|  |  |
| --- | --- |
|  | **Министерство науки и высшего образования Российской Федерации**  **Федеральное государственное бюджетное образовательное учреждение**  **высшего образования**  **«Московский государственный технический университет**  **имени Н.Э. Баумана**  **(национальный исследовательский университет)»**  **(МГТУ им. Н.Э. Баумана)** |

ФАКУЛЬТЕТ \_\_\_\_ИНФОРМАТИКА И СИСТЕМЫ УПРАВЛЕНИЯ\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

КАФЕДРА \_\_\_\_\_\_КОМПЬЮТЕРНЫЕ СИСТЕМЫ И СЕТИ\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

НАПРАВЛЕНИЕ ПОДГОТОВКИ **09.04.01 Информатика и вычислительная техника**

МАГИСТЕРСКАЯ ПРОГРАММА **09.04.01/12 Интеллектуальный анализ больших**

**данных в системах поддержки принятия решений.**

**Отчет**

**по лабораторной работе №** 3

**Название:** Классы. Наследование. Полиморфизм

**Дисциплина:** Языки программирования для работы с большими данными

Студент \_\_\_ИУ6-23М\_\_\_\_ **\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ \_\_\_**А.Д. Зеленский**\_\_\_**

(Группа) (Подпись, дата) (И.О.Фамилия)

Руководитель **\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ \_\_**П. В. Степанов**\_\_\_\_**

(Подпись, дата) (И.О.Фамилия)

**Цель:** ознакомиться с базовыми принципами языка Java для работы с большими данными.

**Вариант 1:**

1. Определить класс Комплекс. Класс должен содержать несколько конструкторов. Реализовать методы для сложения, вычитания, умножения, деления, присваивания комплексных чисел. Создать два вектора размерности n из комплексных координат. Передать их в метод, который выполнит их сложение.
2. Определить класс Квадратное уравнение. Класс должен содержать несколько конструкторов. Реализовать методы для поиска корней, экстремумов, а также интервалов убывания/возрастания. Создать массив объектов и определить наибольшие и наименьшие по значению корни.

**Решение:**

Ниже представлен код класса первой программы, а на рисунке 1 – результат работы программы.

class Complex {  
 public double a,bi;  
 public Complex(){}  
 public Complex(double a1,double b1){  
 this.a=a1;  
 this.bi=b1;  
 }  
 public Complex(int a1,double b1){  
 this.a=a1;  
 this.bi=b1;  
 }  
 public Complex(double a1,int b1){  
 this.a=a1;  
 this.bi=b1;  
 }  
 public Complex(int a1,int b1){  
 this.a=a1;  
 this.bi=b1;  
 }  
 public Complex getPlus (Complex b){  
 Complex c= new Complex();  
 c.a=a+b.a;  
 c.bi=bi+b.bi;  
 return (c);  
 }  
 public Complex Div( Complex cB ) {  
 Complex div = new Complex();  
 double dR, dDen;  
  
 if(Math.abs( cB.a ) >= Math.abs( cB.bi )) {  
 dR = cB.bi/cB.a;  
 dDen = cB.a + dR\*cB.bi;  
 div.a = (a + dR\*bi)/dDen;  
 div.bi = (bi - dR\*a)/dDen;  
 } else {  
 dR = cB.a/cB.bi;  
 dDen = cB.bi + dR\*cB.a;  
 div.a = (dR\*a + bi)/dDen;  
 div.bi = (dR\*bi - a)/dDen;  
 }  
 return (div);  
 }  
  
 public Complex Mult( Complex cB ) {  
 Complex prod = new Complex();  
 prod.a = a\*cB.a - bi\*cB.bi;  
 prod.bi = bi\*cB.a + a\*cB.bi;  
 return (prod);  
 }  
 public String toString() {  
 return "Complex{" + "A=" + this.a + ", bI=" + this.bi +"}";  
 }  
}

