Calculator de Polinoame

Proiect realizat in cadrul

laboratorului de

Tehnici Fundamentale De Programare

CUPRINS:

1. Obiectivul temei
2. Analiza problemei. Cazuri de utilizare
3. Proiectare
4. Implementare
5. Rezultate
6. Concluzii
7. Bibliografie

**Obiectivul temei:**

Dezvoltarea si implementarea unui calculator de polinoame cu o interfata grafica dedicata prin care utilizatorul poate introduce polinoame, selecta operatia matematica ce urmeaza sa fie efectuata si poate vizualiza rezultatul. Calculatorul trebuie sa fie eficient, util si intuitiv pentru ca utilizatorul sa poata avea parte de o experienta de utilizare cat mai fericita.

**Obiective secundare:**

Analiza temei si identificarea cerintelor reprezinta prima etapa in atingerea oricarui obiectiv, dar cu atat mai mult in ceea ce priveste dezvoltarea unei aplicatii software. In acest sens, tema proiectului este un calculator cu ajutorul caruia pot fi efectuate operatii cu polinoame, iar cerintele sunt in directa legatura cu operatiile pe care ne dorim sa le putem efectua asupra polinoamelor introduse. (capitolul II – analiza problemei)

Dezvoltarea calculatorului de polinoame este urmatorul pas in cadrul acestui proiect. Odata identificate cerintele impuse de operatiile dorite si de interfata grafica de care utilizatorul trebuie sa se poata folosi, urmeaza proiectarea claselor si structurilor de date ce urmeaza a fi folosite pentru a duce la bun sfarsit proiectul si a livra aplicatia promisa utilizatorului. (capitolul III – proiectare)

Implementarea calculatorului de polinoame este al treilea obiectiv pe drumul ales pentru a completa proiectul. Aici, deciziile luate in etapa de dezvoltare sunt trecute in cod propriu-zis si aplicatia incepe sa capete contur. In cadrul acestui pas exista posibilitatea de a se reveni la pasul anterior pentru mici modificari legate de aspecte ce nu au fost prevazute de dinainte de a incepe etapa de implementare a proiectului. (capitolul IV – implementare)

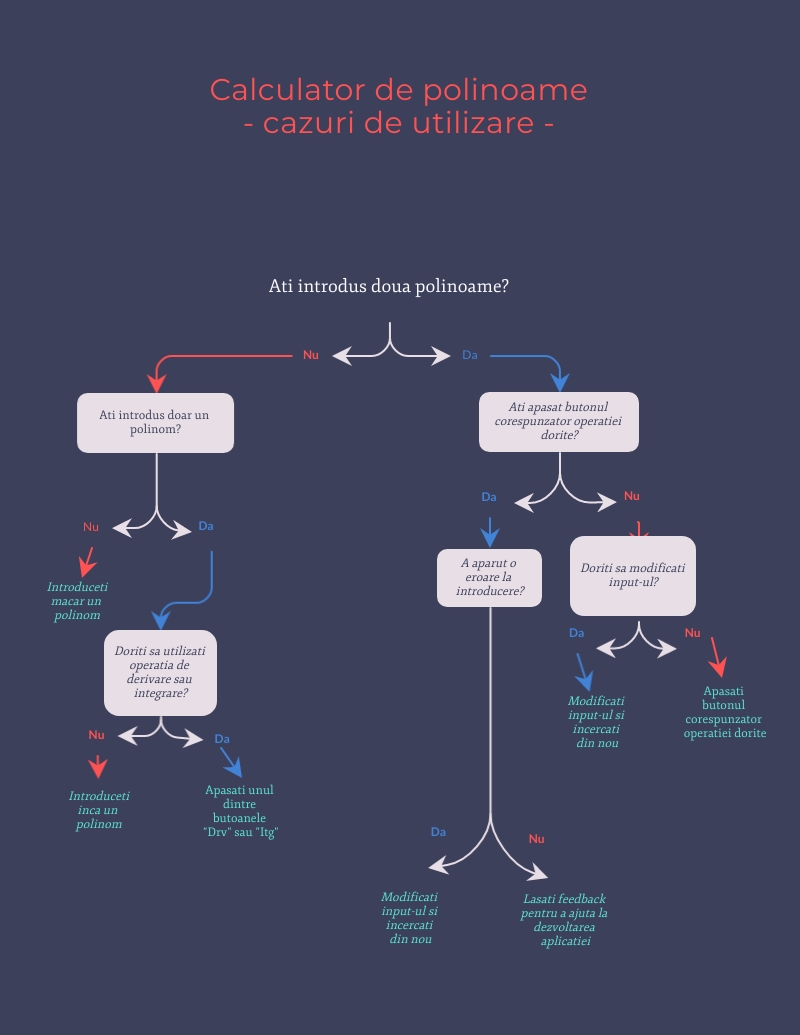
Testarea calculatorului de polinoame reprezinta etapa finala in atingerea obiectivului definit. In cadrul etapei de testare sunt definite teste unitare care sa verifice daca functionalitatea aplicatiei este conform cerintelor definite in prima etapa. In cazul in care exista erori sau vreun test esuaza, se revine la pasii anteriori pentru a se rectifica eroarea produsa, urmand mai apoi sa se ruleze din nou testele. (capitolul V – rezultate)

