|  |  |
| --- | --- |
| **Gerb-BMSTU_01** | **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 (ИУ6) \_\_\_\_\_\_

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 гр. ИУ6-23М **\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ \_\_** Г. Е. Горский **\_\_\_**

(Подпись, дата) (И.О. Фамилия)

Преподаватель  **\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ \_\_**П.В. Степанов **\_\_**

(Подпись, дата) (И.О. Фамилия)

Москва, 2023

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

**Вариант 1:**

1. Определить класс Дробь в виде пары (m,n). Класс должен содержать несколько конструкторов. Реализовать методы для сложения, вычитания, умножения и деления дробей. Объявить массив из k дробей, ввести/вывести значения для массива дробей. Создать массив объектов и передать его в метод, который изменяет каждый элемент массива с четным индексом путем добавления следующего за ним элемента массива.
2. 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.

**Решение:**

import java.util.Random;  
  
public class Drob {  
 private float m;  
 private float n;  
  
 Drob(){  
 this.m = 2; //числитель  
 this.n = 1; // знаменатель  
 }  
 Drob(int m\_, int n\_){  
 this.m=m\_;  
 this.n=n\_;  
 }  
 Drob(int m\_){  
 this.m=m\_;  
 this.n = 1;  
 }  
  
 public float getM() {  
 return m;  
 }  
  
 public float getN() {  
 return n;  
 }  
  
 public void setM(float m) {  
 this.m = m;  
 }  
  
 public void setN(float n) {  
 this.n = n;  
 }  
  
 @Override  
 public String toString() {  
 return "m=" + m +  
 ", n=" + n;  
 }  
  
/\* Определить класс Дробь в виде пары (m,n). Класс должен содержать несколько конструкторов.  
 Реализовать методы для сложения, вычитания, умножения и деления дробей. Объявить массив из k дробей,  
 ввести/вывести значения для массива дробей. Создать массив объектов и передать его в метод,  
 который изменяет каждый элемент массива с четным индексом путем добавления следующего за ним элемента массива. \*/  
 public static void main(String[] args) {  
 Random rnd = new Random();  
 Drob dr1 = new Drob(rnd.nextInt(1,5),rnd.nextInt(1,5) );  
 Drob dr2 = new Drob(rnd.nextInt(1,5));  
 System.out.println(dr1.toString() + "\n" + dr2.toString());  
 System.out.println((Plus(dr1, dr2)));  
 System.out.println((Minus(dr1, dr2)));  
 System.out.println((Multiply(dr1, dr2)));  
 System.out.println((Delete(dr1, dr2)));  
  
  
  
 int k=rnd.nextInt(2,7);  
 Drob[] dr3 = new Drob[k];  
 try {  
 for (int i = 0; i<k; i++){  
 dr3[i] = new Drob(rnd.nextInt(1,5),rnd.nextInt(1,5) );  
 }  
 } catch (OutOfMemoryError e){  
 throw new OutOfMemoryError("виртуальная машина Java не может выделить объект из-за нехватки памяти");  
 }  
  
 try {  
 for (int i = 0; i<k; i++){  
 // dr3[i] = new Drob(rnd.nextFloat(1,5),rnd.nextFloat(1,5) );  
 System.out.println(dr3[i].toString());  
 }  
 } catch (IndexOutOfBoundsException e){  
 throw new IndexOutOfBoundsException("Выход за пределы масисива " + k);  
 }  
  
 Array\_Drob(k, dr3);  
 System.out.println("new array");  
 for (int i = 0; i<k; i++) {  
 System.out.println(dr3[i].toString());  
 }  
 }  
 public static Drob[] Array\_Drob(int k, Drob[] a){  
  
 for (int i=0; i<k-1; i++){  
 if(i%2==0){  
 float tmp1=a[i].getM() + a[i+1].getM();  
 float tmp2=a[i].getN() + a[i+1].getN();  
 a[i].setM(tmp1);  
 a[i].setN(tmp2);  
 }  
 }  
 return a;  
 }  
 public static float Change(Drob a) throws ArithmeticException{  
 float first\_ch = a.getM();  
 float first\_zn = a.getN();  
 float tmp1 =0;  
 // first\_zn = 0;  
  
 tmp1 = first\_ch / first\_zn;  
 if (tmp1 == Float.POSITIVE\_INFINITY || tmp1 == Float.NEGATIVE\_INFINITY) {  
 throw new ArithmeticException("Деление на ноль");  
 }  
 return tmp1;  
 }  
 public static float Plus(Drob a, Drob b){  
 float tmp1 =Change(a);  
 float tmp2 = Change(b);  
 return tmp1 + tmp2;  
 }  
 public static float Minus(Drob a, Drob b){  
 float tmp1 =Change(a);  
 float tmp2 = Change(b);  
 return tmp1 - tmp2;  
 }  
 public static float Multiply(Drob a, Drob b){  
 float tmp1 =Change(a);  
 float tmp2 = Change(b);  
 return tmp1 \* tmp2;  
 }  
 public static float Delete(Drob a, Drob b) throws ArithmeticException {  
 float tmp1 =Change(a);  
 float tmp2 = Change(b);  
 if (tmp2 != 0) {  
 return tmp1 / tmp2;  
 }  
 else {  
 throw new ArithmeticException("Деление на ноль");  
 }  
 }  
}

Результат выполнения представлен на рисунке 1.

