DOCUMENTATIE

Calculator de polinoame

NUME STUDENT: Sipos Bogdan-Cristian
GRUPA: 30225

CUPRINS

1. Obiectivul temei

Obiectivul principal a acestei teme de laborator este de a propune, proiecta si implementa un sistem de procesare si calculare a polinoamelor cu o singura variabila necunoscuta si cu coeficienti numere reale. Acesta trebuie sa efectueze operatiile de adunare, scadere, inmultire si impartire a doua polinoame, precum si operatiile de derivare si integrare.

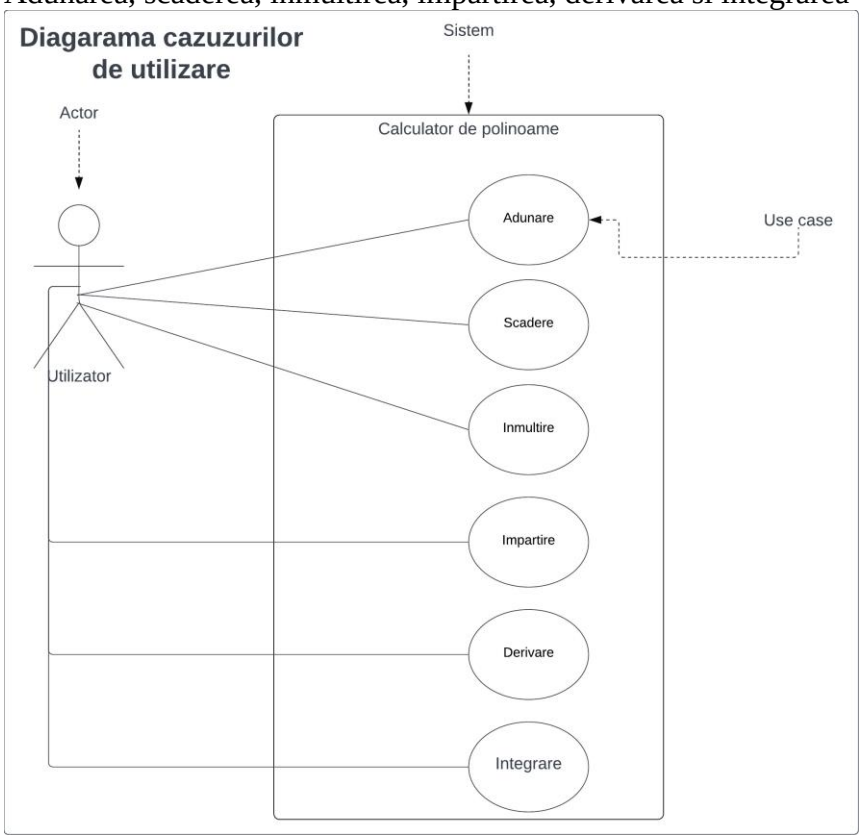
Obiectiv Secundar	Descriere	Capitol
Dezvoltare de use case-uri si scenarii	A fost un ajutor in a identifica necesitatile problemei si sa ilustram interactiunea dintre utilizator si sistem	2
Alegerea structurilor de date	Alegerea anumitor structuri de date a avut scopul de a ne eficientiza programul sau de a ne simplifica metoda de implementare	3
Impartirea pe clase	Impartirea pe clase ne ajuta in: organizarea structurii codului, reutilizarea acestuia, incapsularea si ascunderea detaliilor precum si in partea de intretinere si extindere	3
Dezvoltarea algoritmilor	Dezvoltarea algoritmilor a fost facuta prin traducerea in cod a algoritmilor operatiilor de polinoame din matematica	3
Implementarea solutiei	In implementare am divizat fiecare problema in mai multe subprobleme (in mai multe clase, subclase dar si metode) dar am si realizat o interfata grafica cat mai simpla de utilizat	4
Testare	Testarea a avut rolul de a ne asigura ca metodele implementate sunt corecte prin intermediul alegerii unor exemple de verificat	5

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

Cerintele functionale ale calcalatorului polinomial sunt urmatoarele operatii:

- Citirea unui polinom de la tastatura sub forma 7x^2+8x^3-x+9

- Adunarea, scaderea, inmultirea, impartirea, derivarea si integrarea



Use case: Adunare
Actor principal: Utilizator
Scenariul principal:

