**FACULTATEA DE AUTOMATICĂ ŞI CALCULATOARE**

**Calculator de polinoame**

**Documentație**

**Săsăujan Cristian-Andrei**

**Grupa 30228**

CUPRINS

1. Obiectivul temei.............................................................................................................................. 1

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

3. Proiectare........................................................................................................................................ 5

4. Implementare..................................................................................................................................7

5. Rezultate......................................................................................................................................... 8

6. Concluzii...........................................................................................................................................9

7. Bibliografie.......................................................................................................................................9

**1. Obiectivul temei**

Obiectivul temei de laborator a fost implementarea unui calculator de polinoame, mai exact a două polinoame, cu operațiile de adunare, scadere, înmulțire, împărțire, integrare, derivare, toate acestea fiind implementate într-o interfață grafică, fiecare operație având butonul ei corespunzător. Fiecare polinom reprezintă un input, iar ca output final vom avea un alt polinom pe care s-au efectuat operațiile.

Sub-obiective : Proiectarea (Capitolul 3) unde putem vedea organizarea proiectului, Implementarea (Capitolul 4) unde vom regăsi informații despre clase, Testarea proiectului (Capitolul 5)

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

M-am gândit ca inputul meu, polinomul, să-l parsez în grupuri mai mici numite monoame, de unde să extrag mai departe coeficientul și exponentul, atribute ale clasei Monom, iar în clasa Polinom voi avea o listă de monoame. Parsarea am facut-o cu ajutorul regex-ului, am organizat input-ul în mai multe grupuri apoi verificând mai multe aspecte, de exemplu în cazul în care am x in grupul de regex se va regăsi null si verific daca grupul e null să-mi pună ca coeficient 1, dacă am doar 3 ca input imi va pune exponent 0 , daca am doar x imi va pune exponent 1 fară să introduc eu x^1. Un alt lucru care a avut nevoie de puțin mai multă atenție a fost la scădere, cazul in care am in polinomul 1 de ex 3x^2+2x și în polinomul 2 4x^3+3x^9 să parcurg polinomul 2 și să setez exponentul fiecarui monom cu ‘’-‘’ deoarece nu există elemente care să se reducă sau care să se scadă.

**Cerințe funcționale:**

* Aplicația noastră ar trebui să permită utilizatorului să introducă 2 polinoame, în cele 2 text field-uri.
* Aplicația permite utilizatorului să efectueze operația de adunare a doua polinoame, rezultatul fiind afișat în text field-ul final.
* Aplicația permite utilizatorului să efectueze operația de scădere a doua polinoame, rezultatul fiind afișat în text field-ul final.
* Aplicația permite utilizatorului să efectueze operația de înmulțire a doua polinoame, rezultatul fiind afișat în text field-ul final.
* Aplicația permite utilizatorului să efectueze operația de derivare a unui polinom, în cazul în care se introduc 2 polinoame se va afișa o eroare, dacă nu rezultatul fiind afișat în text field-ul final.
* Aplicația permite utilizatorului să efectueze operația de integrare a unui polinom, în cazul în care se introduc 2 polinoame se va afișa o eroare, dacă nu rezultatul fiind afișat în text field-ul final.

**Use Case**: Operații asupra polinoamelor

**Actor principal** : Utilizatorul

**Șcenariu de success** :

1 .Utilizatorul introduce cele 2 polinoame

2 .Utilizatorul alege o operație(Adunare,Scădere,Înmulțire,Derivare,Integrare,Împărțire)

3. Utilizatorul va putea vedea rezultatul afișat în căsuța text.

**Șcenarii alternative:**

1. În cazul în care utilizatorul dorește să aleagă operația de Derivare sau Integrare utilizatorul va trebui să introduce un singur polinom, dacă utilizatorul introduce 2 polinoame va primi o eroare.