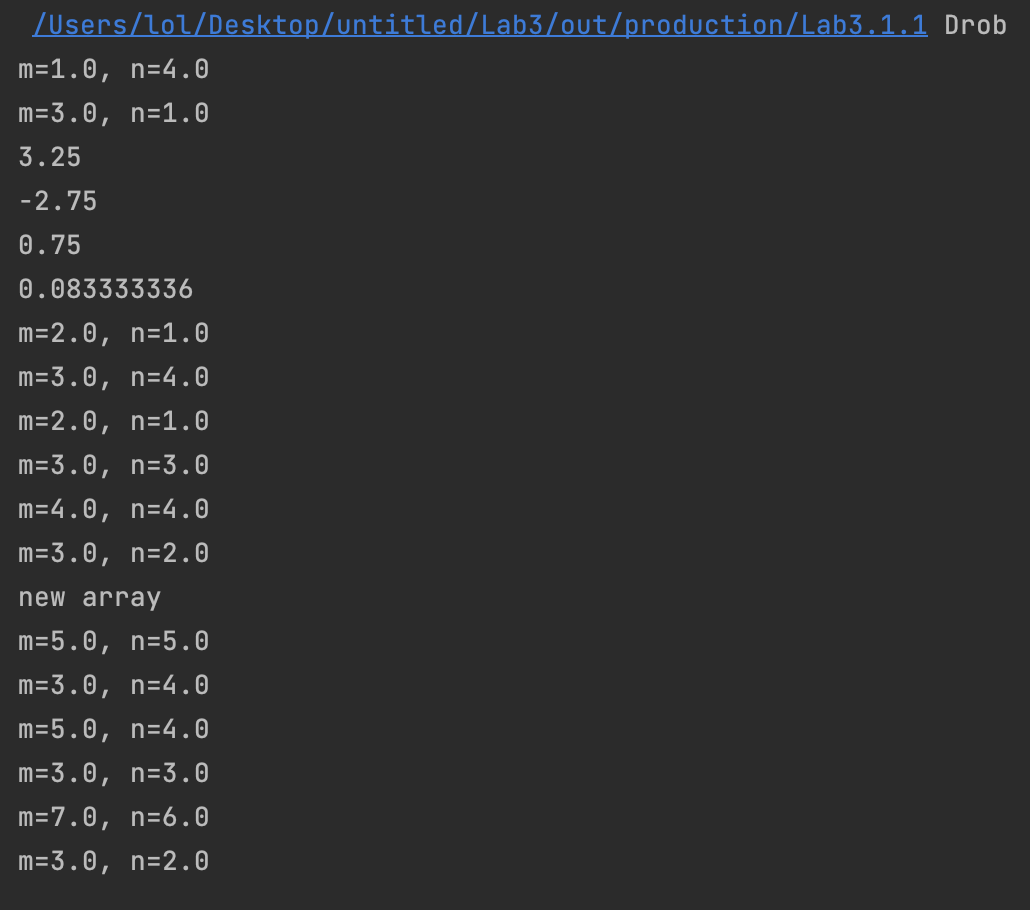
****

Рисунок 1 — пример выполнения

import java.util.Random;  
import java.util.Vector;  
  
public class Complex {  
 private float re;  
 private float im;  
  
 public Complex(float real, float imag) {  
 this.re = real;  
 this.im = imag;  
 }  
 public Complex(float real) {  
 this.re = real;  
 this.im = 0;  
 }  
 public Complex() {  
 this.re = 2;  
 this.im = 1;  
 }  
 @Override  
 public String toString() {  
 if (im == 0) return re + "";  
 if (re == 0) return im + "i";  
 if (im < 0) return re + " - " + (-im) + "i";  
 return re + " + " + im + "i";  
 }  
 public Complex Plus(Complex b) {  
 Complex a = this;  
 float real = a.re + b.re;  
 float imag = a.im + b.im;  
 return new Complex(real, imag);  
 }  
 public Complex Minus(Complex b) {  
 Complex a = this;  
 float real = a.re - b.re;  
 float imag = a.im - b.im;  
 return new Complex(real, imag);  
 }  
 public Complex Multiply(Complex b) {  
 Complex a = this;  
 float real = a.re \* b.re - a.im \* b.im;  
 float imag = a.re \* b.im + a.im \* b.re;  
 return new Complex(real, imag);  
 }  
  
 public Complex reciprocal() throws ArithmeticException{  
 float scale = re \* re + im \* im;  
 if (scale == Float.POSITIVE\_INFINITY || scale == Float.NEGATIVE\_INFINITY) {  
 throw new ArithmeticException("Деление на ноль");  
 }  
 return new Complex(re / scale, -im / scale);  
 }  
 public Complex Delete(Complex b) {  
 Complex a = this;  
 return a.Multiply(b.reciprocal());  
 }  
 public static Vector<Complex> Vect(Vector<Complex> v1, Vector<Complex> v2){  
  
 for (int i=0; i< v1.capacity(); i++){  
 v1.set(i,v1.get(i).Plus(v2.get(i)));  
 }  
  
 return v1;  
 }  
/\* 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. \*/  
 public static void main(String[] args) {  
 Random rnd = new Random();  
 Complex a = new Complex(rnd.nextInt(-5, 5), rnd.nextInt(-5, 5));  
 Complex b = new Complex(rnd.nextInt(-5, 5), rnd.nextInt(-5, 5));  
  
 System.out.println("a = " + a);  
 System.out.println("b = " + b);  
 System.out.println("b + a = " + b.Plus(a));  
 System.out.println("a - b = " + a.Minus(b));  
 System.out.println("a \* b = " + a.Multiply(b));  
 System.out.println("a / b = " + a.Delete(b));  
  
 int n=rnd.nextInt(2,7);  
 Vector<Complex> v1 = new Vector<Complex>(n);  
 Vector<Complex> v2 = new Vector<Complex>(n);  
 try {  
 for (int i = 0; i<n; i++) {  
 v1.add(new Complex(rnd.nextInt(-5, 5), rnd.nextInt(-5, 5)));  
 v2.add(new Complex(rnd.nextInt(-5, 5), rnd.nextInt(-5, 5)));  
 }  
 }catch (OutOfMemoryError e){  
 throw new OutOfMemoryError("виртуальная машина Java не может выделить объект из-за нехватки памяти");  
 }catch (Exception e){  
 System.out.println("Что-то пошло не так");  
 }  
 System.out.println(v1.toString());  
 System.out.println(v2.toString());  
 Vect(v1,v2);  
 System.out.println("new Vector");  
 System.out.println(v1.toString());  
 }

}

Результат выполнения представлен на рисунке 2.

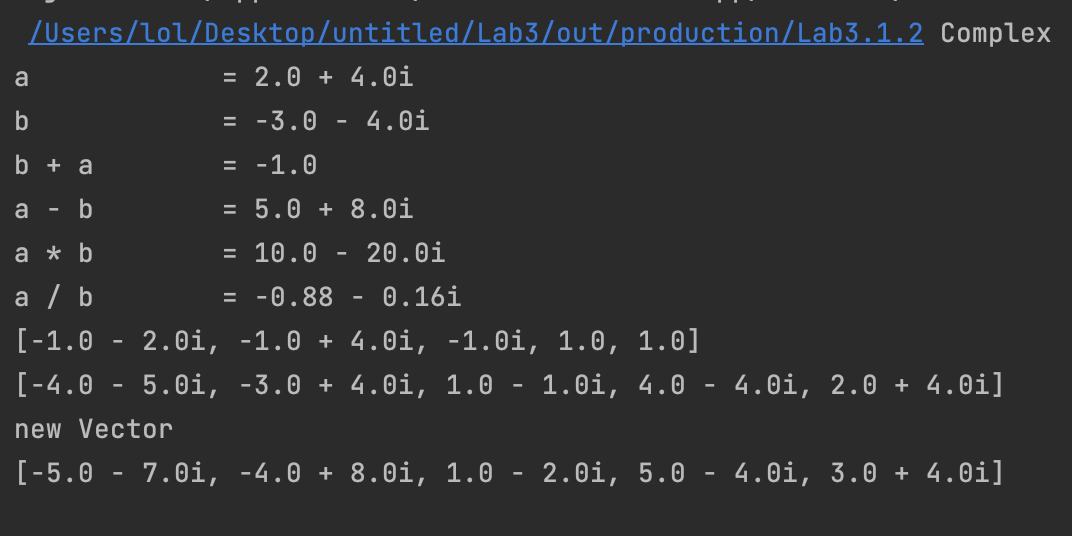


Рисунок 2 — пример выполнения

**Вариант 2:**

1. Phone: id, Фамилия, Имя, Отчество, Адрес, Номер кредитной карточки, Дебет, Кредит, Время городских и междугородных разговоров. Создать массив объектов. Вывести: a) сведения об абонентах, у которых время внутригородских разговоров превышает заданное; b) сведения об абонентах, которые пользовались междугородной связью; c) сведения об абонентах в алфавитном порядке.
2. 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.

