MINISTERUL EDUCAȚIEI, CERCETĂRII, TINERETULUI ȘI SPORTULUI
UNIVERSITATEA TEHNICĂ
DIN CLUJ-NAPOCA

Sistem de procesare a polinoamelor

TEMA 1 – Tehnici de Programare

Nume: Vlad Mărginean
Grupa: 30224

Asistent: Călina Cenan

Cuprins

1.Obiectivul temei

1.1 Cerința problemei

Propuneți, proiectați și implementați un sistem de procesare a polinoamelor de o singură variabilă cu coeficienți întregi. Cu alte cuvinte, se vor introduce două polinoame de la tastatură, prin diferite metode de citire a coeficienților și/sau a gradelor și se va realiza un fel de calculator special pentru a efectua o mulțime de operații asupra acestor polinoame introduse.

1.2 Obiectiv

Obiectivul temei este obținerea unui sistem funcțional de procesare a polinoamelor cu coeficienți întregi. Acesta va trebui să cuprindă funcții standard de calcul, precum:

- Adunarea
- Scăderea
- Inmulțirea
- Impărțirea
- Derivarea
- Integrarea

Pentru a facilita folosirea operațiilor implementate, se va folosi o interfață grafică. Aceasta va conține atât funcționalități pentru introducerea datelor, dar și pentru accesarea operațiilor efectuate pe aceste date:

- Citirea polinoamelor de la tastatură
- Efectuarea operațiilor prin selectarea butoanelor
- Afișarea rezultatului

În partea de calcul a operațiilor vor trebui verificate excepțiile ce pot apărea în decursul introducerii datelor, sau efectuării calculelor pe polinoamele citite:

- Introducerea unui format greșit la citirea polinomului
- Adunarea unui polinom cu un monom sau o constanta (a fost implementată o singură metodă care asigură calcularea corectă a sumei pentru fiecare caz)
- Scăderea a două polinoame egale
- Inmulțirea cu un scalar sau cu 0
- Impărțirea cu 0 sau cu un scalar

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

2.1 Introducere

Proiectul de față a fost realizat în limbajul orientat obiect Java, utilizând IDE-ul Eclipse. Java este unul dintre cele mai răspândite limbaje OOP și oferă flexibilitate și portabilitate aplicațiilor dezvoltate. Acestea pot fi transferate pe orice platformă care are instalată o mașină virtuală Java.

Problema construirii unui sistem de calcul a operațiilor cu polinoame are o complexitate medie, însă ridică dificultăți la implementarea fiecărei excepții care poate apărea în funcție de datele de intrare. Proiectul trebuie să fie unul stabil, intuitiv, pentru a putea fi folosită de un utilizator străin. Acesta trebuie să primească informații de utilizare din interfața programului, dar și să fie atenționat în cazul apariției unor greșeli.

In matematică, un polinom este definit ca o expresie construită dintr-una sau mai multe variabile și constante, legate între ele prin operații aritmetice(adunare, scădere, înmulțire și ridicare la o putere constantă, pozitivă întreagă). Polinoamele sunt construite din termeni numiți monoame, de forma (+-)ax^b. Coeficientul a și exponentul b pot lua orice valori sau pot lipsi complet din expresia monomului. Exponentul 0 va indica un termen liber.

Polinoamele introduse din interfață vor fi de forma:

a1x^b1+a2x^b2+...+anx^bn

Acesta este preluat ca un șir de caractere, iar apoi procesat și despărțit în monoame care ulterior vor forma polinomul supus operațiilor implementate. Coeficienții și exponenții vor fi convertiți intr-un format numeric pentru a facilita executarea metodelor de adunare, scadere, etc.

In interfața grafică, există posibilitatea de a introduce două polinoame pentru a efectua operații, însă pentru operațiile unare (derivare, integrare) se poate introduce un singur polinom, prin scrierea formei în prima locație disponibila și apăsarea butonului „Get Polinom 1”. Pentru a executa operații cu doi parametrii, va fi necesara preluarea datelor din ambele căsuțe de text.