1. Utilizatorul introduce primul polinom
2. Utilizatorul introduce al doilea polinom
3. Utilizatorul apasa pe butonul de adunare(Add)
4. Calculatorul verifica daca forma intrudusa a polinoamelor este una corecta
5. Calculatorul afiseaza rezultatul adunarii

Scenarii alternative:

a) Nu a fost introdus polinomul
   - Polinomul va fi considerat calaculator ca fiind egal cu 0

b) Nu a fost introdus corespunzator polinomul
   - Nu va efectua operatia de adunare si nu va da nimic ca rezultat
   - In acest scenariu ne intoarcem la pasul 1

Use case: Scadere
Actor principal: Utilizator
Scenariul principal:

1. Utilizatorul introduce primul polinom
2. Utilizatorul introduce al doilea polinom
3. Utilizatorul apasa pe butonul de scadere (Substract)

4. Calculatorul verifica daca forma intrudusa a polinoamelor este una corecta
5. Calculatorul afiseaza rezultatul scaderii

Scenarii alternative:

a) Nu a fost introdus polinomul
   - Polinomul va fi considerat calaculator ca fiind egal cu 0

b) Nu a fost introdus corespunzator polinomul
   - Nu va efectua operatia de scadere si nu va da nimic ca rezultat
   - In acest scenariu ne intoarcem la pasul 1

Use case: Inmultire
Actor principal: Utilizator
Scenariul principal:

1. Utilizatorul introduce primul polinom
2. Utilizatorul introduce al doilea polinom
3. Utilizatorul apasa pe butonul de inmultire (Multiply)
4. Calculatorul verifica daca forma intrudusa a polinoamelor este una corecta
5. Calculatorul afiseaza rezultatul inmultirii

Scenarii alternative:

a) Nu a fost introdus polinomul
   - Polinomul va fi considerat calaculator ca fiind egal cu 0

b) Nu a fost introdus corespunzator polinomul
   - Nu va efectua operatia de inmultire si nu va da nimic ca rezultat
   - In acest scenariu ne intoarcem la pasul 1

Use case: Impartire
Actor principal: Utilizator
Scenariul principal:

1. Utilizatorul introduce primul polinom
2. Utilizatorul introduce al doilea polinom
3. Utilizatorul apasa pe butonul de impartire (Divide)
4. Calculatorul verifica daca forma intrudusa a polinoamelor este una corecta
5. Calculatorul afiseaza rezultatul impartirii

Scenarii alternative:

a) Nu a fost introdus unul sau mai multi polinomi
   - Nu va efectua operatia de impartire si nu va da nimic ca rezultat
   - In acest scenariu ne intoarcem la pasul 1

b) Nu a fost introdus corespunzator polinomul
   - Nu va efectua operatia de impartire si nu va da nimic ca rezultat
   - In acest scenariu ne intoarcem la pasul 1

Use case: Derivare
Actor principal: Utilizator
Scenariul principal:

1. Utilizatorul introduce primul polinom

2. Utilizatorul apasa pe butonul de derivare (Derivative)
3. Calculatorul verifica daca forma intrudusa a polinoamelor este una corecta
4. Calculatorul afiseaza rezultatul derivarii

Scenarii alternative:

a) Nu a fost introdus polinomul
   - Polinomul va fi considerat de calaculator ca fiind egal cu 0

b) Nu a fost introdus corespunzator polinomul
   - Nu va efectua operatia de derivare si nu va da nimic ca rezultat
   - In acest scenariu ne intoarcem la pasul 1

Use case: Integrare
Actor principal: Utilizator
Scenariul principal:

1. Utilizatorul introduce primul polinom
2. Utilizatorul apasa pe butonul de integrare (Integral)
3. Calculatorul verifica daca forma intrudusa a polinoamelor este una corecta
4. Calculatorul afiseaza rezultatul integrarii

Scenarii alternative:

a) Nu a fost introdus polinomul
   - Polinomul va fi considerat de calaculator ca fiind egal cu 0

b) Nu a fost introdus corespunzator polinomul
   - Nu va efectua operatia de integrare si nu va da nimic ca rezultat
   - In acest scenariu ne intoarcem la pasul 1

3. Proiectare

Am impartit problema in patru clase de tip obiect pentru a ne usura implementarea: in clasa Monom avem coeficientul si rangul unui monom, in clasa Polinom avem un HashMap care reprezinta un polinom care contine monoame de tip Monom, in clasa View ne-am ocupat de cum arata calculatorul polinomial, in clasa Controller am rezolvat partea interactiva a calculatorului si prelucrarea informatiilor in monoame si apoi in polinoame.

