**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 **09.04.01/07 Интеллектуальные системы анализа, обработки и интерпретации больших данных**

О Т Ч Е Т

|  |  |
| --- | --- |
| **по лабораторной работе № 3** |  |

**Дисциплина:** Классы. Наследование. Полиморфизм

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

|  |  |  |  |
| --- | --- | --- | --- |
| Студент | ИУ6-22М |  | Д.М. Карасёв\_\_ |
|  | (Группа) | (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 Вектор размерности n. Реализовать методы сложения, вычитания, умножения, инкремента, декремента, индексирования. Определить массив из m объектов. Каждую из пар векторов передать в методы, возвращающие их скалярное произведение и длины. Вычислить и вывести углы между векторами.

2. Определить класс Вектор размерности n. Определить несколько конструкторов. Реализовать методы для вычисления модуля вектора, скалярного произведения, сложения, вычитания, умножения на константу. Объявить массив объектов. Написать метод, который для заданной пары векторов будет определять, являются ли они коллинеарными или ортогональными.

**Решение:**

public class Vector {  
 public int[] array;  
 public Vector(int size){  
 this.array = new int[size];  
 for (int i = 0; i < size; i++){  
 this.array[i] = (int) (Math.*random*()\*100);  
 }  
 }  
 public static Vector sum(Vector V1, Vector V2){  
 Vector resultV = null;  
 if(V1.array.length == V2.array.length){  
 resultV = new Vector(V1.array.length);  
 for (int i = 0; i < V1.array.length; i++){  
 resultV.array[i] = V1.array[i] + V2.array[i];  
 }  
 }  
 return resultV;  
 }  
 public static Vector sub(Vector V1, Vector V2){  
 Vector resultV = null;  
 if(V1.array.length == V2.array.length){  
 resultV = new Vector(V1.array.length);  
 for (int i = 0; i < V1.array.length; i++){  
 resultV.array[i] = V1.array[i] - V2.array[i];  
 }  
 }  
 return resultV;  
 }  
 public static Vector multiplication(Vector V1, Vector V2){  
 Vector resultV = null;  
 if(V1.array.length == V2.array.length){  
 resultV = new Vector(V1.array.length);  
 for (int i = 0; i < V1.array.length; i++){  
 resultV.array[i] = V1.array[i] \* V2.array[i];  
 }  
 }  
 return resultV;  
 }  
 public static Vector inc(Vector V1){  
 Vector resultV = new Vector(V1.array.length);  
 for (int i = 0; i < V1.array.length; i++){  
 resultV.array[i] = V1.array[i] + 1;  
 }  
 return resultV;  
 }  
 public static Vector dec(Vector V1){  
 Vector resultV = new Vector(V1.array.length);  
 for (int i = 0; i < V1.array.length; i++){  
 resultV.array[i] = V1.array[i] - 1;  
 }  
 return resultV;  
 }  
 public static int scalar(Vector V1, Vector V2){  
 int scalar = 0;  
 for(int i = 0; i < V1.array.length; i++){  
 scalar += V1.array[i] \* V2.array[i];  
 }  
 return scalar;  
 }  
 public static double length(Vector V1){  
 int length = 0;  
 for (int i = 0; i < V1.array.length; i++){  
 length += Math.*pow*(V1.array[i],2);  
 }  
 return Math.*sqrt*(length);  
 }  
 public static double angle(Vector V1, Vector V2){  
 double angle = *scalar*(V1,V2)/(*length*(V1) \* *length*(V2));  
 return Math.*toDegrees*(Math.*acos*(angle));  
 }  
}