**Analiza problemei**

Problema care a trebuit rezolvata in cadrul acestui proiect este reprezentata de dezvoltarea si implementarea unui calculator de polinoame cu o interfata grafica cu ajutorul careia utilizatorul sa poata efectua operatii cu polinoame si sa poata vizualiza rezultatul. Operatiile care s-au implementat au fost *adunarea, scaderea, inmultirea, impartirea* a doua polinoame si *derivarea* si *integrarea* unui polinom.

Totodata, calculatorul de polinoame trebuie sa aiba o interfata prietenoasa cu utilizatorul astfel incat sa faca experienta de utilizare una placuta si sa poata fi de folos in scopul pe care si-l doreste sa il indeplineasca. Utilizatorul poate introduce doua polinoame, apasand pe unul dintre cele 6 butoane denumite sugestiv, reprezentand fiecare dintre operatiile mentionate mai sus.

In cazul in care unul dintre polinoamele introduse nu poate fi citit corect, calculatorul afiseaza o eroare, aducand la cunostiinta utilizatorului ca ar trebui sa verifice input-ul pentru a se asigura ca este introdus corect. Daca input-ul este recunoscut, in momentul apasarii oricarui buton din cadrul aplicatiei, se efectueaza operatia corespunzatoare butonului apasat si rezultatul este afisat in casuta corespunzatoare rezultatului.

****

* Adunarea polinoamelor: citire polinoame → adunarea coeficientilor monoamelor de acelasi grad → afisarea rezultatului
* Scaderea polinoamelor: citire polinoame → scaderea coeficientilor monoamelor de acelasi grad → afisarea rezultatului
* Inmultirea polinoamelor: citire polinoame → inmultirea fiecarui monom din primul polinom cu fiecare monom din cel de-al doilea monom → afisarea rezultatului
* Impartirea polinoamelor: citire polinoame → impartirea primului monom din polinomul de grad mai mare la primul monom din polinomul cu grad mai mic → inmultirea catului obtinut cu polinomul cu grad mai mic si scaderea rezultatului din polinomul cu grad mai mare → repetarea ultimilor doi pasi pana cand catul are gradul mai mic decat cel de-al doilea polinom → afisarea rezultatului
* Derivarea polinoamelor: citire polinoame → modificarea coeficientilor monoamelor corespunzator operatiei de derivare → modificarea puterilor monoamelor corespunzator operatiei de derivare → modificarea gradului polinomului corespunzator operatiei de derivare → afisarea rezultatului
* Integrarea polinoamelor: citire polinoame → modificarea puterilor monoamelor corespunzator operatiei de integrare → modificarea coeficientilor monoamelor corespunzator operatiei de integrare → modificarea gradului polinomului corespunzator operatiei de integrare → afisarea rezultatului

