УНІВЕРСИТЕТ КОРОЛЯ ДАНИЛА

Кафедра інформаційних технологій та програмної інженерії

КУРСОВА РОБОТА

з Об’єктно-орієнтованого програмування

на тему: Бібліотечні засоби для розв'язування задач лінійної алгебри

Студента II курсу _____ групи

спеціальності 121 «Інженерія програмного забезпечення»

Дронь І.Ю

(прізвище та ініціали)

Керівник _____к.т.н., Пашкевич О.П.

Національна шкала _____

Кількість балів _____ Оцінка: ECTS _____

м. Івано-Франківськ 2021

_____Університет Короля Данила_____

назва вищого навчального закладу

Кафедра __інформаційних технологій та програмної інженерії_____

Дисципліна _Об’єктоно-орієнтоване програмування_____

Спеціальність 121 «Інженерія програмного забезпечення»

_____

Курс_____II_____ Група _____ІПЗс-2019_____ Семестр _____4_____

ЗАВДАННЯ

на курсовий проект (роботу) студента

_____Дронь Ігор Юрійович_____

Прізвище, ім’я, по-батькові

Тема проекту (роботи) Бібліотечні засоби для розв'язування задач лінійної алгебри

_____

_____

__

Строк здачі студентом закінченого проекту (роботи) _____

_____

_____

____

3. Вихідні дані до проекту (роботи)

_____

_____

____

4. Зміст роботи (перелік питань по розділах, які потрібно розробити)

_____

_____

____

5. Дата видачі завдання

_____

Анотація

В даній курсовій роботі була виконана розробка програми на мові програмування Java з використанням об’єктно-орієнтованого програмування , яка включила в себе :

1. Створення класових типів - чисельна квадратна матриця та одновимірний масив динамічного типу із змінними розмірами.

2. Створення перевантажених операцій:

3. Реалізація компонентних методів:

3.1 додавання та віднімання матриць;

3.2 додавання та віднімання векторів;

3.3 множення матриці на вектор;

3.4 обертання матриці;

3.5 порівняння матриць;

3.6 розв'язування системи лінійних рівнянь;

3.7 скалярне множення векторів.

3.8 обчислення визначника матриці;

3.9 обчислення довжини вектору.

Зміст

Вступ

Java — об'єктно-орієнтована мова програмування, випущена 1995 року компанією «Sun Microsystems» як основний компонент платформи Java. З 2009 року мовою займається компанія «Oracle», яка того року придбала «Sun Microsystems». В офіційній реалізації Java-програми компілюються у

байт-код, який при виконанні інтерпретується віртуальною машиною для конкретної платформи.

«Oracle» надає компілятор Java та віртуальну машину Java, які задовольняють специфікації Java Community Process, під ліцензією GNU General Public License.

Мова значно запозичила синтаксис із C і C++. Зокрема, взято за основу об'єктну модель C++, проте її модифіковано. Усунуто можливість появи деяких конфліктних ситуацій, що могли виникнути через помилки програміста та полегшено сам процес розробки об'єктно-орієнтованих програм. Ряд дій, які в C/C++ повинні здійснювати програмісти, доручено віртуальній машині. Передусім Java розроблялась як платформо-незалежна мова, тому вона має менше низькорівневих можливостей для роботи з апаратним забезпеченням, що в порівнянні, наприклад, з C++ зменшує швидкість роботи програм. За необхідності таких дій Java дозволяє викликати підпрограми, написані іншими мовами програмування.

Java вплинула на розвиток J++, що розроблялась компанією «Microsoft». Роботу над J++ було зупинено через судовий позов «Sun Microsystems», оскільки ця мова програмування була модифікацією Java. Пізніше в новій платформі «Microsoft» .NET випустили J#, щоб полегшити міграцію програмістів J++ або Java на нову платформу. З часом нова мова програмування C# стала основною мовою платформи, перейнявши багато чого з Java. J# востаннє включався в версію Microsoft Visual Studio 2005.

Мова сценаріїв JavaScript має схожу із Java назву і синтаксис, але не пов'язана із Java.

У створенні мови програмування Java було п'ять початкових цілей:

1. Синтаксис мови повинен бути «простим, об'єктно-орієнтовним та звичним».
2. Реалізація має бути «безвідмовною та безпечною».
3. Повинна зберегтися «незалежність від архітектури та переносність».
4. Висока продуктивність виконання
5. Мова має бути «інтерпретованою, багатопотоковою, із динамічним зв'язуванням модулів».

Динамічні структури за визначенням характеризуються відсутністю фізичної суміжності елементів структури в пам’яті, непостійністю і непередбачуваністю розміру (кількість елементів) структури в процесі її обробки.

Оскільки елементи динамічної структури розташовуються за не передбачуваними адресами пам’яті, адресу елемента такої структури не можна обчислити за адресою початкового або попереднього елемента. Для встановлення зв’язку між елементами динамічної структури використовуються покажчики, через які встановлюються явні зв’язки між елементами. Таке представлення даних в пам’яті називається зв’язним. Елемент динамічної структури складається з двох полів:

- інформаційного поля або поля даних, в якому містяться ті дані, заради яких і створюється структура;

- поле зв’язку, в якому міститься один або декілька покажчиків, які зв’язують даний елемент з іншими елементами структури.

Коли зв’язне представлення даних використовується для вирішення прикладної задачі, для кінцевого користувача „видимим” робиться тільки вміст інформаційного поля, а поле зв’язку використовується тільки програмістом-розробником. Переваги зв’язного представлення даних:

- можливість забезпечення значної змінності структур;
- розмір структури обмежується тільки доступним об’ємом машинної пам’яті;
- при зміні логічної послідовності елементів структури потрібно виконати не переміщення даних в пам’яті, а тільки корекцію покажчиків.

Разом з тим зв’язне представлення не позбавлене й недоліків, основні з яких:

- робота з покажчиками вимагає більш високої кваліфікації від програміста;
- на поля зв’язку витрачається додаткова пам’ять;
- доступ до елементів зв’язної структури може бути менш ефективним за часом.