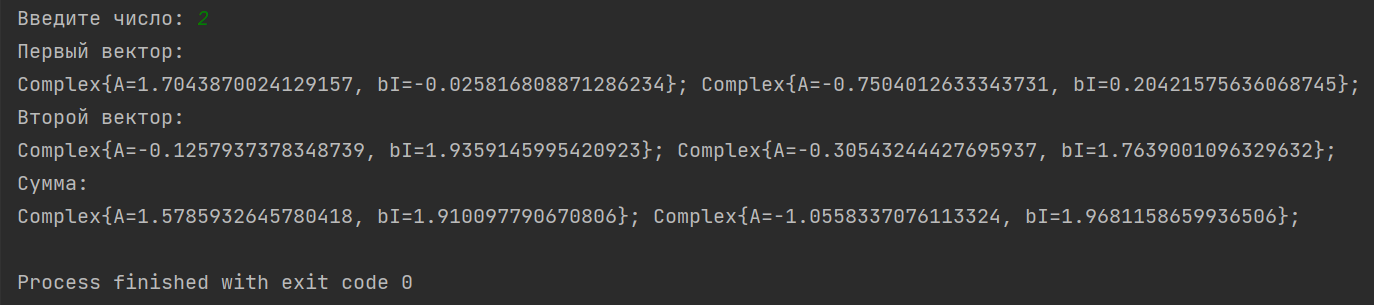


Рисунок 1 – Результат работы программы

Ниже представлен код класса второй программы, а на рисунке 2 – результат работы программы.

class QuadEquat {  
 DecimalFormat df = new DecimalFormat("###.##");  
 double a,b,c,x1,x2,d;  
 public QuadEquat(){};  
 public QuadEquat(double a,double b, double c){  
 this.a=a;  
 this.b=b;  
 this.c=c;  
 };  
 public QuadEquat( int a, int b, int c){  
 this.a=a;  
 this.b=b;  
 this.c=c;  
 };  
  
 public void Root(){  
 this.d=this.b\*this.b-4\*this.a\*this.c;  
 if (d>=0){  
 this.x1=(-b+Math.sqrt(d))/(2\*a);  
 this.x2=(-b+Math.sqrt(d))/(2\*a);  
 }  
  
 }  
  
 public void getQuadEquat() {  
 System.out.println("Уравнение: "+ df.format(this.a)+ "X^2 + " +df.format(this.b)+"X + "+df.format(this.c));  
 }  
 public void getRoot(){  
 if (d>=0){  
 System.out.println("Корни уравнения: "+ df.format(this.x1)+ " и " +df.format(this.x2)+";");  
 }  
 else{  
 System.out.println("Корней уравнения не существует! ");  
 }  
 }  
 public void MinMax() {  
double x1=-b/(2\*a);  
double y1=a\*x1\*x1+b\*x1+c;  
if (this.a>0) {  
 System.out.println("Уравнение имеет минимум в точке (" + df.format(x1) + ";" + df.format(y1) + ");");  
}  
else {System.out.println("Уравнение имеет максимум в точке (" + df.format(x1) + ";" + df.format(y1) + ");");}  
 }  
  
 public void MinMaxSpan() {  
 double x1=-b/(2\*a);  
 if (this.a>0) {  
 System.out.println("Уравнение возрастает на промежутке -∞ до " + df.format(x1) + " и далее убывает до +∞ ;");  
 }  
 else {System.out.println("Уравнение убывает на промежутке -∞ до " + df.format(x1) + " и далее возрастает до +∞ ;");;}  
 }  
}

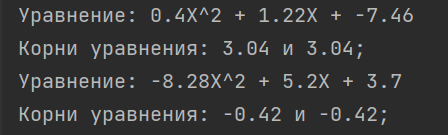


Рисунок 2 – Результат работы программы

**Вариант 2:**

Создать классы, спецификации которых приведены ниже. Определить конструкторы и методы setТип(), getТип(), toString(). Определить дополнительно методы в классе, создающем массив объектов. Задать критерий выбора данных и вывести эти данные на консоль.

