop. Datorita acestui fapt, in loc sa scriem toate operatiile intr-o singura clasa am descompus in 7 clase diferite (cum se poate vedea si pe diagram UML de mai jos), dintre care doar ModelTest nu tine de implementarea propriu-zisa a problemei(in aceasta clasa sunt scrise cateva teste Junit pentru a verifica corectitudinea metodelor din clasa Model).



Clasa Monom este probabil cea mai simpla clasa. In aceasta este definit doar o structura de date in care se afla doi intregi: un coeficient si un exponent. Cu ajutorul acestuia ne vom defini un sir de monoame care in final va fi polinomul nostru. In mod normal, avem create getter-e si setter-e.

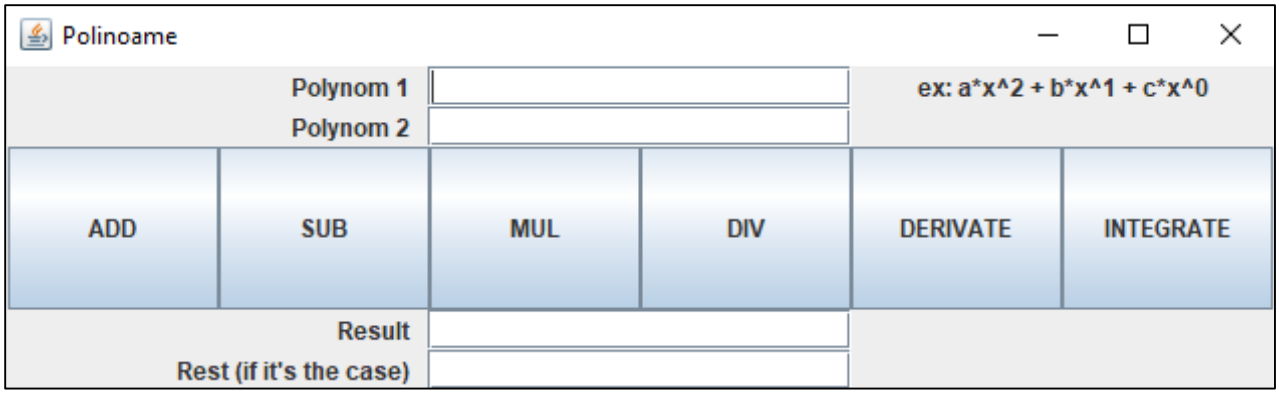
```
public class Monom {
	private int coef;
	private int exp;
	public Monom(int coef, int exp) {
		super();
		this.coef = coef;
		this.exp = exp;
	}
	public double getCoef() {
		return coef;
	}
	public void setCoef(int coef) {
		this.coef = coef;
	}
	public int getExp() {
		return exp;
	}
	public void setExp(int exp) {
		this.exp = exp;
	}

}
```

Urmatoare clasa relative simpla este cea de ErrorTest, in care sunt implementate rezolvarea exceptiilor/erorilor care pot aparea pe parcursul rularii (divizare cu zero, input lasat gol, introducerea unor caractere care nu ar trebui introduse (litere, alte semne)). Daca programul depisteaza o astfel de exceptie, va afisa un mesaj pe consola care va spune utilizatorului ce a gresit mai apoi va inchide interfata.

```
public class ErrorTest {
	public void Test(String p1) {
		if (p1.length()<5)
		{
			System.out.println("Possible trying to work w/ 0 or w/ wrong input!");
			System.exit(0);
		}
		else if (p1.length() == 0) {
			System.out.println("Empty Input!");
			System.exit(0);
		} else {
			String verif = "0123456789x^*+-";
			for (int i = 0; i < p1.length(); i++) {
				if (verif.indexOf(p1.charAt(i)) == -1) {
					System.out.println("Wrong Characters Used!");
					System.exit(0);
				}
```

```
                }
            }
        }
}
```

Daca rulam programul, ne va afisa o interfata grafica, in care vor fi 4 textfield-uri, doua in care vom introduce polinoamele, si doua in care vor aparea rezultatele pe urma operatiilor, si 6 butoane, fiecare responsabil pentru o operatie in parte.