Останній недолік є найбільш серйозним і саме ним обмежується застосування зв’язного представлення даних. Якщо в суміжному представленні даних для обчислення адреси будь-якого елемента у всіх випадках достатньо номера елемента і інформації, яка міститься в описі структури, то для зв’язного представлення адреса елемента не може бути обчислена з початкових даних. Опис зв’язної структури містить один або

декілька покажчиків, які дозволяють увійти до структури, далі пошук необхідного елемента виконується проходженням ланцюжком покажчиків від елемента до елемента. Тому зв’язне представлення практично ніколи не застосовується в задачах, де логічна структура даних має вигляд вектора або масиву – з доступом за номером елемента, але часто застосовується в задачах, де логічна структура вимагає іншої початкової інформації доступу (таблиці, списки, дерева і т.д.).

Я не вважаю цю програму актуальною на сьогоднішній день через велику кількість програм що реалізують окремі функції цієї програми ,проте як завдання для курсової роботи це в міру складна задачка на перевірку знань.

1. Переваги мови Java для написання програм

Простота, інтерпретація, перенесення, незалежність від архітектури. До простих мов програмування належать ті, які працюють з інтерпретатором (наприклад, Бейсік). Перші персональні комп’ютери поставлялися з

інтерпретатором Бейсіка. Сьогодні їхнє місце займають HTML і мови сценаріїв для Web. Вивчати і використовувати мови програмування, які компілюються, набагато складніше, ніж ті, які інтерпретуються. Тобто є мови програмування для професіоналів. Наприклад, C++, де використання покажчиків і керування пам'яттю є складними не тільки для початківців, але й для досвідчених програмістів. Одна стрічка програми, яка звертається до недозволеного місця в пам'яті, може спричинити до збоїв не тільки програми, але й комп'ютера загалом. Java − це мова, програми якою компілюються та інтерпретуються, і водночас вона має просту структуру мови високого рівня. Написана програма компілюється в проміжну форму − байткод. Пізніше ця програма виконується, тобто інтерпретується виконавчим середовищем Java. Байткод дуже відрізняється від машинного коду, який є послідовністю нулів та одиничок. Байткод − це набір інструкцій, які подібні до команд Асемблера. Машинний код комп'ютер виконує безпосередньо, а байткод потрібно інтерпретувати. Тому машинний код можна використати тільки на конкретній платформі, для якої його 8 скомпільовано. Байткод можна виконувати на довільній платформі, на якій є виконавче середовище Java. Саме ця можливість і робить програми на Java незалежними від архітектури. Так як байткоди є проміжною формою програми, то його інтерпретація вимагає незначних витрат. Байткод створено для машини, яка реально не існує. Цю машину називають віртуальною Java-машиною (JVM), вона існує тільки в пам'яті комп'ютера. Створення компілятором Java байткоду для неіснуючої машини − це тільки половина процесу, який забезпечує незалежність від архітектури. Другу частину виконує інтерпретатор Java, який виконує роль посередника між віртуальною Java-машиною та реальним комп'ютером.

Архітектура мови для розподіленого мережевого середовища. Головною вимогою щодо мови для роботи в розподіленому просторі комп’ютерів (наприклад, в Internet) − це можливість працювати на різнорідних і розподілених платформах.

Мова Java є пристосованою до перенесення завдяки підтримці стандартів IEEE для структур даних, наприклад, цілих чисел, чисел з плаваючою комою і рядків. До мови Java зачислено безпосередньо підтримку таких розповсюджених протоколів як FTP, HTTP, що забезпечує сумісність під час роботи в мережі. Java забезпечує розподілену роботу за допомогою механізму виклику віддалених методів (RMI), тобто дає змогу використовувати об’єкти, розташовані на локальних і віддалених машинах. Багатопоточність. У багатопоточних операційних системах для кожного застосування (процесу) надається окрема захищена область пам’яті, в якій зберігаються коди програми і дані. А час одного процесора квантується між цими процесами. З метою запуску процесу або переключення з одного на інший на рівні операційної системи необхідно виконати значний об’єм роботи. Тому для розробників прикладних програм спеціально створили "полегшену" версію системного процесу − потік. Найбільшою проблемою, пов’язаною з процесами і потоками, є їхнє функціонування під керівництвом конкретної операційної системи. Спеціалісти компанії Sun зробили потоки частиною мови програ- 9 мування. Тому багатопоточне застосування, написане мовою Java, працюватиме і в операційних середовищах Windows, Unix, MacOs. Висока продуктивність. Інтерпретатор Java може виконувати байткоди зі швидкістю, яка наближається до швидкості виконання коду, відкомпільованого до машинного формату, що досягається завдяки використовування

інтерпретатором багатьох потоків виконання. Наприклад, доки комп’ютер чекає на введення даних, фонові потоки можуть зайнятися очищенням пам’яті. Стійкість до помилок. Java − це мова строгого використання типів, що зумовлює зменшення числа помилок під час написання програми. У мові Java відбувається

автоматична перевірка виконання граничних умов під час роботи з масивами і стрічками, які в Java є класами. У Java немає арифметики покажчиків, а керування пам’яттю здійснюється автоматично. Програмний код, написаний мовою Java не може зіслатися на пам’ять поза простором програми або зробити помилку внаслідок вивільнення пам’яті і тим самим вичерпати всю пам’ять. Зазначимо, що у Java організовано процес автоматичного збирання сміття, тобто об’єктів, на які більше ніхто не вказує. Безпека. Функції забезпечення безпеки дуже важливі для розподілених мереж з безліччю вірусів, "троянських коней" і т. п. Для реалізації цієї мети розробники мови Java створили механізм, який отримав назву пісочниці (sandbox): - перевірку на рівні JVM; - захист на рівні мови; - інтерфейс Java Sucurity (цифрового підпису)

2.Огляд класів

2.1Матриці та дії над ними

```
class matrix {
    int  x=2, y=2;
    int[][] Matrix = new int[2][2];
    void Mfilling() {
        for (x = 0; x < 2; x++) {
            for (y = 0; y < 2; y++) {
                Matrix[x][y] = x + y;
                System.out.print(Matrix[x][y]);}
            System.out.println();  } }
   void Mobertanna() {
        int A = Matrix[0][0] * Matrix[1][1] - Matrix[0][1] * Matrix[1][0];
        System.out.println(A);
        int temp;
        temp = Matrix[0][0];
        Matrix[0][0] = Matrix[1][1];
        Matrix[1][1] = temp;
        Matrix[0][1] = (-Matrix[0][1]);
        Matrix[1][0] = (-Matrix[1][0]);
        for (x = 0; x < 2; x++) {
            for (y = 0; y < 2; y++) {
                Matrix[x][y] = Matrix[x][y] / A;
```

```
            System.out.print(Matrix[x][y]);}
        System.out.println();
}}}
```

В цьому класі я створюю матрицю 2x2 і заповнюю її значеннями використовуючи метод Mfiling який зразу виводить значення в консоль (якщо не заповнити матрицю подальші дії з нею будуть викликати помилки).