1. Car: id, Марка, Модель, Год выпуска, Цвет, Цена, Регистрационный номер. Создать массив объектов. Вывести: a) список автомобилей заданной марки; b) список автомобилей заданной модели, которые эксплуатируются больше n лет; c) список автомобилей заданного года выпуска, цена которых больше указанной.
2. Product: id, Наименование, UPC, Производитель, Цена, Срок хранения, Количество. Создать массив объектов. Вывести: a) список товаров для заданного наименования; b) список товаров для заданного наименования, цена которых не превосходит заданную; c) список товаров, срок хранения которых больше заданного.

**Решение:**

Ниже представлен кусок кода первой программы, а на рисунке 3 – результат работы программы.

System.out.print("Б) Введите модель и срок эксплуатации: ");  
s = input.nextLine();  
int n = input.nextInt();  
Date dt=new Date();  
int year=dt.getYear()+1900;  
System.out.println("Подходящие товары: ");  
for (int i=0;i<4;i++){  
 String s1=cars[i].getModel().toUpperCase(Locale.ROOT);  
 int ey=cars[i].getYear();  
 if (s1.equals(s.toUpperCase(Locale.ROOT)) & ((year - ey) > n)){  
 System.out.println(cars[i].toString());  
 }  
}  
  
System.out.println();

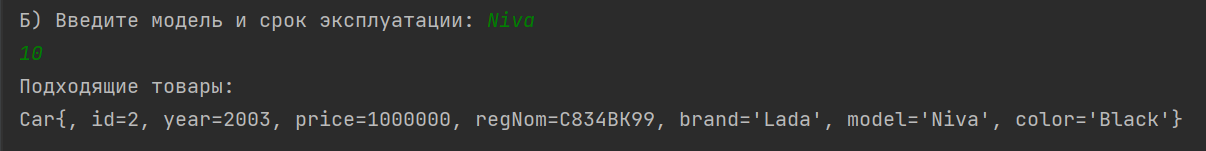


Рисунок 3 – Результат работы программы

Ниже представлен часть кода второй программы, а на рисунке 4 – результат работы программы.

System.out.print("А) Введите наименование: ");  
String s = input.nextLine().toString();  
System.out.println("Подходящие товары: ");  
for (int i=0;i<4;i++){  
 String s1=products[i].getName().toUpperCase(Locale.ROOT);  
 if (s1.equals(s.toUpperCase(Locale.ROOT))){  
 System.out.println(products[i].toString());  
 }  
}  
System.out.println();

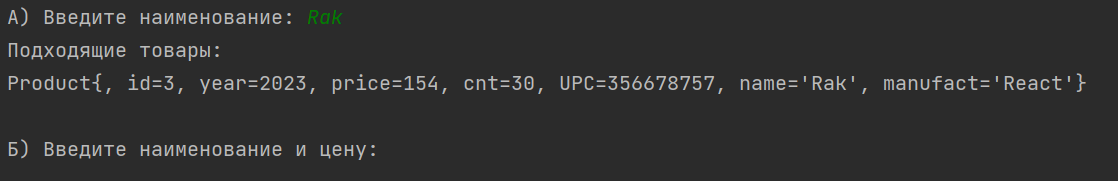


Рисунок 4 – Результат работы программы

**Вариант 3:**

Создать приложение, удовлетворяющее требованиям, приведенным в задании. Аргументировать принадлежность классу каждого создаваемого метода и корректно переопределить для каждого класса методы equals(), hashCode(), toString().

1. Создать объект класса Пианино, используя класс Клавиша. Методы: настроить, играть на пианино, нажимать клавишу.

Создать объект класса Фотоальбом, используя класс Фотография. Методы: задать название фотографии, дополнить фотоальбом фотографией, вывести на консоль количество фотографий.

