Cristian Flaviu-Remus
Group 30423

Homework1

1)Objectives

The main purpose of this program is to implement the following operation on polynomials:

- Addition
- Subtraction
- Multiplication
- Division
- Integration
- Derivation

Secondary Objectives in order to implement the previous operation:

- Store Monomials
- Store Polynomials
- Order Polynomials in order of the degree
- Delete Monomials with coefficient 0
- Get the Monomial with the biggest degree
- Multiply a Polynomial with a Monomial
- Etc.

Specification: polynomials will have only one variable and integer coefficients.

Every method will be described in detail later on.

2**) Problem analysis, scenarios and use cases:**

The program expects interaction with a user in order to introduce data and specify a flow of instructions. The user have to introduce in a specified text field the value of each polynomial and right after that he must press the save button in order to save the data introduced in each text field.

The polynomial have to respect a certain format, each monomial should have the form D1X^D2 where D1 represents the coefficient of the monomial (positive or negative) and D2 represents the degree of the monomial (only positive). If the format is not corresponding with the one specified before the program will show an error message instead of the value of the desired polynomial.

After saving the polynomial the user can choose what operation should be performed. The result of the operation will be displayed on the bottom of the page right after the message”Result : ”.

1) Use case “Addition”(description):
    a. The user opens the program and he has to introduce the first and the second polynomial and save them by pressing the button SaveFirstPolynomial respectively SaveSecondPolynomial .System will check if the input is introduced correct. User has to select the “+” button and the result will be printed on the bottom of the page.
2) Use case “subtraction”(description):
    a. The user opens the program and he has to introduce the first and the second polynomial and save them by pressing the button SaveFirstPolynomial respectively SaveSecondPolynomial .System will check if the input is introduced correct. User has to select the “-” button and the result will be printed on the bottom of the page.
3) Use case “multiplication”(description):
    a. The user opens the program and he has to introduce the first and the second polynomial and save them by pressing the button SaveFirstPolynomial respectively SaveSecondPolynomial .System will check if the input is introduced correct. User has to select the “*” button and the result will be printed on the bottom of the page.
4) Use case “division”(description):
    a. The user opens the program and he has to introduce the first and the second polynomial and save them by pressing the button SaveFirstPolynomial respectively SaveSecondPolynomial .System will check if the input is introduced correct. User has to select the “/” button and the result will be printed on the bottom of the page. If the denominator is 0 an error message will be displayed .

5) Use case “integration”(description):
    a. The user opens the program and he has to introduce the first and the second polynomial and save them by pressing the button SaveFirstPolynomial respectively SaveSecondPolynomial .System will check if the input is introduced correct. User has to select the “I” button and the result will be printed on the bottom of the page.

6) Use case “derivation”(description):
    a. The user opens the program and he has to introduce the first and the second polynomial and save them by pressing the button SaveFirstPolynomial respectively SaveSecondPolynomial .System will check if the input is introduced correct. User has to select the “D” button and the result will be printed on the bottom of the page.

Alternative scenarios:

There are two scenarios in which the system does not produce an output:

1)The user does not respect the specified format for introducing the polynomials.
In this case, the system will prompt the user with a warning message (“Invalid input!”).