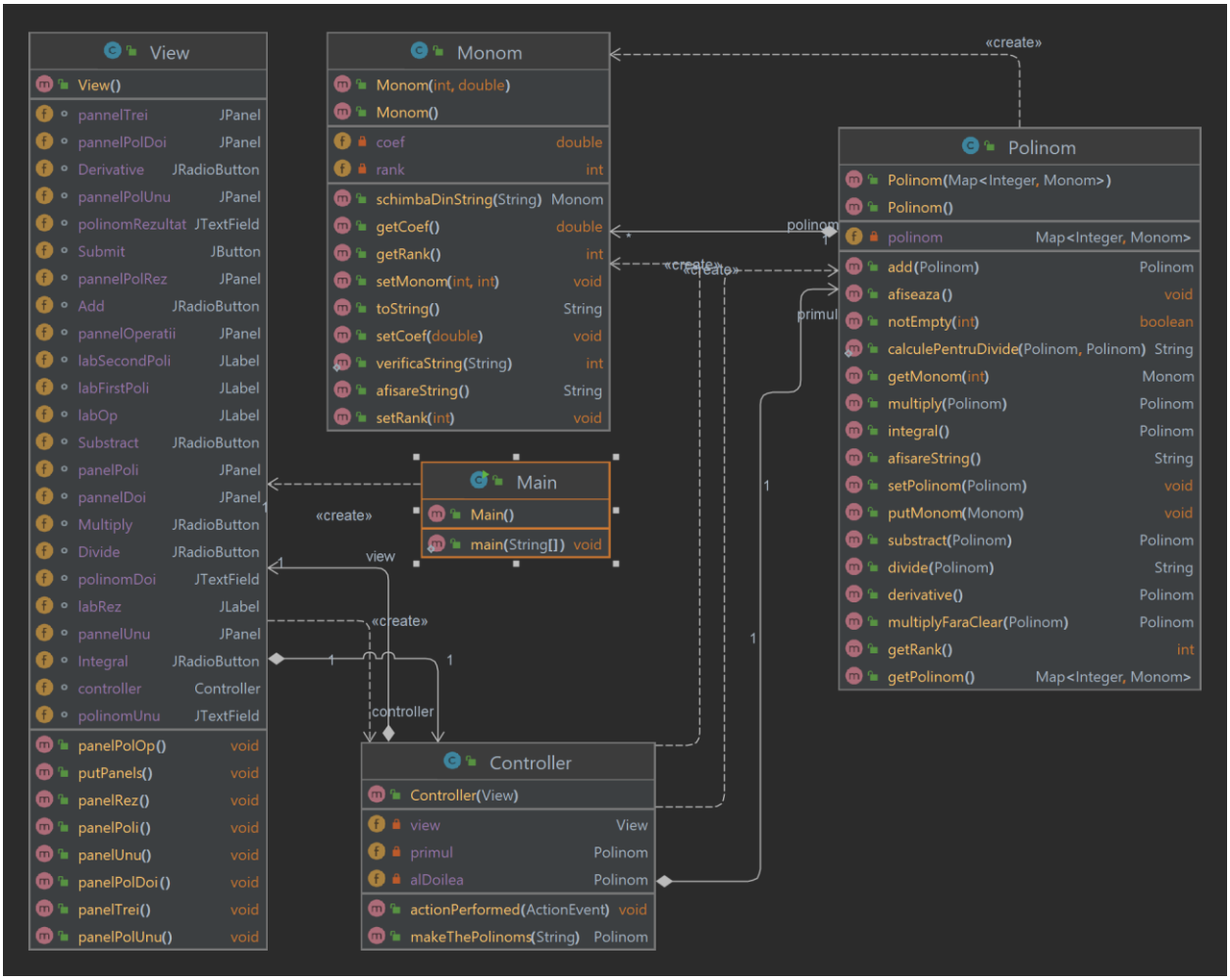
La structuri de date am folosit un HashMap pentru stocarea monoamelor intr-un polinom datorita anumitor avantaje precum:

- Eficienta: Operațiile de bază cum ar fi inserarea, ștergerea și căutarea sunt eficiente și au o complexitate medie de timp constant (O(1)). Cu toate acestea, în cazuri rare, complexitatea poate deveni O(n), dar acest lucru este extrem de rar

- Dimensiune ajustabila: aceasta se poate redimensiona automat pe măsură ce numărul de elemente crește sau scade, asigurându-se că performanța rămâne optimă indiferent de dimensiunea datelor.

