

**FACULTATEA DE AUTOMATICĂ ŞI CALCULATOARE**

**CATEDRA CALCULATOARE**

FUNDAMENTAL PROGRAMMING TECHNIQUES

ASSIGNMENT 1

Polynomial calculator

Student: Torzsa Gabriela

An studiu: II, semestrul II

Grupa: 30226

CUPRINS

* Obiectivul temei
* Analiza problemei, modelare, scenarii, cazuri de utilizare
* Proiectare
* Implementare
* Rezultate
* Concluzii
* Bibliografie
* Obiectivul temei

Obiectivul temei / proiectului a fost să proiectăm și să implementăm un calculator de polinoame care poate realiza diferite operații, cum ar fi: adunare, scădere, înmulțire, împărțire, derivare și integrare de polinoame. Având două polinoame introduse de utilizator și în funcție de operația aleasă de acesta, aplicația va returna/afișa pe ecran rezultatul operației efectuate. Interacțiunea dintre program și utilizator se realizează prin intermediul unei interfețe grafice (Graphical User Interface).

* Analiza problemei, modelare, scenarii, cazuri de utilizare
* Ce este un polinom?

Un polinom este un șir format din mai multe monoame, monoamele fiind formate dintr-un coeficient și o putere. Ex: x^2+3x+2x

Programul care va fi realizat, va lua datele introduse de utilizator și le va trece printr-o serie de transformări, respectiv operații, care ulterior vor genera rezultatul final.

* Modelare

Cerința problemei ne sugerează să folosim în definirea polinoamelor o colecție de monoame, sub forma unui ArrayList<>, deoarce pe măsura ce vom extrage monoamele din șirul de caractere dat de utilizator de la tastatură, le vom adăuga în lista care formează polinoamele. Pentru ca programul să funcționeze corect, utilizatorul va trebui să respecte un anumit format în scrierea polinoamelor, de exemplu, în locul scrierii clasice a unui polinom astfel ”x^2+3x+2”, acesta va trebui defapt să rescrie sub forma ” 1x^2+3x^1+2x^0”. Adică, vor trebui scrise inclusiv puterile de 1 și 0 a monoamelor, respectiv dacă coeficientul monomului este 1, acesta va trebui scris în fața monomului respectiv.

* Scenarii

Scenariile posibile care pot apărea sunt introducerea incorectă a datelor de intrare de către utilizator, adică scrierea polinomului în format greșit celui pe care îl acceptă programul. Astfel, la apăsarea oricărui buton care realizează o operație pe polinoame va genera un mesaj, care îl va atenționa pe utilizator că datele introduse nu sunt în conformitate cu cerințele programului, mesajul afișat pe ecran fiind ”Empty polynomials / wrong format!”. Astfel, ca programul să funcționeze corect, utilizatorul v trebui sa respecte formatul cerut de aplicație.

În cazul în care utilizatorul respectă formatul, programul va rula corect, iar rezultatul afișat pe ecran va fi cel dorit.

* Cazuri de utilizare

În funcție de butonul apăsat de utilizator, adică operația pe care acesta dorește să o efectueze, programul va afișa într-o zonă specială, destinată afișării rezultatului, rezultatul operației. În cazul în care utilizatorul dorește să efectueze derivare sau integrare de polinom, el trebuie să știe că polinomul care va fi integrat sau derivat va fi primul polinom (Polynomial P(x)) – acest fapt este specificat și pe butoanele aplicației. Când utilizatorul dorește să închidă aplicația, acest lucru se va realiza prin apăsarea butonului x din colțul ferestrei aplicației.

* Proiectare
  + Decizii de proiectare

Programul a fost gândit în așa fel încât să fie cât mai simplu structurat, fără a scrie clase cu prea multe linii de cod inutile, care ar fi dus la o citire a codului mai greoaie.

Astfel am ales să scriu doar 5 clase, fiecare având scop bine definit. Clasele sunt: Monomial, Polynomial, Regex, Controller, View si Main (rolul fiecărei dintre aceste clase îl voi detalia în secțiunea *Implementare*).

Pe scurt, clasa Monomial se ocupă de crearea de monoame, clasa Polynomial, care are rolul de Model în arhitectura MVC (Model View Controller), în care se află metodele pentru operațiile pe polinoame, clasa Regex, care împarte în grupuri string-ul introdus de utilizator în (coeficient)(x)(putere), clasa Controller, care se ocupă de acțiunile fiecărui buton în parte, clasa View în care am creat interfața grafică și clasa Main în care se realizează legătura dintre Model, View și Controller. Clasa Polynomial și clasa Monomial le-am adăugat în pachetul *model*, clasa Regex și clasa Controller se găsesc în pachetul *controller*, iar clasa View am creat-o în pachetul *view.* Clasa Main a rămas in pachetul *default.*

* Diagrama UML

A picture containing text, electronics

Description automatically generated

Sd. Ds. D. d d d d d d d d d. d d d d d d d d d d d dd d. dd d. d d d d d d d d d d d d d d d d d. d d d d d d d d d d

Sd. Ds. D. d d d d d d d d d. d d d d d d d d d d d dd d. dd d. d d d d d d d d d d d d d d d d d. d d d d d d d d d d