**Решение:**

import java.util.Arrays;  
import java.util.Random;  
import java.util.Scanner;  
public class Phone {  
 static final int def = 10;  
 int ID;  
 String surname;  
 String name;  
 String otch;  
 String address;  
 int number\_credit;  
 int debet;  
 int credit;  
 int localtime;  
 int globaltime;  
  
 @Override  
 public String toString() {  
 return "ID=" + ID +  
 ", surname='" + surname + '\'' +  
 ", name='" + name + '\'' +  
 ", otch='" + otch + '\'' +  
 ", address='" + address + '\'' +  
 ", number\_credit=" + number\_credit +  
 ", debet=" + debet +  
 ", credit=" + credit +  
 ", localtime=" + localtime +  
 ", globaltime=" + globaltime;  
 }  
  
  
 public int getID() {  
 return ID;  
 }  
  
 public void setID(int ID) {  
 this.ID = ID;  
 }  
  
 public String getSurname() {  
 return surname;  
 }  
  
 public void setSurname(String surname) {  
 this.surname = surname;  
 }  
  
 public String getName() {  
 return name;  
 }  
  
 public void setName(String name) {  
 this.name = name;  
 }  
  
 public String getOtch() {  
 return otch;  
 }  
  
 public void setOtch(String otch) {  
 this.otch = otch;  
 }  
  
 public String getAddress() {  
 return address;  
 }  
  
 public void setAddress(String address) {  
 this.address = address;  
 }  
  
 public int getNumber\_credit() {  
 return number\_credit;  
 }  
  
 public void setNumber\_credit(int number\_credit) {  
 this.number\_credit = number\_credit;  
 }  
  
 public int getDebet() {  
 return debet;  
 }  
  
 public void setDebet(int debet) {  
 this.debet = debet;  
 }  
  
 public int getCredit() {  
 return credit;  
 }  
  
 public void setCredit(int credit) {  
 this.credit = credit;  
 }  
  
 public int getLocaltime() {  
 return localtime;  
 }  
  
 public void setLocaltime(int localtime) {  
 this.localtime = localtime;  
 }  
  
 public int getGlobaltime() {  
 return globaltime;  
 }  
  
 public void setGlobaltime(int globaltime) {  
 this.globaltime = globaltime;  
 }  
 public void Show(){  
 Phone a = this;  
 if(a.localtime > def){  
 System.out.println("a) сведения об абонентах, у которых время внутригородских разговоров превышает заданное: \n" +  
 "ID=" + ID +  
 ", surname='" + surname + '\'' +  
 ", name='" + name + '\'' +  
 ", otch='" + otch + '\'' +  
 ", localtime=" + localtime );  
 }  
 if(a.globaltime>0){  
 System.out.println("b) сведения об абонентах, которые пользовались междугородной связью: \n" +  
 "ID=" + ID +  
 ", surname='" + surname + '\'' +  
 ", name='" + name + '\'' +  
 ", otch='" + otch + '\'' +  
 ", globaltime=" + globaltime );  
 }  
 }  
 public static void main(String[] args) {  
 Random rnd = new Random();  
 Scanner myObj = new Scanner(System.in);  
 int k=rnd.nextInt(2,4);  
 // int k =2;  
 Phone[] ph = new Phone[k];  
 for (int i = 0; i<k; i++){  
 ph[i] = new Phone();  
 ph[i].ID=i;  
 System.out.println(" new user");  
 myObj.nextLine();  
 System.out.println("Input surname: ");  
 ph[i].surname = myObj.nextLine();  
 System.out.println("Input name: ");  
 ph[i].name = myObj.nextLine();  
 System.out.println("Input otch: ");  
 ph[i].otch = myObj.nextLine();  
 System.out.println("Input address: ");  
 ph[i].address = myObj.nextLine();  
 System.out.println("Input number credit card: ");  
 ph[i].number\_credit = myObj.nextInt();  
 System.out.println("Input debet: ");  
 ph[i].debet = myObj.nextInt();  
 System.out.println("Input credit: ");  
 ph[i].credit = myObj.nextInt();  
 System.out.println("Input debet: ");  
 ph[i].debet = myObj.nextInt();  
 System.out.println("Input localtime: ");  
 ph[i].localtime = myObj.nextInt();  
 System.out.println("Input globaltime: ");  
 ph[i].globaltime = myObj.nextInt();  
 }  
 System.out.println("-------------------------------------------");  
 for (int i = 0; i<k; i++){  
 System.out.println(ph[i].toString());  
 }  
 System.out.println("-------------------------------------------");  
 for (int i = 0; i<k; i++){  
 ph[i].Show();  
 }  
 System.out.println("-------------------------------------------");  
 Arrays.sort(ph, (a, b) -> a.name.compareTo(b.name));  
 for (int i = 0; i<k; i++){  
 System.out.println(ph[i].toString());  
 }  
 }  
}

Результат выполнения представлен на рисунке 3.