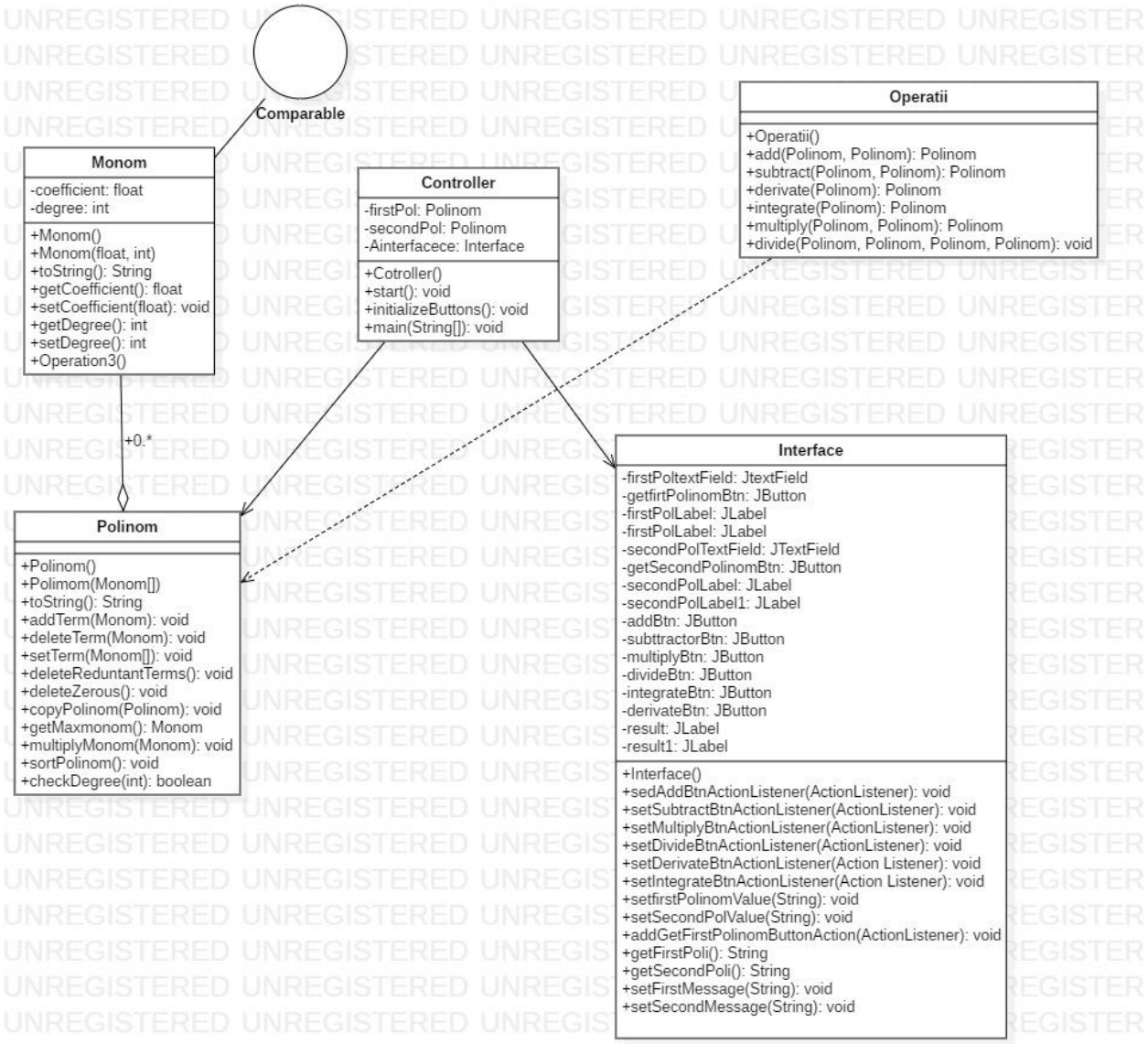
2)The user tries a division where secondPolynomial is 0 . In this case, the system will prompt a

Warning message “Division by 0”.

3) Designing the project (design decision, UML diagrams, classes, relation between classes, packages, GUI)

The project is composed by 4 packages: structures, operations , assInterface and Exception. The package structures contains the classes Monom and Polinom which are used for storing the data. The package operations contains one class Operatii which has 6 methods , one for each operation (Addition, Subtraction …) .The package assInterface is composed from 2 classes Interface and Controller. Interface class contains all the components for the GUI and the Controller class is used for specifying what does every button from Interface, it also contains the main program which include a instance of Interface. The package Exception has only one class DivideByZeroException which extends

RunTimeException.