Метод Mobertanna створений для обертння матриці 2x2 він знаходить визначник матриці , матрицю доповнень після чого транспонує її і ділить елементи га визначник . Врезультаті ми отримуємо обернену матрицю

```
class Mplusminus  {
   public static void Mplus(matrix a, matrix b) {
      System.out.println("||");
      int [][] Msum = new int[2][2];
      for (int x = 0; x < 2; x++) {
         for (int y = 0; y < 2; y++) {
            Msum[x][y] = a.Matrix[x][y] + b.Matrix[x][y];
            System.out.print(Msum[x][y]);      }
         System.out.println(); }  }
   public static void Mminus(matrix a, matrix b) {
      System.out.println("||");
      int [][] Msum = new int[2][2];
      for (int x = 0; x < 2; x++) {
         for (int y = 0; y < 2; y++) {
            Msum[x][y] = a.Matrix[x][y] - b.Matrix[x][y];
```

```
            System.out.print(Msum[x][y]);
        }
        System.out.println();
    } }}
```

Клас Mplusminus створений для двох методів а саме : Mplus та Mminus.

Вони отримують два обєкти типу matrix і проводять над ними операції а саме:додавання відповідних елементів та віднімання відповідних елементів.Резулльтат виводиться в консоль.

```
class Mporiv{
  public static void Mpor(matrix a,matrix b){
     int temp1=0,x,y;
      if (a.x== b.x || a.y==b.y){
        for ( x = 0; x < 2; x++) {
           for ( y = 0; y < 2; y++) {
              if (a.Matrix[x][y]==b.Matrix[x][y]){
                 temp1++;
                 if (temp1 == 4){
                 System.out.println("Матриці "+a+" і "+b+" рівні");}
              }  }  }  }  }  }
```

Клас Mporiv має в собі метод Mpor що отримує два обєкти matrix і порівнює елементи матриць якщо їхні розміри одинакові . Цей метод вважає дві матриці рівними якщо кількість рівних елементів(temp1) рівна загальній кількості елементів в матриці.

2.2Вектори та дії над ними

```
class vector  {
   int[] v = new int[3];
void Vfiling (){
   for (int i = 0;i<v.length;i++){
      v [i]=i+i;
      System.out.print(v[i]);  }
   System.out.println();}
void Vlong (vector a){
   System.out.println("long vector:"+a.v.length);}}
```

Клас vector це шаблон для створення вектора який ми заповнюємо в медоді Vfiling і одразу виводимо результат в консоль.Також в цьому класі є метод що виводить кількість елементів в векторі.

```
class Vplusminus{
  public static void plus(vector a,vector b){
      int [] c = new int[2];
      for (int i = 0;i<c.length;i++){
         c [i]=a.v[i]+b.v[i];
         System.out.print(c[i]);    }
      System.out.println();  }
   public static void minus(vector a,vector b){
      int [] c = new int[2];
      for (int i = 0;i<c.length;i++){
         c [i]=a.v[i]-b.v[i];
         System.out.print(c[i]);    }
      System.out.println(); }}
```

Клас Vplusminus по аналогії з Mplusminus створений для методів що домдають чи віднімають відповідні елементи (plus і minus) паралельно виводячи результат в консоль.Ці методи отримують два обєкти типу vector.

```
class scalar{
   public static void Vscalar(vector a,vector b){
      vector c=new vector();
      for (int x = 0; x < a.v.length; x++) {
            c.v[x]=a.v[x]*b.v[x];
            System.out.print(c.v[x]+" ");
      } }}
```

Клас scalar створений для методу Vscalar що отримує два елементи типу vector . В методі я створив новий обєкт типу vector в який буду записувати результат множення відповідних елементів двох матриць і виводити результат в консоль

2.3 Змішані дії з матрицями і векторами

```
class linriv{
   public static void linr(vector b,matrix A){
      int d = A.Matrix[0][1]*A.Matrix[1][0]-A.Matrix[1][1]*A.Matrix[0][0];
      int x1 =(A.Matrix[0][1]*b.v[1]-A.Matrix[1][1]*b.v[0])/d;
      int x2 =(A.Matrix[1][0]*b.v[0]-A.Matrix[0][0]*b.v[1])/d;
      System.out.println(x1 +" "+ x2); }}
```

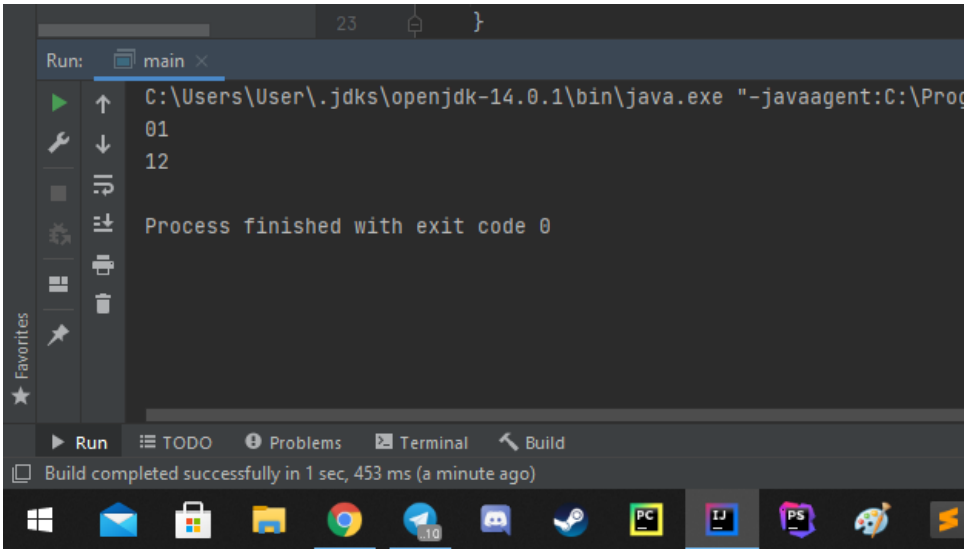
Клас linriv створений для методу linr що отримує два обєкти різних типів (vector і matrix).Першим етапом цей метод знаходить визначник матриці а другим знаходить елементи результуючої матриці і виводить її в консоль.

```
class VMmnogenna{
    public static void VM(vector a,matrix b){
        vector c = new vector();
        for (int x=0;x<b.x;x++) {
            c.v[x] = a.v[0] * b.Matrix[x][0] + a.v[1] * b.Matrix[x][1];   }
         System.out.println(c.v[0]+"||"+c.v[1]); }}
```