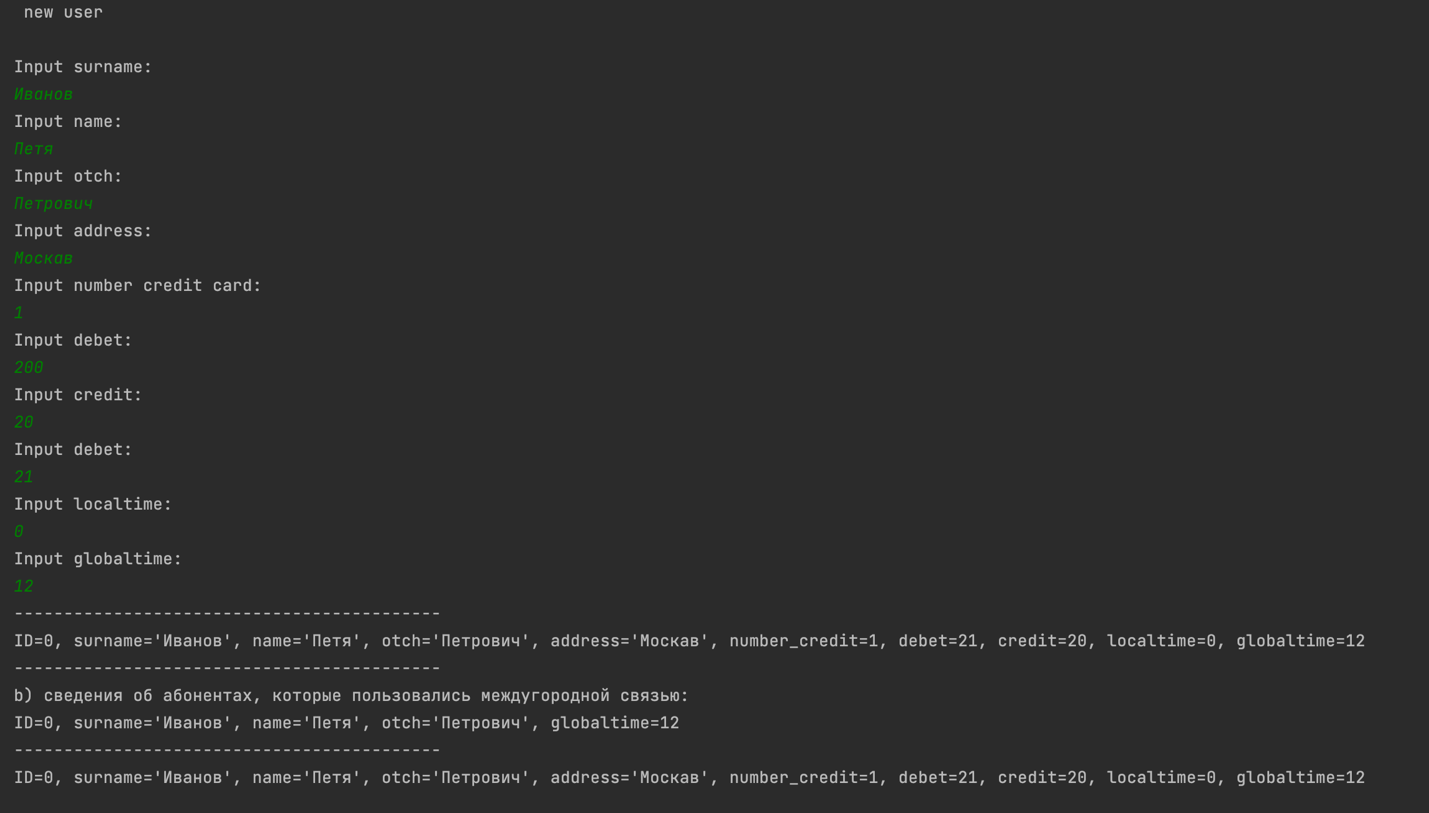


Рисунок 3 — пример выполнения

import java.util.Random;  
import java.util.Scanner;  
  
public class Car {  
  
 int ID;  
 String mark;  
 String model;  
 int year;  
 String color;  
 int price;  
 String reg\_num;  
  
 @Override  
 public String toString() {  
 return "Car{" +  
 "ID=" + ID +  
 ", mark='" + mark + '\'' +  
 ", model='" + model + '\'' +  
 ", year=" + year +  
 ", color='" + color + '\'' +  
 ", price=" + price +  
 ", reg\_num='" + reg\_num + '\'' +  
 '}';  
 }  
  
  
 public int getID() {  
 return ID;  
 }  
  
 public void setID(int ID) {  
 this.ID = ID;  
 }  
  
 public String getMark() {  
 return mark;  
 }  
  
 public void setMark(String mark) {  
 this.mark = mark;  
 }  
  
 public String getModel() {  
 return model;  
 }  
  
 public void setModel(String model) {  
 this.model = model;  
 }  
  
 public int getYear() {  
 return year;  
 }  
  
 public void setYear(int year) {  
 this.year = year;  
 }  
  
 public String getColor() {  
 return color;  
 }  
  
 public void setColor(String color) {  
 this.color = color;  
 }  
  
 public int getPrice() {  
 return price;  
 }  
  
 public void setPrice(int price) {  
 this.price = price;  
 }  
  
 public String getReg\_num() {  
 return reg\_num;  
 }  
  
 public void setReg\_num(String reg\_num) {  
 this.reg\_num = reg\_num;  
 }  
 public void Show(String mark, String model, int n, int m, int price) {  
 Car a = this;  
 if (a.mark.contains(mark)) {  
 System.out.println("a) список автомобилей заданной марки: \n" +  
 "ID=" + ID +  
 ", mark='" + a.mark + '\'' +  
 ", model='" + a.model + '\'' +  
 ", year='" + year + '\'' +  
 ", reg num=" + reg\_num);  
 }  
 if (a.model.contains(model) && (2023 - a.year > n)){  
 System.out.println("b) список автомобилей заданной модели, которые эксплуатируются больше n лет: \n" +  
 "ID=" + ID +  
 ", mark='" + a.mark + '\'' +  
 ", model='" + a.model + '\'' +  
 ", year='" + year + '\'' +  
 ", reg num=" + reg\_num);  
 }  
 if (a.year == m && a.price > price){  
 System.out.println("c) список автомобилей заданного года выпуска, цена которых больше указанной: \n" +  
 "ID=" + ID +  
 ", mark='" + a.mark + '\'' +  
 ", model='" + a.model + '\'' +  
 ", year='" + year + '\'' +  
 ", price='" + a.price + '\'' +  
 ", reg num=" + reg\_num);  
 }  
 }  
 public static void main(String[] args) {  
 Random rnd = new Random();  
 Scanner myObj = new Scanner(System.in);  
 //int k=rnd.nextInt(2,4);  
 int k =2;  
 Car[] ph = new Car[k];  
 for (int i = 0; i<k; i++){  
 ph[i] = new Car();  
 ph[i].ID=i;  
 System.out.println(" new car ");  
 System.out.println("Input mark: ");  
 ph[i].mark = myObj.nextLine();  
 System.out.println("Input model: ");  
 ph[i].model = myObj.nextLine();  
 System.out.println("Input color: ");  
 ph[i].color = myObj.nextLine();  
 System.out.println("Input reg number: ");  
 ph[i].reg\_num = myObj.nextLine();  
 System.out.println("Input year: ");  
 ph[i].year = myObj.nextInt();  
 System.out.println("Input price: ");  
 ph[i].price = myObj.nextInt();  
 myObj.nextLine();  
 }  
  
 String mark = "BMW";  
 String model = "3";  
 int n = 7;  
 int m = 2010;  
 int price = 100;  
  
  
 System.out.println("-------------------------------------------");  
 for (int i = 0; i<k; i++){  
 System.out.println(ph[i].toString());  
 }  
 System.out.println("-------------------------------------------");  
 for (int i = 0; i<k; i++){  
 ph[i].Show(mark, model, n, m, price);  
 }  
 }  
}

Результат выполнения представлен на рисунке 4.

