Facultatea de Automatica si Calculatoare

Departamentul de Calculatoare si Tehnologia Informatiei

**Polynomial Calculator**

Student : Ștef Paula-Elena

Grupa : 30221

An academic : 2020-2021

Cuprins :

1. Obiectivul temei…………………………………………………………………3
2. Analiza problemei, modelare, scenarii, cazuri de utilizare….3
3. Proiectare………………………………………………………………………….5
4. Implementare……………………………………………………………………6
5. Rezultate…………………………………………………………………………..10
6. Concluzii……………………………………………………………………………
7. Bibliografie……………………………………………………………………….

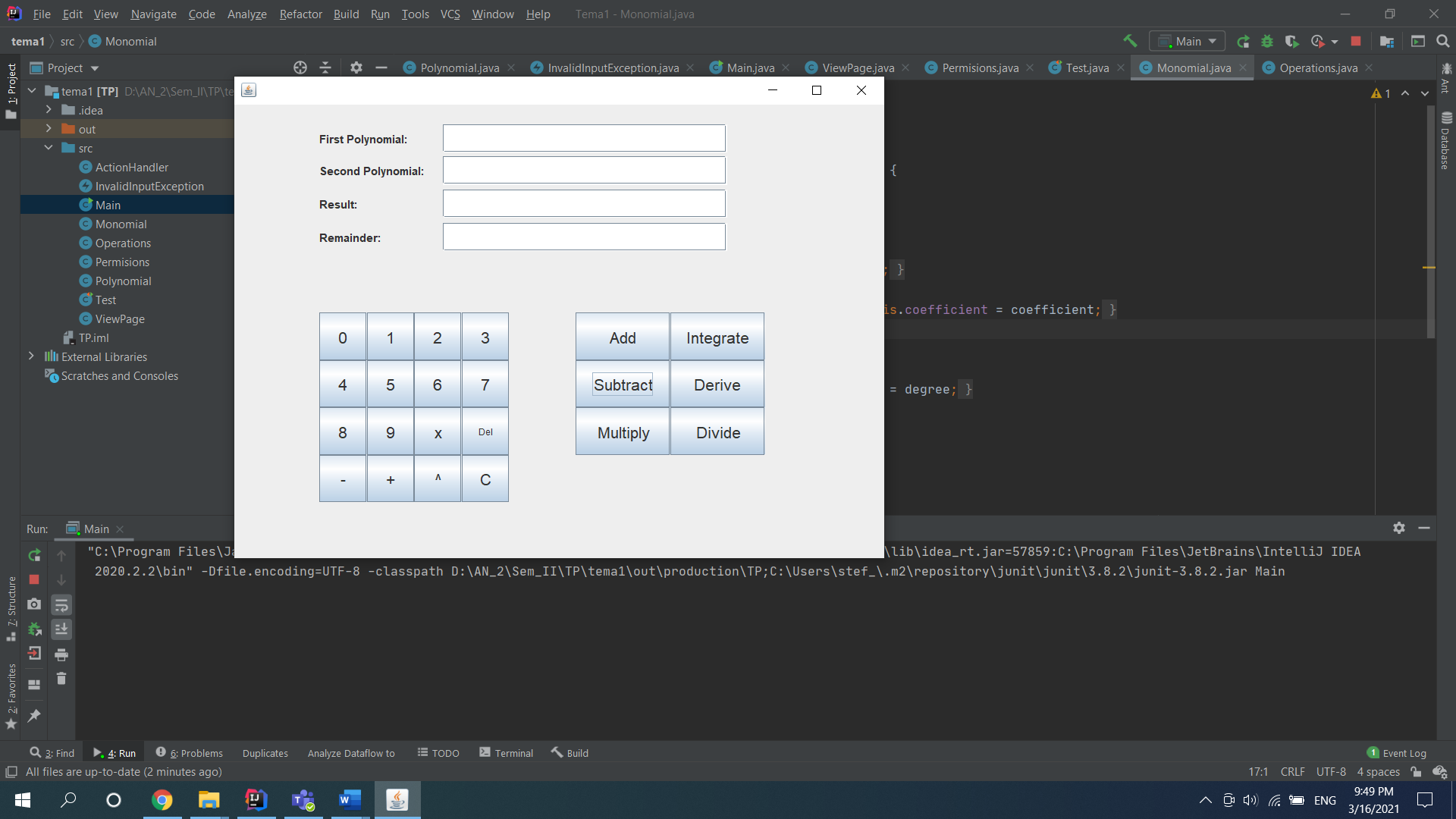
1. **Obiectivul temei**

Se doreste proiectarea si implementarea unui calculator de polinoame, utilizand un limbaj de programare orientat pe obiect si care prezinta o interfata grafica prin intermediul careia utilizatorul va putea insera polinoame cu coeficienti intregi si va alege operatia pe care doreste sa o efectueze, fiind afisat rezultatul corespunzator.

Operatiile pe care utilizatorul le poate aplica asupra polinoamelor sunt:

* Adunarea
* Scaderea
* Inmultirea
* Impartirea
* Integrarea
* Derivarea

1. **Analiza problemei, modelare, scenarii, cazuri de utilizare**