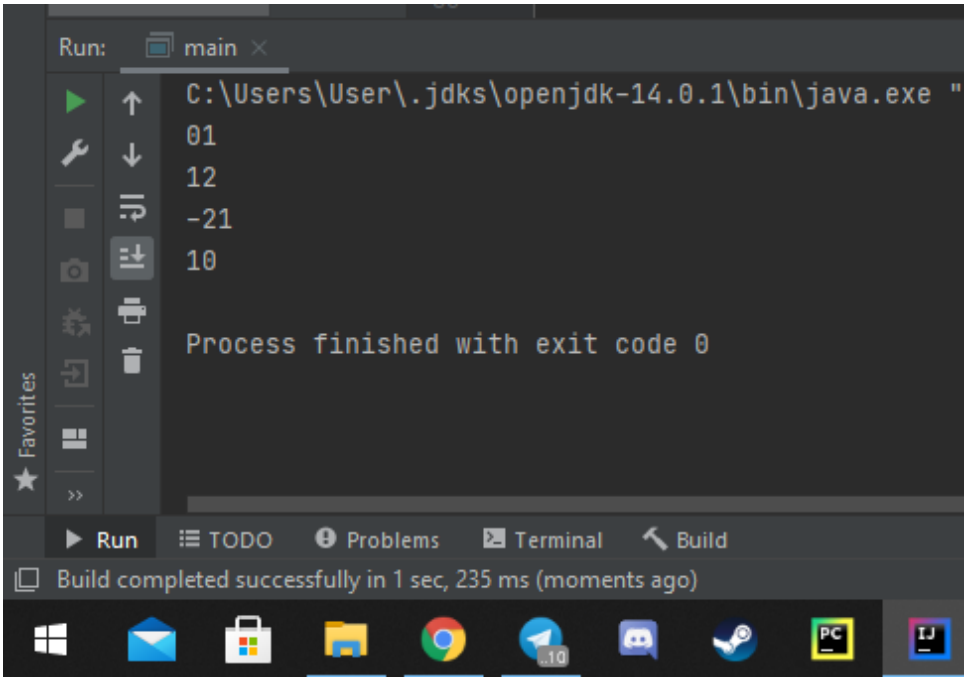
Клас VMmnogenna створений для методу VM що отримує два обєкти типу vector і matrix . В методі я створив обєкт типу vector в який я буду записувати результат множення вектора на матрицю.