public class lr3\_task1\_v1 {  
 public static void main(String[] args) {  
 Scanner in = new Scanner(System.*in*);  
 System.*out*.println("Введите кол-во векторов:");  
 int size\_array = in.nextInt();  
 Vector[] myArray = new Vector[size\_array];  
 System.*out*.println("Укажите размер вектора:");  
 int size\_vector = in.nextInt();  
 for (int i = 0; i < size\_array; i++){  
 myArray[i] = new Vector(size\_vector);  
 System.*out*.println(Arrays.*toString*(myArray[i].array));  
 }  
 for (int i = 0; i < size\_array; i++){  
 for (int j = i+1; j < size\_array; j++){  
 System.*out*.println();  
 System.*out*.println("Вычисления для векторов: " + Arrays.*toString*(myArray[i].array) + " " + Arrays.*toString*(myArray[j].array));  
 System.*out*.println("Скалярное произведение: " + Vector.*scalar*(myArray[i],myArray[j]));  
 System.*out*.println("Длина первого вектора: " + Vector.*length*(myArray[i]));  
 System.*out*.println("Длина второго вектора: " + Vector.*length*(myArray[j]));  
 System.*out*.println("Угол между векторами(в градусах): " + Vector.*angle*(myArray[i],myArray[j]));  
 }  
 }  
 }  
}

На рисунке 1 представлен результат выполнения.

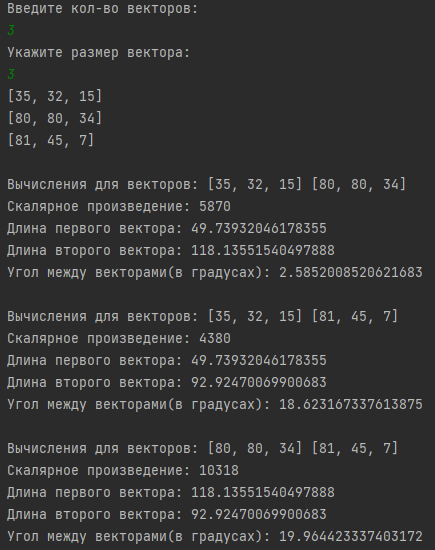


Рисунок 1 – Результат выполнения

public class lr3\_task2\_v1 {  
 public static void main(String[] args) {  
 Scanner in = new Scanner(System.*in*);  
 System.*out*.println("Введите кол-во векторов:");  
 int size\_array = in.nextInt();  
 Vector[] myArray = new Vector[size\_array];  
 System.*out*.println("Укажите размер вектора:");  
 int size\_vector = in.nextInt();  
 for (int i = 0; i < size\_array; i++) {  
 myArray[i] = new Vector(size\_vector);  
 System.*out*.println(Arrays.*toString*(myArray[i].array));  
 }  
 for (int i = 0; i < size\_array; i++) {  
 for (int j = i + 1; j < size\_array; j++) {  
 System.*out*.println();  
 System.*out*.println("Вычисления для векторов: " + Arrays.*toString*(myArray[i].array) + " " + Arrays.*toString*(myArray[j].array));  
 Vector.*CollinearAndOrto*(myArray[i], myArray[j]);  
 }  
 }  
 }  
}