Dupa cum se poate observa si in imagine, avem 2 campuri pentru inserarea polinoamelor asupra carora dorim sa efectuam operatii: *First Polynoomial*, respectiv *Second Polynomial* (pentru operatiile de integrare si derivare se va folosi campul First Polynomial, fiind nevoie de un singur polinom ) . In aceste campuri se vor introduce polinoamele sub o forma generalizata de ex:

P(x) = ±ax^b ± cx^d ± … ± ex ± f

, unde a,c,..,e,f reprezinta coeficientii intregi ai polinomului si pot lipsi in cazul in care dormi sa avem coeficientul 1 (ex. x^2), iar ^b,^d.. reprezinta gradele (numere naturale) polinomului, care din nou pot lipsi daca dormi sa avem gradul 1 sau putem avea doar coeficientul daca dorim gradul 0(ex. x+1). Polinoamele pot fi scrise in orice ordine a gradelor, acestea putand fii si repetate, atata timp cat combinatiile respecta tiparul ±ax^b.

Pentru a insera polinoamele se pot folosi butoanele din partea stanga care au rolul de a insera in casutele asignate polinoamelor numerele sau semnele corespunzatoare (de ex. numere: 1,2,6..; semne : +,-,^ , sau x ). De asemenea avem un buton pentru a sterge cate un element in cazul greselilor (Delete) si de asemenea un buton “Clear” (C) prin care se elibereaza toate campurile.