Перевірка

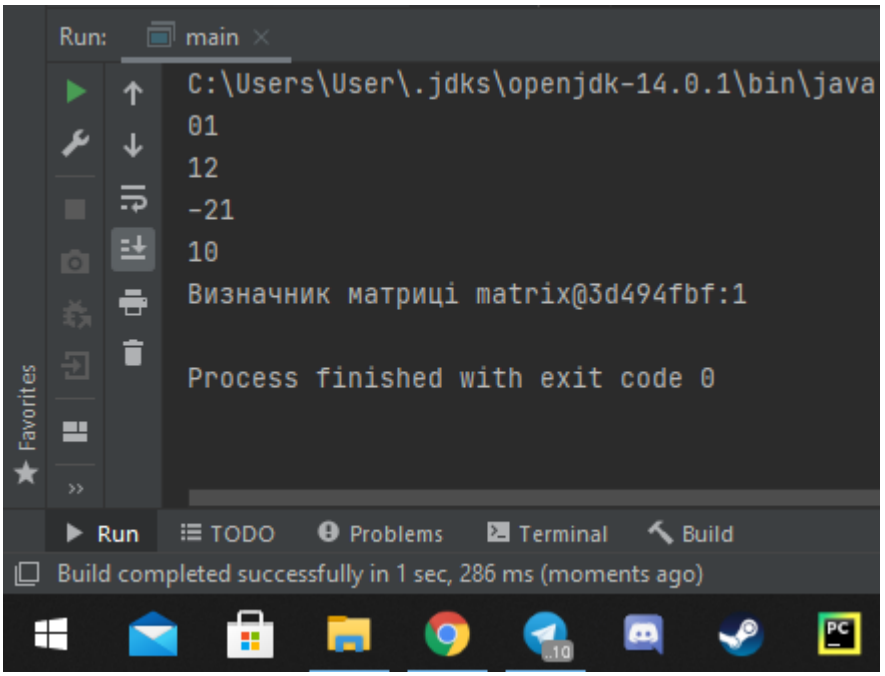
1.Створення матриці 2x2



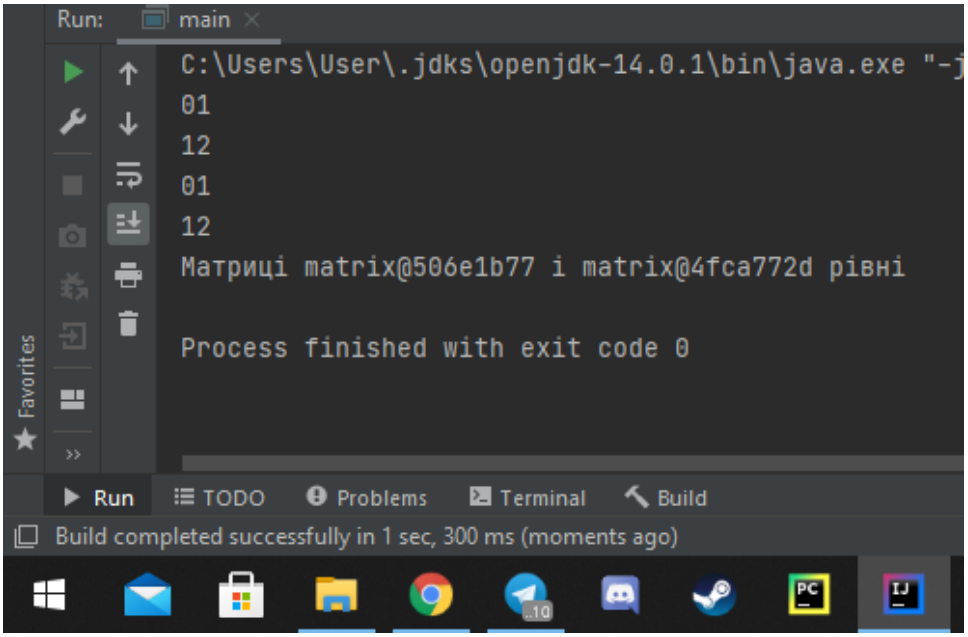
2.Обертання матриці



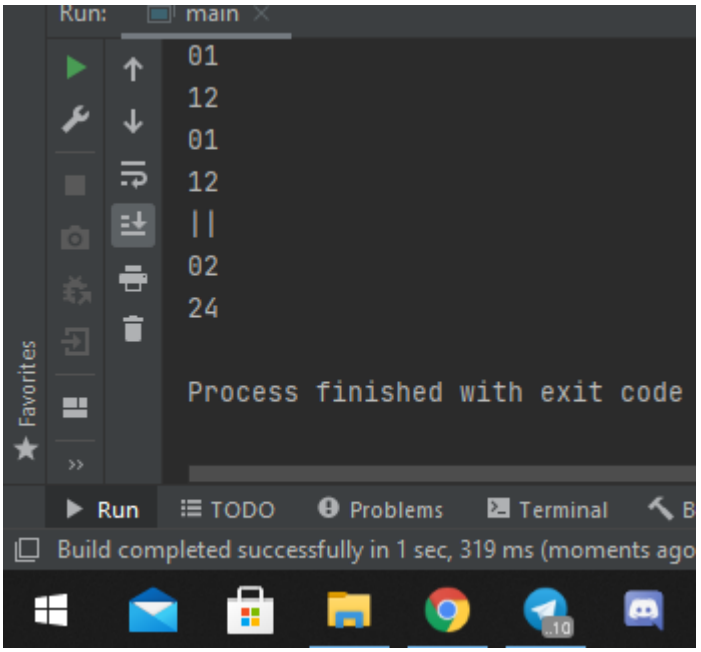
3.Визначник матриці



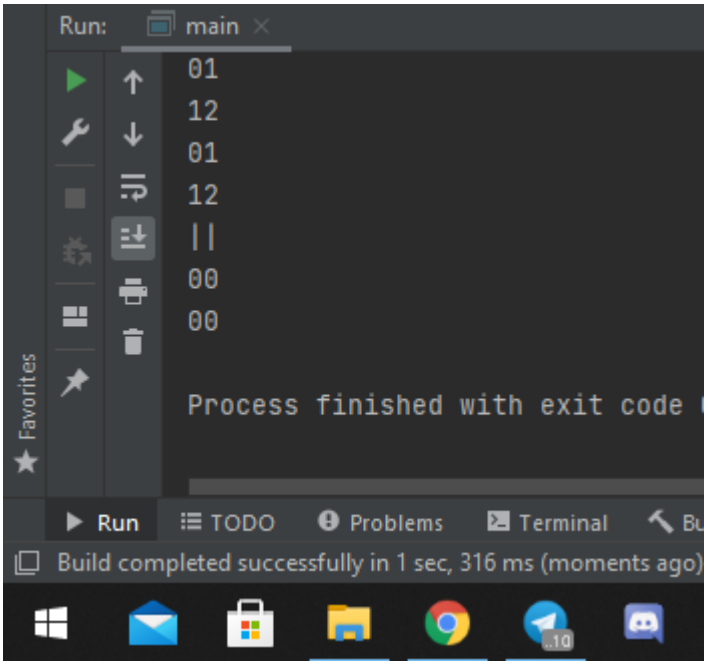
4.Порівняння матриць(матриця n1 заповнена тимиж значеннями)