Rezultatele operațiilor vor fi returnate în interfața grafică prin intermediul a încă două casuțe de text (deoarece împărțirea polinoamelor necesită afișarea câtului și restului). Celelalte operații vor afișa rezultatul în prima locație.

2.2 Modelare

Pentru a obține un model curat, care să respecte principiile OOP dar și să implementeze toate funcționalitățile cerute, a fost făcută o analiză a problemei și, folosind structura obținută, problema a fost împărțită în clase legate între ele, fiecare având un rol anume, fie în implementarea operațiilor, fie în construirea interfeței cu utilizatorul.

Clasele au fost grupate în două pachete de bază:

- GUI (Graphic User Interface) – pachetul care conține clasa folosită pentru creerea ferestrei de interfață și legarea metodelor pentru calculul operațiilor astfel încât acestea să fie accesibile din exterior fără a fi nevoie de cunoștințe programatorice.
- Model – conține o multitudine de clase care fac posibilă realizarea operațiilor cerute.

  Printre acestea, grupul de clase format din Monom, Polinom și ComparatorPutere ajută la creerea și păstrarea polinomului obținut prin convertirea șirului introdus din interfață.

  O altă grupare de clase, UnaryOperation, Derivate și Integrate, reprezintă implementările operațiilor care au nevoie de un singur polinom ca parametru.

  În paralel, clasele BinaryOperation, Add, Subtract, Multiply și Division fac posibilă calcularea operațiilor cu doi parametrii.

Clasele din pachetul Model sunt:

- Monom:

Este utilizată pentru păstrarea unu element fundamental al polinomului, de forma ax^b, unde a reprezintă coeficientul, iar b exponentul monomului. Polinomul va fi reprezentat de o sumă a acestor monoame.

Coeficientul este stocat ca un element de tip double, iar exponentul este un numar întreg

Pe lângă constructorul clasei și metodele de get/set parametrii, clasa Monom conține metoda toString din clasa Object, dar suprascrisă pentru a

putea afișa un obiect de tip Monom. Aceasta construiește un string în funcție de parametrii obiectului: coeficient și exponent.

- Polinom:

Reprezintă clasa care stă la baza întregului proiect, împreună cu Monom, deoarece conține un ArrayList de Monoame, reprezentând totalitatea elementelor de tip [coeficient, putere] introduse pentru un polinom.

Constructorul clasei are doar rolul de a inițializa lista de monoame.

Aceasta conține metode pentru a facilita accesul din exterior (*addMonom*, *getList*, *getGrad*, *clone*). Gradul polinomului se va modifica după fiecare inserare a unui monom.

Metoda *addMonom* are rolul de a adăuga, pe rând, monoamele primite din sirul de caractere primite la intrare. Se verifică dacă, coeficientul monomului este diferit de 0, dar și dacă mai există un monom de același grad, deja prezent în polinom. In acest caz, nu se va adăuga un nou monom, ci se vor aduna coeficienții.

- ComparatorPutere:

Este o clasă folosită în sortarea listei de monoame, în ordinea descrescătoare a puterilor fiecăruia. Acesta este un proces important, pentru a evita apariția unor complicații în implementarea operațiilor propriu-zise.

- Operation:

Operation este o interfață prezentă în proiect pentru a face o legătură logică între clasele UnaryOperation și BinaryOperation.

- UnaryOperation:

Reprezintă o clasă părinte pentru operațiile unare, conținând metoda compute (și ea abstractă) pentru a lega toate operațiile ce se execută cu un singur parametru.

Aceasta obligă clasele care o moștenesc să implementeze metoda declarată *abstract.* In acest mod, se va evita creerea unei noi clase de operație unară, fără implementarea unei metode de calcul a rezultatului.

- Derivate:

Extinde clasa UnaryOperation și implementează metoda *compute* pentru calculul derivatei unui polinom.

Pentru fiecare monom prezent în lista polinomului se aplică derivata

(ax^b)’ = (a*b)x^(b-1)

- Integrate:

Clasa Integrate moștenește, la rândul ei UnaryOperation, folosind un singur parametru la intrare.