Sd. Ds. D. d d d d d d d d d. d d d d d d d d d d d dd d. dd d. d d d d d d d d d d d d d d d d d. d d d d d d d d d d

Sd. Ds. D. d d d d d d d d d. d d d d d d d d d d d dd d. dd d. d d d d d d d d d d d d d d d d d. d d d d d d d d d d

Sd. Ds. D. d d d d d d d d d. d d d d d d d d d d d dd d. dd d. d d d d d d d d d d d d d d d d d. d d d d d d d d d d

* Interfața grafică (GUI)

Interfața grafică conține:

* + - * Label-uri pentru a ști ce trebuie completat în TextField-uri, respectiv eticheta unde se va afișa rezultatul
      * TextField-uri în care se vor introduce datele de intrare, adică cele două polinoame, sub forma de string și repsectând formatul
      * TextArea unde se va afișa rezultatul operațiilor pe cele două polinoame
      * Butoane pentru fiecare operație: adunare, scădere, înmulțire, împărțire, derivare, integrare

Graphical user interface, application

Description automatically generated

* Implementare

Am implementat codul într-un mod cât mai simplu și structurat, astfel încât în momentul în care un alt programator se va uita peste program, acesta să-l poată citi cu ușurință și să-l înțeleagă.

Clasele proiectului:

1. Clasa Monomial

În clasa Monomial, am declarat atributele unui monom, coeficientul și puterea. Am scris un constructor fără parametri în care se instanțiază atributele cu valoarea 0, respectiv, un constructor cu doi parametri, coeficientul și puterea, care va instanția atributele cu valorile specifice, rezultate din transformarea stringurilor în monoame care vor forma un polinom.

Din cauza faptului că atributele au indentificator *private*, am scris gettere și settere pentru a putea le putea accesa din afara clasei.

Java code:

Text

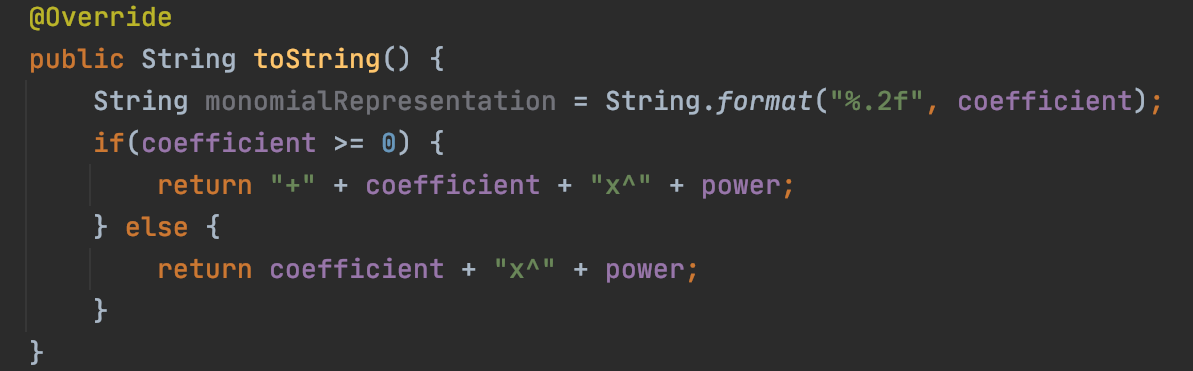
Description automatically generated

Metodele implementate în această clasă sunt: addition, subtraction, respectiv multiplication, care realizează adunarea, scăderea și înmulțirea a două monoame.

Text

Description automatically generatedJava code:

Pentru afișarea sub formă de string a monoamelor pe ecran, am suprascris funcția *toString()*, adăugând adnotația *@Override*

Java code:

1. Clasa Polynomial

Clasa Polynomial reprezintă defapt Modelul programului.

Clasa are ca singur atribut o colecție (de tipul List<>) de monoame, care reprezintă un polinom.

Reprezentând modelul arhitecturii MVC, în clasă am implementat operațiile efetuate pe cele două polinoame date ca intrări de către utilizator.

Descrierea operațiilor:

* + - Adunarea de polinoame

Există 2 cazuri de adunare:

* + - 1. Când indecșii de parcurgere a polinoamelor sunt ambii mai mici decât lungimea polinoamelor (numarul de monoame) se verifică puterile monoamelor curente. În cazul în care puterea monomului din polinomul P este mai mare decât puterea monomului din polinomul Q, atunci se adaugă în polinomul rezultat monomul din polinomul P. Dacă, puterea monomului din al doilea polinom este mai mare, atunci se adaugă acest monom la rezultat. Iar în cazul în care puterile celor două monoame sunt egale, se apelează funcția de adunare din clasa Monomial, iar monomul rezultat se adaugă în polinomul pentru rezultat.
      2. În cazul în care unul dintre polinoame a fost parcurs complet, iar celălalt mai are termeni, îi adăugăm la rezultat.
    - Scăderea de polinoame

Codul pentru scăderea polinoamelor este același cu cel al adunării lor, diferența făcând-o cazul în care monomul din al doilea polinom are puterea mai mare decat puterea monomului din primul polinom și când polinomul al doilea mai are termeni de parcurs, iar primul polinom fost parcurs complet. În acest caz, coeficientul se înmulțește cu valoarea -1.