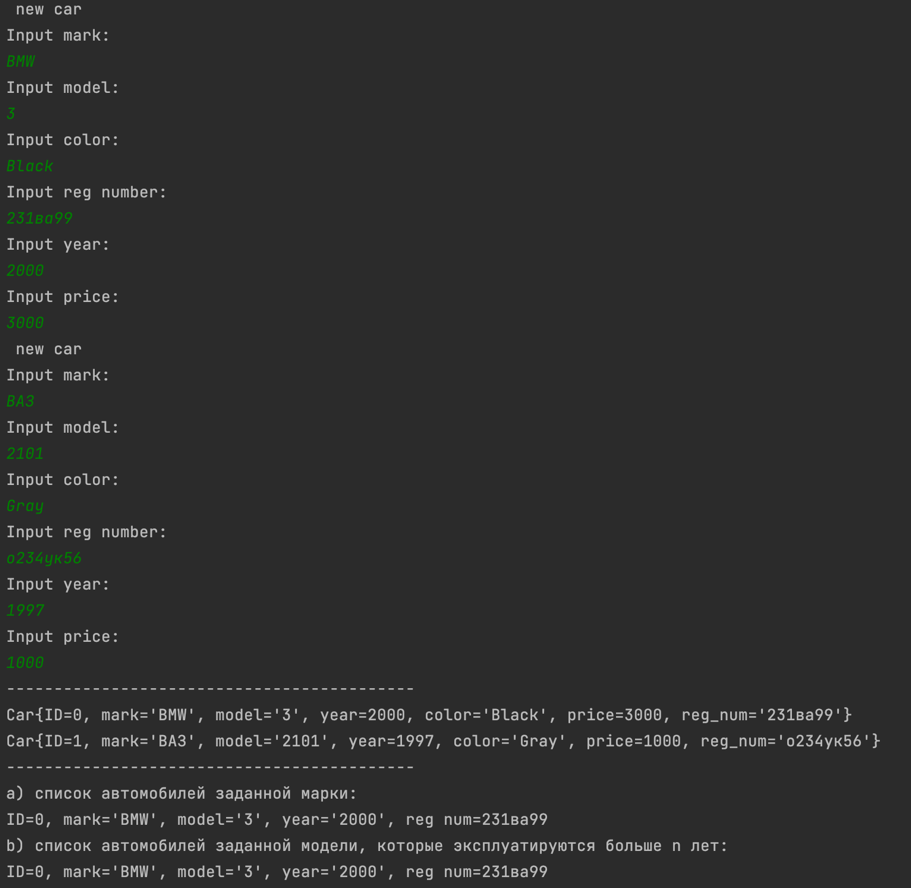


Рисунок 4 — пример выполнения

**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 Дерево, используя классы Лист. Методы: зацвести, опасть листьям, покрыться инеем, пожелтеть листьям.
2. Создать объект класса Пианино, используя класс Клавиша. Методы: настроить, играть на пианино, нажимать клавишу.

**Решение:**

public class Main {  
 public static void main(String[] args) {  
 Tree a = new Tree("Береза", 9);  
 Tree b = new Tree();  
 List a1 = new List();  
 List b1 = new List(5);  
 a.Zacvesti();  
 a.FallList();  
 System.out.println(a.toString());  
 System.out.println(b1.toString());  
 System.out.println(a.equals(b));  
 System.out.println(a1.equals(b1));  
 System.out.println(a.hashCode());  
 System.out.println(a1.hashCode());  
 }  
}  
class List {   
 int list;  
 //пара конструкторов  
 List() {  
 list = 10;  
 }  
 @Override  
 public boolean equals(Object ob)  
 {  
 if (ob == this) {  
 return true;  
 }  
  
 if (ob == null || ob.getClass() != getClass()) {  
 return false;  
 }  
  
 List p = (List) ob;  
  
 return Objects.equals(list, p.list);  
 }  
 @Override  
 public int hashCode()  
 {  
 return Objects.hash(list);  
 }  
 @Override  
 public String toString() {  
 return "List{" +  
 "list=" + list +  
 '}';  
 }  
  
 List(int count) {  
 this.list = count;  
 }  
}  
  
class Tree extends List {  
 String tree; //поле класса  
 Tree() { //конструктор  
 super();  
 tree = "Береза";  
 }  
  
 Tree(String tree, int count) {  
 super(count);  
 this.tree = tree;  
 }  
 void Zacvesti() {  
 System.out.println("Зацвести");  
 }  
 void FallList() { //сменить колесо  
 super.list = 0;  
 }  
 void Inei() {  
 System.out.println("Покрыться инеем");  
 }  
 void YellowList() {  
 System.out.println("Пожелтеть");  
 }  
 @Override  
 public boolean equals(Object ob)  
 {  
 if (ob == this) {  
 return true;  
 }  
  
 if (ob == null || ob.getClass() != getClass()) {  
 return false;  
 }  
  
 Tree p = (Tree) ob;  
  
 return Objects.equals(tree, p.tree);  
 }  
 @Override  
 public int hashCode()  
 {  
 return Objects.hash(tree);  
 }  
  
 @Override  
 public String toString() {  
 return "Tree{" +  
 "tree='" + tree + '\'' +  
 ", list=" + list +  
 '}';  
 }  
}

Результат выполнения представлен на рисунке 5.

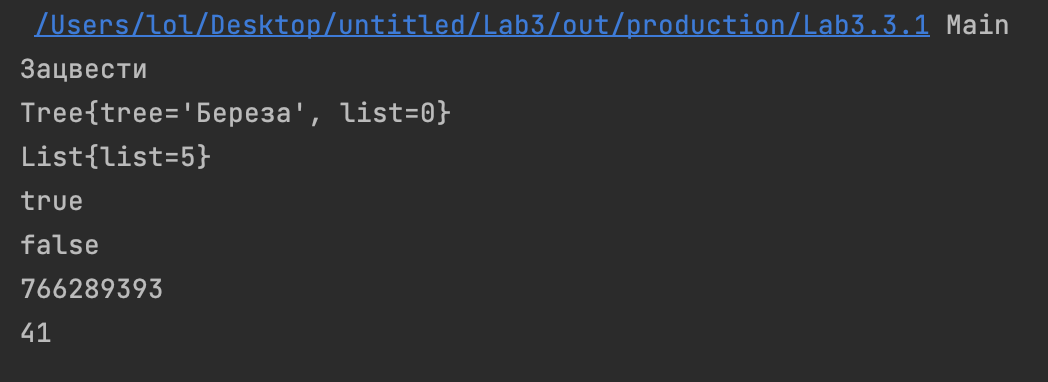


Рисунок 5 — пример выполнения

import java.util.Objects;  
  