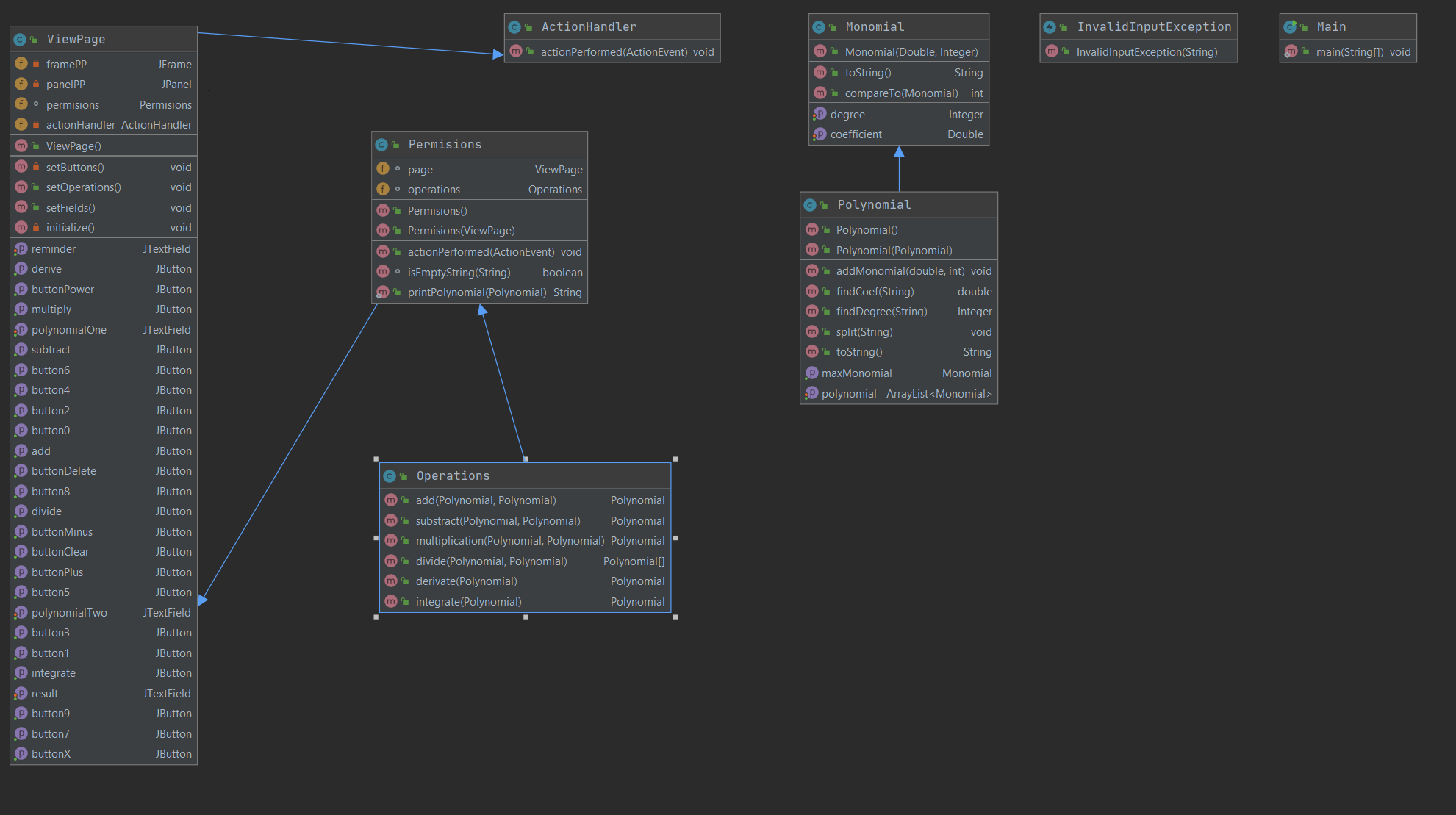
Odata ce polinoamele sunt inserate corespunzator, se va putea alege operatia dorita prin butoanele din partea dreapta cu nume sugestive, tinandu-se cont ca pentru operatiile de derivare si integrare si va folosi primul camp (First Polynomial) si de asemenea pentru operatia de impartire este necesar ca a doilea polinom sa fie corespunzator ( sa existe sis a nu fie 0) , in caz contrar operatia nefiind realizata si se va afisa un mesaj de eroare.

Rezultatul va fi afisata in campul *Result* tot sub forma unui polinom, in ordinea gradelor.In cazul operatiei de impartire este necesar inca un camp *Remainder* in cazul in care impartirea are un rest(si aici rezultatul e afisat sub forma unui polinom).

In cazul inserarii unor polinoame ce nu sunt in conformitate cu forma generala a unui polinom se va afisa un mesaj “Invalid Input”, si nu se vor efectua operatiile.

1. **Proiectare**

Am realizat aceasta aplicatie utilizand abstractizarea datelor, conform carora s-au format clasele : Polynnomial , Monomial, Operations, ViewPage, Permissions, ActionHandler, InvalidInputException, Main, Test in intermediul lor fiind utilizate structuri de date de tip ArrayList, extensii JFrame si TestCase, interfete ActionListener si Comparable apartinand pachetelor java.util.\*, java.awt.\*, java.swing\* si junit.framework.\*, combinate oferind aplicatiei caracteristica de user-friendly, aceasta fiind usoara de inteles si simpla de folosit.



1. **Implementare**

* Clasa *Monomial* contine doua campuri private “coefficient” si “degree” care reprezinta coeficientul fiecarui monom, numar real (l-am ales de tip Double deoarece desi coeficientii de intrare sunt numere intregi, in urma operatiilor de impartire/derivare/integrare putem obtine si coeficienti reali ), respectiv gradul fiecarui coefficient, numar natural si un constructor (pe care l-amsuprascris0 care are ca parametrii un coefficient si un grad pentru a simplifica crearea monoamelor.