Clasa responsabila pentru aceasta interfata este View:

```
import java.awt.BorderLayout;
import java.awt.GridLayout;

import javax.swing.*;

public class View {

    public static void main(String[] args) {
        JFrame frame = new JFrame("Polinoame");

        frame.setSize(650, 200);
        frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
        frame.setLayout(new BorderLayout());
        JPanel panel = new JPanel();
        JPanel panel2 = new JPanel();
        JPanel panel3 = new JPanel();

        panel.setLayout(new GridLayout(2, 3));
        JLabel lbl = new JLabel("Polynom 1   ", JLabel.RIGHT);
        panel.add(lbl);
        JTextField txt = new JTextField("");
        txt.setSize(10, 100);
        panel.add(txt);

        JLabel lblaux = new JLabel("ex: a*x^2 + b*x^1 + c*x^0", JLabel.CENTER);
        panel.add(lblaux);
        JLabel lbl2 = new JLabel("Polynom 2   ", JLabel.RIGHT);
        panel.add(lbl2);
```

```
                JTextField txt2 = new JTextField("");
                txt2.setSize(10, 100);
                panel.add(txt2);

                panel2.setLayout(new GridLayout(1, 6));
                JButton butAdd = new JButton("ADD");
                JButton butSub = new JButton("SUB");
                JButton butMul = new JButton("MUL");
                JButton butDiv = new JButton("DIV");
                JButton butDeriv = new JButton("DERIVATE");
                JButton butInteg = new JButton("INTEGRATE");

                panel3.setLayout(new GridLayout(2, 3));
                JLabel lbl3 = new JLabel("Result   ", JLabel.RIGHT);
                panel3.add(lbl3);
                JTextField txt3 = new JTextField("");
                txt.setSize(10, 100);
                panel3.add(txt3);

                JLabel lblspa = new JLabel("");
                panel3.add(lblspa);
                JLabel lbl4 = new JLabel("Rest (if it's the case)   ", JLabel.RIGHT);
                panel3.add(lbl4);
                JTextField txt4 = new JTextField("");
                txt4.setSize(10, 100);
                panel3.add(txt4);

                Controller cont = new Controller();
                cont.add(butAdd, txt, txt2, txt3);
                cont.sub(butSub, txt, txt2, txt3);
                cont.mul(butMul, txt, txt2, txt3);
                cont.div(butDiv, txt, txt2, txt3, txt4);
                cont.deriv(butDeriv, txt, txt3);
                cont.integ(butInteg, txt, txt3);

                panel2.add(butAdd);
                panel2.add(butSub);
                panel2.add(butMul);
                panel2.add(butDiv);
                panel2.add(butDeriv);
                panel2.add(butInteg);

                frame.add(panel, BorderLayout.NORTH);
                frame.add(panel2, BorderLayout.CENTER);
                frame.add(panel3, BorderLayout.SOUTH);
                frame.setVisible(true);
        }
}
```

In primele doua casute se vor introduce polinoamele in formatul specificat in parte dreapta sus, dupa care se apasa oricare buton, pentru a efectua operatia dorita. Odata apasat pe buton se va apela clasa Controller, in care se vor salva stringurile de input, si se va apela metoda ceruta de catre utilizator:

```
import javax.swing.JButton;
import javax.swing.JTextField;

public class Controller {

    Model button = new Model();

    public void add(JButton butAdd, JTextField txt, JTextField txt2, JTextField txt3) {
        butAdd.addActionListener(e -> {
            String p1 = txt.getText();
            String p2 = txt2.getText();
            button.AdunareScadere(p1, p2, txt3, 0);
        });
    }

    public void sub(JButton butSub, JTextField txt, JTextField txt2, JTextField txt3) {
        butSub.addActionListener(e -> {
            String p1 = txt.getText();
            String p2 = txt2.getText();
            button.AdunareScadere(p1, p2, txt3, 1);
        });
    }

    public void mul(JButton butMul, JTextField txt, JTextField txt2, JTextField txt3) {
        butMul.addActionListener(e -> {
            String p1 = txt.getText();
            String p2 = txt2.getText();
            button.Inmultire(p1, p2, txt3);
        });
    }

    public void div(JButton butDiv, JTextField txt, JTextField txt2, JTextField txt3, JTextField txt4) {
        butDiv.addActionListener(e -> {
            String p1 = txt.getText();
            String p2 = txt2.getText();
            button.Inpartire(p1, p2, txt3, txt4);
        });
    }

    public void deriv(JButton butDeriv, JTextField txt, JTextField txt3) {
        butDeriv.addActionListener(e -> {
            String p1 = txt.getText();
            button.Derivare(p1, txt3);
        });
    }
```

```
	public void integ(JButton butInteg, JTextField txt, JTextField txt3) {
		butInteg.addActionListener(e -> {
			String p1 = txt.getText();
			button.Integrare(p1, txt3);
		});
	}

}
```