**Решение:**

Ниже представлен код класса первой программы, а на рисунке 5 – результат работы программы.

class Piano extends Key{  
 Key[] keys;  
 int noms;  
 public Piano(){}  
 public void Setup (int noms){  
 if (noms>0) {  
 this.keys=new Key[noms];  
 this.noms=noms;  
 for (int i = 0; i < noms; i++) {  
  
 this.keys[i]=new Key(i+1);}  
  
 System.out.println("Пианино настроен как " + noms + " клавишный!");  
 }  
 else {System.out.println("Пианино невохможно настроить!");}  
 }  
  
 public void playPiano (int[] key){  
 for (int i=0;i<key.length;i++){  
 if ((key[i]<=this.noms+1)&(key[i]>0)){  
 System.out.print("Играет "+ key[i]+ " клавиша; ");  
 } else System.out.print("Промах; ");  
 System.out.println();  
 }  
 }  
  
 public void playKey (int key){  
  
 if ((key<=this.noms+1)&(key>0)) {  
 System.out.println("Зажата " + key + " клавиша; ");  
 } else{ System.out.print("Нет такой клавиши; ");}  
 System.out.println();  
 }  
  
 @Override  
 public boolean equals(Object o) {  
 if (this == o) return true;  
 if (o == null || getClass() != o.getClass()) return false;  
 if (!super.equals(o)) return false;  
 Piano piano = (Piano) o;  
 return noms == piano.noms && Arrays.equals(keys, piano.keys);  
 }  
  
 @Override  
 public int hashCode() {  
 int result = Objects.hash(super.hashCode(), noms);  
 result = 31 \* result + Arrays.hashCode(keys);  
 return result;  
 }  
  
 @Override  
 public String toString() {  
 return "Piano{" +  
 "keys=" + Arrays.toString(keys) +  
 ", noms=" + noms +  
 '}';  
 }  
}

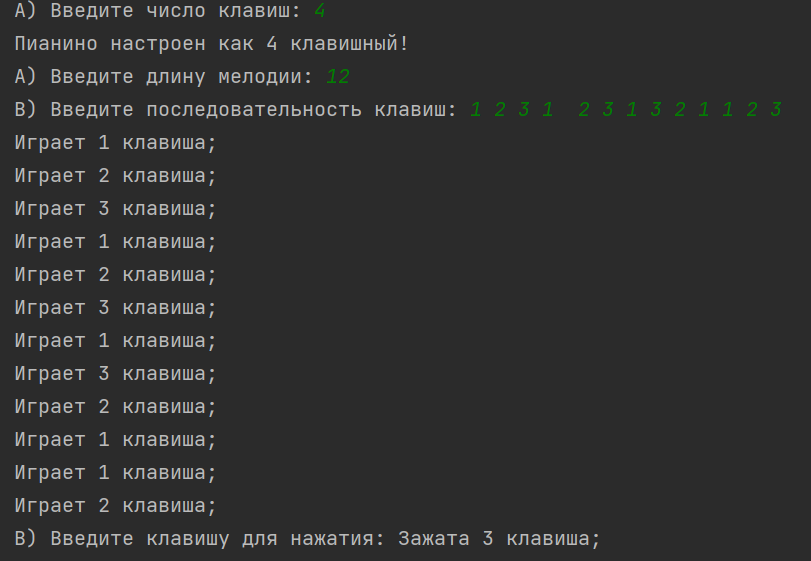


Рисунок 5 – Результат работы программы

Ниже представлен код класса второй программы, а на рисунке 6 – результат работы программы.

class Photo {  
 String photo;  
 public Photo(){}  
 public Photo(String n){  
 this.photo=n;  
 }  
  
 @Override  
 public boolean equals(Object o) {  
 if (this == o) return true;  
 if (!(o instanceof Photo photo1)) return false;  
 return photo == photo1.photo;  
 }  
  