2. Dacă utilizatorul nu introduce nimic, se va afișa un mesaj de eroare.

**3. Proiectare**

Pentru adunare și scădere, înmulțire verific de fiecare dată ca ambele input-uri sa fie introduse dacă nu sunt introduse, voi afisa un mesaj de eroare "Introduceti ambele polinoame."

* **compareTo**

Am implementat totodata în clasa Monom metoda compareTo care o voi folosi în toate operațiile cu Collections.sort() pentru a aranja lista de monoame dupa exponentul cel mai mare, pentru a reda forma polinomului.

* **Parsarea**

Parsarea am facut-o cu ajutorul regex-ului, în ea îmi extrăgeam cu ajutorul pattern-ului coeficienții si exponenții care îi salvam în lista de monoame.

* **Adunarea**

La adunare primul lucru am inițializat o lista de monoame care îmi reprezintă polinomul meu final, iar m-ai apoi am extras din textfield fiecare polinom intr-un string diferit, care m-ai apoi l-am prelucrat in monoame cu ajutorul metodei de parsare. Pentru a realiza adunarea celor 2 polinoame am parcurs cele 2 liste de monoame în paralel si verificam cu un if daca au același exponent atunci să mi se adune coeficenții între ei , iar mai apoi eliminam monomul pe care s-a făcut deja adunarea deoarece mai apoi, pe monoamele care nu s-au efectuat operații acestea să le adaug la rezultatul final(în lista de monoame) fără modificări, să le sortez cu Collections.sort , să creez un obiect nou de instanță polinom la care pasez mai departe lista mea de monoame, mai apoi afisând în textfield-ul rezultatului final polinomul final.

* **Scăderea**

La scădere primul lucru am inițializat o lista de monoame care îmi reprezintă polinomul meu final, iar m-ai apoi am extras din textfield fiecare polinom intr-un string diferit, care m-ai apoi l-am prelucrat in monoame cu ajutorul metodei de parsare. Pentru a realiza adunarea celor 2 polinoame am parcurs cele 2 liste de monoame în paralel si verificam cu un if daca au același exponent atunci realizam scăderea coeficientului din input1 cu coeficientul din input2, iar mai apoi scot monoamele care s-au scăzut din lista lor , rămânand doar cele asupra cărora nu s-a realizat operația. Pe monoamele asupra cărora nu s-a realizat operația le voi parcurge si le voi seta coeficientul cu minus in caz ca de exemplu avem x^2 ca input1 si 3x^4+6x^5 în input2 nu se efectuează nici-o operație asupra inputului2 atunci doar le inversez semnul. Toată informația va fi colectată intr-o lista de monoame, care va fi sortată cu ajutorul metodei compareTo care a fost implementată în clasa Monom astfel având permisiunea sa folosim Collections.sort, cu ajutorul căreia vom sorta dupa exponentul cel mai mare polinomul final care va fi afișat în textField-ul din interfața grafică.

* **Înmulțirea**

La înmulțire primul lucru am inițializat o lista de monoame care îmi reprezintă polinomul meu final, și o altă listă de monoame care mă v-a ajuta să adun monoamele în cazul în care în lista mea finală se mai regăsesc monoame cu aceelași exponent, iar m-ai apoi am extras din textfield fiecare polinom intr-un string diferit. Am parcurs cu 2 for each-uri ambele polinoame și adăugam într-un alt polinom coeficientul inputului unu plus coeficientul inputului doi, iar ca exponent păstram coeficientul inputului 1. Acest polinom obținut il dădeam ca argument metodei adunaMonoame, care de exemplu dacă dau ca input unu : 3x+2x^2 și ca input doi : x+2 atunci după ce se realizează înmulțirea termen cu termen vom ajunge la rezultatul 3x^2+6x+2x^3+4x^2 , unde se poate observa ca avem 2 termeni cu exponent 2, acestia 2 vor trebui să se adune, de acest lucru ocupându-se metoda adunaMonoame, care îmi parcurge rezultatul 3x^2+6x+2x^3+4x^2 și caută dacă există exponenți egali, dacă se gasesc, o să le adune coeficienții. Toată informația va fi colectată intr-o lista de monoame, care va fi sortată cu ajutorul metodei compareTo care a fost implementată în clasa Monom astfel având permisiunea sa folosim Collections.sort, cu ajutorul căreia vom sorta dupa exponentul cel mai mare polinomul final care va fi afișat în textField-ul din interfața grafică.

