DOCUMENTAȚIE

TEMA 1

NUME STUDENT: Vicsi Márk-Ottó

GRUPA:30221

# CUPRINS

[1. Obiectivul temei 3](file:///C:\Users\bastard\AppData\Local\Temp\PT2021-2022_Documentation_Template.doc#_Toc95297885)

[2. Analiza problemei, modelare, scenarii, cazuri de utilizare 3](file:///C:\Users\bastard\AppData\Local\Temp\PT2021-2022_Documentation_Template.doc#_Toc95297886)

[3. Proiectare 4](file:///C:\Users\bastard\AppData\Local\Temp\PT2021-2022_Documentation_Template.doc#_Toc95297887)

[4. Implementare 3](file:///C:\Users\bastard\AppData\Local\Temp\PT2021-2022_Documentation_Template.doc#_Toc95297888)

[5. Rezultate 3](file:///C:\Users\bastard\AppData\Local\Temp\PT2021-2022_Documentation_Template.doc#_Toc95297889)

[6. Concluzii 3](file:///C:\Users\bastard\AppData\Local\Temp\PT2021-2022_Documentation_Template.doc#_Toc95297890)

[7. Bibliografie 3](file:///C:\Users\bastard\AppData\Local\Temp\PT2021-2022_Documentation_Template.doc#_Toc95297891)

# Obiectivul temei

Proiectarea și implementarea unui calculator de polinoame cu o interfață grafică dedicată, prin care utilizatorul poate introduce polinoame, selecta operația matematică (de exemplu adunare, scădere, înmulțire, împărțire, derivată, integrare) care urmează să fie efectuată și vizualiza rezultatul.

Obiective secundare:

* Capitolul 2: Analiza problemei și identificarea cerințelor (use case diagram, cerințe funcționale)
* Capitolul 3: Proiectarea calculatorului polinomial (alegerea structurilor de date, împărțirea pe clase)
* Capitolul 4: Implementarea calculatorului polinomial
* Capitolul 5: Rezultatele după testarea calculatorului polinomial

# Analiza problemei, modelare, scenarii, cazuri de utilizare

Cerințele funcționale sunt:

* Calculatorul de polinoame ar trebui să permită utilizatorilor să introducă polinoame.
* Calculatorul de polinoame ar trebui să verifice corectitudinea datelor de intrare furnizate de către utilizator.
* Calculatorul de polinoame ar trebui să permită utilizatorilor să selecteze operația matematică dorită.
* Calculatorul de polinoame ar trebui să adune două polinoame.
* Calculatorul de polinoame ar trebui să adune două polinoame.
* Calculatorul de polinoame ar trebui să scadă două polinoame.
* Calculatorul de polinoame ar trebui să înmulțească două polinoame.
* Calculatorul de polinoame ar trebui să împartă două polinoame.
* Calculatorul de polinoame trebuie să calculeze derivata unui polinom.
* Calculatorul de polinoame trebuie să calculeze integrala nedeterminată a unui polinom.

Diagram

Description automatically generatedDiagrama de cazuri de utilizare (use case diagrams) este o sinteză vizuală a interacțiunilor și a relațiilor din cadrul unui sistem. Aceste diagrame arată o viziune foarte largă a unui sistem. O diagramă de caz de utilizare include câteva componente cheie, cum ar fi: actorii, care sunt persoane care au o relație în cadrul sistemului. Aceștia interacționează cu fiecare etapă a procesului. Cazurile de utilizare, adesea reprezentate printr-un oval sau un cerc care cuprinde un text, descriu o funcție a sistemului pe care este un actor sau o persoană o poate începe sau utiliza. Actorii se pot conecta la cazurile de utilizare și la alte persoane printr-o varietate de legături de comunicare care reprezintă diferite relații, cum ar fi: aserțiuni, generalizări, extinderi, incluziuni (Team, 2021)

Exemplu de caz de utilizare: operație matematică pe două polinoame

Actor primar: utilizatorul

Scenariu de succes:

1. Utilizatorul introduce cele 2 polinoame cu ajutorul interfeței grafice.
2. Utilizatorul selectează operațiunea dorită.
3. Utilizatorul apasă butonul “EVALUATE”
4. Calculatorul de polinomial efectuează operația aleasă asupra polinoamelor și afișează rezultatul.