**Proiectare**

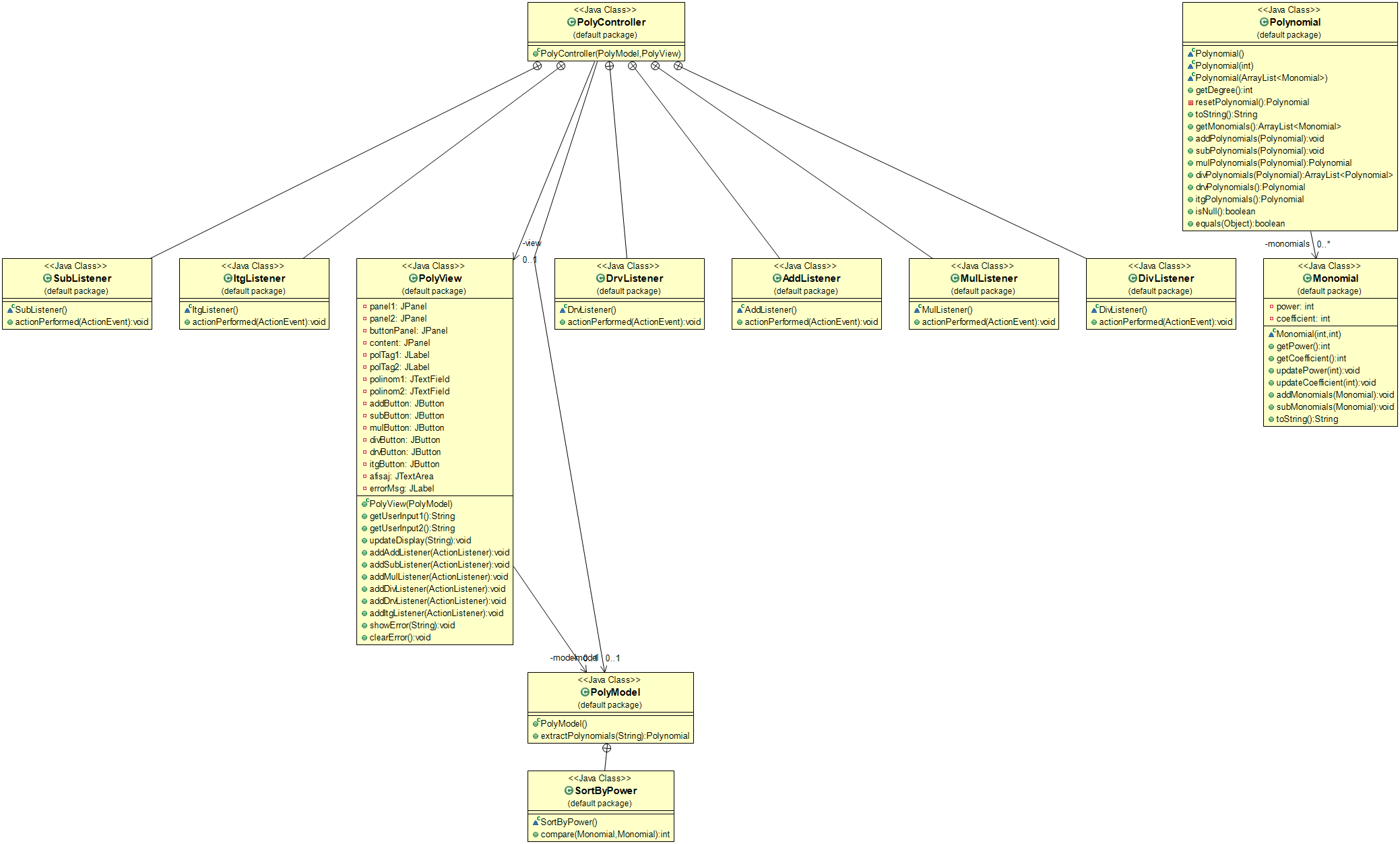
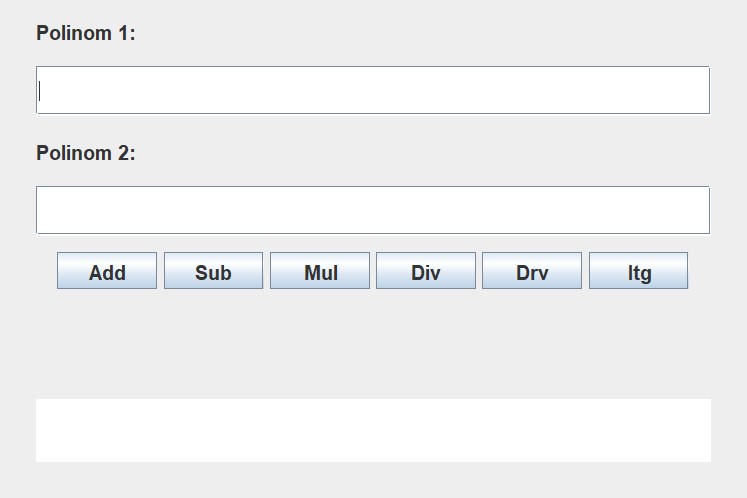
****