* **Derivare**

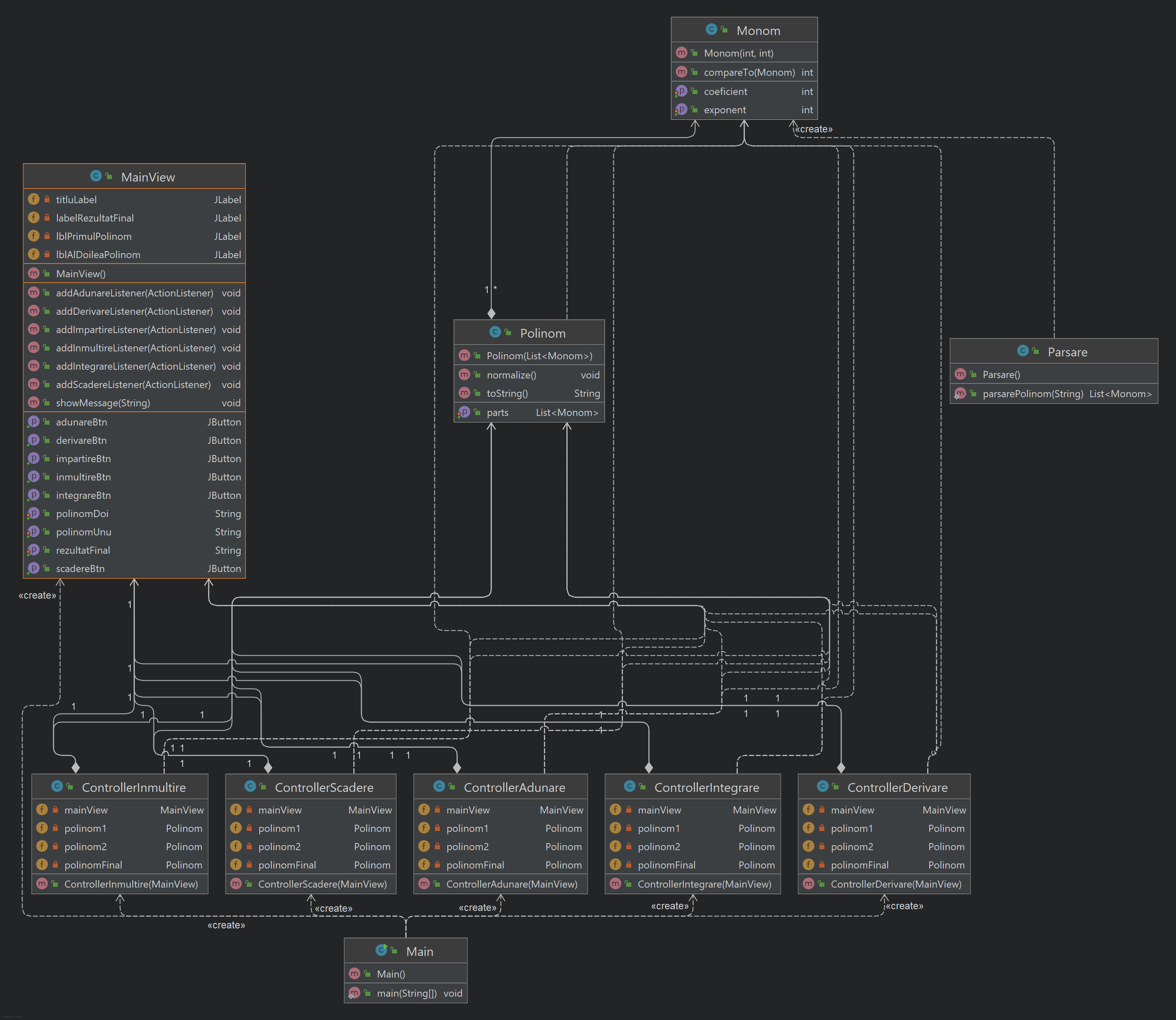
Pentru derivare am introdus un try catch care îmi aruncă o excepție în cazul în care utilizatorul a introdus in ambele textField-uri un polinom, în cazul în care se introduc 2 polinoame se va afisa pe ecran intr-o căsuță "Introduceti un singur polinom pentru derivare". Am pus această condiție deoarece voi deriva un singur polinom, ori cel din input unu ori cel din input doi. Am procedat la fel ca și la operațiile anterioare, am preluat în 2 string-uri input unu sau inputul doi, care le- am parsat creând o listă de monoame, cu ajutorul căreia am inițializat polinomul unu și polinomul doi. Următorul pas este parcurgerea celor 2 polinoame, unul dintre ele va fi empty, iar la fiecare pas adaug intr-o listă de monoame, la coeficient, coeficientul+exponentul monomului parcurs, iar la exponent voi scădea o unitate. Toată informația va fi colectată intr-o lista de monoame, care va fi sortată cu ajutorul metodei compareTo care a fost implementată în clasa Monom astfel având permisiunea sa folosim Collections.sort, cu ajutorul căreia vom sorta dupa exponentul cel mai mare polinomul final care va fi afișat în textField-ul din interfața grafică.