4)Implementation

Package structures

1)Monom is a class used to store the monomials. The coefficient of the monomial will be stored in a variable of type float (coefficient) and the degree in a variable of type int (degree).I choose to store the coefficients in a variable of type float even if the problem says that the polynomial will have only integer coefficients because from dividing two polynomials may result some coefficients which are not integer.

It implements the interface comparable in order to be able to sort the Polynomials which are

composed of Monomials and override the method toString() in order to have a nice print of the Monomials.

2)Polinom is the class used to store the polynomials. It stores the polynomial as a list of type ArrayList of Monoms named term.It has the following methods:

a) deleteRedundantTerms

```
public void deteleteReduntantTerms() {

		ArrayList<Monom> elementToDelete = new ArrayList<>();
		for (Monom i : term) {
			for (Monom j : term) {
				if (i.getDegree() == j.getDegree() && i != j) {

					i.setCoefficient(i.getCoefficient() + j.getCoefficient());

					j.setCoefficient(0f);
					j.setDegree(0);
					elementToDelete.add(j);
				}
			}
		}

		term.removeAll(elementToDelete);

	}
```

If a polynomial has 2 or more monomials of the same degree this method will reduce them to one monomial with the coefficient being the sum of the coefficients with the same degree.

b) deleteZerous

```
public void deleteZerous()
	{
		List<Monom> deleleteList=new ArrayList<>();
		for(Monom i:this.getTerm())
		{
			if(i.getCoefficient()==0)
			{
				deleleteList.add(i);
			}
		}
		this.getTerm().removeAll(deleleteList);
	}
```

It deletes every Monom which have as coefficient zero by iterating trough every element of the ArrayList and checking if it has zero as coefficient or not. Every element with coefficient zero is added to a new list where all the elements which will be deleted are stored temporally .

c) copyPolinom()

```
public void copyPolinom(Polinom aPolinom) {
		this.getTerm().clear();
		for (Monom aMonom : aPolinom.getTerm()) {
			this.addTerm(new Monom(aMonom.getCoefficient(), aMonom.getDegree()));
		}

	}
```

Copy a polynomial into another polynomial

d) getMaxMonom

```
public Monom getMaxmonom() {
		Monom maxMonom = new Monom();
		for (Monom aMonom : this.getTerm()) {
			if (aMonom.getDegree() > maxMonom.getDegree())
				maxMonom = new Monom(aMonom.getCoefficient(), aMonom.getDegree());

		}

		return maxMonom;
	}
```

Iterates though every elements of the Array list and store the monomial with the biggest degree and coefficient different from zero. This method will return that term.

e) multiplyMonom

```
public void  multiplyMonom(Monom aMonom)
	{

		for(Monom bMonom:this.getTerm())
		{
			bMonom.setCoefficient(bMonom.getCoefficient()*aMonom.getCoefficient());
			bMonom.setDegree(aMonom.getDegree()+bMonom.getDegree());
		}
	}
```

This method will multiply every element of a polynomial with a specified Monomial by iterating though that polynomial and multiplying the coefficients and adding the degree of the specified Monomial with every element from the ArrayList.

f) sortPolinom

```
public void sortPolinom() {
		Collections.sort(term);
	}
```

This method will sort the elements of the polynomial in increasing order of the degree

g) check degree

```
public boolean checkDegree(int degree) {

		for (Monom i : term) {
			if (i.getDegree() == degree)
				return true;
		}

		return false;
```

```
}
```

This method check if a specified degree exist in a certain Monomial

h) toString

```
public String toString() {

		String string = "";
		for (Monom a : term) {
			if (a.getCoefficient() > 0)
				string = string + "+ ";

			string = string + a.toString();

		}

		if (string.length() == 0)
			return new String("0X^0");
		if (term.get(0).getCoefficient() < 0)
			return string;
		return string.substring(1, string.length() - 1);
	}
```