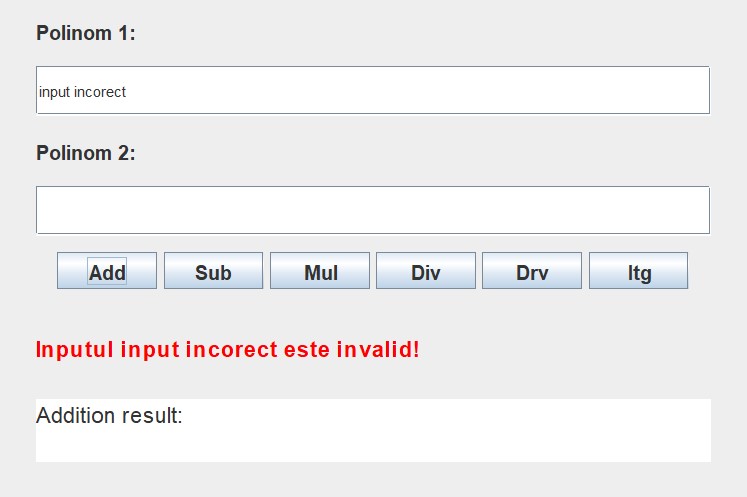
Diagrama de clase

La baza calculatorului de polinoame stau clasele care se ocupa cu interfata grafica si clasele care se ocupa de modelarea propriu-zisa a polinoamelor. Un polinom este implementat ca o lista de zero sau mai multe monoame, ordonate in ordinea descrescatoare a puterilor. La construirea polinomului se adauga si monoamele inexistente fizic (coeficient 0) pentru a face mai usoara efectuarea operatiilor: un polinom 5X^3 + 3X^1 introdus ca input va fi stocat ca fiind 5X^3 + 0X^2 + 3X^1 + 0X^0.



Screenshot din aplicatie

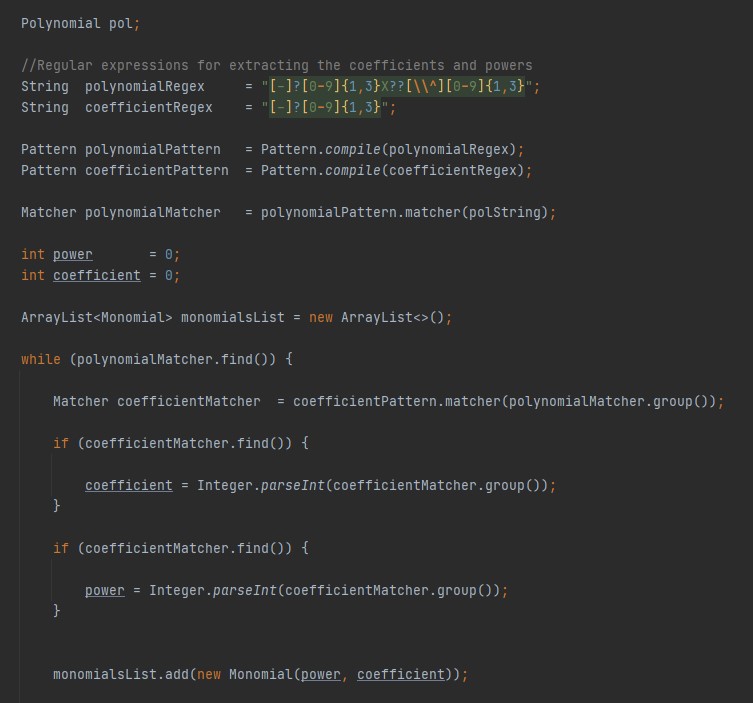
Am decis ca un aspect minimalist al aplicatiei ar imbunatati considerabil experienta de utilizare a calculatorului de polinoame. Astfel, introducand doar sase butoane pentru efectuarea operatiilor dorite, doua text field-uri pentru introducerea polinoamelor si o fereastra simpla pentru afisarea rezultatului obtinut dupa efectuarea operatiei cerute, am obtinut o interfata simpla dar completa si functionala.