* **Integrare**

Pentru integrare am introdus un try catch care îmi aruncă o excepție în cazul în care utilizatorul a introdus in ambele textField-uri un polinom, în cazul în care se introduc 2 polinoame se va afisa pe ecran intr-o căsuță "Introduceti un singur polinom pentru integrare". Am pus această condiție deoarece voi deriva un singur polinom, ori cel din input unu ori cel din input doi. Am procedat la fel ca și la operațiile anterioare, am preluat în 2 string-uri input unu sau inputul doi, care le- am parsat creând o listă de monoame, cu ajutorul căreia am inițializat polinomul unu și polinomul doi. Următorul pas este parcurgerea celor 2 polinoame, unul dintre ele va fi empty, iar la fiecare pas adaug intr-o listă de monoame, la coeficient, coeficentul monomului , iar la exponent, exponentul monomului care il voi incrementa cu 1. Totodata pentru a integra corect am folosit o afisare care imi va afisa monomul, totul supra exponentul incrementat cu 1, iar dupa ce s-au integrat toate monoamele pe rând și toată informația care a fost colectată intr-o lista de monoame, care va fi sortată cu ajutorul metodei compareTo care a fost implementată în clasa Monom astfel având permisiunea sa folosim Collections.sort, cu ajutorul căreia vom sorta dupa exponentul cel mai mare polinomul final, am adaugat si un “+C” care reprezinta constanta arbitrară de integrare, toate acestea vor fi afisate in textField-ul polinomului final.

Aplicația este creată cu ajutorul pattern-ului MVC (model-view-controller), model, view, controller fiind fiecare individual un package.