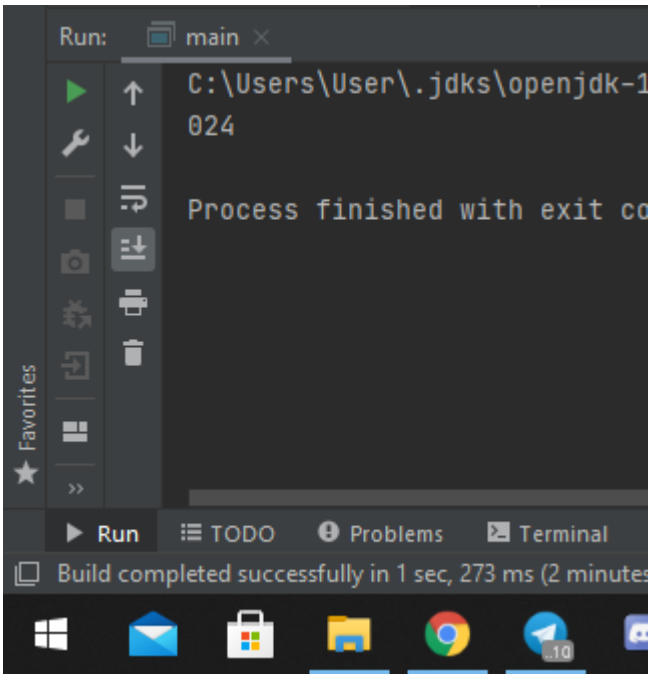
5.Додавання матриць



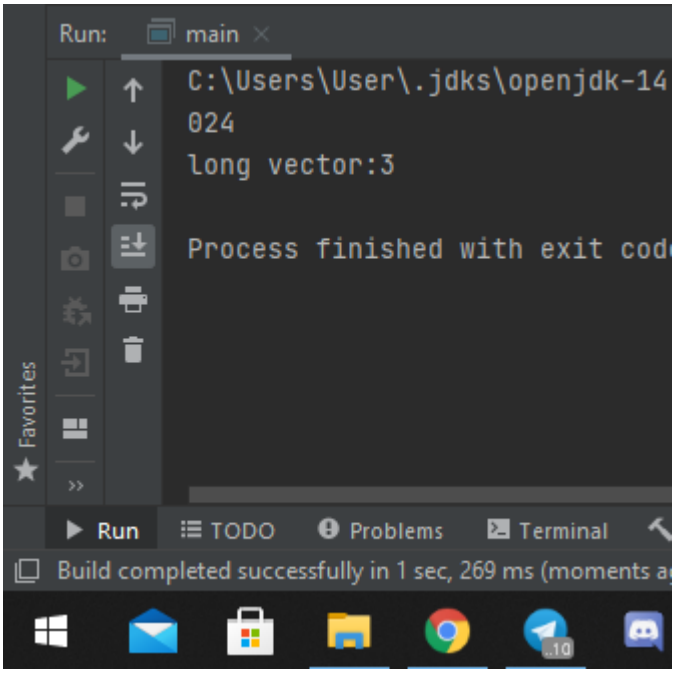
6.Віднімання матриць



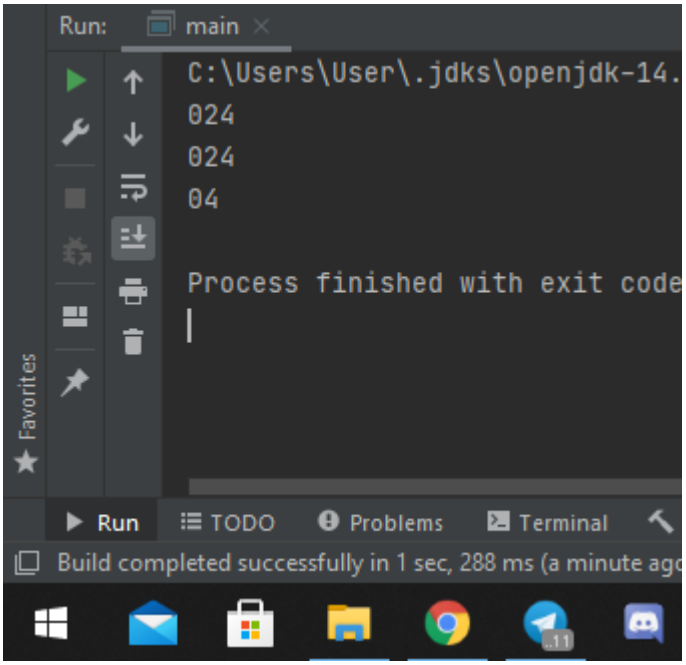
7.Створення вектору



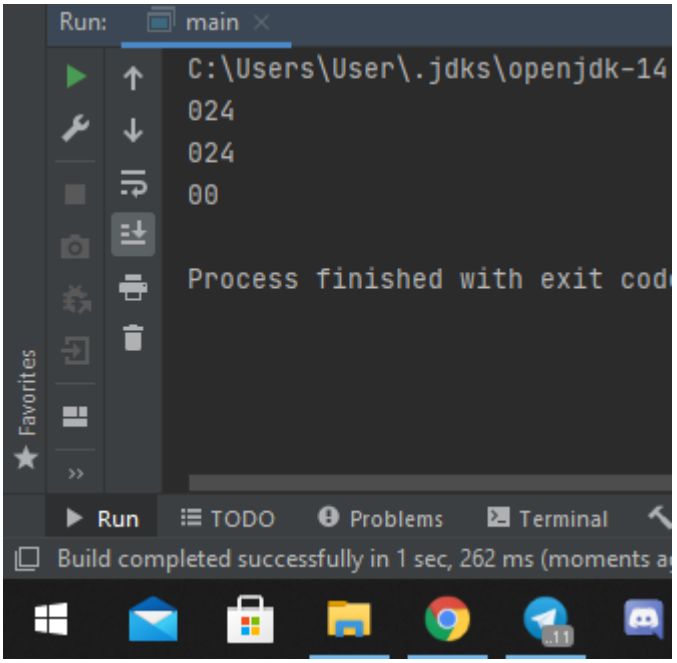
8.Кількість елементів в векторі



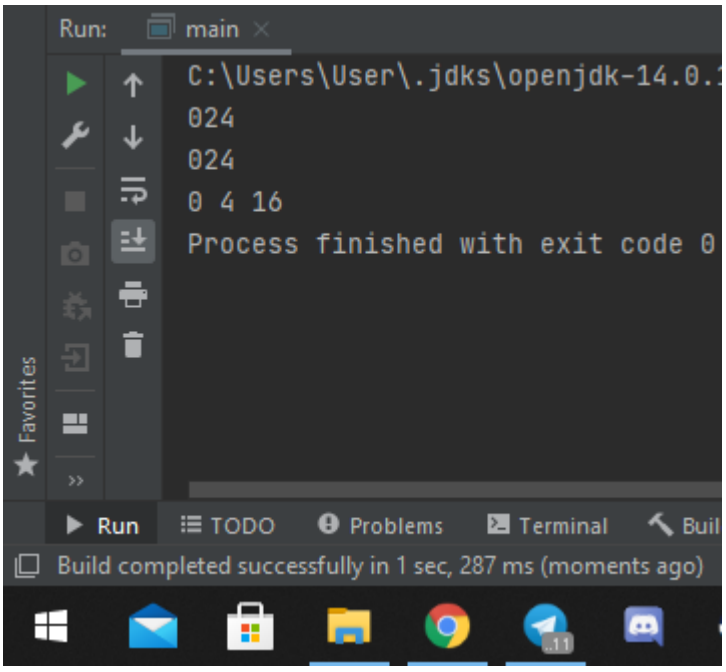
9.Додавання векторів



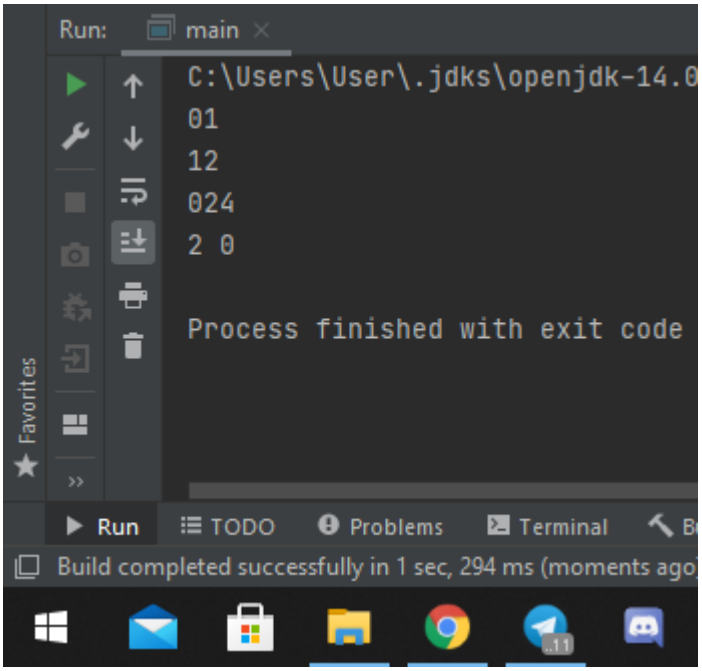
10.Віднімання векторів



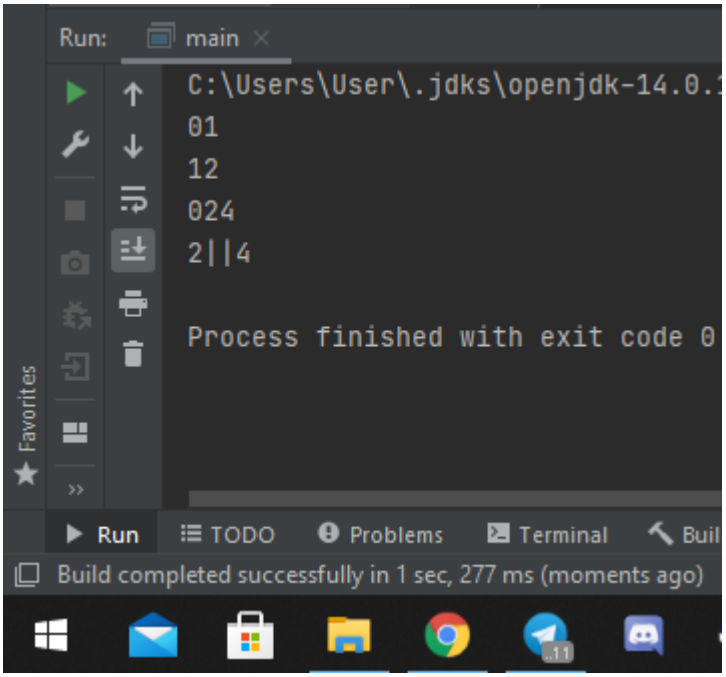
11.Множення векторів



12.Лінійне рівняння



13.Множення вектора на матрицю



Інструкція користувача

Після створення нових матриць та векторів їх обовязково потрібно заповнити методом Mfilling/ Vfiling.

Список функцій програми:

n.Mobertanna()--обертає матрицю n;

Mplusminus.Mplus(n, n1)--додає дві матриці;
Mplusminus.Mminus(n1,n)--віднімає матриці;

v.Vlong(v)--показує кількість елементів в векторі;

linriv.linr(v,n)--розвязує систуму лінійних рівнянь де v-вільні члени, а n-це матриця звільними коефіцієнтами;

VMmnogenna.VM(v,n)--множить матрицю на вектор;

Vplusminus.plus(v,v1)--додає вектори;
Vplusminus.minus(v,v1)--віднімає вектори;

Висновок

В ході курсової роботи я на практиці застосував свої знання з об’єктно-орієнтованого програмування. Поглибив розуміння основних принципів об’єктно-орієнтованої ідеології програмування. Відпрацював на практиці основні методи та засоби об’єктно-орієнтованого програмування. Навчився розробляти ієрархію класів. Практично застосував такі поняття, як інкапсуляція, наслідування, перевантаження та перевизначення функцій. Набув практичних навиків роботи.