Screenshot cu input incorect introdus

Totodata am adaugat functionalitatea unei erori care informeaza utilizatorul in momentul in care acesta introduce un input nerecunoscut. Aceasta eroare dispare in cazul in care inputul este introdus corect mai tarziu, iar daca inputul devine invalid eroarea reapare.

La preluarea input-ului de la utilizator s-au folosit doua string-uri pentru a efectua recunoasterea sablonului unui polinom: mai intai sunt gasite grupurile de caractere din sirul introdus care au forma unui monom, iar mai apoi, pentru fiecare monom gasit, sunt identificate coeficientul si puterea. Dupa identificarea coeficientului si puterii monomului, se construieste un obiect avand proprietatile abia gasite si este introdus intr-o lista care urmeaza sa contina monoamele identificate si construite. La final, lista este sortata in ordine descrescatoare a puterilor, pentru a fi siguri ca polinomul este ordonat corect.

Dupa obtinerea listei de monoame introduse de utilizator, se apeleaza constructorul clasei Polynomial pentru a obtine un polinom la fel ca cel din input, dar sub forma de obiect asupra caruia se pot efectua operatii si apela metode, spre deosebire de sirul de caractere introdus ca input.

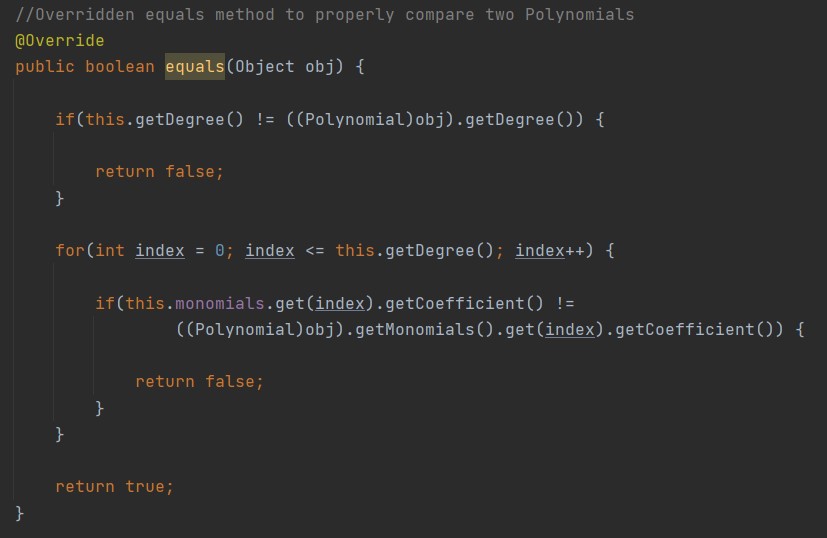
Codul care converteste inputul intr-o lista de monoame

**Implementare**

Aplicatia este implementata urmarind modelul structural MVC (Model – View – Controller), care separa logica aplicatiei, de interfata grafica si de elementele care fac legatura intre cele doua. In acest sens din Model fac parte clasele PolyModel, Polynomial, Monomial; View-ul este alcatuit doar din clasa PolyView; iar Controller-ul este alcatuit, la randul lui, doar din clasa PolyController.