public class Vector {  
 public float[] array;  
 public Vector(float[] array)  
 {  
 this.array = array;  
 }  
 public Vector(int size){  
 this.array = new float[size];  
 for (int i = 0; i < size; i++){  
 this.array[i] = (int) (Math.*random*()\*100);  
 }  
 }  
 public static Vector sum(Vector V1, Vector V2){  
 Vector resultV = null;  
 if(V1.array.length == V2.array.length){  
 resultV = new Vector(V1.array.length);  
 for (int i = 0; i < V1.array.length; i++){  
 resultV.array[i] = V1.array[i] + V2.array[i];  
 }  
 }  
 return resultV;  
 }  
 public static Vector sub(Vector V1, Vector V2){  
 Vector resultV = null;  
 if(V1.array.length == V2.array.length){  
 resultV = new Vector(V1.array.length);  
 for (int i = 0; i < V1.array.length; i++){  
 resultV.array[i] = V1.array[i] - V2.array[i];  
 }  
 }  
 return resultV;  
 }  
 public static Vector multiplication(Vector V1, float con){  
 Vector resultV = new Vector(V1.array.length);  
 for (int i = 0; i < V1.array.length; i++){  
 resultV.array[i] = V1.array[i] \* con;  
 }  
 return resultV;  
 }  
 public static Vector inc(Vector V1){  
 Vector resultV = new Vector(V1.array.length);  
 for (int i = 0; i < V1.array.length; i++){  
 resultV.array[i] = V1.array[i] + 1;  
 }  
 return resultV;  
 }  
 public static Vector dec(Vector V1){  
 Vector resultV = new Vector(V1.array.length);  
 for (int i = 0; i < V1.array.length; i++){  
 resultV.array[i] = V1.array[i] - 1;  
 }  
 return resultV;  
 }  
 public static int scalar(Vector V1, Vector V2){  
 int scalar = 0;  
 for(int i = 0; i < V1.array.length; i++){  
 scalar += V1.array[i] \* V2.array[i];  
 }  
 return scalar;  
 }  
 public static double length(Vector V1){  
 int length = 0;  
 for (int i = 0; i < V1.array.length; i++){  
 length += Math.*pow*(V1.array[i],2);  
 }  
 return Math.*sqrt*(length);  
 }  
 public static double angle(Vector V1, Vector V2){  
 double angle = *scalar*(V1,V2)/(*length*(V1) \* *length*(V2));  
 return Math.*toDegrees*(Math.*acos*(angle));  
 }  
 public static void CollinearAndOrto(Vector V1, Vector V2){  
 float k = 0;  
 for (int i = 0; i < V1.array.length; i++){  
 if ((V1.array[i] != 0) && (V2.array[i] != 0)){  
 k = V2.array[i]/V1.array[i];  
 }  
 }  
 if (Arrays.*equals*(V2.array,Vector.*multiplication*(V1,k).array)){  
 System.*out*.println("Данные векторы коллинеарны");  
 }  
 else {  
 System.*out*.println("Данные векторы не коллинеарны");  
 }  
 if (Vector.*scalar*(V1,V2) == 0){  
 System.*out*.println("Векторы ортогональны");  
 }  
 else {  
 System.*out*.println("Векторы не ортогональны");  
 }  
 }  
}

На рисунке 2 представлен результат выполнения.

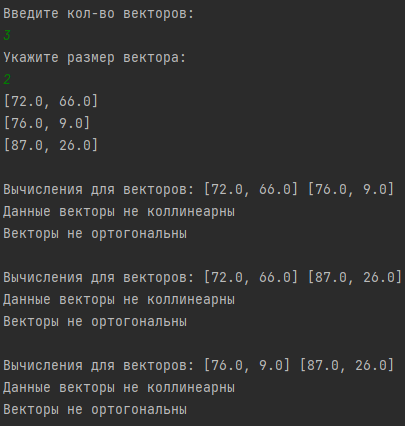


Рисунок 2 – Результат выполнения

**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. Student: id, Фамилия, Имя, Отчество, Дата рождения, Адрес, Телефон, Факультет, Курс, Группа. Создать массив объектов. Вывести: a) список студентов заданного факультета; b) списки студентов для каждого факультета и курса; c) список студентов, родившихся после заданного года; d) список учебной группы.

2. Customer: id, Фамилия, Имя, Отчество, Адрес, Номер кредитной карточки, Номер банковского счета. Создать массив объектов. Вывести: a) список покупателей в алфавитном порядке; b) список покупателей, у которых номер кредитной карточки находится в заданном интервале.

