DOCUMENTAȚIE

TEMA 1

Pașcan Daniela Maria

Grupa 9

Cuprins

1. Obiectivul temei.........................................................................................................................3
2. Analiza problemei, modelare, cazuri de utilizare.......................................................................3
3. Proiectare....................................................................................................................................6
4. Implementare..............................................................................................................................9
5. Rezultate, testare......................................................................................................................15
6. Concluzii..................................................................................................................................16
7. Bibliografie...............................................................................................................................17

1.Obiectivul temei

Dorim să implementăm un calculator de polinoame cu interfață grafică, unde utilizatorul poate introduce unul sau două polinoame și să aleagă dintr-o listă de operații pe cea pe care dorește să o efectueze, urmând să primească rezultatul tot sub formă de polinom.

Obiective secundare:

1*. Analiza problemei, modelare, scenariu, cazuri de utilizare-* aici se vor reprezenta cerițele funcționale și cele non-funcționale, precum și use-case-urile

2*. Proiectare-* în această secțiune se va discuta despre diagrama UML a proiectului și despre organizarea acestuia în pachete și clase

1. *Implementare-* aici se va descrie în detalui fiecare clasă și metodele importante ale acesteia
2. *Rezultate, testare*- la final vom discuta despre testarea unitară cu JUnit

2.Analiza problemei, modelare, cazuri de utilizare

**2.1.Analiza problemei**

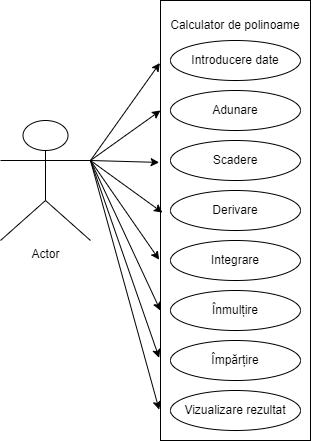
***Cerințe funcționale***

* Calculatorul de polinoame trebuie să permită utilizatorului să introducă polinoamele dorite
* Calculatorul trebuie să permită alegerea operației dintr-o listă de operații
* Calculatorul trebuie să adune două polinoame
* Calculatorul trebuie să scadă două polinoame
* Calculatorul trebuie să înmulțească două polinoame
* Calculatorul trebuie să împărțească două polinoame
* Calculatorul trebuie să deriveze un polinom
* Calculatorul trebuie să integreze un polinom
* Calculatorul trebuie să afișeze rezultatul după selectarea operației dorite

***Cerințe non-funcționale***

* Calculatorul de polinoame trebuie să fie intuitiv și simplu de utilizat de către utilizator
* Calculatorul trebuie să aibă o bună performanță ( timpul de răspuns să fie minim, utilizarea sa să meargă linear)
* Calculatorul trebuie sa aibă scalabilitate ( să poată lucra cu un volum mai mare de informație și să permită mărirea sau extinderea sa)
* Calculatorul trebuie să țină cont de integritatea datelor, să afișeze rezultate corecte, complete și consistente și să păstreze datele neschimbate indiferent de cat timp sunt stocate sau cat de des sunt accesate
* Calculatorul trebuie să fie ușor de întreținut și ușor de reparat
* Calculatorul trebuie să țină cont de securitatea datelor

**2.2.Cazuri de utilizare**

****

**Caz de utilizare**: Adunarea polinoamelor

Actor primar: Utilizatorul

Scenariu de succes:

1. Utilizatorul inserează două polinoame în interfața grafică
2. Utilizatorul apasă butonul specific pentru operația de adunare
3. Calculatorul de polinoame efectuează operația de adunare și o afișează în caseta cu rezultatul

Secvență alternativă:

* Utilizatorul inserează polinoamele într-un mod incorect(ex nu introduce coeficienții polinomului sub formă de variabilă întreagă)
* Scenariul se întoarce înapoi în punctul 1

**Caz de utilizare**: Scăderea polinoamelor

Actor primar: Utilizatorul

Scenariu de succes:

1. Utilizatorul inserează două polinoame în interfața grafică
2. Utilizatorul apasă butonul specific pentru operația de scădere
3. Calculatorul de polinoame efectuează operația de scădere și o afișează în caseta cu rezultatul

Secvență alternativă:

* Utilizatorul inserează polinoamele într-un mod incorect
* Scenariul se întoarce înapoi în punctul 1