Scenariu alternativ: polinoame incorecte

* Utilizatorul introduce polinoame incorecte (de exemplu, polinoame cu 2 sau mai multe variabile, gradul unui termen este negativ, introduce un text sau caracter)
* Scenariul revine la subpunctul 1 din scenariul de succes precizat mai sus

# Proiectare

**Modelul de proiectare MVC** (Model-View-Controller) atribuie obiectelor dintr-o aplicație unul dintre cele trei roluri: model, vizualizare (view) sau controler. Modelul definește nu numai rolurile pe care le joacă obiectele în aplicație, ci și modul în care obiectele comunică între ele. Fiecare dintre cele trei tipuri de obiecte este separat de celelalte prin limitări abstracte și comunică cu obiectele din celelalte tipuri prin aceste limitări. Colecția de obiecte de un anumit tip MVC dintr-o aplicație este uneori denumită strat / nivel - de exemplu, strat /nivel de model.

Avantajele adoptării acestui model sunt numeroase. Multe obiecte din aceste aplicații tind să fie mai ușor de reutilizat, iar interfețele lor tind să fie mai bine definite. De asemenea, aplicațiile care au un design MVC sunt mai ușor de dezvoltat decât alte aplicații.

Obiectele Model încapsulează datele specifice unei aplicații și definesc logica și calculul care manipulează și procesează aceste date. În acest caz, clasele care implementează datele aplicației sunt Polynomial și Monomial. Mai multe despre acestea sunt discutate în capitolul următor. Aplicația este concepută astfel încât obiectul model să nu aibă o legătură explicită cu obiectele de vizualizare care prezintă datele sale și permit utilizatorilor să editeze aceste date - nu este preocupat de problemele legate de interfața cu utilizatorul și de prezentare.

Comunicarea: Acțiunile utilizatorului în stratul de vizualizare care creează sau modifică datele sunt comunicate prin intermediul unui obiect controler și au ca rezultat crearea sau actualizarea unui obiect model. Atunci când un obiect model se modifică (de exemplu, calculatorul a calculat un nou polinom) acesta notifică un obiect controler, care actualizează obiectele de vizualizare corespunzătoare.

Un obiect de vizualizare este un obiect dintr-o aplicație pe care utilizatorii îl pot vedea. Un obiect de vizualizare știe cum să se deseneze și poate răspunde la acțiunile utilizatorului. Unul dintre scopurile principale ale obiectelor de vizualizare este de a afișa datele din obiectele model ale aplicației și de a permite editarea acestor date. În ciuda acestui fapt, obiectele de vizualizare sunt de obicei decuplate de obiectele model într-o aplicație MVC. Aplicația este realizată cu JavaFX, iar obiectele View sunt create automat de FXMLLoader pe baza documentului FXML furnizat, astfel încât nu există clase pentru GUI în acest proiect.

Comunicare: Obiectele de vizualizare află despre modificările aduse datelor modelului prin intermediul obiectelor controlerului aplicației și comunică modificările inițiate de utilizator - de exemplu, un nou polinom introdus în câmpul de intrare - prin intermediul obiectelor controlerului către obiectele modelului aplicației.

Un obiect controler acționează ca un intermediar între unul sau mai multe obiecte de vizualizare ale unei aplicații și unul sau mai multe obiecte model ale acesteia. Astfel, obiectele controler sunt un canal prin care obiectele de vizualizare află despre modificările din obiectele model și viceversa. De asemenea, obiectele controler pot îndeplini sarcini de configurare și coordonare pentru o aplicație și pot gestiona ciclurile de viață ale altor obiecte.