/\*  
Создать объект класса Пианино, используя класс Клавиша.  
Методы: настроить, играть на пианино, нажимать клавишу.  
 \*/  
public class Main {  
 public static void main(String[] args) {  
 Piano a = new Piano("Yamaha", 9);  
 Piano b = new Piano();  
 Button a1 = new Button();  
 Button b1 = new Button(5);  
 a.Set();  
 a.Play();  
 b.Presbut(4);  
 System.out.println(b.toString());  
 System.out.println(b1.toString());  
 System.out.println(a.equals(b));  
 System.out.println(a1.equals(b1));  
 System.out.println(a.hashCode());  
 System.out.println(a1.hashCode());  
 }  
}  
class Button { //колесо  
 int but;  
 //пара конструкторов  
 Button() {  
 but = 1;  
 }  
 @Override  
 public boolean equals(Object ob)  
 {  
 if (ob == this) {  
 return true;  
 }  
  
 if (ob == null || ob.getClass() != getClass()) {  
 return false;  
 }  
  
 Button p = (Button) ob;  
  
 return Objects.equals(but, p.but);  
 }  
 @Override  
 public int hashCode()  
 {  
 return Objects.hash(but);  
 }  
 @Override  
 public String toString() {  
 return "Button{" +  
 "Button=" + but +  
 '}';  
 }  
  
 Button(int num) {  
 this.but = num;  
 }  
}  
  
class Piano extends Button {  
 String Piano;  
 Piano() { //конструктор  
 super();  
 Piano = "Yamaha";  
 }  
  
 Piano(String Piano, int num) {  
 super(num);  
 this.Piano = Piano;  
 }  
 void Set() {  
 System.out.println("Настроить");  
 }  
 void Play() {  
 System.out.println("Играть на пианино");  
 }  
 void Presbut(int n) {  
 System.out.println("Нажать клавишу:" + n);  
 this.but = n;  
 }  
 @Override  
 public boolean equals(Object ob)  
 {  
 if (ob == this) {  
 return true;  
 }  
  
 if (ob == null || ob.getClass() != getClass()) {  
 return false;  
 }  
  
 Piano p = (Piano) ob;  
  
 return Objects.equals(Piano, p.Piano);  
 }  
 @Override  
 public int hashCode()  
 {  
 return Objects.hash(Piano);  
 }  
  
 @Override  
 public String toString() {  
 return "Piano{" +  
 "Piano='" + Piano + '\'' +  
 ", Button=" + but +  
 '}';  
 }  
}

Результат выполнения представлен на рисунке 6.

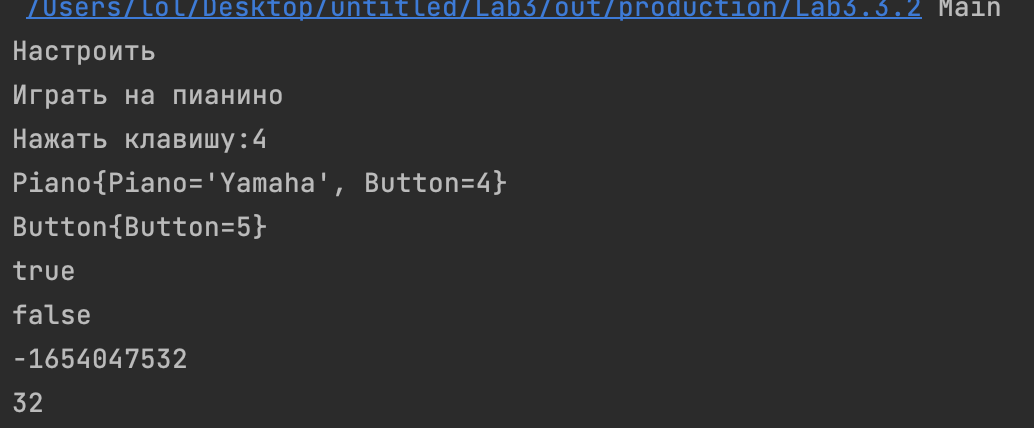


Рисунок 6 — пример выполнения

**Вариант 4:**

1. Система Телефонная станция. Абонент оплачивает Счет за разговоры и Услуги, может попросить Администратора сменить номер и отказаться от услуг. Администратор изменяет номер, Услуги и временно отключает Абонента за неуплату.
2. 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.

**Решение:**

public class Telefon\_Station {  
  
  
 public static void main(String[] args) {  
 Random rnd = new Random();  
 Scanner myObj = new Scanner(System.in);  
 String str1 = "";  
 String name\_uslugi = "";  
 boolean podcluchen = true;  
 int ID = 0;  
 int summa = 0;  
 int summa1 = 0;  
 Uslugi uslugi = new Uslugi();  
 Chet chet1= new Chet();  
 ArrayList<Chet> chets = new ArrayList<>(1);  
 Abonent abonent1 = new Abonent();  
 ArrayList<Abonent> abonents = new ArrayList<>(1);  
 //abonents.add(new Abonent());  
 Admin admin = new Admin();  
 System.out.println("Добро пожаловать! Выберите действие:\n" +  
 "1) Добавить услугу\n" +  
 "2) Добавить счет\n" +  
 "3) Добавить абонента\n" +  
 "4) Пополнить счет\n" +  
 "5) Добавить услугу на счет\n" +  
 "6) Сменить номер\n" +  
 "7) Установить неуплату\n" +  
 "8) Отказкаться от услуги\n" +  
 "9) Вывод данных об абоненте" +  
 "0) Выход");  
 while (true){  
 System.out.println("Выберете действие: ");  
 int tmp = myObj.nextInt();  
 switch (tmp) {  
 case 1:  
 System.out.print("Введите название: ");  
 name\_uslugi = myObj.nextLine()+ myObj.nextLine();  
 uslugi = new Uslugi(name\_uslugi, podcluchen);  
 System.out.println("услуга создана");  
 break;  
 case 2:  
 //создать счет  
 int number\_phone = rnd.nextInt();  
 chet1= new Chet(ID, number\_phone, summa);  
 chets.add(chet1);  
 ID++;  
 System.out.println("счет создан");  
 break;  
 case 3:  
 System.out.print("ФИО: ");  
 String str = myObj.nextLine()+ myObj.nextLine();  
 System.out.println("Выбере счет: " + chets.toString());  
 int str\_1 = myObj.nextInt();  
 abonent1 = new Abonent(str, chets.get(str\_1));  
 abonents.add(abonent1);  
 System.out.println("абонент создан");  
 break;  
  