**Caz de utilizare**: Înmulțirea polinoamelor

Actor primar: Utilizatorul

Scenariu de success:

1. Utilizatorul inserează două polinoame în interfața grafică
2. Utilizatorul apasă butonul specific pentru operația de înmulțire
3. Calculatorul de polinoame efectuează operația de înmulțire și o afișează în caseta cu rezultatul

Secvență alternativă:

* Utilizatorul inserează polinoamele într-un mod incorect
* Scenariul se întoarce înapoi în punctul 1

**Caz de utilizare**: Împărțirea polinoamelor

Actor primar: Utilizatorul

Scenariu de success:

1. Utilizatorul inserează două polinoame în interfața grafică
2. Utilizatorul apasă butonul specific pentru operația de împărțire
3. Calculatorul de polinoame efectuează operația de împărțire și o afișează în caseta cu rezultatul

Secvență alternativă:

* Utilizatorul inserează polinoamele într-un mod incorect
* Scenariul se întoarce înapoi în punctul 1

Secvență alternativă:

* Utilizatorul inserează al doilea polinom ca fiind 0
* În caseta cu rezultatul se va afișa un mesaj de eroare deoarece nu se poate efectua împărțirea la 0
* Scenariul se întoarce înapoi la pasul 1

Secvență alternativă:

* Utilizatorul introduce primul polinom cu gradul mai mic decât al doilea polinom introdus
* În caseta cu rezultatul se va afișa un mesaj de eroare deoarece nu putem efectua împărțirea unui polinom cu grad mai mic la un polinom cu grad mai mare
* Scenariul se întoarce înapoi la pasul 1

**Caz de utilizare**: Derivarea polinomului

Actor primar: Utilizatorul

Scenariu de success:

1. Utilizatorul inserează un polinom în interfața grafică
2. Utilizatorul apasă butonul specific pentru operația de derivare
3. Calculatorul de polinoame efectuează operația de derivare și o afișează în caseta cu rezultatul

Secvență alternativă:

* Utilizatorul inserează polinomul într-un mod incorect
* Scenariul se întoarce înapoi în punctul 1

**Caz de utilizare**: Integrarea polinomului

Actor primar: Utilizatorul

Scenariu de success:

1. Utilizatorul inserează un polinoame în interfața grafică
2. Utilizatorul apasă butonul specific pentru operația de integrare
3. Calculatorul de polinoame efectuează operația de integrare și o afișează în caseta cu rezultatul

Secvență alternativă:

* Utilizatorul inserează polinomul într-un mod incorect
* Scenariul se întoarce înapoi în punctul 1

3.Proiectare

Programarea orientată pe obiecte este o metodă de programare care organizează programul in funcție de date sau obiecte, în loc de funcții și logică. Un obiect poate fi definit ca un câmp de date care are atribute și comportament unic. Programarea orientată pe obiecte se concentrează mai degrabă pe obiectele pe care dezvoltatorii doresc să le manipuleze decât pe logica necesară pentru a le manipula.

Am început prin a determina pachetele de care avem nevoie pentru proiect și am grupat mai apoi clasele în pachete.

**3.1.Modelare**

***Designul sistemului***

Calculator de Polinoame

2x^2+3x+5

4x^3-5x-1

Operație

Rezultat

**3.2.Organizarea în pachete**

INTERFAȚA

OPERAȚII

MODEL

Pachetul pentru interfață conține clasele ce formează împreună interfața grafică cu utilizatorul.

Pachetul cu operații conține toate clasele ce implementează operațiile matematice pe polinoame, de asemenea, mai conține și metoda responsabilă de convertirea unui polinom într-un string.

Pachetul model conține clasele ce modelează aplicația, în cazul de față, avem clasa Polinom ce folosește un TreeMap.

**3.3.Organizarea pe clase**

CONTROLLER

VIEW

INTERFAȚA

FUNCȚII

MAINCLASS

OPERAȚII

POLINOM

MODEL

**3.4.Diagrama UML de clase**

Diagram

Description automatically generated

4.Implementare

***Clasa Polinom***

Această clasă reprezintă un polinom matematic, reprezentat ca o colecție de monoame (termeni algebrici de grad unic). Polinomul este stocat într-un obiect TreeMap care este o colecție sortată de chei și valori. Cheile sunt gradul fiecărui monom, iar valorile sunt coeficienții corespunzători.