* + - Înmulțirea polinoamelor

Parcurgând cu două bucle for, prima polinomul P, iar a doua polinomul Q, înmulțim toate monoamele din primul polinom cu fiecare monom din al doilea. Rezultatul obținut se memorează într-o variabilă auxiliară, iar la final adaugăm toate monoamele rezultate în polinomul dedicat rezultatului.

* + - Derivarea polinoamelor

Aplicația a fost implementată calculând doar derivarea primului polinom (lucru specificat pe buton). Algoritmul ia fiecare putere a fiecărui monom, o înmulțește cu coeficientul, iar apoi setează coeficientul monomului respectiv cu valoarea nouă calculată. Puterea monomului va scădea cu 1.

* + - Integrarea polinoamelor

Asemenea derivării, operația de integrare se va aplica doar primului polinom. Am declarat două variabile temporare, una pentru coeficientul rezultat, iar alta pentru puterea nouă rezultată. Valoarea noului coeficientului va fi valoarea puterii monomului împărțită cu valoarea coeficientului actual, iar puterea monomului va crește cu 1. Noul monom va fi adăugat polinomului dedicat rezultatului

* + - Împărțirea nu este implementată

1. Clasa Regex

Această clasă realizează spargerea și gruparea șirului de caractere dat de utilizator ca input și transformă caracterele în monoame. Adică primul grup al regular expression-ului, va reprezenta coeficientul monomului. Conținutul grupului va fi transformat în double și valoarea va fi atribuită unei variabile reprezentant coeficientul, iar al treilea grup va fi transformat intr-o valoare de tip int și atribuită unei variabile reprezentând puterea monomului. Al doilea grup reprezintă x-ul, adică litera utilizată pentru monom. În final, aceste valori vor defini un monom, pe care îl vom adauga polinomului.

Java code:

Text

Description automatically generated

Regular expression

1. Clasa Controller

Clasa Controller realizează legatura dintre View si Model (clasa Polynomial). Clasa implementează ascultătorii (listenerii) butoanelor, adică ce se va întâmpla în momentul în care se apasă un buton. In fiecare clasă, instanțiem două polinoame, respectiv unul (pentru derivare și integrare), în care stocăm polinoamele generate de clasa Regex. În final, rezultatul operației efectuate se va afișa în zona dedicată rezultatului în interfață.

Text

Description automatically generated

Text

Description automatically generated

* + - Dacă utilizatorul nu introduce nimic sau formatul nu este corespunzător, atunci se va afișa un mesaj de eroare.



Metodă în clasa View

1. Clasa View

Text

Description automatically generatedÎn clasă, am implementat interfața grafică, utilizând pachetul *swing*. Am declarat componentele și le-am inițializat.

Am creat 3 panel-uri (unul pentru zona de input, unul pentru zona de butoane, unul pentru zona de afișare rezultat) + panel-ul principal în care le vom adăuga la final pe toate. Am dat n titlu ferestrei, am ales ca metodă de închidere a aplicației să fie *EXIT\_ON\_CLOSE*.

Text

Description automatically generated

Text

Description automatically generatedUltimul pas în scrierea clasei View este adăugarea listenerilor pentru butoane.

1. Clasa Main

În clasă am creat un obiect instanța clasei View, un obiect instanța clasei Polynomial (Model) și un obiect instanța clasei Controller. Pentru ca interfața grafică să fie vizibilă am folosit metoda *setVisible(true)*.

Text

Description automatically generated

* Rezultate

Text

Description automatically generatedAm testat corectitudinea operațiilor folosind o clasă de test folosind JUnit. Pentru fiecare operație, am implementat câte o metodă care verifică prin *AssertEquals* egalitatea dintre parametri, importând în prealabil pachetul

Java code pentru metodele de testare pe operații:

Text

Description automatically generated

În fiecare metodă (adnotată cu *@Test*), se instanțiază două polinoame (adunare, scădere, inmulțire), respectiv un polinom (derivare, integrare) și o instață de Regex (pentru spargerea și transformarea stringului sub formă de polinom). În final, verificăm cu *AssertEquals* dacă rezultatul din polinomul P (primul) în format *toString* este sau nu egal cu string-ul dat ca al doilea parametru al metodei.

Rularea testelor compilat cu succes, astfel punându-se o bifă în dreptul metodei.

Graphical user interface, text, application

Description automatically generated

1. Concluzii

Aplicația este una destul de simplă, care realizează adunarea, scăderea, înmulțirea, derivarea, integrarea a două polinoame. Interfața grafică este ușor de înțeles, iar interacțiunea utilizatorului cu programul este una *user friendly*. Pentru îmbunătățirea aplicației se mai pot implementa și alte operații, cum ar fi: împărțirea, derivare multiplă, calcul de limite și integrale.

1. Bibliografie
   * + Laboratoare și cursuri POO, an II, sem I
     + <https://stackoverflow.com>