La fiecare aplelare, metoda *compute* integrează fiecare monom din ArrayList-ul prezent în Polinom. Se aplică operația de integrare după formula:

(ax^b) -> (a/b+1)x^(b+1)

- BinaryOperation:

Cu ajutorul acestei clase, se face legătura între operațiile care au nevoie de doi parametrii de intrare, de tip Polinom.

Pe lângă metoda compute, aceasta mai cuprinde două metode numite *compute2* (o metodă pentru efectuarea unei operații cu un monom și un polinom) și *getResult*, metodă care facilitează efectuarea operației de împărțire. Impărțirea are nevoie de doi parametrii la intrare, însă returnează doi parametrii: cât și rest.

Următoarele 4 clase moștenesc clasa BinaryOperation

- Add:

Clasa face posibilă efectuarea operației de adunare, printr-un algoritm simplu similar cu cel de inserare a monoamelor în listă.

- Subtract:

Prin propria implementare a metodei *compute,* clasa Subtract realizează operația de scădere între 2 polinoame, sau polinom și monom.

- Multiply:

Aceasta este una dintre clasele în care au trebuit testate situații speciale, precum înmulțirea cu 0. Prin verificarea listei unuia dintre operanzi, ne dăm seama daca acesta este polinomul nul și se va returna rezultatul 0.

Algoritmul de implementare este unul simplu, parcurgând fiecare listă în parte și înmulțirea element cu element a monoamelor.

- Division:

Este clasa care a ridicat cele mai multe probleme, algoritmul având o complexitate ridicată și folosind două dintre operațiile deja implementate (înmulțirea și scăderea).

Pentru realizarea împărțirii a fost implementată metoda matematică pentru obținerea câtului și restului. Codul tratează corect fiecare situație, precum grad(P1)<grad(P2), returnând câtul 0 și restul P1, dar și alte situații ce pot apărea pe parcurs.

3.Scenarii

Titlu: Alegere și calcul operații

3.1 Sumar

Trebuie să stabilim de ce date avem nevoie la intrare, cum le obținem pentru a le putea utiliza ușor și manipula prin metodele implementate în structura proiectului. Avem nevoie de o structură simplă pentru stocare și destul de accesibilă astfel încât metodele de calcul să fie optime ca spațiu de stocare, înțelegere și utilizare în interfață.

Pentru fiecare din operații avem nevoie, pentru fiecare monom, de coeficientul și puterea acestuia, datele fiind preluate de la utilizator. Pentru aceasta, am ales stocarea într-un ArrayList de monoame, fiind o soluție accesibilă. Pentru a facilita stocarea coeficienților reali după efectuarea unor operații precum *integrarea* sau *împărțirea,* aceștia vor fi stocați într-o structura de tip *double* în toate polinoamele.

După achiziția datelor de la utilizator, fiecare String va fi procesat prin următorul algoritm, prezent în clasa GraphicInterface. Numele metodei este *buildPolynomial(String s):*

Prin structura *try { } catch {}* se încearcă obținerea unei structuri de tipul căutat (ax^b), iar în caz contrar se va afișa în câmpul de rezultat mesajul: „Polinom incorect!”.

In interiorul *try { }* regăsim pașii de căutare a structurii de monom:

```
polin = polin.replace("-", "+-");
String token[] = polin.split("\\+");
```

Prin acest pas, șirul de la intrare este separat în bucățele, limita fiecărei părți fiind semnul care o succede.

```
if (!(monom.equals("")))
```

Se verifică dacă String-ul primit nu este nul. In caz contrar, polinomul nu va fi creat. In schimb, dacă stringul conține elemente, algoritmul continuă cu verificările succesive.

Se va testa dacă „monomul” începe cu ‘x’. In acest caz, coeficientul va fi setat la valoarea 1 și urmează să verificăm dacă elementul are sau nu un exponent. Se va testa caracterul de dupa x și se verifică dacă este ‘^’. Dacă aceasta se confirmă, vom continua cu citirea exponentului.

In cazul în care șirul nu are ca prim element pe 'x', se încearcă citirea coeficientului, până se ajunge la caracterul 'x' sau la finalul șirului. In continuare, se va face aceeași testare pentru existența exponentului.