Constructorul acestei clase primește un obiect TreeMap care conține deja termenii polinomului sau creează un polinom gol cu un singur monom, gradul 0 cu coeficientul 0.

Avem in această clasă metoda "adaugaMonom", care permite adăugarea unui monom nou în polinom, specificând gradul și coeficientul acestuia.

Tot aici avem și o metodă de creare a polinoamelor care folosește regex-ul. O expresie regex este o secvență de caractere care specifică un model de potrivire în text. De obicei, astfel de modele sunt folosite de algoritmii de căutare a șirurilor de caractere pentru operațiunile de găsire sau de găsire și înlocuire pe șiruri de caractere sau pentru validarea intrărilor.

În acest proiect, folosesc o expresie regex pentru a forma monoamele ce urmează să fie adăugate în polinom și încă două expresii pentru a scoate din monom coeficientul și rangul.

La final, metoda getPolinom() ne returnează polinomul sub formă de TreeMap.

***Clasa Funcții***

Clasa Funcții definește un polinom și implementează diferite operații matematice pe polinoame, cum ar fi adunarea, scăderea, înmulțirea, derivarea, integrarea și reducerea unui polinom.

Constructorul acestei clase primește un obiect TreeMap care conține deja termenii polinomului sau creează un polinom gol cu un singur monom, gradul 0 cu coeficientul 0.

Se regăsesc următoarele metode:

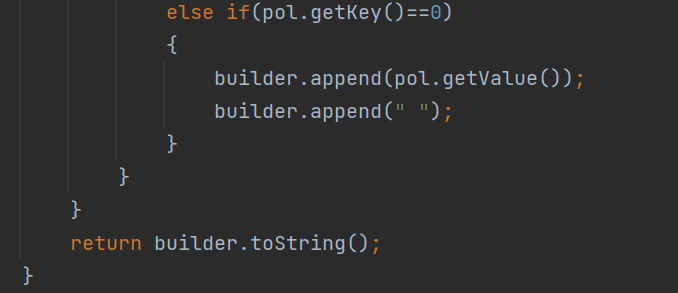
* **afisarePolinom**

Metoda primește un obiect de tip Polinom ca parametru și returnează un șir de caractere care reprezintă polinomul.

În cadrul metodei, se creează un obiect StringBuilder care va fi folosit pentru a construi șirul de caractere care va fi returnat. Apoi, se parcurge polinomul obținut din obiectul Polinom primit ca parametru, folosind o iterație prin maparea de perechi cheie-valoare.

Text

Description automatically generated



* **adunaPolinoame**

Această metodă adună două obiecte de tipul Polinom. Se vor folosi două bucle for pentru a itera peste toate monoamele din cele două polinoame. În prima buclă, se adună valorile corespunzătoare acelorași monoame, iar în a doua buclă se adaugă toate monoamele care nu au fost adăugate în primul pas.