- Acces rapid: Odată ce cheia este cunoscută, valoarea corespunzătoare poate fi obținută rapid, fără a fi nevoie să parcurgem toată structura. Acest lucru face HashMap ideal pentru accesul rapid la date în funcție de chei.
- Permite valori nule și chei nule: permite atât cheile, cât și valorile să fie nule, ceea ce oferă o mai mare flexibilitate în gestionarea datelor.
- Iterare eficientă: Iterarea este una eficientă și permite accesul la toate perechile cheie-valoare într-un timp optim.

Singura interfata pe care am implementat-o a fost ActionListener pentru tratarea evenimentelor de buton.



Algoritmii pe care i-am implementat au fost:

1. Cel de adunare a polinoamelor

- Am luat prima data monoamele a caror putere se gasea in ambele polinoame si le-am adunat iar rezultatul l-am introdus intr-un polinom nou. In acesta am adunat dupa aceea monoamele ramase din ambele polinoame.

2. Cel de scadere a polinoamelor
   - Am luat prima data monoamele a caror putere se gasea in ambele polinoame si le-am scazut iar rezultatul l-am introdus intr-un polinom nou. In acesta am adunat dupa aceea monoamele ramase in monumul din care am scazut si am scazut monoamele ramase in celalalt.
3. Cel de inmultire a polinoamelor
   - Am inmultit fiecare monom a unui polinom cu monomul celuilalt monom
4. Cel de impartire a polinoamelor
   - Se ordoneaza cele doua polinoamele (o sa le numim a si b) dupa puterile descrescatoare ale nedeterminantei x
   - Se face impartirea polinomului mai mare (in cazul nostru a) la polinomul de grad mai mic
   - Se imparte primul termen a lui f la primul termen al lui g si se obtine astfel primul termen al catului
   - Se inmulteste rezultatul obtinut cu impartitorul g si se scade acest produs din deimpartitorul f. Acest calcul ne da primul rest al impartirii
   - Se repeta procedeul luand primul rest ca deimpartit
   - Algoritmul se termina cand restul este mai mic ca impartitorul
   - Cea ce ramane fiind restul intregii impartiri
5. Cel de derivare a polinoamelor
   - Am folosit formula de derivare (coef * x^n)' = (n-1) * coef * x^(n-1) pentru fiecare monom in parte din primul polinom
6. Cel de integrare a polinoamelor
   - Am folosit formula de integrare ∫(coef * x^n) = coef / (n+1) * x^(n+1) pentru fiecare monom in parte din primul polinom

4. Implementare

Clasa Monom contine:

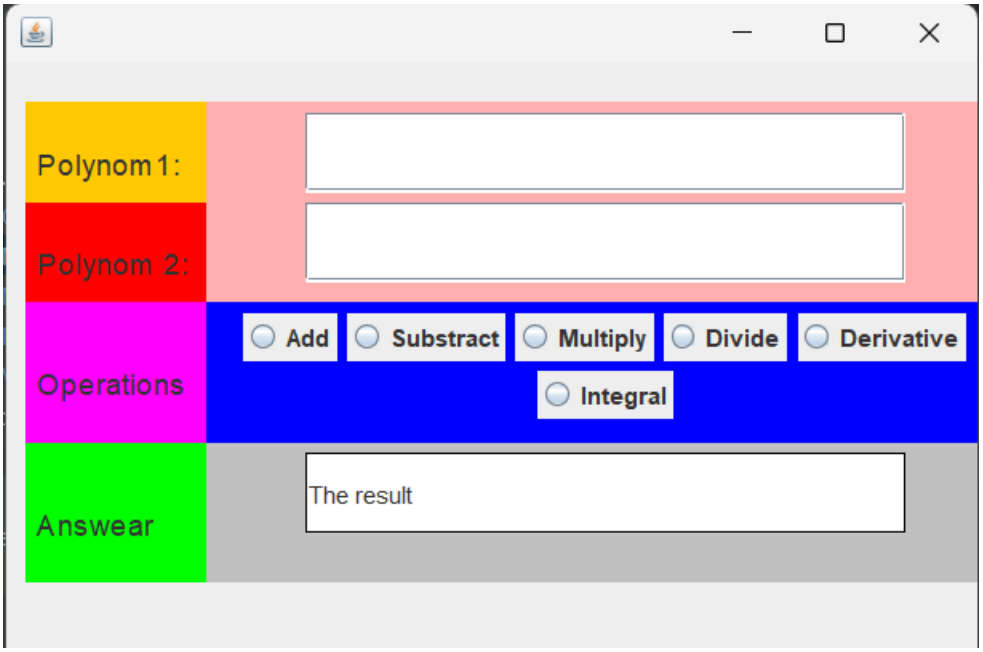
1. Campuri:
   - coef, de tip double, reprezinta coeficientul unui monom
   - rank, de tip intreg, reprezinta puterea unui monom
2. Metode:
   - getCoef returneaza coeficientul monomului
   - getRank returneaza puterea monomului
   - setCoef face ca coeficientul sa ia valoarea parametrului metodei
   - setRank faca ca puterea sa ia valoarea parametrului metodei
   - verificaString primeste ca parametru un sir care reprezinta un monom si verifica in ce forma se afla acesta si returneaza o valoarea de tip int care reprezinta in caz trebuie sa fie tratat sirul