La finalul algoritmului, având coeficientul și exponentul stocate în două șiruri, acestea vor fi transmise metodei *addMonom:*

```
builtP.addMonom(new Monom(Double.parseDouble(parametriiMonom[0]),
                          Integer.parseInt(parametriiMonom[1])));
```

După construirea polinomului pe structura dată, urmează selectarea operației. De la unul dintre butoanele interfeței se va activa o ramură în interiorul metodei *actionPerformed*. Fiecare buton are asignat un șir pentru a recunoaște o acțiune asupra acestuia. Prin intermediul codului *buton.equals(String s)* vom testa fiecare buton în parte printr-o succesiune de if-uri. In corpul acestora regăsim apelurile de operații și printarea rezultatelor în interfața grafică.

```
if (button.equals("Add")) {
// Aduna polinom1 + polinom2
BinaryOperation binary1 = new Add();
textField3.setText(binary1.compute(p1, p2).toString());
}
```

In continuare, se poate introduce o intrare schimbată sau efectuarea altor operații pe aceleași polinoame de intrare.

4.Cazuri de utilizare

Deși proiectul nu prezintă o complexitate majoră, acesta ar putea fi folosit în procesul de învățare, în școli, pentru testarea rezultatelor operațiilor, în momentul în care se rezolvă aplicații ale operațiilor pe polinoame. Operațiile de bază ar fi foarte ușor de verificat în acest mod.

O altă alternativă ar putea fi testarea digitalizată a elevilor, prin mici modificări aduse proiectului și instruirea utilizatorilor în introducerea rezultatului. Acesta ar fi calculat în interiorul soft-ului și verificat cu rezultatul introdus într-o căsuță de text.

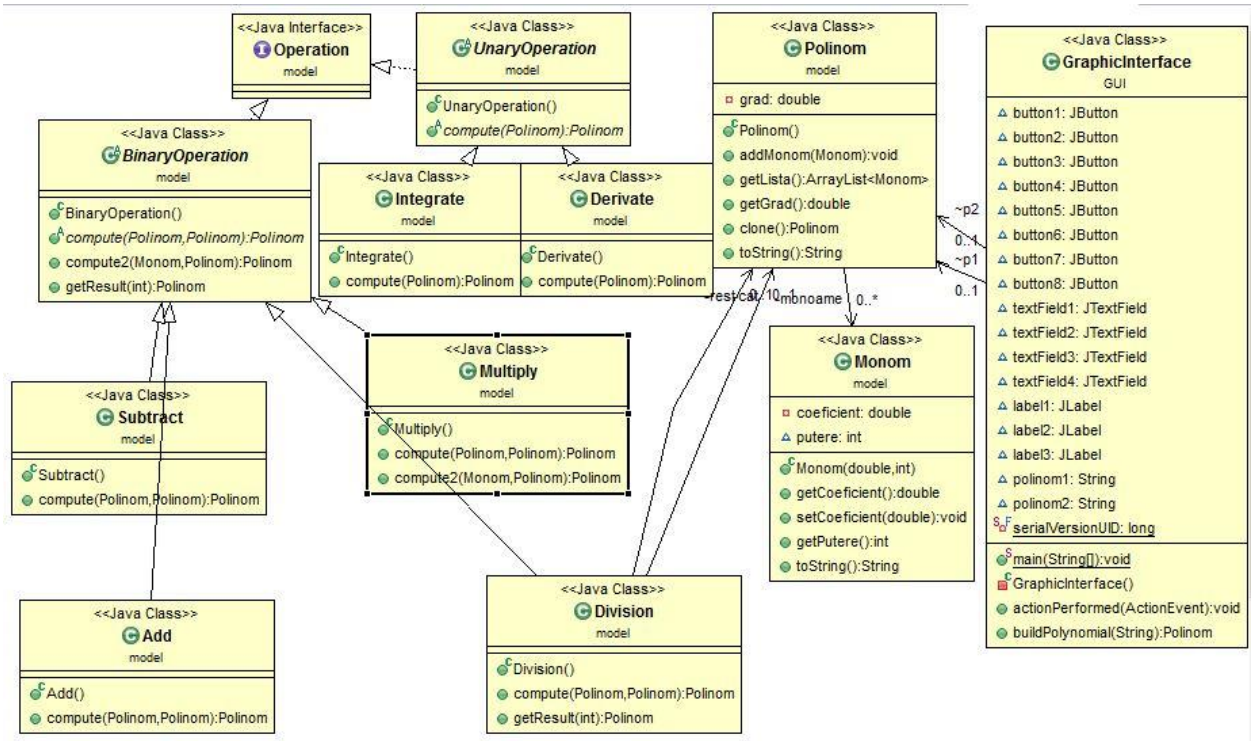
In construcția proiectului s-a urmărit realizarea unei structuri simple, dar cu multe funcționalități, care să îndeplinească exact rolul pentru care a fost creat, calculul operațiilor de bază.