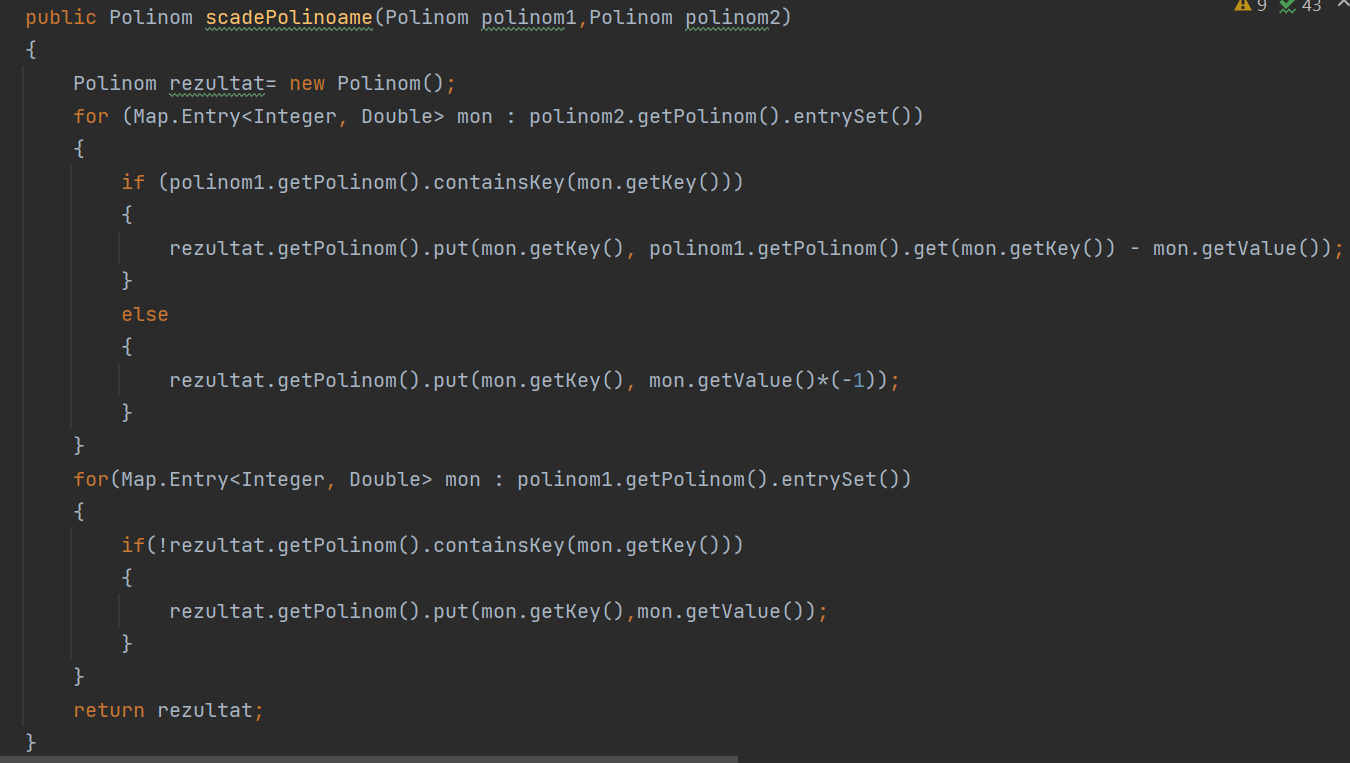
Dacă prin adunarea a două monoame de același grad se va obține zero, atunci monomul cu acel grad va fi eliminat din polinomul rezultat.

Text

Description automatically generated

* **scadePolinoame**

Metoda are același principiu de funcționare ca și metoda folosită pentru adunarea polinoamelor.



* **inmultestePolinoame**

Această metodă implementează operația de înmulțire a două polinoame. Mai întâi, verifică dacă unul dintre polinoame este un polinom nul și în acest caz returnează un polinom nul ca rezultat al înmulțirii. În caz contrar, metoda iterează prin toate monoamele din primul polinom și pentru fiecare dintre acestea, înmulțește cu fiecare monom din al doilea polinom, rezultând un nou polinom.

Aceste polinoame rezultate sunt adunate utilizând metoda adunaPolinoame(), care grupează monoamele de grad egal. La final, metoda returnează polinomul rezultat al înmulțirii.

Text

Description automatically generated

* **derivarePolinom**

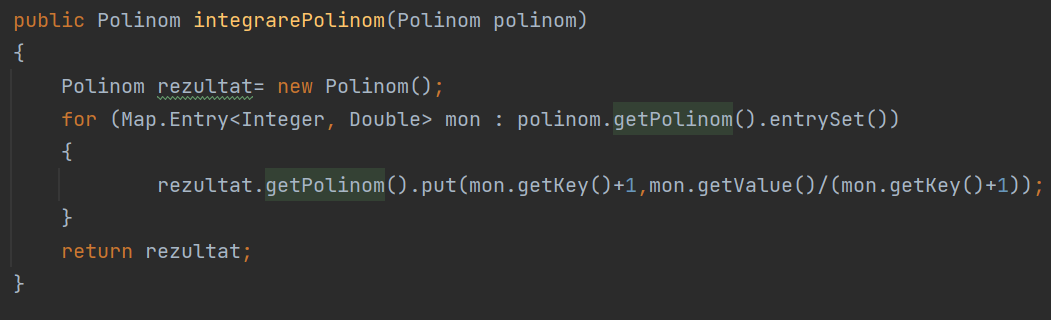
Această metodă calculează derivata polinomului dat și returnează rezultatul sub formă de un alt polinom. Pentru fiecare monom din polinom, metoda calculează noul grad și coeficient, conform regulilor de derivare a funcțiilor polinomiale. Monomul constant, de grad 0, va fi eliminat din polinomul rezultat, deoarece derivata sa este zero.