Cand se apeleaza clasa Model, indiferent de care operatie este vorba, sunt efectuate doi pasi: se rescrie stringul de la input asa fel incat in loc de "-" se pune "+-". Aceasta este un trick prin care ne face treaba mai usoare cand "spargem" stringurile, care totodata este pasul doi, in care luam stringul rescris, il spargem in functie de "+", mai apoi in funcite de "*x^". Dupa acestea, gasim coeficientii si exponentii pe care le vom salva intr-un monom, mai apoi adaugam monoamele intr-un ArrayList, care va constitui polinomul cu care vom lucra in continuare. Codul pentru acesti pasi (apare un pic urat din cauza versiunii de Microsoft Office):

```
ArrayList<Monom> polinom1 = new ArrayList<>();
ArrayList<Monom> polinom2 = new ArrayList<>();

	public String Reface(String p1) {
		for (int i = 1; i < p1.length(); i++) {
			if (p1.charAt(i) == '-' && i != 0) {
				p1 = p1.substring(0, i) + "+" + p1.substring(i);
				i++;
			}
		}
		return p1;
	}

	public ArrayList<Monom> Descompune(String p1) {
		ArrayList<Monom> polaux = new ArrayList<>();
		ErrorTest err=new ErrorTest();
		err.Test(p1);
		String[] arr = p1.split("\\+", -2);
		int c = 0;
		int e = 0;
		int db = 0;
		for (String a : arr) {
			{
				String[] arr1 = a.split("\\^", -3);
				db = 0;
				c = 0;
				e = 0;
				for (String a1 : arr1) {
					if (db == 0) {
						if (a1.charAt(0) == '-') {
							for (int i = 1; i < a1.length() - 2; i++) {
								c *= 10;
								c = c +
Integer.parseInt(String.valueOf(a1.charAt(i)));
							}
```

```
                                                c = -c;
                                        } else {
                                                for (int i = 0; i < a1.length() - 2; i++) {
                                                        c *= 10;
                                                        c = c +
Integer.parseInt(String.valueOf(a1.charAt(i)));
                                                }
                                        }

                                } else {
                                        for (int i = 0; i < a1.length(); i++) {
                                                e *= 10;
                                                e = e +
Integer.parseInt(String.valueOf(a1.charAt(i)));

                                        }
                                }
                                db++;
                        }
                        Monom m = new Monom(c, e);
                        polaux.add(m);
                }
        }
        return polaux;
    }
```

In urmatoarele vor fi descrise codurile pentru fiecare operatie in parte. Adunarea si Scaderea au fost implementate in aceeasi functie, fiindca se aseamana foarte tare. In antetul functiei a fost introdus inca un parametru intreg care va fi 0 pentru adunare, si 1 pentru scadere. Metodele AdunareScadere si Inmultire am descris in doua parti pentru ca la divizare vor trebui refolosite, si acest lucru trebuia sa fie independent de metodele in care se refac si se descompun stringurile initial.