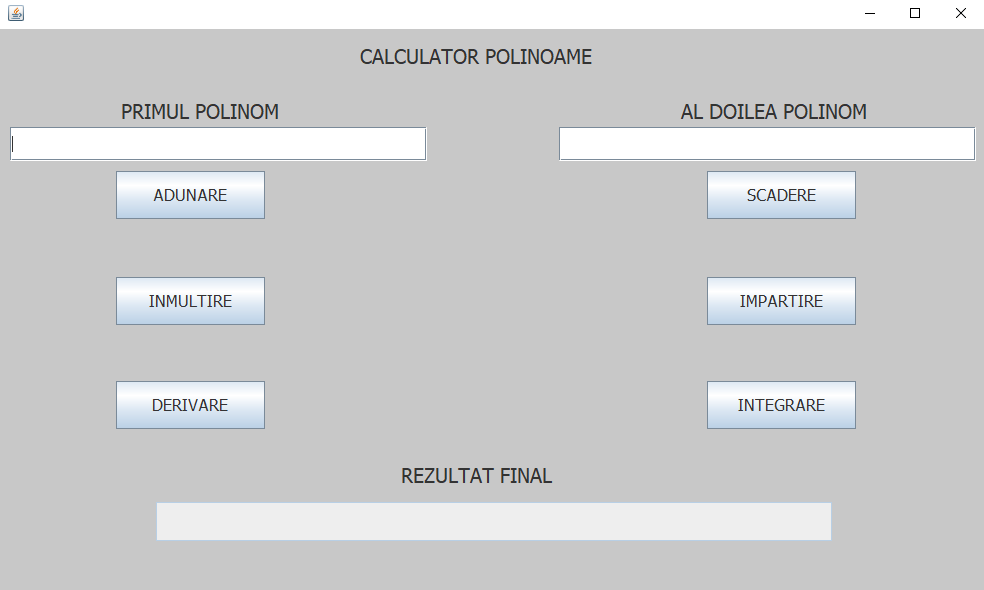
* Model: Această parte a controlatorului manipulează operațiunile logice și de utilizare de informație (trimisă dinainte de către rangul său superior) pentru a rezulta de o formă ușor de înțeles.
* View: Acestui membru al familiei îi corespunde reprezentarea grafică, sau mai bine zis, exprimarea ultimei forme a datelor: interfața grafică ce interacționează cu utilizatorul final. Rolul său este de a evidenția informația obținută până ce ea ajunge la controlor.
* Controller: Cu acest element putem controla accesul la aplicația noastră. Pot fi fișiere, scripturi (eng. scripts) sau programe, in general orice tip de informație permisă de interfață. În acest fel putem diversifica conținutul nostru de o formă dinamică și statică, în același timp.

**Diagrama UML**

Am folosit clasa ArrayList<T>, care este o colecție Java și implementează toate metodele din interfața Collections, pentru a reține o listă de monoame, care este atribut al clasei polinom.

Clasa Monom implementează interfața Comparable, ce are ca tip generic un obiect de tip Monom, cu ajutorul ei am creat metoda compareTo de care m-am ajutat la afișarea polinomului final.

**4. Implementare**

* Clasa Main: în această clasă am instanțiat view-ul si controller-ul pentru fiecare aplicație.
* Clasa Monom: în această clasă am ca atribute un exponent și un coeficient care mai departe cu ajutorul constructorului îmi va crea obiecte de tip monom cu exponent și coeficient, totodata în această clasă am si gettere și settere, dar și metoda compareTo
* Clasa Polinom: în acestă clasă am ca atribut o listă de monoame cu ajutorul căreia realizez polinomul, totodata în această clasă am si gettere și settere, dar și metoda compareTo
* Clasa MainView: în această clasă sunt prezente toate butoanele, textfield-urile, toate label-urile , tot ce ține de interfața grafică
* Clasa ControllerAdunare : în această clasă am actionlistener-ul pe butonul de adunare, care cu ajutorul metodei din această clasă va aduna cele 2 polinoame.
* Clasa ControllerScădere : în această clasă am actionlistener-ul pe butonul de scădere, care cu ajutorul metodei din această clasă va scădea cele 2 polinoame.
* Clasa ControllerÎnmulțire : în această clasă am actionlistener-ul pe butonul de înmulțire, care cu ajutorul metodei din această clasă va înmulți cele 2 polinoame.
* Clasa ControllerDerivare : în această clasă am actionlistener-ul pe butonul de derivare, care cu ajutorul metodei din această clasă va deriva unul din cele 2 polinoame.
* Clasa ControllerDerivare : în această clasă am actionlistener-ul pe butonul de integrare, care cu ajutorul metodei din această clasă va integra unul din cele 2 polinoame.