Pe langa acestea mai exista clasa MainClass, care contine metoda Main() si este clasa principala de unde se porneste aplicatia, clasa InputNotPolynomialException, care este o exceptie personalizata pentru cazul in care este introdus un input care nu poate fi tradus intr-un polinom si clasa TestClass, care este clasa unde sunt implementate testele unitare folosind Junit.

Esenta operatiilor cu polinoame consta in modelarea corecta a polinoamelor si implicit si a monoamelor regasite in interiorul acestora. De acest lucru se ocupa clasele Monomial si Polynomial. Clasa Monomial are ca parametri pentru obiecte puterea si coeficientul unui monom. Pe langa gettere si settere, se remarca metodele **addMonomials** si **subMonomials**, care aduna, respectiv scad, un monom la/din un altul si metoda **toString** suprascrisa pentru a putea vizualiza mai usor cum trebuie monoamele.

 Clasa Polynomial are un singur camp ce reprezinta o lista de monoame din care este alcatuit polinomul. Constructorul principal al clasei preia o lista de monoame si creaza lista finala de monoame care formeaza de fapt polinomul. Metoda **resetPolynomial** trateaza situatia in care raman monoame goale, inutile (de forma 0X^3) la inceputul listei de monoame, lucru care dezechilibreaza algoritmii pentru operatii. Metoda suprascrisa **toString** se ocupa de convertirea corecta a obiectelor de tip Polynomial in text. Tot in cadrul acestei clase sunt implementate si metodele propriu-zise care se ocupa de rezolvarea operatiilor cerute: **addPolynomials**, **subPolynomials**, **mulPolynomials**, **divPolynomials**, **drvPolynomials**, **itgPolynomials**. Ultima metoda din cadrul clasei Polynomial este metoda suprascrisa **equals**, care ajuta in compararea corecta a doua polinoame, comparand coeficientii si puterile monoamelor din cadrul celor doua polinoame, doar in cazul in care gradele acestora coincid.

Suprascrierea metodei equals

Clasa PolyModel se ocupa de convertirea inputului de la utilizator in monoame si implicit in polinoame, folosind regular expressions si pattern matching pentru a efectua conversia. Metoda care efectueaza conversia este **extractPolynomials**, iar la finalul conversiei este folosita clasa interna *SortByPower* care are implementata metoda compare, pentru a putea compara doua monoame si a realiza ordonarea acestora in cadrul listei generate in urma convertirii inputului.

Clasa PolyController se ocupa de preluarea comenzilor de la butoanele din aplicatie si apelarea functiilor din celelalte clase care se ocupa de efectuarea operatiilor cerute. Aceasta contine referinte catre PolyModel si PolyView pentru a putea comunica cu acestea si a-si juca rolul de controller in arhitectura MVC. In interiorul acestei clase sunt prezente sase clase interne, care implementeaza interfata ActionListener si preiau apasarea butoanelor pentru operatii si actualizeaza afisajul rezultatului sau al erorii, dupa caz.

Clasa InputNotPolynomialException extinde clasa Exception si este o exceptie personalizata care ajuta la afisarea corecta a erorii. In cadrul acestei clase, constructorul primeste input-ul gresit de la utilizator si creaza un mesaj personalizat care sa ajute in depistarea greselii la introducere, iar suprascrierea metodei **toString** ajuta la vizualizarea corecta a mesajului erorii.

**** Clasa PolyView este clasa care se ocupa de interfata grafica cu care interactioneaza utilizatorul. Aceasta contine ca parametri cele sase butoane **addButton**, **subButton**, **multButton**, **divButton**, **drvButton**, **itgButton**, cele doua text field-uri pentru introducerea polinoamelor: **polinom1**, **polinom2**, afisajul pentru rezultat: **afisaj**, si etichetele care denumesc field-urile pentru polinoame si eroarea: **polTag1**, **polTag2**, **errorMsg**.