5. Proiectare (diagrame UML, proiectare clase, structuri de date, interfete, relatii, pachete, interfata utilizator)

5.1 Diagrame UML

Diagrama UML reprezintă cel mai bun mod de a reprezenta ierarhia de clase în cazul structurilor complexe. Modelarea conceptuală este procesul principal de indentificare a conceptelor importante în construcție. In tehnica OOP, modelarea se realizează prin diagrama claselor.

Diagrama pentru proiectul de față este prezentată în figura de mai jos:



O clasă se reprezintă printr-o căsuţă împărţită în trei. În partea de sus este notat numele clasei, în partea mediană atributele, iar în partea de jos operaţiile sale.

Intre clasele GraphicInterface și Polinom există o relație de asociere. Această asocire dintre 2 clase se datorează faptului că putem naviga de la obiectele unei clase la obiectele celeilalte clase. Clasa GraphicInterface conține două instranțe ale clasei Polinom, cu ajutorul căreia apelam funcțiile de calcul operații în ActionListenerii butoanelor. Săgeata este orientarea spre clasa Polinom, pentru că aceasta e independentă de GraphicalInterface.

5.2 Structuri de date

In implementarea proiectului s-au folosit următoarele structuri de date:

- String – pentru citirea polinoamelor din interfața grafică
- Double – pentru păstrarea coeficienților întregi și reali ai polinoamelor
- int – pentru păstrarea exponenților fiecărui monom
- ArrayList<Monom> - pentru păstrarea unei liste de monoame, partea esențială în construcția polinomului.

5.3 Proiectare clase

Aplicația a fost modelată în două părți: partea de Interfață grafică și implementarea concretă a structurii Polinomului și a operațiilor.

Pachetul model conține următoarele clase:

- Polinom:
  Prin metoda *addMonom*, putem adăuga un monom la ArrayList-ul polinomului prin specificarea prarametrilor: coeficient și exponent.
  Alte două metode folosite sunt getLista și getGrad. Acestea facilitează obținerea listei de monoame din exterior, dar și a gradului polinomului pentru alte verificări.
  Metoda clone ajută la creerea unui alt obiect de tip Polinom cu aceeași listă de monoame.

In final, metoda toString() returnează un șir construit prin apelarea metodei toString din clasa Monom pentru fiecare element prezent în ArrayList-ul *monoame.*

- ComparatorPutere:
  Această clasă implementează interfața Comparator, pentru a putea suprascrie metoda compare pentru obiectele de tip Polinom. Acest lucru ne este de ajutor la sortarea listei de monoame dintr-un polinom.
- Add:
  Este o clasă care moștenește BinaryOperation și implementează operația de adunare a două monoame prin metoda compute.
- Derivate:
  Moștenește clasa UnaryOperation și prin metoda compute, realizează derivarea unui polinom primit ca parametru.
- Integrate:
  La fel ca Derivate, această clasă extinde UnaryOperation și prin suprascrierea metodei compute, realizează integrarea polinomului primit ca parametru.
- Division:
  Are una din cele mai complexe metode compute, deoarece a fost necesară implementarea algoritmului „long division" pentru polinoame și tratarea tututror excepțiilor ce pot interveni în acest proces.
  Totodată, prin metoda getResult, vom putea selecta unul dintre rezultatele împărțirii din clasa Division : câtul sau restul.
- Multiply:
  Suprascrierea metodei compute în această clasă a avut ca scop realizarea înmulțirii dintre două polinoame și returnarea ca rezulatat a unui nou polinom.
  La fle ca la Division, au trebuit tratate diferite situații speciale care puteau interveni în timpul efectuării operației (înmulțire cu scalar, cu 0, cu monom).
- Subtract:
  Această clasă realizează prin metoda compute scăderea a două polinoame. Situația specială este cazul în care avem două polinoame egale ca parametrii, iar rezultatul va trebui să fie 0.