Використані джерела

В.В. Подбєльский, С.С. Фомін “Програмування на мові Сі”

М. Уейт, С. Прата, Д. Мартін “Мова Сі”

П. Кіммел “Borland C++ 5”

Джесс Ліберті “Освой самостоятельно C++ за 21день”

Гіберт Шілдт “Язык С для проффесионалов ”

Эккель Б. Философия Java.: 4-е изд. – СПб.: Питер, 2009. – 640 с.

Хорстманн К. С., Корнелл Г. Библиотека профессионала. Java 2. Том 1. Основы, 7-е изд.: Пер. с англ. – М.: Издательский дом "Вильямс", 2007. – 896 с.

Хорстманн, К., С., Корнелл, Г. Библиотека профессионала. Java 2. Том 2. Тонкости программирования, 7-е изд.: Пер. с англ. – М.: Издательский дом "Вильямс", 2007. – 1168 с.

Шилдт Г. Полный справочнык по Java. Java SE 6 Edition, 7-е изд.: Пер. с англ. – М.: Издательский дом "Вильямс", 2007. – 1034 с.

Флэнаган Д. Java. Справочник, 4-е изд.: Пер. с англ. – Спб.:

Абельсон Х. Структура и интерпретация компьютерных программ / Абельсон Х., Джеральд Дж. С., Сассман Дж. / М.: Добросвет, 2006. – 608 с.

Хорстманн К. С., Корнелл Г. Библиотека профессионала. Java 2. Том 1. Основы, 7-е изд.: Пер. с англ. – М.: Издательский дом "Вильямс", 2007. – 896 с.

Хорстманн, К., С., Корнелл, Г. Библиотека профессионала. Java 2. Том 2. Тонкости программирования, 7-е изд.: Пер. с англ. – М.: Издательский дом "Вильямс", 2007. – 1168 с.

Зубов В.С. Справочник программиста. Базовые методы решения графовых задач и сортировки.- М.: "Филинъ", 1999.- 256 с.

В.В.Белов, Е.М.Воробьев, В.Е.Шаталов. Теория графов.- М.: Высш. шк., 1976.- 392 с.