3.1 Adunare si Scadere

```
public void AdunareScadere(String p1, String p2, JTextField txt3, int o) {
        p1 = Reface(p1);
        p2 = Reface(p2);
        polinom1 = Descompune(p1);
        polinom2 = Descompune(p2);
        ArrayList<Monom> rez = new ArrayList<>();
        rez=AS_prop(polinom1, polinom2,o);
        String fin = "";
        for (Monom i : rez)
                if (i.getCoef() != 0) {
                        if (i.getCoef() > 0)
                                fin += "+";
                        fin += Integer.toString((int) i.getCoef()) + "*x^" +
Integer.toString((int) i.getExp());
                }
        txt3.setText(fin);
    }
```

```
public ArrayList<Monom> AS_prop(ArrayList<Monom> p1,ArrayList<Monom> p2, int o)
	{
		ArrayList<Monom> rez = new ArrayList<>();
		int c = 0, e = 0;
		for (Monom i : p1) {
			for (Monom j : p2) {
				if (i.getExp() == j.getExp()) {
					if (o == 0)
						c = (int) (i.getCoef() + j.getCoef());
					else
						c = (int) (i.getCoef() - j.getCoef());
					e = i.getExp();
					i.setExp(-1);
					j.setExp(-1);
					Monom m = new Monom(c, e);
					rez.add(m);
				}
			}
		}
		for (Monom i : p1)
			if (i.getExp() >= 0)
				rez.add(i);
		for (Monom i : p2)
			if (i.getExp() >= 0) {
				if (o == 1) {
					int aux = (int) i.getCoef();
					aux = -aux;
					i.setCoef(aux);
				}
				rez.add(i);
			}
		//sortarea elementelor in functie de exponent
		Collections.sort(rez, new Criteriu());
		return rez;
```

3.2 Inmultire

```
public void Inmultire(String p1, String p2, JTextField txt3) {
		p1 = Reface(p1);
		p2 = Reface(p2);
		polinom1 = Descompune(p1);
		polinom2 = Descompune(p2);
		ArrayList<Monom> rez = new ArrayList<>();
		rez=Inmultire_prop(polinom1, polinom2);
		String fin = "";
		for (Monom i : rez)
			if (i.getCoef() != 0) {
				if (i.getCoef() > 0)
					fin += "+";
				fin += Integer.toString((int) i.getCoef()) + "*x^" +
Integer.toString((int) i.getExp());
			}
		txt3.setText(fin);
```

```
	}

	public ArrayList<Monom> Inmultire_prop(ArrayList<Monom> p1,ArrayList<Monom> p2)
	{
		ArrayList<Monom> rez = new ArrayList<>();
		int c = 0, e = 0;
		for (Monom i : p1) {
			for (Monom j : p2) {
				c = (int) (i.getCoef() * j.getCoef());
				e = (int) (i.getExp() + j.getExp());
				Monom m = new Monom(c, e);
				rez.add(m);
			}
		}
		int ii = 0, jj = 0;
		for (Monom i : rez) {
			jj = 0;
			for (Monom j : rez) {
				if (i.getExp() == j.getExp() && ii != jj) {
					int m = (int) (i.getCoef() + j.getCoef());
					i.setCoef(m);
					j.setCoef(0);
				}
				jj++;
			}
			ii++;
		}
		Collections.sort(rez, new Criteriu());
		return rez;
	}
```

3.3 Inpartire

```
public void Inpartire(String p1, String p2, JTextField txt3, JTextField txt4) {
		p1=Reface(p1);
		p2=Reface(p2);
		polinom1=Descompune(p1);
		polinom2=Descompune(p2);
		ArrayList<Monom> cat = new ArrayList<>();
		ArrayList<Monom> cat2 = new ArrayList<>();
		ArrayList<Monom> rez = new ArrayList<>();
		while(grad(polinom1)>=grad(polinom2))
		{
			Monom impartit=polinom1.get(0);
			Monom impartitor=polinom2.get(0);
			Monom aux=new
Monom((int)impartit.getCoef()/(int)impartitor.getCoef(),impartit.getExp()-
impartitor.getExp());
			cat.add(aux);
			if (cat2.size()!=0)
				cat2.remove(0);
			cat2.add(aux);
			rez=Inmultire_prop(cat2,polinom2);
```

```
                        polinom1=AS_prop(polinom1,rez,1);
                        if (impartit.getCoef()==0)
                                        polinom1.remove(0);

                }
                String fin="";
                String fin1="";
                for (Monom i : cat)
                        if (i.getCoef() != 0) {
                                if (i.getCoef() > 0)
                                        fin += "+";
                                fin += Integer.toString((int) i.getCoef()) + "*x^" +
Integer.toString((int) i.getExp());
                        }
                for (Monom i : polinom1)
                        if (i.getCoef() != 0) {
                                if (i.getCoef() > 0)
                                        fin1 += "+";
                                fin1 += Integer.toString((int) i.getCoef()) + "*x^" +
Integer.toString((int) i.getExp());
                        }
                txt3.setText(fin);
                txt4.setText(fin1);
        }
```