This method override the method toString in order to obtain a nice print for the Polynomial

i) addTerm

```
public void addTerm(Monom aMonom) {
		term.add(new Monom((aMonom.getCoefficient()), aMonom.getDegree()));
	}
```

This method add a monomial to the polynomial .First it will check if a monomial with the same degree exist, if so it will add the coefficients of the monomials with the same degree,unless there doesn't exist monomial with the same degree a new monom will be created and added to the list

Package Operation

1. add

```
public static Polinom add(Polinom a, Polinom b) {
				Polinom cPolinom = new Polinom();
				cPolinom.copyPolinom(a);

				for (Monom i : b.getTerm()) {
					if (!cPolinom.checkDegree(i.getDegree()) ) {
						cPolinom.addTerm(i);

					} else {
						for (Monom j : cPolinom.getTerm()) {
							if (j.getDegree() == i.getDegree()) {
								j.setCoefficient(j.getCoefficient() + i.getCoefficient());
							}
						}
```

```
                }

                }
                cPolinom.sortPolinom();
                cPolinom.deleteZerous();
                return cPolinom;
        }
```

This method is used for adding polynomials(polynomial a,polynomial b).First we create a new polynomial c in order to not modify the content of a or b ,after that we copy the content of the first polynomial a in c .The next step consist on iterating through polynomial b and adding each monomial from b to c.

2. subtract

```
public static Polinom substract(Polinom a, Polinom b) {
        Polinom polinom = new Polinom();
        for (Monom i : b.getTerm()) {
                polinom.addTerm(new Monom(i.getCoefficient() * (-1), i.getDegree()));
        }

        return Operatii.add(a, polinom);
}
```

This method operates the subtraction of polynomials. It the consist of creating a new polynomial (polinom) and adding every monomial from b with opposite sign for coefficient. So the subtraction will be equivalent with an addition . After that we call the operation add(a, polinom) and return the value returned from that operation

3.multiply

```
public static Polinom multiply(Polinom aPolinom, Polinom bPolinom) {
        Polinom cPolinom = new Polinom();

        for (Monom bMonom : bPolinom.getTerm()) {
                for (Monom aMonom : aPolinom.getTerm()) {
                        Monom aux = new Monom();

                        aux.setCoefficient(aMonom.getCoefficient() * bMonom.getCoefficient());
                        aux.setDegree(aMonom.getDegree() + bMonom.getDegree());
                        cPolinom.addTerm(aux);
                }
        }
        cPolinom.sortPolinom();


        return cPolinom;
}
```

Method multiply takes two polynomials (aPolinom and bPolinom) , and returns through polynomial cPolynom the product aPolinom X bPolinom .It uses an auxiliary variable aux in order to not modify the monomial from any term. At the end of the method cPolinom will be sorted in order to obtain a prettier format for the polynomial

3. divide

```
public static void divide(Polinom firstPol, Polinom bPolinom, Polinom cat, Polinom rest) throws DivideByZeroException{
                        Monom multiplyMonom = new Monom();
                        Polinom aux = new Polinom();
                        Polinom aux1 = new Polinom();
                        Polinom aPolinom = new Polinom();
                        aPolinom.copyPolinom(firstPol);
                        ArrayList<Monom> deleteList = new ArrayList<>();
                        if (bPolinom.getMaxmonom().getDegree() == 0)
                                {
                                throw new DivideByZeroException("error");
                                }
                        else {
                                while (aPolinom.getMaxmonom().getDegree() >= bPolinom.getMaxmonom().getDegree() ) {
                                        aux.copyPolinom(bPolinom);
                                        multiplyMonom.setDegree(aPolinom.getMaxmonom().getDegree() -
bPolinom.getMaxmonom().getDegree());
                                        multiplyMonom.setCoefficient(aPolinom.getMaxmonom().getCoefficient() /
bPolinom.getMaxmonom().getCoefficient());
                                        cat.addTerm(multiplyMonom);
                                        aux.multiplyMonom(multiplyMonom);
                                        aux1.copyPolinom(Operatii.substract(aPolinom, aux));

                                        aPolinom.copyPolinom(aux1);

                                        for (Monom iMonom : aPolinom.getTerm())
                                                if (iMonom.getCoefficient() == 0.0f)
                                                        deleteList.add(iMonom);
                                        aux.getTerm().clear();
                                        aPolinom.getTerm().removeAll(deleteList);
                                }
                                rest.copyPolinom(aPolinom);
                        }
                        }
```