 case 4:  
 System.out.print("Введите сумму: ");  
 summa1 = myObj.nextInt();  
 System.out.println("Выбере абонента: " + abonents.toString());  
 int str\_2 = myObj.nextInt();  
 abonents.get(str\_2).pay\_chet(summa1);  
 System.out.println("счет пополнен");  
 break;  
 case 5:  
 System.out.print("Введите название: ");  
 name\_uslugi = myObj.nextLine()+ myObj.nextLine();  
 //chet1.uslugi\_chet(name\_uslugi, true);  
 System.out.println("Выбере абонента: " + abonents.toString());  
 int str\_3 = myObj.nextInt();  
 admin.add(abonents.get(str\_3), name\_uslugi);  
 System.out.println("услуга добавлена" + abonents.get(str\_3) + " +" + name\_uslugi);  
 break;  
 case 6:  
 System.out.println("Выбере абонента: " + abonents.toString());  
 int str\_4 = myObj.nextInt();  
 admin.change\_number(abonents.get(str\_4));  
 System.out.println("номер поменян");  
 break;  
 case 7:  
 System.out.println("Выбере абонента: " + abonents.toString());  
 int str\_5 = myObj.nextInt();  
 abonents.get(str\_5).pay\_chet(-summa1-2);  
 System.out.println("неуплата установлена");  
 admin.neuplata( abonents.get(str\_5));  
 break;  
 case 8:  
 System.out.print("Введите название: ");  
 name\_uslugi = myObj.nextLine()+ myObj.nextLine();  
 System.out.println("Выбере абонента: " + abonents.toString());  
 int str\_6 = myObj.nextInt();  
 admin.otkaz(abonents.get(str\_6), name\_uslugi);  
 System.out.println("отказ прошел");  
 break;  
 case 9:  
 System.out.println("Вывод данных абонента\n"+  
 abonents.toString());  
  
 case 0:  
 break;  
  
 }  
 if (tmp == 0){  
 break;  
 }  
 }  
 }  
}

Результат выполнения представлен на рисунке 7.