6. Implementare și testare

Implementarea a început prin construcția claselor Monom și Polinom, apoi gândirea metodei de stocare a monoamelor într-o listă. S-a urmărit eficiența și ușurința la acces.

Tot odata în această clasă mai avem și metode de determinare a gradului și a coeficienților și o metodă de tipul String cu ajutorul căreia construim polinomul ca un String pentru a-l putea afișa în forma lui matematică.

Testarea a fost realizată prin introducerea din interfață a diferitelor situații care pot duce la erori în metodele apelate, astfel că s-a procedat la verificarea tuturor acestor erori și în funcție de rezultat, schimbarea sau nu a codului din metode. Câteva teste necesare în acest proces au fost:

- Adunarea unui polinom cu un monom
- Adunarea cu o constanta
- Adunarea cu 0
- Scăderea polinoamelor egale
- Scăderea pentru eliminarea unui element din polinomul inițial
- Inmulțirea cu 0
- Inmulțirea cu un scalar
- Inmulțirea cu un monom
- Impărțirea cu 0
- Impărțirea cu un polinom egal
- Impărțirea cu un scalar

7.Interfața. Mod de utilizare

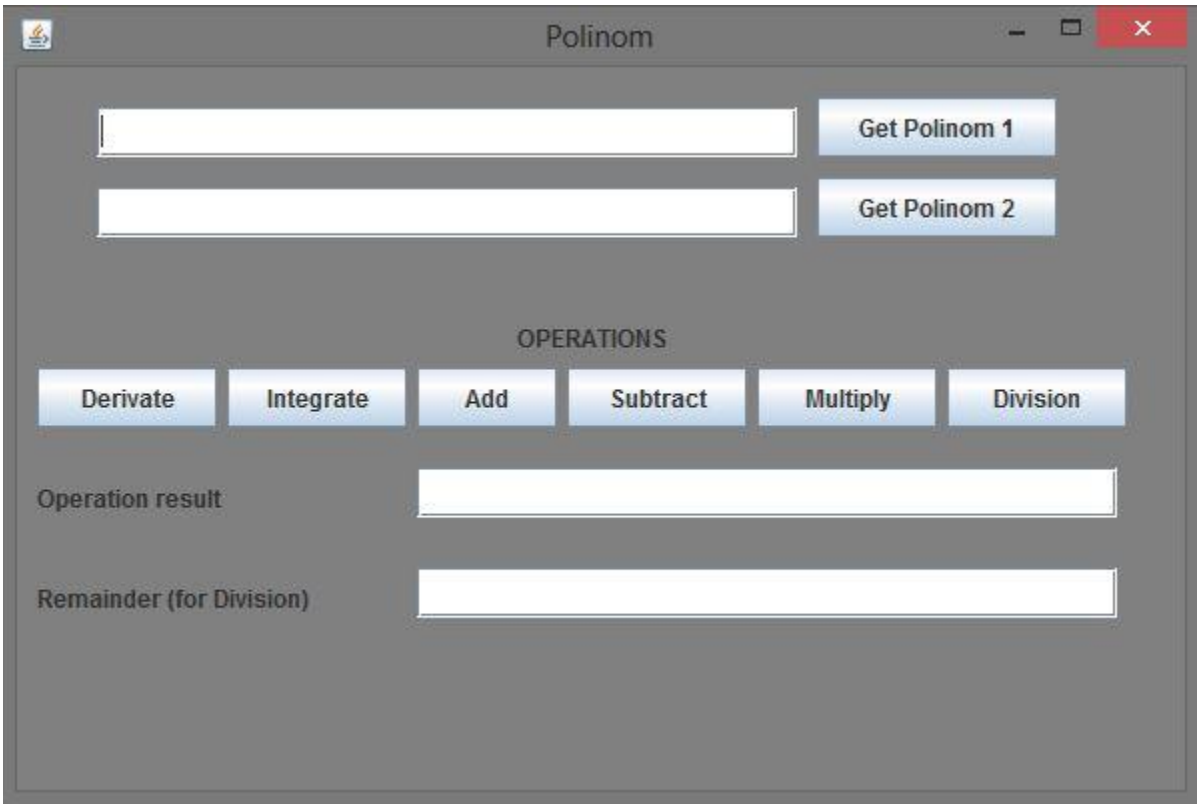
Interfata conține patru câmpuri, utilizate pentru introducerea polinoamelor asupra cărora se dorește să se aplice operațiile și pentru afișarea rezultatelor. A treia căsuță este folosită pentru a afisa rezultatele, iar cea de-a patra serveste doar în cazul în care operația aleasă este împartirea, caz în care ne sunt returnate două polinoame: câtul și restul, caseta a patra afișând-ul pe cel din urmă.