This method take as parameters 4 polynomials, first two representing the operands and the other two representing the quote and the remainder. It throws an exception in case the secondPolynomial bPolinom is 0 .

4) integrate

```
public static Polinom integrate(Polinom Polinom) {
                        Polinom aPolinom=new Polinom();
                        aPolinom.copyPolinom(Polinom);
                        for (Monom a : aPolinom.getTerm()) {
                                a.setCoefficient(a.getCoefficient() * 1 / (a.getDegree() + 1));
                                a.setDegree(a.getDegree() + 1);
                        }
                        return aPolinom;
        }
```

This method take as parameters a polynomial(Polinom) which is copied into another polynomials in order to not modify the Polynomial P .It returns the integrated polynomial value of the polynomial Polinom.

5) derivate

```
public static Polinom derivate(Polinom Polinom) {

                        Polinom aPolinom=new Polinom();
                        aPolinom.copyPolinom(Polinom);
                        List<Monom> deleteList=new ArrayList<>();
                        for (Monom a : aPolinom.getTerm()) {
                        if(a.getDegree()!=0) {
                                a.setCoefficient(a.getCoefficient() * a.getDegree());
                                a.setDegree(a.getDegree() - 1);
                                }
                        else
                                {
                                deleteList.add(a);
                        }
                }
                aPolinom.getTerm().removeAll(deleteList);
                return aPolinom;

        }
```

This method return the integrated value of the polynomial Polinom ,which is transmited as parameter.It copy the value of the polynomial Polinom in the polynomial APolinom in order to not modify the value of the Polinom.The method use in auxiliary list ,deleteList,for deleting all monomials which have the degree equals to zero before the derivation.

Package exceptions

1)DivideByZeroException

It extends the class RuntimeExceptions. It is used to signal when the user try to do a division by zero.

Package assInterface

1) Interface

This class is used for implementing the graphical user interface. It contains all the components from the GUI used in this project(JLabel, JButton, JTextField). For each Polynomial there exist a JTextField where the user type the value of the polynomial, a JLabel bellow the JTextField where the user can see the value stored in each Polynomial and a JButton used the save the value typed in the JTextField SaveButton .For each operation described before there exist a button with a specific symbol on it, like for addition the button will have the symbol “+”.This class contains a lot of methods in order to modify the value for each JLabel and also to specify what each button does.

2) Controller

This class contains a variable of type Interface , 2 Polynomials which are used to store value of the Polynomial introduced by the user and also 1 boolean variable for each polynomial, which is false until a value is stored in the polynomial .It contains a method (initialize buttons) which implements the function of each button from the interface.

5)Result

Each operation is tested using a JUnit Test, by providing two predefined polynomials, performing each operation, and testing to see if the answer matches the expected output.

In each tested case, the tested output is the string returned by the method toString() from class Poly.

Each tested case performed as expected, returning no errors and matching the expected output.

6)Conclusions

In conclusion, all of the primary and secondary objectives were met, the result being a program that performs basic operations on polynomials, that also has a graphical user interface for interaction with a user.

Further Improvements can be brought both to the interface and to the functionality of the program. Program can be modified in order to be more efficient and also the interface can have more suggestive message to the user when he is doing something wrong .

7)Bibliography

https://www.youtube.com/watch?v=8lT00iLntFc&t=151s
https://www.vogella.com/tutorials/JavaRegularExpressions/article.html