Comunicarea: Un obiect controler interpretează acțiunile utilizatorului efectuate în obiectele de vizualizare și comunică datele noi sau modificate la nivelul modelului. Atunci când obiectele modelului se modifică, un obiect controler comunică noile date ale modelului către obiectele de vizualizare, astfel încât acestea să le poată afișa (Apple Documentation Archive, 2018)

**Ca structura de dată** principală aplicația folosește ArrayList-ul. Termenii polinomului sunt reprezentate printr-o listă de monoame, astfel clasa Polinoame conține un List de tip Monomial. Clasa Monomial conține ca variabile un număr întreg, care reprezintă gradul și un număr double care reprezintă coeficientul. Acesta din urmă este de tip double deoarece la anumite operații coeficienții pot fi numere zecimale.

**Ca algoritm de sortare** aplicația utilizează metoda Collection.sort(), creată pentru colecții Java precum List, pentru a sorta termenii polinomului în ordinea descrescătoare a gradelor. Această implementare este garantată ca fiind stabilă (elementele egale nu vor fi reordonate în urma sortării - ceea ce nu este cazul în cazul sortării polinoamelor), adaptivă, iterativă, de tip "mergesort", care necesită mult mai puține comparații decât n lg(n) atunci când șirul de intrare este sortat parțial. (Oracle Java Documentations, n.d.)

Graphical user interface, application

Description automatically generated**Diagrama de clase**

Cum se vede și din diagrama de clase, un Polinom conține mai multe (o listă de) obiecte de tip Monomial. Fiecare polinom și monom este construită prin clasa Builder, fiind singura clasă care creează polinoame sau monoame noi. PolMath și Controller fac apel de fiecare dată la metodele clasei Builder pentru a instanția noi obiecte.

# Implementare

**Clasa Controller**

* Text

  Description automatically generated with low confidenceVariabila *inputCurrent* indică caseta de intrare selectată de utilizator, în care va fi introdus polinomul. În mod implicit, la începutul aplicației, aceasta arată către caseta de intrare superioară.
* *input1* și *input2* sunt referințe către câmpurile de intrare
* Funcția *mathInput()* este declanșată când utilizatorul introduce prin interfața grafică o cifră de la 0 la 9, simboluri matematice precum "+", "-", "." sau variabila X, care va fi completată automat cu un "^" de către aplicație. La pornirea aplicației, în mod implicit, căsuțele de introducere a datelor sunt completate cu textul "polynom1", "polynom2". Înainte ca în câmpul de introducere ales să apară caracterul butonului apăsat, funcția șterge textul afișat la inițializare. Dacă în câmpul de intrare au fost introduse deja 40 de caractere, atunci funcția nu mai inserează caractere noi. Această opțiune este realizată cu ajutorului variabilei *inputCurrentLength.*
* Funcțiile *delete()* / *deleteAll()* sunt funcții care șterg un singur caracter / întregul câmp de intrare.
* Funcția *changeInput()* schimbă sursa variabilei *inputCurrent*
* Funcția *parseInput()* verifică cu ajutorul unei expresii regex dacă utilizatorul a introdus un polinom corect. Dacă expresia introdusă nu este corespunzător tiparului / modelului, atunci funcția afișează un mesaj de eroare și returnează un polinom gol.
* Când utilizatorul apasă butonul “EVALUATE”, funcția *calculate()* apelează operația matematică aleasă pentru polinoamele introduse și afișează rezultatul pe câmpul de ieșire a interfeței grafice. Utilizatorul este avertizat dacă nu a ales o operație matematică.

Observații cu privire la formatul de intrare:

* Calculatorul așteaptă la intrare monoame de formă ±NUMĂR1X^NUMĂR2, unde
  + NUMĂR1 este un număr întreg (-5, 6, etc), zecimal (5.6, -8.888, .05, etc) sau nimic (implicit unu)
  + NUMĂR2 este un număr natural, care poate fi precedat de zerouri ( ex. X^005 este X^5)
* Calculatorul la versiunea actuală nu acceptă constante, ele trebuie să fie urmate de X^0 (ex. X^2+2 nu este acceptat de calculator, doar X^2+2X^0).

**Clasa Builder**

Graphical user interface, text, application

Description automatically generatedEste o clasă care construiește noi polinoame și monoame.