De asemenea, pentru cele doua variabile avem getter-i si setter-i pentru a utiliza si accesa mai usor valorile respective si o metoda toString folosita pentru verificari pe parcursul implementarii.De asemenea am suprascris si metoda compareTo, din interfata Comparable, pe care o folosim pentru a compara gradele monoamele si o utilizam cand dormi sa “ordonam” polinoamele in functie de grade.

* Clasa *Polynomial* contine un camp privat de tip ArrayList<Monomial> “polynomial” in care retinem toate monoamele ce apartin unui polinom, doi constructori : unul fara parametrii care doar initializeaza o noua lista de monoame si unul care primeste ca parametru un polinom ce este parcurs monom cu monom si il copiaza. Avem metoda “addMonomial” prin intermediul careia adaugan cate un monom in lista polynomial, avand grija sa verificam daca nu avem deja un monom cu acelasi grad (daca avem vom aduna doar ceoficientii acestora). Dupa care avem 3 metode care impreuna ajuta la trecerea din string-ul primit de la utilizator, care are forma generala a unui polinom, in lista de monoame necesara pentru efectuarea operatiilor. Ca si metoda principala avem “split” in care impartim stringurile in substringuri de forma “±ax^b”: mai intai adaugam semnul “+” in fata oricarui coefficient ce are semnul “-“(e negativ) deoarece “spargerea” polinomului se face dupa semnul “//+” dar avem nevoie sa pastram semnul de “–“, dupa care pentru fiecare substring obtinut se apeleza metodele de findCoeff si findDegree. La nivelul acestora se cauta ceoficientii si gradele monoamelor, tinandu-se cont de cazurile speciale ( coeficientul e 1/ gradul e 0 etc ). De asemenea mai avem o metoda getMaxMonomial prin intermediul careia se returneaza monomul cu gradul cel mai mare si o metoda toString tot pentru afisare.
* Clasa Operations la nivelul careia nu avem decat implementarea operatiilor pe care dorim sa le aplicam asupra polinoamelor inserate, fiecare returnand un nou polinom (sau doua in cazul impartirii) care reprezinta rezultatul. Metoda “add” realizeaza adunarea a doua polinoame (primate ca argumente) parcurgand primul polinom(care e sub forma de lista), iar pentru fiecare monom din primul polinom se aduna coeficientul intr-o variabila si se parcurge si al doilea polinom.Daca in cel de-al doilea polinom se gaseste un monom cu grad corespunzator monomului din primul polinom atunci in variabila definite vom aduna si coeficientul acestuia, vom elimina dintr-o copie monomul respectiv celui de al doilea polinom si vom iesi din al doilea for; in ambele cazuri se adauga un nou monom la rezultat cu variabila pe post de coefficient si gradul corespunzator monomului actual din primul polinom. Dupa se mai face o parcurgere a copiei polinomului doi pentru a nu se pierde monoame si se sorteaza resultatul in functie de grade. Se merge pe acelasi principu si in cazul scaderii in metoda “subtract” doar ca vom scadea coeficientii in loc sa ii adunam.In cazul metodei pentru inmultire “multiplication” se parcurge primul polinom si fiecare monom se inmulteste cu fiecare monom cu cel de-al doilea polinom ( se inmultesc coeficientii si se aduna gradele ) rezultatul fiind un polinom care se va aduna pentru fiecare monom din primul polinom la resultatul final.

Functia de impartire. Metoda “divide” efectueaza impartirea celor doua polinoame , si are ca rezultat 2 polinoame : catul si restul. Aici se face pe rand impartirea monomului cu cel mai mare grad din primul polinom la monomul cu cel mai mare grad din cel de-al doilea polinom ( impartind coeficientii si scazand gradele) rezultand un monom care este inmultit cu al doilea polinom si scazut din primul polinom pentru a obtine restul. Acest lucru se repeat pana cand gradul maxim al restului obtinut este mai mic decat gradul maxim al celui de-al doilea polinom. Metoda “derivate” deriveaza polinomul dupa formula de derivare a functiei putere, si anume se inmulteste coeficientul fiecarui monom cu gradul acestuia si ii descrestem gradul cu 1. Aceasta metoda are ca arguent doar un polinom, asemenea metodei de integrare, “integrate”. Aici coeficientul fiecarui monom este impartit la gradul acestuia + 1 si ii crestem gradul cu 1 obtinand un nou polinom ca rezultat.