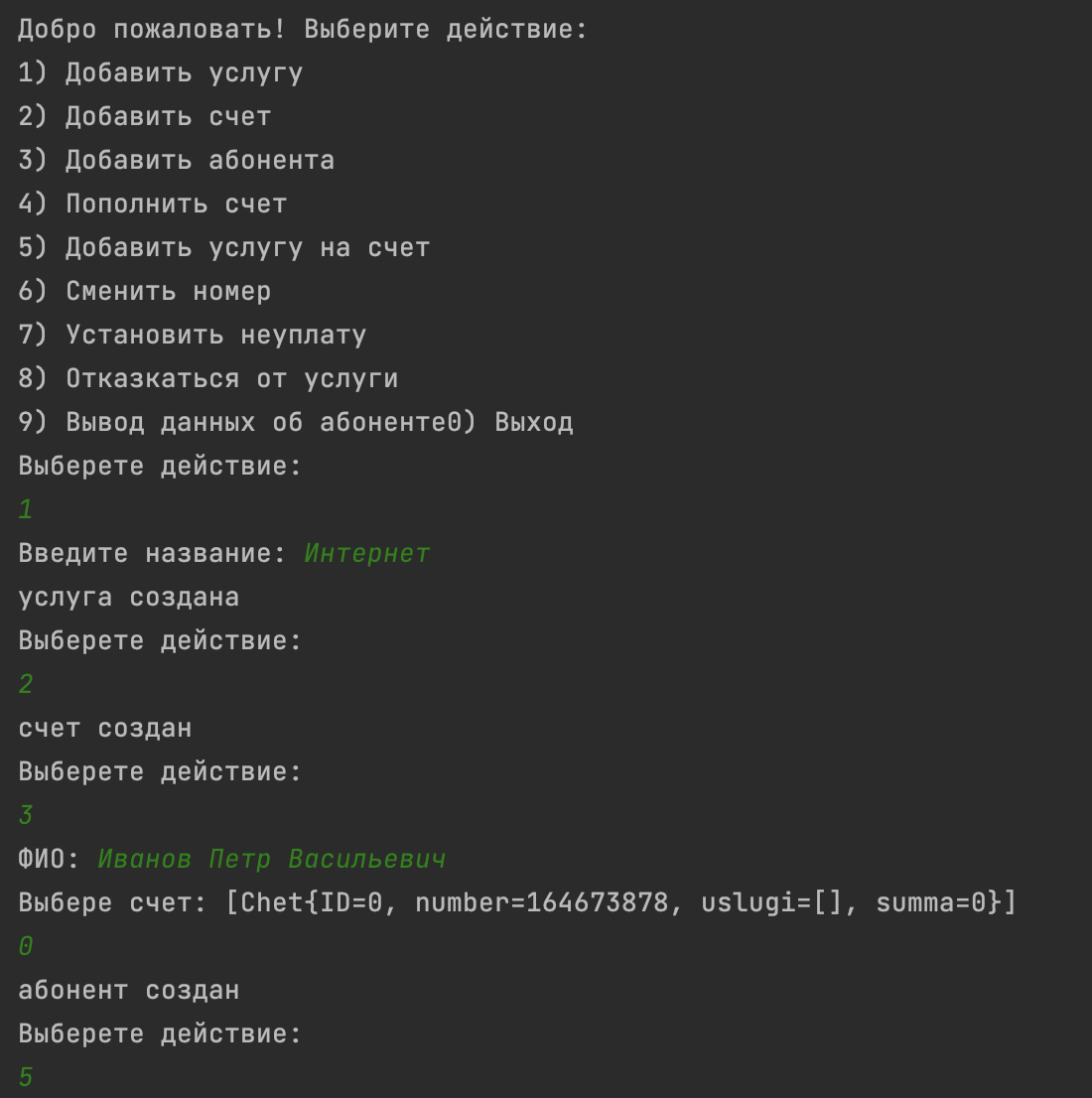


Рисунок 7 — пример выполнения

public class AutoBase {  
  
  
 public static void main(String[] args) {  
  
 Scanner myObj = new Scanner(System.in);  
 Driver driver = new Driver();  
 ArrayList<Driver> drivers = new ArrayList<Driver>(1);  
 Car car = new Car();  
 ArrayList<Car> cars = new ArrayList<>(1);  
 Flight flight= new Flight();  
 ArrayList<Flight> flights= new ArrayList<Flight>(1);  
 Disprtcher disprtcher= new Disprtcher();  
 String car\_name = "";  
 String driver\_name = "";  
 String flight\_name = "";  
 System.out.println("Добро пожаловать! Выберите действие:\n" +  
 "1) Добавить авто\n" +  
 "2) Добавить водителя\n" +  
 "3) Добавить рейс\n" +  
 "4) Распределение заявки и авто\n" +  
 "5) Заявка на ремонт\n" +  
 "6) Отстронить водителя\n" +  
 "7) Отметка о рейсе и состоянии авто\n" +  
 "8) Вывод данных диспетчера\n" +  
 "0) Выход");  
 while (true){  
 System.out.println("Выберете действие: ");  
 int tmp = myObj.nextInt();  
 switch (tmp) {  
 case 1:  
 System.out.print("Введите название авто: ");  
 car\_name = myObj.nextLine()+ myObj.nextLine();  
 car = new Car(car\_name);  
 cars.add(car);  
 System.out.println("авто добавлено");  
 break;  
 case 2:  
 System.out.print("Введите ФИО: ");  
 driver\_name = myObj.nextLine()+ myObj.nextLine();  
 driver = new Driver(driver\_name);  
 drivers.add(driver);  
 System.out.println("водитель создан");  
 break;  
 case 3:  
 System.out.print("Введите рейс: ");  
 flight\_name = myObj.nextLine()+ myObj.nextLine();  
 flight = new Flight(flight\_name);  
 flights.add(flight);  
 System.out.println("рейс создан");  
 break;  
 case 4:  
 System.out.print("Выберите рейс: ");  
 System.out.println(flights.toString());  
 int flight\_name1 = myObj.nextInt() ;  
 System.out.print("Выберите авто: ");  
 System.out.println(cars.toString());  
 int car\_name1 = myObj.nextInt();  
 System.out.print("Выберете водителя: ");  
 System.out.println(drivers.toString());  
 int driver\_name1 = myObj.nextInt();  
 disprtcher.zayavka(cars.get(car\_name1), flights.get(flight\_name1), drivers.get(driver\_name1));  
 break;  
 case 5:  
 System.out.print("Выберите авто: ");  
 System.out.println(cars.toString());  
 int car\_name2 = myObj.nextInt();  
 System.out.print("Выберете водителя: ");  
 System.out.println(drivers.toString());  
 int driver\_name2 = myObj.nextInt();  
 disprtcher.remont(cars.get(car\_name2), drivers.get(driver\_name2));  
 break;  
 case 6:  
 System.out.print("Выберете водителя: ");  
 System.out.println(drivers.toString());  
 int driver\_name3 = myObj.nextInt();  
 disprtcher.otstran(drivers.get(driver\_name3));  
 drivers.remove(driver\_name3);  
 break;  
 case 7:  
 System.out.print("Выберите рейс: ");  
 System.out.println(flights.toString());  
 int flight\_name4 = myObj.nextInt() ;  
 System.out.print("Выберите авто: ");  
 System.out.println(cars.toString());  
 int car\_name4 = myObj.nextInt();  
 System.out.print("Выберете водителя: ");  
 System.out.println(drivers.toString());  
 int driver\_name4 = myObj.nextInt();  
 System.out.print("Введите состояние авто: ");  
 String sost\_car = myObj.nextLine()+ myObj.nextLine();  
 System.out.print("Введите состояние рейса: ");  
 String sost\_fl = myObj.nextLine()+ myObj.nextLine();  
 disprtcher.otmetka(cars.get(car\_name4), drivers.get(driver\_name4), flights.get(flight\_name4), sost\_fl, sost\_car);  
 break;  
 case 8:  
 System.out.println(disprtcher.toString());  
 break;  
  
 case 0:  
 break;  
  
 }  
 if (tmp == 0){  
 break;  
 }  
 }  
  
 }  
}

Результат выполнения представлен на рисунке 8.

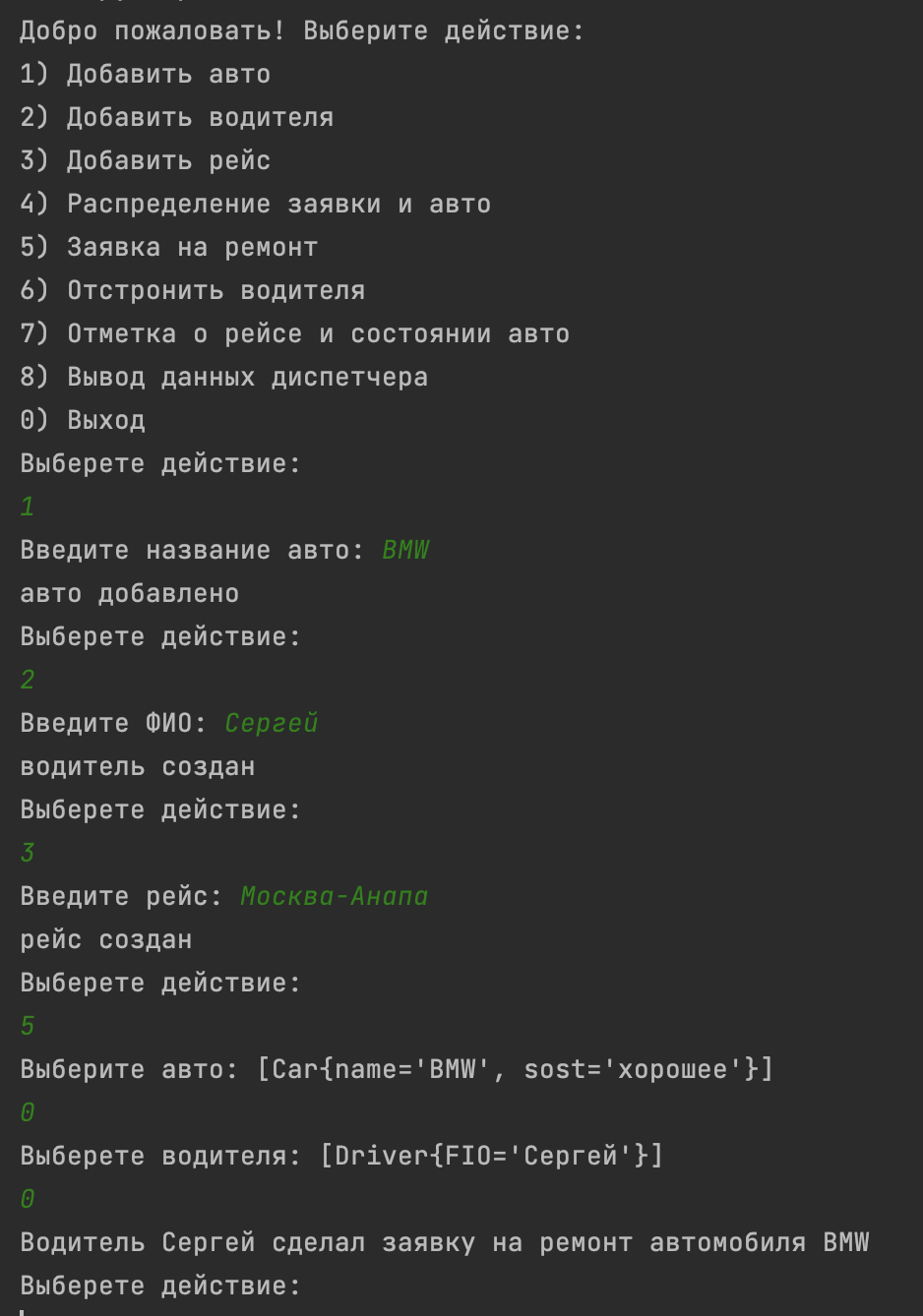


Рисунок 8 — пример выполнения

**Вывод:** были разработаны классы и методы согласно вариантам.