**Решение:**

public class Student {  
 private static int *Counter* = 0;  
 private int id, Course, Group;  
 private String LastName, FirstName, Patronymic, Address, Telephone, Faculty;  
 private LocalDate DateOfBirthday;  
 public Student( String lastName, String firstName, String middleName, LocalDate birthDate, String address, String phone, String faculty, int course, int group) {  
 this.id = *Counter*++;  
 this.LastName = lastName;  
 this.FirstName = firstName;  
 this.Patronymic = middleName;  
 this.DateOfBirthday = birthDate;  
 this.Address = address;  
 this.Telephone = phone;  
 this.Faculty = faculty;  
 this.Course = course;  
 this.Group = group;  
 }  
 //getters  
 public String getLastName(){  
 return this.LastName;  
 }  
 public String getFirstName(){  
 return this.FirstName;  
 }  
 public String getAddress(){  
 return this.Address;  
 }  
 public String getTelephone(){  
 return this.Telephone;  
 }  
 public String getPatronymic(){  
 return this.Patronymic;  
 }  
 public String getFaculty(){  
 return this.Faculty;  
 }  
 public int getCourse(){  
 return this.Course;  
 }  
 public int getGroup(){  
 return this.Group;  
 }  
 public LocalDate getDateOfBirthday(){  
 return this.DateOfBirthday;  
 }  
 public int getId(){  
 return this.id;  
 }  
 //setters  
 public void setId(int id) {  
 this.id = id;  
 }  
 public void setAddress(String address) {  
 this.Address = Address;  
 }  
 public void setCourse(int course) {  
 this.Course = course;  
 }  
 public void setDateOfBirthday(LocalDate dateOfBirthday) {  
 this.DateOfBirthday = dateOfBirthday;  
 }  
 public void setFaculty(String faculty) {  
 Faculty = faculty;  
 }  
 public void setTelephone(String telephone) {  
 Telephone = telephone;  
 }  
 public void setFirstName(String firstName) {  
 FirstName = firstName;  
 }  
 public void setGroup(int group) {  
 Group = group;  
 }  
 public void setLastName(String lastName) {  
 LastName = lastName;  
 }  
 public void setPatronymic(String patronymic) {  
 Patronymic = patronymic;  
 }  
 public String toString() {  
 return "Student{" +  
 "id=" + id +  
 ", lastName='" + LastName + '\'' +  
 ", firstName='" + FirstName + '\'' +  
 ", middleName='" + Patronymic + '\'' +  
 ", birthDate=" + DateOfBirthday +  
 ", address='" + Address + '\'' +  
 ", phone='" + Telephone + '\'' +  
 ", faculty='" + Faculty + '\'' +  
 ", course=" + Course +  
 ", group='" + Group + '\'' +  
 '}';  
 }  
}

public class lr3\_task1\_v2 {  
 public static void main(String[] args) {  
 StudentArray studentArray = new StudentArray();  
 studentArray.addStudent(new Student( "Ivanov", "Ivan", "Ivanovich", LocalDate.*of*(2000, 5, 1), "Moscow", "123456789", "Mathematics", 2, 101));  
 studentArray.addStudent(new Student( "Petrov", "Petr", "Petrovich", LocalDate.*of*(1999, 3, 15), "Moscow", "987654321", "Physics", 3, 201));  
 studentArray.addStudent(new Student( "Sidorov", "Sidor", "Sidorovich", LocalDate.*of*(2001, 7, 20), "St. Petersburg", "555555555", "Mathematics", 1, 101));  
 studentArray.addStudent(new Student( "Kuznetsov", "Igor", "Igorevich", LocalDate.*of*(1998, 12, 10), "Moscow", "111111111", "Physics", 3, 202));  
 System.*out*.println("Full list of students");  
 studentArray.printAllStudents();  
 System.*out*.println();  
 System.*out*.println("List of students of the faculty of Mathematics");  
 studentArray.printStudentsCurrentFaculty("Mathematics");  
 System.*out*.println();  
 System.*out*.println("List of students by all courses and faculty");  
 studentArray.printStudentsByFacultyAndCourse();  
 System.*out*.println();  
 System.*out*.println("List of students born after a given year");  
 studentArray.printStudentsBirthAfterDate(LocalDate.*of*(2000, 4, 1));  
 System.*out*.println("Study group list");  
 studentArray.printStudentsCurrentlyGroup(3, 201, "Physics");  
 }  
}

На рисунке 3 представлен результат.