* In clasa *ViewPage* avem implementata interfata grafica prin intermediul careia utilizatorul poate folosi calculatorul. In cadrul acesteia avem campuri private care reprezinta butoane si text field-uri ce sunt asezate pe un panel la diferite coordonate, avand anumite dimensiuni si mesaje representative scopului fiecaruia.

De asemenea avem cateva metode in cadrul carora am initializat si setat toate campurile folosite pe categorii : in metoda setButtons am creat butoanele pe care le folosim pentru a insera polinoamele in text field-uri, fiecare efectuand metoda addActionListener prin intermediul careia i se acorda “rolul” si de asemenea l-eam etichetat corespunzator : butoanele 0,1… 9 care scriu cifrele, butonul de delte care sterge cate un character pe rand, butonul de Clear care goleste text field-urile etc.;

dupa care in metoda setOperations avem aceleasi setari a unor butoane dar acestea au rolul de a selecta operatia pe care utilizatorul vrea sa o efectueze asupra polinoamelor : “Add” pentru adunare, “ Subtract “ pentru scadere, “Multiply” pentru inmultire, “Divide” pentru impartire, “ Integrate” pentru integrare si “Derive” pentru derivare; in metoda setFields avem cele 4 text field-uri : 2 pentru a scrie cele doua polinoame asupra carora se realizeaza operatiile si 2 pentru afisarea rezultatelor ( sunt 2 deoarece pentru impartire vom avea 2 polinoame ca rezultat ). Toate aceste metode sunt apelate in metoda initialize, dupa ce mi-am creat si setat frame-ul si panel-ul pe care le voi adauga.La randul ei aceasta metoda este apelata in constructorul vid al clasei. In cadrul acestei clase mai avem 2 campuri private: permissions si actionHandler prin intermediul carora se fac legaturi cu clasele care controleaza evenimentele asupra campurilor mentionate mai sus(prin apelarea pentru fiecare a metodei addActionListener care are ca paramtru unul dintre aceste campuri). Mai avem de asemenea getter-ii si setter-ii campurilor.

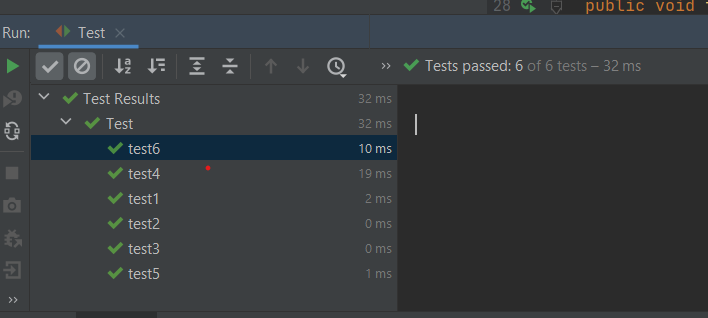
* In clasa *Permissions* avem un camp private de tip ViewPage prin care se face lagatura cu interfata grafica, si un camp de tip operations pentru a putea realiza

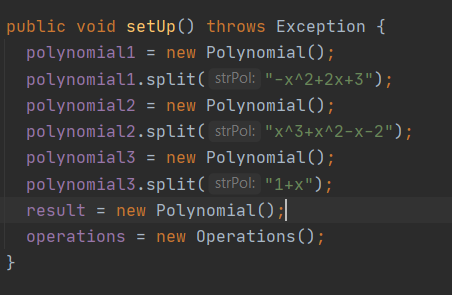
operatiile asupra polinoamelor.Avem un constructor prin care se initializeaza campul view si o metoda suprascrisa “actionPerformed” care pe baza sirurilor de caractere care reprezinta polinoamele preluate din text field-urile din page, genereaza lista monoamelor pentru cele doua polinoame ( sau unul daca este cazul ), verifica daca s-au introdus polinoame corespunzatoare, initializeaza polinoamele rezultat, testeaza daca datele introduse sunt eronate si afiseaza mesajul aferent in caz de adevar.Dupa care verifica pe rand fiecare buton din interfata grafica pentru a gasi sursa care a declansat aceasta metoda si pentru a efectua operatia dorita de utilizator, genereaza polinomul/polinoamele rezultat si il/le transpunerea in sir de caractere, dupa care le afiseaza. Aceasta transpunere din polinom in sir de caractere este realizata prin intermediul metodei statice din aceasta clasa “printPolynomial” care primeste un polinom,il parcurge monom cu monom si adauga la un string in functie de coeficienti si grade caracterul/caracterele corespunzatoare pentru a crea o forma generala a polinomului ca si string si il returneaza.Aceasta clasa este folosita pentru butoanele de operatii (cele din dreapta) si pentru butonul de delete.