- schimbaDinString primeste ca ca parametru un sir care reprezinta un monom, foloseste functia verificaString pentru a stabili cazul in care trebuie tratat acel sir si apoi in functie de acesta va crea si returna un obiect de tip Monom
- afisareString are rolul de a pune intr-o forma matematica monomul sub forma unui sir pe care apoi il returneaza

Clasa Polinom contine:

1. Campuri:
   - Map<Integer, Monom> polinom este un Map care stocheaza mai multe monoame de tip Monom intr-un HashMap, cheile fiind puterea monoamelor
2. Metode:
   - setPolinom care are ca parametru un obiect Polinom care este atribuit Polinomului actual
   - putMonom are ca parametru un Monom care este introdus in HashMap-ul polinom
   - getMonom are ca paremetru o valoare de tip intreg care reprezinta puterea monomului pe care vrem sa il gasim. Aceasta il gaseste si il returneaza sub forma de obiect Monom
   - notEmpty are ca paremetru o valoare de tip intreg care reprezinta puterea monomului pe care vrem sa il gasim. Daca nu avem nici un monom care sa contina acea putere vom returna fals daca avem vom returna adevarat
   - getRank are rolul de a gasi cea mai mare putere a polinomului pe care apoi o returneaza
   - afisareString parcurge tot HashMap-ul cu monoame si le introduce intr-un sir pentru o afisare mai frumoasa. Daca nu avem nici un monom sirul va contine doar numarul zero. La final returneaza sirul realizat.
   - add aduna parametrul de tip Polinom cu Polinomul care a apelat metoda. Prima data am parcurs Map-ul Polinomul care a apelat metoda si daca ambele Polinoame aveau Monoame pe aceasi cheie adunam coeficientii monoamelor intre ei si punam monomul rezultat intr-un nou Polinom c. Apoi am adaugat in acesta monoamele celor doua polinoame care au ramas(cele care nu au avut putere comuna). La final am golit HashMap-urile Polinoamelor intre care am facut adunarea si am returnat Polinomul c.
   - substrtact scade parametrul de tip Polinom din Polinomul care a apelat metoda. Prima data am parcurs Map-ul Polinomul care a apelat metoda si daca ambele Polinoame aveau Monoame pe aceasi cheie scadeam coeficientii monoamelor intre ei si punam monomul rezultat intr-un nou Polinom c. Apoi am adaugat in acesta monoamele ramase in Polinomul care a apelat functia si monoamele care au ramas din celalalt dar inversand semnul coeficientilor. La final am golit HashMap-urile Polinoamelor intre care am facut scaderea si am returnat Polinomul c.
   - multiply are ca parametru un Polinom b. Am parcurs ambele polinoame pe rand si am inmultit fiecare monom a Polinomului care a apelat metoda cu inmultit fiecare monom a lui b. Fiecare monom rezultat din inmultire a fost introdus intr-un nou Polinom care apoi a fost returnat. Polinoamele care au fost inmultite le-au fost golite HashMap-urile.
   - multiplyFaraClear face acelasi lucru ca multiply dar nu mai goleste HashMap-urile Polinoamelor.