Implementare interfeței utilizator : interfața cuprinde 4 etichete(Primul polinom, Al doilea polinom, Calculator polinoame, Rezultat final) , 6 butoane (adunare, scădere, înmulțire, împărțire, derivare, integrare) și 3 textfield-uri. Fiecare buton are un action. Fiecare buton are un listener, fiecare textfield are un getter si un setter. În textfield-ul rezultatului final nu este permisă scrierea. Totodată i-am oferit o culoare gri contentpanel-ului.

**5. Rezultate**

Cu ajutorul Junit-ului am testat mai multe cazuri, 4 cazuri pe operație, fiecare test conținând o chichiță. Am să atașez cate un caz pentru fiecare operație.

void adunare1*()  
{* mainView=new MainView*()*;  
 mainView.setPolinomUnu*(*"x^2+2"*)*;  
 mainView.setPolinomDoi*(*"x^2-2"*)*;  
 controllerAdunare=new ControllerAdunare*(*mainView*)*;  
 mainView.getAdunareBtn*()*.doClick*()*;  
 *assertEquals(*"+2x^2", mainView.getRezultatFinal*())*;  
*}*

void scadere1*()  
{* mainView=new MainView*()*;  
 mainView.setPolinomUnu*(*"x-2"*)*;  
 mainView.setPolinomDoi*(*"x-2"*)*;  
 controllerScadere=new ControllerScadere*(*mainView*)*;  
 mainView.getScadereBtn*()*.doClick*()*;  
 *assertEquals(*"0", mainView.getRezultatFinal*())*;  
*}*

void inmultire1*()  
{* mainView=new MainView*()*;  
 mainView.setPolinomUnu*(*"x-2"*)*;  
 mainView.setPolinomDoi*(*"x+2"*)*;  
 controllerInmultire=new ControllerInmultire*(*mainView*)*;  
 mainView.getInmultireBtn*()*.doClick*()*;  
 *assertEquals(*"+x^2-4", mainView.getRezultatFinal*())*;  
*}*

void derivare1*()  
{* mainView=new MainView*()*;  
 mainView.setPolinomUnu*(*"3x^2+6x"*)*;  
 //mainView.setPolinomDoi("3x+2x^2");  
 controllerDerivare=new ControllerDerivare*(*mainView*)*;  
 mainView.getDerivareBtn*()*.doClick*()*;  
 *assertEquals(*"+6x^1+6", mainView.getRezultatFinal*())*;  
*}*

void integrare1*()  
{* mainView=new MainView*()*;  
 //mainView.setPolinomUnu("5x^-2+6");  
 mainView.setPolinomDoi*(*"5x^-2"*)*;  
 controllerIntegrare=new ControllerIntegrare*(*mainView*)*;  
 mainView.getIntegrareBtn*()*.doClick*()*;  
 *assertEquals(*"[+(5x^-1)/-1]+C", mainView.getRezultatFinal*())*;  
*}*

**6. Concluzii**

În concluzie, după munca depusă la aplicație am învățat sa folosesc regex-ul, nu știam de această metodă de parsare, totodata m-am perfecționat în folosirea for each-ului, deobicei nu îl foloseam. Această temă ne exemplifică cât de bine trebuie să fie organizată o aplicație dar și cât de bine trebuie programatorul să își împartă fiecare task în parte. O posibilitate de extindere a aplicației poate să fie ca polinoamele să accepte ca exponenți toate numerele din mulțimea numerelor reale, la fel și pentru coeficienți.

**7. Bibliografie**

<https://ro.wikipedia.org/wiki/Model-view-controller>

[**https://regex101.com/r/OuQKd6/1**](https://regex101.com/r/OuQKd6/1)

[**https://users.utcluj.ro/~igiosan/teaching\_poo.html**](https://users.utcluj.ro/~igiosan/teaching_poo.html)

[**https://www.youtube.com/watch?v=flpmSXVTqBI&t=443s**](https://www.youtube.com/watch?v=flpmSXVTqBI&t=443s)