 @Override  
 public int hashCode() {  
 return Objects.hash(photo);  
 }  
  
 @Override  
 public String toString() {  
 return "Photo{" +  
 "photo=" + photo +  
 '}';  
 }  
}  
class PhotoAlbom extends Photo{  
 ArrayList<Photo> photos=new ArrayList<Photo>();  
 public PhotoAlbom(){}  
 public void setName (String photo,int n) {  
 if (this.photos.size() < n) {  
 for (int i = photos.size() ; i < n -1; i++) {  
 this.photos.add(new Photo("Zero photo"));  
 }  
 this.photos.add(n - 1, new Photo(photo));  
 }  
 }  
  
 public void addPhoto (String photo){  
 this.photos.add(new Photo(photo));  
 }  
  
 public int getCount (){  
 return this.photos.size();  
 }  
  
 @Override  
 public boolean equals(Object o) {  
 if (this == o) return true;  
 if (!(o instanceof PhotoAlbom that)) return false;  
 if (!super.equals(o)) return false;  
 return Objects.equals(photos, that.photos);  
 }  
  
 @Override  
 public int hashCode() {  
 return Objects.hash(super.hashCode(), photos);  
 }  
  
 @Override  
 public String toString() {  
 return "PhotoAlbom{" +  
 "photos=" + photos +  
 '}';  
 }  
}

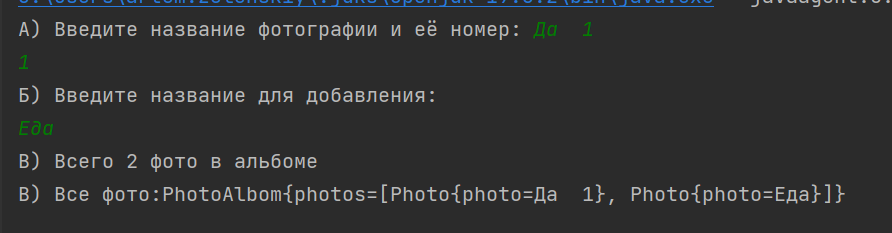


Рисунок 6 – Результат работы программы

**Вариант 4:**

Построить модель программной системы.

1. Система Автобаза. Диспетчер распределяет заявки на Рейсы между Водителями и назначает для этого Автомобиль. Водитель может сделать заявку на ремонт. Диспетчер может отстранить Водителя от работы. Водитель делает отметку о выполнении Рейса и состоянии Автомобиля.
2. Система Интернет-магазин. Администратор добавляет информацию о Товаре. Клиент делает и оплачивает Заказ на Товары. Администратор регистрирует Продажу и может занести неплательщиков в «черный список».

**Решение:**

Ниже представлен кусок кода первой программы.

public void zayavka(Car car, Flight flight, Driver driver){  
  
 Driver driver1 = driver;  
 this.drivers.add(driver1);  
 Car car1 = car;  
 this.cars.add(car1);  
 Flight flight1 = flight;  
 this.flights.add(flight1);  
 System.out.println("Водитель " + driver + " заявлен на рейс "+ flight + " на автомобиле "+ car);  
 System.out.println(drivers.toString() + cars.toString() + flights.toString());  
}

Ниже представлен часть кода второй программы.

public void renameProduct(Product p, String s){  
 Product pr=p;  
 this.products.remove(pr);  
 pr.setName(s);  
 this.products.add(pr);  
}  
  
public void setBL(Client c){  
 this.clients.remove(c);  
 c.setSost("В бане");  
 System.out.println("Клиент " + c.getName() + " попадает в черный список");  
}  
  
public void setSELL(Client c, Order o){  
 this.clients.get( this.clients.indexOf(c)).setOrder(o,"Оплачен");  
 System.out.println("Заказ " + o.getOrderID() + " оплачен клиентом " +c.getName());  
}

**Вывод:** были получены базовые представления о работе языка java с классами, наследованием и полиморфизмом. Были написаны программы согласно вариантам.