- calculePentruDivide se dau doua Polinoame a si b, unde b este impartit la a. Daca a are o singur monom si acesta are puterea 0 atunci Polinomul rezultat c va contine moanoamele lui b cu coeficientul fiecarui monom impartit la coeficientul monomului lui a. Daca nu, se imparte primul termen a lui b la primul termen al lui a si se obtine astfel primul termen al catului care se introduce in noul Polinom c. Se inmulteste rezultatul obtinut cu impartitorul a si se scade acest produs din deimpartitorul b, acest calcul ne da primul rest al impartirii. Se repeta procedeul luand primul rest ca deimpartit iar algoritmul se termina cand restul este mai mic ca impartitorul. Cea ce ramane fiind restul intregii impartiri. Se transforma Polinomul c intr-un sir caruia i se concateneaza si restul tot sub forma de sir si se face returnarea acestuia.
- divide determina care polinom are o putere mai mare, cel transmis ca parametru sau cel care a apelat metoda, si apoi apeleaza metoda calculePentruDivide si returneaza sirul rezultat.



Clasa View este o extensie a clasei Java JFrame si contine:

1. Campuri:
    - controller obiect de tip Controller care se ocupa de partea interactiva a interfetei grafice
    - polinomUnu, polinomDoi, polinomRezultat care sunt obiecte de tip JTextField si ne permite scrierea de text si navigarea in casuta sa text
    - labRez, labFirstPoli, labSecondPoli, labOp sunt obiecte de tip JLabel care sunt ca niste etichete text care nu pot fi modificate de utilizator
    - panelPoli, pannelUnu, pannelTrei, pannelPolUnu,pannelPolDoi, pannelOperatii, pannelPolRez sunt obiecte de tip JPanel care reprezinta suprafete pe care se pot atasa tot felul de obiecte
    - Add, Substract, Multiply, Derivative, Integral, Divide sunt JRadioButton care sunt niste butoane care permit doar alegerea unei variante pe rand marcand cu o bulina plina alegerea facuta

2. Metode:
    - panelPoli creaza partea cu fundal roz a calculatorului, panelPoli, unde avem si cele doua casete text polinumUnu si polinomDoi unde introducem cele doua polinoame pe care le vom folosi. In aceasta metoda am reglat de asemenea si dimensiunile la JPanel si la JTextField-uri.
    - panelUnu creza partea cu fundal albastru a calculatorul, pannelUnu, unde avem grupate butoanele: Add, Substract, Multiply, Divide, Derevative si Integral care reprezinta operatiile pe care le putem face pe calculator. Fiecare buton: apare pe GUI cu numele sau, au ActionListener in controller si reprezinta operatiile pe care le putem face cu polinoamele.
    - panelTrei creaza partea cu fundal gri a calculatorului, pannelTrei, unde avem si caseta text polinomRezultat unde introducem polinoamul rezultat dupa ce executam o operatie. In aceasta metoda am reglat de asemenea si dimensiunile la JPanel si la JTextField.
    - panelPolUnu creaza partea portocalie a calculatorului, pannelPolUnu, unde avem JLabel-ul labFirstPoli unde am scris Polynom 1. Aici am reglat si dimensiunile celor doua componente.
    - panelPolDoi creaza partea rosie a calculatorului, pannelPolDoi, unde avem JLabel-ul labSecondPoli unde am scris Polynom 2. Aici am reglat si dimensiunile celor doua componente.
    - panelPolOp creaza partea magenta a calculatorului, pannelPolOp, unde avem JLabel-ul labOp unde am scris Operations. Aici am reglat si dimensiunile celor doua componente.
    - panelRez creaza partea verde a calculatorului, pannelRez, unde avem JLabel-ul labRez unde am scris Answear. Aici am reglat si dimensiunile celor doua componente.
    - putPanels are rolul de a ajusta dimensiunea frame-ului, de a ne asigura ca atunci cand apasam pe butonul de inchidere de pe coltul ferestrei programul nostru nu va mai rula, de a face fereastra vizbila si de adauga fiecare JPanel creat in metodele de mai sus pe JFrame

Clasa Controller contine:

1. Interfata ActionListener
2. Campuri:
    - view care este un obiect de tip View care se ocupa de cum arata calculatorul nostru polinomial
    - primul si alDoilea care sunt obiecte de tip Polinom
3. Metode:
    - makeThePolinoms care primeste un sir ca parametru care contine un polinom scris sub forma matematica si folosind un regex il desparte siruri care sunt monoame si apoi folosind functia schimbaDinString, a Monoamelor, transforma aceste siruri in obiecte de tip Monom care sunt apoi introduse intr-un obiect Polinom prin metoda putMonom. Acest Polinom rezultat este returnat.
    - actionPerformed are ca parametru ActionEvent care ne va spune ce buton a fost apasat si implicit de operatie se doreste sa fie efectuata cu polinoamele. Daca

apasam butoanele Add, Substract, Multiply sau Divide atunci vom folosi functiile setPolinom si makeThePolinoms pentru a pune valorile introduse in casetele text de langa Polynom1 si Polynom2 in Polinoamele primul si alDoilea. Apoi apelam operatia pe care am ales-o(primul.add(alDoilea), primul.substract(alDoilea), primul.multiply(alDoilea), primul.divide(alDoilea)) iar rezultatul ei de forma sir o sa-l introducem in cea de a treia caseta de langa Answear. Daca apasam butoanele Derivative sau Integral atunci vom folosi functiile setPolinom si makeThePolinoms pentru a pune valorea introdusa in caseta text de langa Polynom1 in Polinomul primul. Apoi apelam operatia pe care am ales-o(primul.derivative(), primul.integral()) iar rezultatul ei de forma sir o sa-l introducem in cea de a treia caseta de langa Answear.

Clasa OperationsTest contine testarile de Tip JUnit pentru fiecare operatie a calculatorului de polinoame imparte folosind metodele: addTest(), substractTest(), MultiplyTest(), DivideTest(), DerivativeTest() si IntegralTest().

Clasa Main contine metoda main care determina inceperea programului nostru.

5. Rezultate

Am facut teste pentru operatiile dintre polinoame Add, Substract, Multiply, Divide, Derivative, Integral cu ajutorul functiilor addTest(), substractTest(), MultiplyTest(), DivideTest(), DerivativeTest() si IntegralTest(), toate trecand cu succes.

Pentru addTest() am folosit polinomul 5x^6-7x^7+4x^2-9+x si 3+x^2 polinomul pe care le-am adunat si ne-am asteptat la rezultatul -7.0x^7+5.0x^6+5.0x^2+1.0x-6.0.

Pentru substractTest() am folosit polinomul 5x^6-7x^7+4x^2-9+x si 3+x^2 polinomul pe care le-am scazut si ne-am asteptat la rezultatul -7.0x^7+5.0x^6+3.0x^2+1.0x-12.0.

Pentru multiplyTest() am folosit polinomul 5x^6-7x^7+4x^2-9+x si 3+x^2 polinomul pe care le-am inmultit si ne-am asteptat la rezultatul -7.0x^9+5.0x^8-21.0x^7+15.0x^6+4.0x^4+1.0x^3+3.0x^2+3.0x-27.0.

Pentru devideTest() am folosit polinomul 5x^6-7x^7+4x^2-9+x si 7+x polinomul pe care le-am impartit si ne-am asteptat la rezultatul -7.0x^6+54.0x^5-378.0x^4+2646.0x^3-18522.0x^2+129658.0x-907605.0+(+6353226.0)/(+1.0x+7.0).

Pentru derivativeTest() am folosit polinomul 5x^6-7x^7+4x^2-9+x l-am derivat si ne-am asteptat la rezultatul -0.875x^8+0.7142857142857143x^7+1.3333333333333333x^3+0.5x^2-9.0x.

Pentru integralTest() am folosit polinomul 5x^6-7x^7+4x^2-9+x l-am integrat si ne-am asteptat la rezultatul -49.0x^6+30.0x^5+8.0x+1.0.

6. Concluzii

In concluzie, sunt de parere ca acest proiect mi-a aprofundat cunostiintele in tot ce inseamna limbajul Java, implementarea paradigmelor OOP si creearea unui program cu o interfata grafica. De asemenea m-au ajutat sa imi reamintesc tehnicile de programare invatate semestrul trecut. Calculator polinomial poate fii in viitor dezvoltat intr-un calaculator care sa poate rezolva probleme si ecuatii mai complicate cu functii geometrice.

7. Bibliografie

1. https://stackoverflow.com/questions/36490757/regex-for-polynomial-expression
2. https://www.youtube.com/watch?v=Kmgo00avvEw&t=1356s&ab_channel=BroCode