Utilizatorul va trebui să introducă polinoamele sub forma generală: "a1x^b1 + a2x^b2 + ..." . Dacă polinomul are gradul n, nu trebuie specificate toate monoamele mai mici decat acel n, doar cele care au

coeficientul diferit de zero. De asemenea, între coeficient și "x" nu este necesar să punem "*". Pentru operațiile de derivare și integrare, polinomul asupra căruia dorim să facem modificări se introduce în prima căsuță.

Odată ce datele de intrare au fost introduse, următorul pas este să alegem din partea de jos butonul corespunzător operației dorite, fiecare dintre aceste butoane fiind denumit cu numele operatiei pe care o execută. In cazul în care introducem datele intr-un format necorespunzator, se va afisa un mesaj de eroare in casuta numarul 3 (câmpul polinomului Rezultat). Dupa ce alegem butonul dorit, programul realizează operația și afișează în căsuțele 3 și 4 rezultatele obținute.

In cazul in care vrem sa introducem alte polinoame, va trebui să le înlocuim pur și simplu cu altele și să selectăm din nou „Get Polinom".



8. Rezultate

Prin realizarea acestui proiect, consider că am reusit să creez o modalitate simplă de gestionare și efectuare a operatiilor cu polinoame de orice grad. Interfața grafică asigură unui utilizatorilor neexperimentat posibilitatea de a efectua aceste operații, întrucât este un proiect ușor de folosit. Rezultatele sunt afișate într-o forma naturală și ușor de înțeles, iar butoanele aduc un plus de flexibilitate acestei aplicatii.

9. Concluzie

Acest proiect a fost un bun prilej de a-mi reaminti și aplica noțiunile studiate în cadrul cursului de Programare Orientată pe Obiect, utilizând paradigmele POO. De asemenea, mi-a folosit la familiarizarea cu limbajul Java, la exersarea descompunerii unei probleme complexe în subprobleme mai simple (o analiză a datelor de input și a celor de output). De asemenea, m-a ajutat la îmbunătățirea cunoștintelor despre componentele unei interfete si la înțelegerea modului de dezvoltare a unei aplicații ce respectă arhitectura Model View Controller.

10. Dezvoltări ulterioare

Se pot aduce o serie de dezvoltări ulterioare, care ar putea face proiectul mai flexibil, mai operațional și cu cât mai multe funcționalități. Câteva dintre acestea ar fi:

- Adăugarea de noi operații: calculul rădăcinii, desenarea graficului
- Imbunătățirea interfeței cu utilizatorul
- Eficientizarea algoritmilor utilizați pentru o gestiune mai bună a spațiului și timpului

11. Bibliografie

[1] http://www.purplemath.com/modules/polydiv2.htm
[2] http://stackoverflow.com/questions/3481828/how-to-split-a-string-in-java
[3] https://www.youtube.com/user/MrGIZDICH/videos