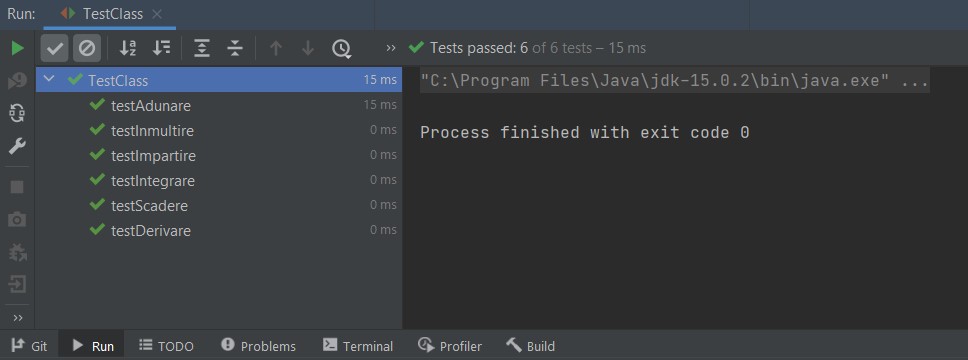
Parametrii clasei PolyView

In cadrul constructorului clasei sunt setate si adaugate toate elementele ce tin de interfata si aranjarea in cadrul ferestrei. In plus, clasa contine metode pentru adaugarea listenerilor pentru butoanele pentru operatii cat si metoda **updateDisplay**, care reimprospateaza afisajul pentru rezultat, metoda **showError**, care face vizibila eroarea pe display si metoda **clearError**, care sterge eroarea de pe display.

**Rezultate**

Dupa implementarea aplicatiei, unul dintre cei mai importanti pasi este testarea corespunzatoare a functionalitatii. Pentru a testa aplicatia, am folosit framework-ul Junit, care permite rularea unor teste predefinite pentru a putea verifica corectitudinea operatiilor implementate fara a realiza acest lucru manual de fiecare data cand o schimbare este adusa codului. Scenariile de testare au fost implementate in cadrul clasei TestClass. Pentru fiecare operatie este introdus un set de date prestabilit si un rezultat calculat manual fata de care se face verificarea rezultatului obtinut prin utilizarea calculatorului implementat.

Astfel, metodele de test sunt **testAdunare**, **testScadere**, **testInmultire**, **testImpartire**, **testDerivare**, **testIntegrare**, la finalul carora este plasat un „assert” care determina daca testul trece sau nu. In cadrul acestui „assert”, este folosita metoda equals care este suprascrisa in cadrul clasei Polynomial.

 In cazul in care toate testele au succes, aplicatia functioneaza corect din punct de vedere al testelor implementate si este cu un pas mai aproape de a fi finalizata. In caz contrar, trebuie urmarit sirul de dezvoltare pentru a descoperi sursa de erori.

Toate testele definite au trecut cu succes

**Concluzii**

In urma implementarii proiectului, am consolidat cunostiintele dobandite in cadrul cursului in legatura cu folosirea modelului arhitectural Model View Controller si modelarea problemei in clase si structuri de date care sa fie utilizate eficient in cadrul aplicatie. Totodata, am avut ocazia sa imbunatatim competentele si cunostiintele ce tin de redactarea unei documentatii complete si corecte pe baza unui proiect.

Din punct de vedere al dezvoltarii ulterioare a calculatorului de polinoame implementat exista doua posibilitati de dezvoltare evidente, care au aparut in urma lucrului la proiect dar si a utilizarii aplicatiei realizate: in primul rand s-ar putea adauga posibilitatea memorarii rezultatelor calculate si folosirea acestora in urmatoarele operatii ce se doresc a fi efectuate; in al doilea rand se pot adauga diverse alte operatii cu polinoame care sa imbunatateasca experienta utilizatorului si utilitatea aplicatiei.

**Bibliografie**

InteliJ Idea: mediul de dezvoltare utilizat pentru implementarea si testarea proiectului.

Visme: unealta online folosita pentru crearea flowchart-ului

<https://dashboard.visme.co/v2/login#/infographics?type=flowchart>

Swing: <https://docs.oracle.com/javase/tutorial/uiswing/>

Java Regular Expressions: <https://docs.oracle.com/javase/tutorial/essential/regex/>

<https://www.baeldung.com/regular-expressions-java>

Junit: <https://www.vogella.com/tutorials/JUnit/article.html>

https://www.baeldung.com/junit-5

Java Naming Conventions: <https://google.github.io/styleguide/javaguide.html>