Text

Description automatically generated

* **integrarePolinom**

Această metodă primește un obiect Polinom ca parametru și returnează o altă instanță de Polinom, care este rezultatul integrării polinomului de intrare. În implementare, pentru fiecare monom din polinomul de intrare, se calculează integrala sa prin adăugarea monomului rezultat la gradul monomului original plus unu și împărțirea valorii sale cu gradul plus unu. Astfel, metoda returnează polinomul integrat.



* **reducePolinom**

Această metodă primește ca parametru un polinom și elimină toate monoamele cu coeficientul egal cu 0, apoi returnează un polinom rezultat din această operație.

Text

Description automatically generated

* **invPolinom**

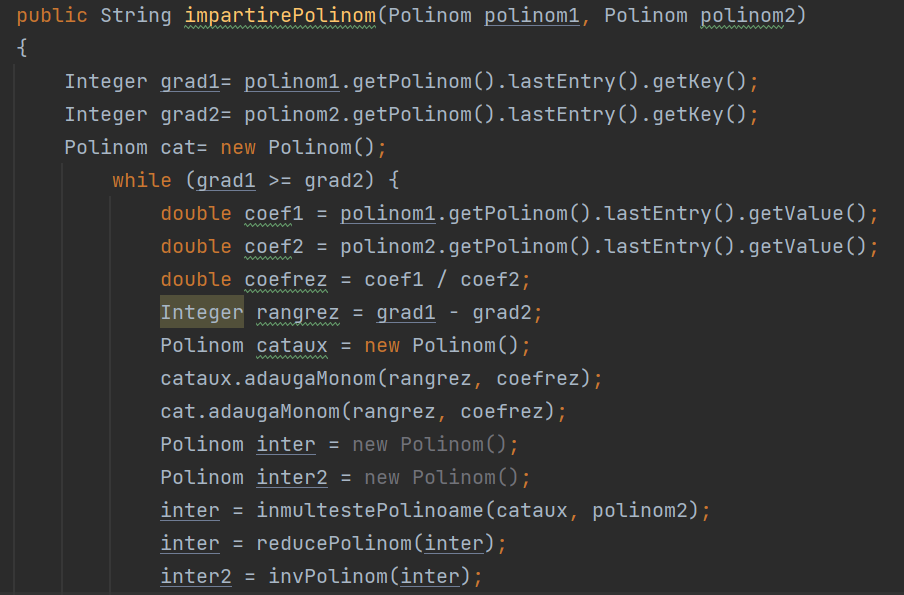
Această metodă primește un obiect Polinom și returnează același obiect cu toți coeficienții înmulțiți cu -1, ceea ce duce la inversarea semnelor tuturor termenilor din polinom.

Text

Description automatically generated

* **impartirePolinom**

Această metodă realizează împărțirea polinoamelor prin aplicarea algoritmului de împărțire polinomială. Împărțirea se realizează pe baza gradelor polinoamelor, iar la fiecare pas se determină coeficientul de la cel mai mare termen al polinomului împărțit prin coeficientul de la cel mai mare termen al polinomului împărțitor, se calculează termenul corespunzător al polinomului rezultat și se face scădere între polinoame. Acest proces continuă până când gradul polinomului rămas de împărțit devine mai mic decât gradul polinomului împărțitor. La final, se returnează un șir de caractere reprezentând rezultatul împărțirii, obținut prin concatenarea polinomului cât și a restului.



Text

Description automatically generated

* **impartireString**

Această metodă concatenează stringul câtului și stringul restului, obținute după efectuarea împărțirii.

***Clasa View***

Această clasă face parte din pachetul care formează interfața grafică cu utilizatorul.

O interfață grafică este o fereastră ce îi acordă utilizatorului accesul la mai multe opțiuni sau interacțiuni cu computerul prin intermediul butoanelor, căsuțelor de text, a listelor etc. Această interfață este controlată prin mouse , tastatură sau chiar cu degetul dacă ecranul este interactiv.

În cadrul acestui proiect, utilizatorul poate comunica cu programul pe care l-am implementat prin introducerea de polinoame în căsuțe text. După introducerea acestora, utilizatorul are posibilitatea de a alege ce operație dorește să aplice celor două polinoame. De exemplu, dacă polinoamele sunt introduse corect, acestea pot fi adunate sau împărțite, însă poate fi introdus și doar un singur polinom care mai apoi să fie derivat sau integrat.