Оре О. Теория графов.- М.: 1980.

А.А.Зыков. Основы теории графов.- М.: Наука, 1987.- 384 с.

Ю.Н.Тюрин, А.А.Макаров. Анализ данных на компьютере.- М.: ИНФРА-М, Финансы и стат., 1995.- 384 с.

Л.И.Турчак. Основы численных методов.- М.: Наука, 1987.- 320 с.

В.Н.Нефедов, В.А.Осипова. Курс дискретной математики: Учеб. пос.- М.: Изд-во МАИ, 1992.- 264 с.

Bloch J. Effective Java: Programming Language Guide. Publisher: Addison Wesley, 2001. – 180 p.

https://www.bestreferat.ru/referat-141723.html

https://www.bibliofond.ru/view.aspx?id=702020

Додаток А

```
public class main {
   public static void main(String args[]) {
       matrix n = new matrix();
        n.Mfilling();
     // matrix n1 = new matrix();
      //n1.Mfilling();
       //Mporiv.Mpor(n,n1);
      // Mplusminus.Mplus(n, n1);
      // Mplusminus.Mminus(n1,n);
     vector v = new vector();
      v.Vfiling();
      //v.Vlong(v);
    // linriv.linr(v,n);
```

```
        // vector v1 = new vector();
        // v1.Vfiling();
        // scalar.Vscalar(v,v1);
       //  VMmnogenna.VM(v,n);
        // Vplusminus.plus(v,v1);
       //  Vplusminus.minus(v,v1);
    }
}


class Mporiv{
   public static void Mpor(matrix a,matrix b){

       int temp1=0,x,y;
        if (a.x== b.x || a.y==b.y){
            for ( x = 0; x < 2; x++) {
                for ( y = 0; y < 2; y++) {

                    if (a.Matrix[x][y]==b.Matrix[x][y]){
                        temp1++;
                        if (temp1 == 4){
                        System.out.println("Матриці "+a+" і "+b+" рівні");}

                    }

                    }
                }
```

```
        }
    }
}


class matrix {
    int  x=2, y=2;
    int[][] Matrix = new int[2][2];

    void Mfilling() {

        for (x = 0; x < 2; x++) {
            for (y = 0; y < 2; y++) {
                Matrix[x][y] = x + y;
                System.out.print(Matrix[x][y]);
            }
            System.out.println();
        }

    }

    void Mobertanna() {
        int A = Matrix[0][0] * Matrix[1][1] - Matrix[0][1] * Matrix[1][0];

        int temp;
        temp = Matrix[0][0];
        Matrix[0][0] = Matrix[1][1];
        Matrix[1][1] = temp;
```

```
        Matrix[0][1] = (-Matrix[0][1]);
        Matrix[1][0] = (-Matrix[1][0]);

        for (x = 0; x < 2; x++) {
            for (y = 0; y < 2; y++) {
                Matrix[x][y] = Matrix[x][y] / A;
                System.out.print(Matrix[x][y]);
            }
            System.out.println();
        }

    }


}

class Mplusminus  {
    public static void Mplus(matrix a, matrix b) {
        System.out.println("||");
        int [][] Msum = new int[2][2];
        for (int x = 0; x < 2; x++) {
            for (int y = 0; y < 2; y++) {
                Msum[x][y] = a.Matrix[x][y] + b.Matrix[x][y];
                System.out.print(Msum[x][y]);
            }
            System.out.println();
        }
```

```
    }
    public static void Mminus(matrix a, matrix b) {
        System.out.println("||");

        int [][] Msum = new int[2][2];
        for (int x = 0; x < 2; x++) {
            for (int y = 0; y < 2; y++) {
                Msum[x][y] = a.Matrix[x][y] - b.Matrix[x][y];
                System.out.print(Msum[x][y]);
            }
            System.out.println();
        }
    }
}

class arraylist  {
    ArrayList<Integer> a = new ArrayList();

    void arrayint() {
        a.add(1);
        a.add(12);

        System.out.println(a.get(1));
        System.out.println(a.size());


    }
}
```

```
class vector  {
    int[] v = new int[3];
void Vfiling (){
    for (int i = 0;i<v.length;i++){
        v [i]=i+i;
        System.out.print(v[i]);
    }
    System.out.println();
}
void Vlong (vector a){
    System.out.println("long vector:"+a.v.length);
}
}
class Vplusminus{
   public static void plus(vector a,vector b){
        int [] c = new int[2];
        for (int i = 0;i<c.length;i++){
            c [i]=a.v[i]+b.v[i];
            System.out.print(c[i]);
        }
        System.out.println();
    }
    public static void minus(vector a,vector b){
        int [] c = new int[2];
        for (int i = 0;i<c.length;i++){
            c [i]=a.v[i]-b.v[i];
```

```
            System.out.print(c[i]);
        }
        System.out.println();
    }
}

class VMmnogenna{
    public static void VM(vector a,matrix b){
        vector c = new vector();
        for (int x=0;x<b.x;x++) {
            c.v[x] = a.v[0] * b.Matrix[x][0] + a.v[1] * b.Matrix[x][1];
        }
        System.out.println(c.v[0]+"||"+c.v[1]);

    }
}

class linriv{
    public static void linr(vector b,matrix A){
        int d = A.Matrix[0][1]*A.Matrix[1][0]-A.Matrix[1][1]*A.Matrix[0][0];
        int x1 =(A.Matrix[0][1]*b.v[1]-A.Matrix[1][1]*b.v[0])/d;
        int x2 =(A.Matrix[1][0]*b.v[0]-A.Matrix[0][0]*b.v[1])/d;
        System.out.println(x1 +" "+ x2);
    }
}
class vuznachnuk{
    public static void vuz(matrix A){
```

```
        int d = A.Matrix[0][1]*A.Matrix[1][0]-A.Matrix[1][1]*A.Matrix[0][0];
        System.out.println("Визначник матриці "+A+":"+d);
    }
}
class scalar{
    public static void Vscalar(vector a,vector b){
        vector c=new vector();
        for (int x = 0; x < a.v.length; x++) {
            c.v[x]=a.v[x]*b.v[x];
            System.out.print(c.v[x]+" ");
        }
    }
}
```