* Polinoamele sunt construite din obiecte de tip String. Dacă la intrare funcția *polynomial()* primește null, un string gol sau un string care conține doar caractere albe, atunci returnează un polinom gol. Metoda *polynomial()* împarte șirul de caractere primit ca parametru în termeni, fiecare dintre ei reprezentând un monom. Apoi din fiecare termen este creat o nouă instanță de tip monom, care este adăugat apoi la lista polinomului nou creat. Metoda verifică dacă utilizatorul a specificat sau nu coeficientul monomului. În cel de-al doilea caz, monomului se atașează implicit coeficientul 1 sau -1. Polinoamele returnate sunt sortate descrescător după gradul lor, iar elementele cu coeficienți nuli sunt șterse din lista de monoame.
* Monoamele sunt construite din două variabile, primul fiind un număr întreg, care reprezintă gradul și a doua este număr zecimal care reprezintă coeficientul. Dacă prima variabilă este null sau negativ, i se va atribui variabilei valoare 1. Dacă coeficientul este null / NaN (not a number) / infinit, atunci i se va atribui variabilei valoarea 1,0.

**Clasa PolMath**

Graphical user interface, text, application, chat or text message

Description automatically generatedEste clasa responsabilă pentru implementarea operațiilor matematice asupra polinoamelor. Metodele *add()*, *subtract()*, *multiply()*, *divide()*, *derivativeOf()*, *integralOf()* returnez un nou polinom la fiecare apel, astfel parametrii de intrare sunt intacți (parametrii sunt marcați ca final). Polinoamele sunt sortate descrescător după gradul lor înainte de a fi returnate de metode.

* Funcția *addMonom()* adăugă on monom unui polinom. Dacă termenul deja apare în polinom, atunci este modificat doar coeficientul termenului respectiv. În caz contrat este creat un nou monom și este adăugat listei de monoame. După fiecare inserare, pe obiectul de tip polinom este apelat metoda *refactor()* pentru eliminarea termenilor cu coeficienți nuli.
* Metoda *multiply()* este supraîncărcat deoarece un polinom se poate înmulți cu un alt polinom sau cu un monom.
* Metoda *divide()* se deosebește de celelalte prin faptul că returnează o listă cu două polinoame. Primul este câtul, al doilea este restul. Dacă unul dintre polinoame este gol (fără monoame) atunci metoda returnează o listă cu două polinoame goale.
* Metoda *integralOf()* returnează un polinom integrat indefinit fără constantă, aceasta fiind adăugată de clasa Controller ulterior ca șir de caracter înainte de a fi afișat pe interfața grafică.

**Graphical user interface, application

Description automatically generatedClasa Polynomial**

Această clasă are trei metode importante:

* Metoda *sort()* sortează termenii polinomului în ordine descrescătoare după gradul lor. Returnează referința la obiectul respectiv.
* Metoda *refactor()* elimină termenii cu coeficienți nuli ( coeficienți cu valori mai mici decât 0.001 – constantă definită în clasa CONSTANTS)
* Metoda *toString()* definește formatul în care polinomul apare pe interfața grafică. Dacă polinomul este gol (nu conține monoame) atunci metoda returnează ”0”. Formatul este următorul:

[Semn][Număr1 rotunjit la 2 zecimale]X^[Număr2 întreg]

Această rotunjire cauzează probleme discutate la capitolul următor.

# Rezultate

JUnit este unul dintre cele mai populare framework-uri de testare a unităților din ecosistemul Java. Versiunea JUnit 5 conține o serie de inovații interesante, cu scopul de a susține noile caracteristici din Java 8 și de mai sus, precum și de a permite multe stiluri diferite de testare.