După ce butonul pentru o operație este apăsat, în căsuța text rezervată rezultatului va fi afișat polinomul rezultat. Se poate mai apoi apăsa pe orice alt buton, fapt care va schimba rezultatul din căsuță în cel corespunzător noii operații alese.

Pentru dezvoltarea acestei clase am folosit JAVA Swing.

Am creat mai întâi o metodă care deschide fereastra interfeței, iar în această metodă sunt apelate alte două metode, una pentru pregătirea panoului ce conține casetele text în care se află cele două polinoame sau rezultatul și o altă metodă ce pregătește panoul ce conține butoanele pentru fiecare operație.

După toate implementările necesare am ajuns la fereastra următoare:

Diagram

Description automatically generated with low confidence

***Clasa Controller***

Această clasă are rolul de a lega pachetul cu operații de interfața grafică creată in clasa View. Aici, primim acțiunile de la mouse sau de la tastatură și afișăm rezultatul în funcție de comanda care a fost dată. Clasa conține mai multe metode, fiecare pentru a efectua o anumită operație matematică cu polinoame. Fiecare metodă extrage string-uri din câmpurile de introducere a datelor ale View și le transformă în obiecte Polinom, apoi efectuează operația corespunzătoare utilizând metodele din clasa Functii și afișează rezultatul în câmpul corespunzător din View.

Metoda actionPerformed() este o metodă din interfața ActionListener și este suprascrisă aici pentru a identifica comanda acțiunii (ADUNARE, SCADERE, etc.) și pentru a apela metoda corespunzătoare de efectuare a operației cu polinoame.

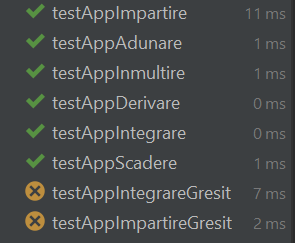
5.Rezultate, testare

JUnit este un framework open source pentru testare unitară în Java. Testarea unitară este procesul de a verifica fiecare componentă individuală a unei aplicații software pentru a verifica dacă aceasta funcționează conform specificațiilor sale. Scopul testării unitare este de a identifica și corecta erorile din timp, astfel încât să nu apară probleme mai târziu în ciclul de dezvoltare.

Prin testare unitară, verificăm funcționalitatea aplicației și ne asigurăm că rezultatele sunt cele la care ne așteptăm. Testarea este foarte importantă pentru a asigura integritatea programului pe care l-am implementat.

În acest proiect am folosit assertEquals pentru a verifica fiecare operație pe care am implementat-o. Am făcut atât teste ce dau rezultatul corect pentru a verifica dacă metodele funcționează cum trebuie, cât și teste pentru rezultate introduse greșit pentru a verifica dacă comparația s-a făcut corect.

Rezultatele testării unitare sunt următoarele:



6.Concluzii

Implementarea unui calculator de polinoame poate fi o modalitate excelentă de a învăța și de a aplica conceptele de programare orientată pe obiecte, de a învăța algoritmi și structuri de date și de a exersa abilitățile de programare în Java.

Din această temă am învățat să stăpânesc mai bine lucrul cu interfața grafică și modul în care se structurează un proiect pe pachete sub forma Model-View-Controller.

Pentru a dezvolta mai mult aplicația creată, s-ar putea adăuga mai multe acțiuni pe polinoame sau s-ar putea introduce, asemenea unui calculator digital, butoane pentru fiecare cifră și simbol, excluzând astfel de tot introducerea polinomului de la tastatură.

Am putea implementa și operații doar pe numere, secțiunea de polinoame fiind una specială.

7.Bibliografie

<https://www.tutorialspoint.com/junit/junit_overview.htm>

<https://www.diagrams.net/blog/move-diagrams-net>

<https://topic.alibabacloud.com/a/use-a-regular-expression-regex-to-match-polynomials-polynomial-regexpolynomial_1_31_32668196.html>

<https://regexr.com/>

<https://www.baeldung.com/java-matcher-find-vs-matches>

<https://blog.jetbrains.com/idea/2020/09/writing-tests-with-junit-5/>

<https://coolors.co/eac435-345995-03cea4-fb4d3d-ca1551>

<https://colorsui.com/view-palette/?palette=f5ea95-fc8e5b-fc5956-c93e4f-3d1734>

<https://www.rapidtables.com/web/color/RGB_Color.html>

<https://www.programiz.com/java-programming/treemap>