3.4 Derivare

```
public void Derivare(String p1, JTextField txt3) {
                p1 = Reface(p1);
                polinom1 = Descompune(p1);
                for (Monom i : polinom1) {
                        if (i.getExp() == 0)
                                i.setCoef(0);
                        int c = 0, e = 0;
                        c = (int) i.getCoef();
                        e = (int) i.getExp();
                        i.setCoef(c * e);
                        e--;
                        i.setExp(e);

                }
                String fin = "";
                for (Monom i : polinom1)
                        if (i.getCoef() != 0) {
                                if (i.getCoef() > 0)
                                        fin += "+";
                                fin += Integer.toString((int) i.getCoef()) + "*x^" +
Integer.toString((int) i.getExp());
                        }
                txt3.setText(fin);
        }
```

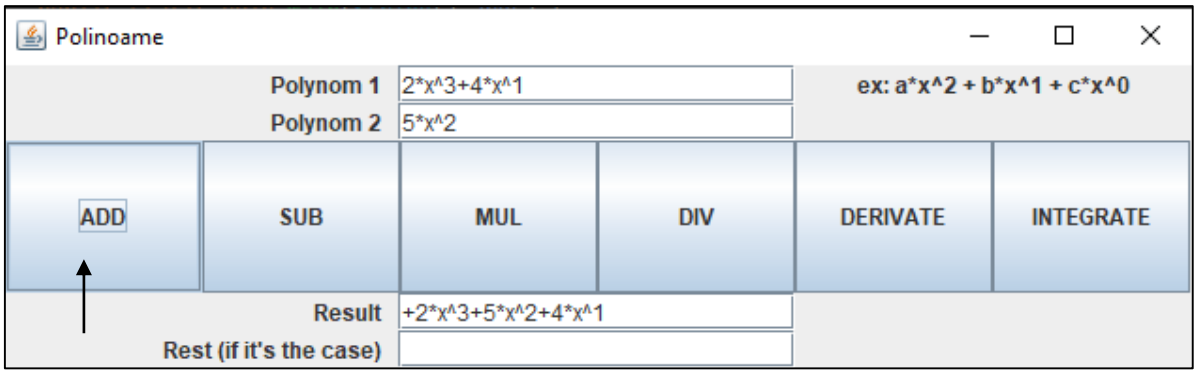
3.5 Integrare

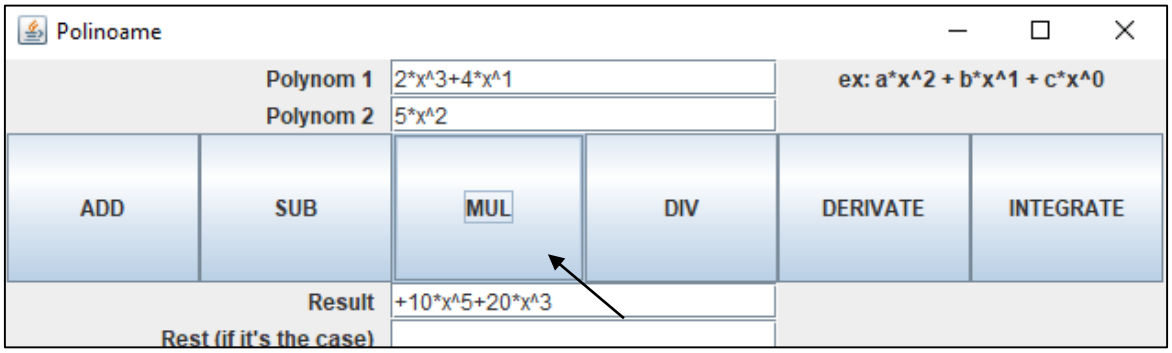
```
public void Integrare(String p1, JTextField txt3) {
		p1 = Reface(p1);
		String fin = "";
		polinom1 = Descompune(p1);
		for (Monom i : polinom1) {
			int c = 0, e = 0;
			c = (int) i.getCoef();

			e = (int) i.getExp();
			e++;
			i.setExp(e);
			if (i.getCoef() > 0)
				fin += "+";
			int aux = lnko(abs(c), abs(e));
			e /= aux;
			c /= aux;
			if (e > 1)
				fin += Integer.toString((int) c) + "/" + Integer.toString((int)
e) + "*x^"
					+ Integer.toString((int) i.getExp());
			else
				fin += Integer.toString((int) c) + "*x^" + Integer.toString((int)
i.getExp());
		}
		txt3.setText(fin);
}
```