* In clasa *ActionHandler* doar este suprascrisa de asemenea metoda de “actionPerformed” astfel incat in urma apasarii unui buton se va prelua textul butonului respective, se va gasi field-ul unde se doreste sa se scrie si si va aduga la textul din field, textul butonului apasat, exceptie facand butonul de delete care doar sterge ultimul character din string. Aceasta clasa este utilizata pentru butoanele din stanga prin intermediul carora se scriu polinoamele dorite. De asemenea pentru ca butoanele sa functioneze correct am “dezactivat” focusarea acestora pentru a se putea stii unde se doreste sa se scrie.
* Clasa InvalidInputException este o clasa care mosteneste din clasa Exception si in care apelam doar super-ul acesteia si o folosim pentru a intercepta diferite greseli de scriere a polinoamelor introduce.
* Clasa Main are o singura metoda, si anume cea ‘main’ cu rolul de a pune in miscare/rula/executa aceasta aplicatie pentru a fi utilizata
* Clasa Test are rolul de a testa aplicatia dezvoltata si de a demonstra corectitudinea operatiilor asupra polinoamelor. Contine trei campuri private: (tip Operations) ‘operation’ cu ajutorul careia se va selecta operatia menita testarii. S-a utilizat un mod de testare fix, in care s-au introdus polinoame asupra carora s-au efectuat deja operatiile, astfel cunoscandu-se in prealabil rezultatele corecte, urmand numai a fi comparate cu cele returnate de program. Defineste metode precum ‘setUp’ pentru a reincarca inaintea fiecarei testari operatia si polinoamele initiale, ‘tearDown’ pentru a “distruge” referintele campurilor cu scopul de a evita evenimentul de suprapunere a datelor si “test1”,”test2”.. pentru testarea efectiva a fiecarei operatii.

1. **Rezultate**

Pentru testarea operatiilor pe polinoame s-a utilizat framework-ul JUnit cu metoda de testare fixa (rezultate cunoscute inainte de efectuarea operatiilor), ca date de testare au fost introduse trei polinoame(au fost folosite in mai multe grupari) generate astfel incat sa acopere pe cat posibiltoate cazurile speciale cat si cel general pentru o buna verificare. In urma acestor teste se poate constata si trasa concluzia care justifica faptul ca aceste metode prezinta atat corectitudine de sintaxa cat si de implementare/rulare iar utilizarea aplicatiei este pe cat se poate de sigura.





1. **Concluzii**

Din aceasta tema s-a putut invata abilitatea de manipulare eficienta a datelor, de a le structura intr-un mod cat mai adecvat si usor de inteles si implementarea lor intr-un mod eficient atat din punctul de vedere al memoriei utilizate cat si din cel ai timpului necesar de rulare, calculare si furnizare a unor date corecte din punct de vedere matematic. Dezvoltarile ulterioare ale acestei aplicatii pot fi adaugarea a mai multor operatii, de exemplu gasirea radacinilor polinoamelor, calcularea transformatei Laplace etc.

1. **Bibliografie**

* notitele din cursurile si laboratoarele materiei ‘Tehnici de programare fundamentale’
* <https://stackoverflow.com/questions/15754374/how-to-use-buttons-to-type-in-a-text-field>
* <https://stackoverflow.com/questions/32730600/dividing-polynomials-in-java>
* <https://stackoverflow.com/questions/950409/how-to-parse-this-string-in-java>
* <https://www.geeksforgeeks.org/split-string-java-examples/>
* <https://docs.oracle.com/javase/tutorial/uiswing/>