Testare adunare:

|  |  |  |  |
| --- | --- | --- | --- |
| Cazuri testate | Caz specific | | Rezultat |
| Polinom gol+  Polinom gol | Null | “ ” | 0 |
| Polinom gol + Polinom | Null | X^2+8X^2 | +9.00X^2 |
| Polinom + Polinom | 0.2X^4+5.6X^1-X^100+.88X^23 | -8.88X^2+74.881X^6+  34X^45-X^7 | -1.00X^100+34.00X^45+0.88X^23-1.00X^7+74.88X^6+0.20X^4-8.88X^2+5.60X^1 |

Testare scădere:

|  |  |  |  |
| --- | --- | --- | --- |
| Cazuri testate | Caz specific | | Rezultat |
| Polinom gol –  Polinom gol | Null | ” ” | 0 |
| Polinom gol - Polinom | Null | X^2+8X^2 | -9.00X^2 |
| Polinom -Polinom | 0.2X^4+5.6X^1-X^100+.88X^23 | -8.88X^2+74.881X^6+  34X^45-X^7 | -1.00X^100-34.00X^45+0.88X^23+1.00X^7-74.88X^6+0.20X^4+8.88X^2+5.60X^1 |
| Polinom -Polinom | 5X^2+3X^0-58.77X^12-15.4X^25 | 5X^2+6X^0-58.769X^12-15.4X^25 | -0.00X^12-3.00X^0 |

Observație: pentru coeficienți mai mici decât 0.05 metoda *toString()* afișează 0.00. Această este o problemă care trebuie tratată în implementările viitoare.

Test multiplicare:

|  |  |  |  |
| --- | --- | --- | --- |
| Cazuri testate | Caz specific | | Rezultat |
| Polinom gol\*  Polinom gol | Null | ” ” | 0 |
| Polinom gol \* Polinom | Null | X^2+8X^2 | 0 |
| Polinom \*Polinom | 0.2X^4+5.6X^1-X^100+.88X^23 | -8.88X^2+74.881X^6  +34X^45-X^7 | -34.00X^145+1.00X^10774.88X^106+8.88X^102+29.92X^68+  6.80X^49+190.40X^46-0.88X^30+65.90X^29-7.81X^25-0.20X^11+14.98X^10-5.60X^8+419.33X^7-1.78X^6-49.73X^3 |

Test diviziune:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Cazuri testate | Caz specific | | Rezultat | |
| Null / polinom | „ ” | X^2+1X^0 | 0 | 0 |
| Polinom / null | 5.66X^1+X^4 | null | 0 | 0 |
| Polinom / polinom | X^2+5X^1+2X^0 | X^1-3X^0 | +1.00X^1+8.00X^0 | +22.00X^0 |

Test derivare:

|  |  |  |
| --- | --- | --- |
| Cazuri testate | Caz specific | Rezultat |
| Polinom gol | „ ” | 0 |
| Polinom | 5X^2+3X^0-58.77X^12-15.4X^25 | -385.00X^24-705.24X^11+10.00X^1 |

Test integrare:

|  |  |  |
| --- | --- | --- |
| Cazuri testate | Caz specific | Rezultat |
| Polinom gol | Null | 0 |
| Polinom | 0.2X^4+5.6X^1-X^100+.88X^23 | -0.01X^101+0.04X^24+0.04X^5+2.80X^2 |

# Concluzii

Calculatorul calculează cu succes operațiile cu polinoame. A trecut de faza de testare, tratând cazuri specifice cu nul și polinoame goale.

Funcționalități care ar putea fi incluse în proiect:

* o interacțiune mai ușoară între date și client ( eliminarea necesității termenului X^0 la câmpurile de intrare, un format de ieșire a polinoamelor mai ușor de citit de către utilizator )
* rezolvarea problemei cu afișarea numerelor zecimale mici ( folosirea tipului BigDecimal în loc de Double )

1. Bibliografie

*Apple Documentation Archive*. ( 2018, 04 06). Retrieved from Cocoa Core Competencies: https://developer.apple.com/library/archive/documentation/General/Conceptual/DevPedia-CocoaCore/MVC.html

*Oracle Java Documentations*. (n.d.). Retrieved from Oracle Help Center: https://docs.oracle.com/javase/7/docs/api/java/util/Collections.html#sort(java.util.List)

Team, I. E. (2021, April 22). *Indeed*. Retrieved from https://www.indeed.com/career-advice/career-development/use-case-diagram