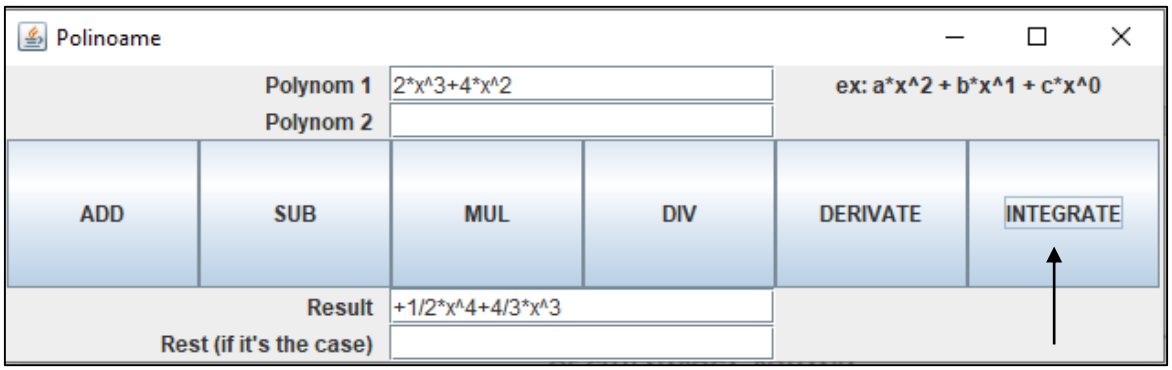
Mai sunt cateva metode auxuliare, de ex lnko (gasirea celui mai mare divisor comun) pentru a face mai frumoasa outputul la integrare, abs (returneaza valoarea absoluta a unui numar), grad (care returneaza gradul unui polinom).

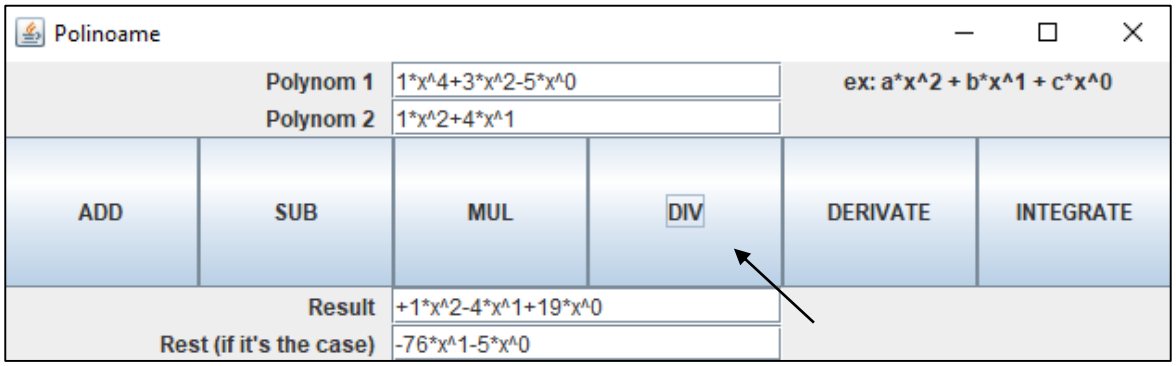
4. Rezultate

La rulare vor aparea rezultate de genul:



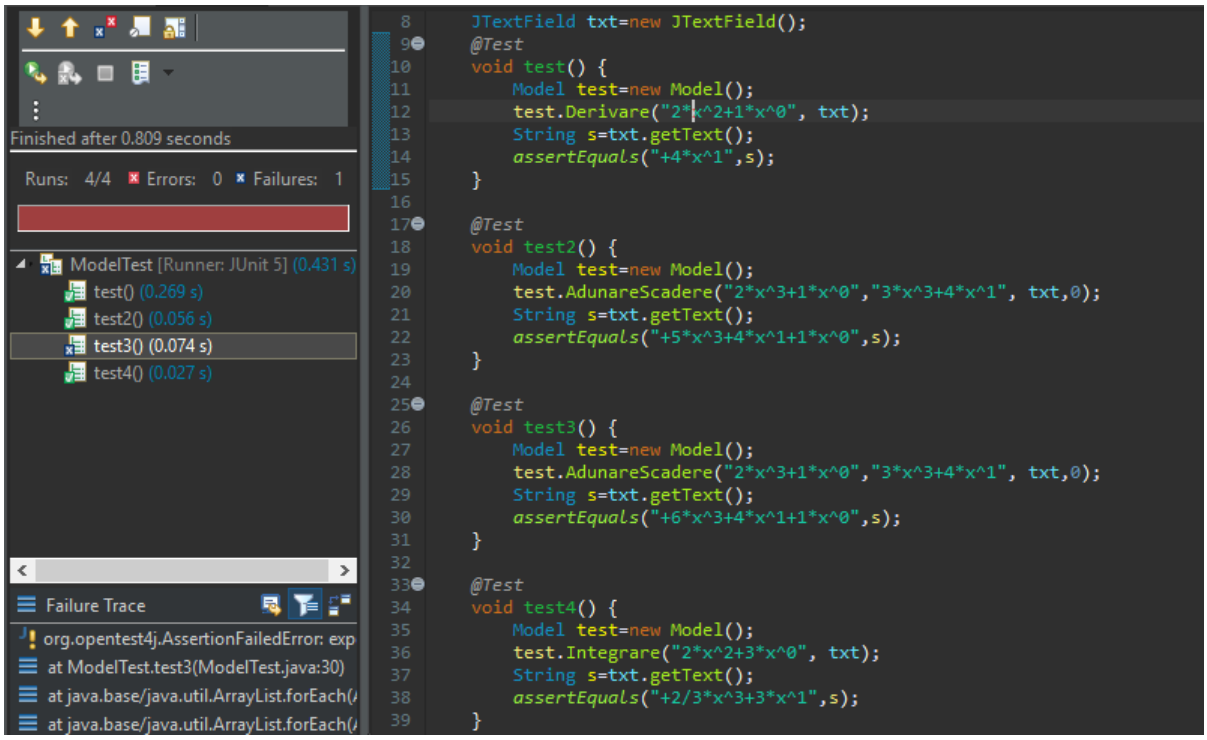






$$
\begin{array}{r}
x^2-4x+19 \\
x^2+4x\,\overline{)\,x^4+\quad 3x^2-\quad 5} \\
\underline{x^4+4x^3} \\
-4x^3+3x^2 \\
\underline{-4x^3-16x^2} \\
19x^2\quad -5 \\
\underline{19x^2+76x} \\
-76x-5
\end{array}
$$

Si cateva teste din JUnit:



5. Concluzii

Pe parcursul temei am invatat multe chestii noi, am reusit sa-mi aprofundez cunostintele in domeniul oop si java care imi vor prinde bine la temele ce urmeaza. Daca este sa vorbim despre dezvoltarile care se pot face, primul lucru ar fi sa fie inbunatatit metoda de input (adica in loc de 1*x^2+2*x^1+3*x^0 sa fie scris doar x^2+2x+3), si eventual atunci cand primeste o eroare de input, mesajul sa fie afisat intr-o fereastra pop-up in loc de consola (aceasta ar face posibil ca dupa inchiderea ferestrei pop-up sa se introduca din nou datele fara a reporni si a relansa codul sursa).

6. Bibliografie

https://www.youtube.com/watch?v=ai4ld6Kd9mc

https://www.youtube.com/watch?v=I8XXfgF9GSc

Plug in-ul folosit pentru UML:

http://www.objectaid.com/update/current