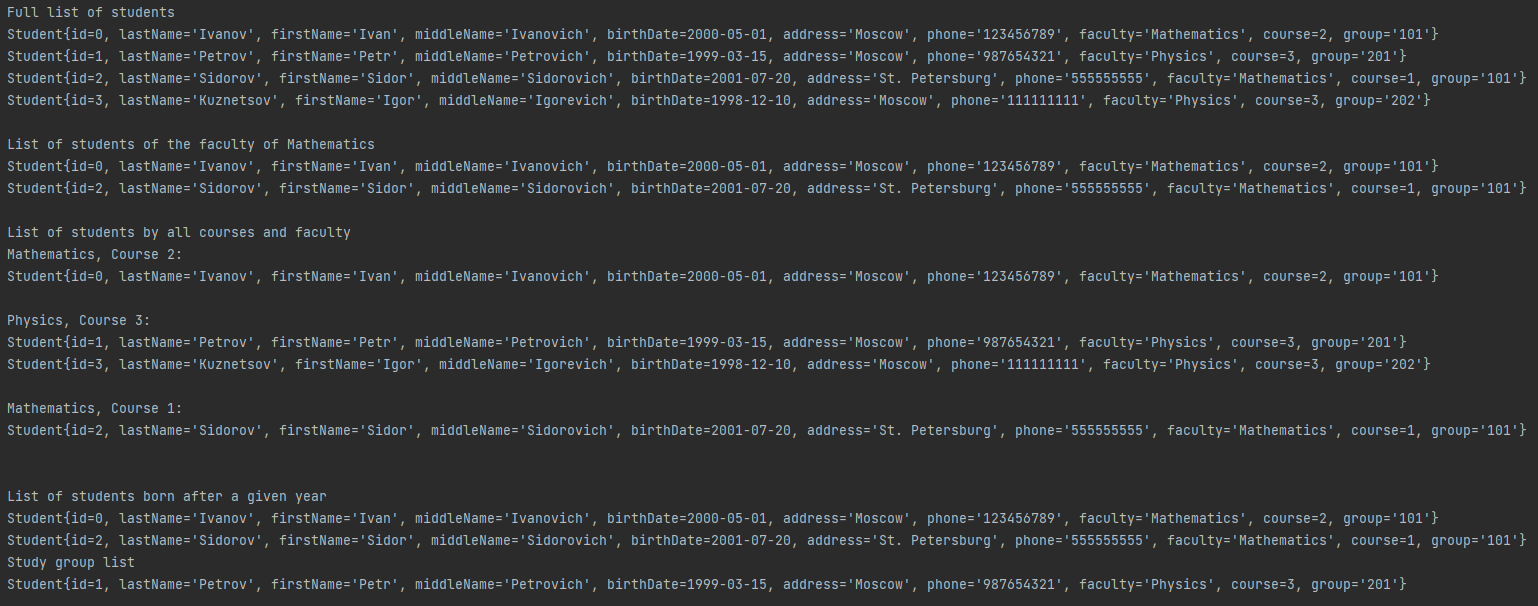


Рисунок 3 – Результат выполнения

package lr3\_task2\_v2;  
  
public class Customer {  
 private int id;  
 private String lastName;  
 private String firstName;  
 private String middleName;  
 private String address;  
 private int creditCardNumber;  
 private int bankAccountNumber;  
  
 public Customer(int id, String lastName, String firstName, String middleName, String address,  
 int creditCardNumber, int bankAccountNumber) {  
 this.id = id;  
 this.lastName = lastName;  
 this.firstName = firstName;  
 this.middleName = middleName;  
 this.address = address;  
 this.creditCardNumber = creditCardNumber;  
 this.bankAccountNumber = bankAccountNumber;  
 }  
  
 public int getId() {  
 return id;  
 }  
  
 public void setId(int id) {  
 this.id = id;  
 }  
  
 public String getLastName() {  
 return lastName;  
 }  
  
 public void setLastName(String lastName) {  
 this.lastName = lastName;  
 }  
  
 public String getFirstName() {  
 return firstName;  
 }  
  
 public void setFirstName(String firstName) {  
 this.firstName = firstName;  
 }  
  
 public String getMiddleName() {  
 return middleName;  
 }  
  
 public void setMiddleName(String middleName) {  
 this.middleName = middleName;  
 }  
  
 public String getAddress() {  
 return address;  
 }  
  
 public void setAddress(String address) {  
 this.address = address;  
 }  
  
 public long getCreditCardNumber() {  
 return creditCardNumber;  
 }  
  
 public void setCreditCardNumber(int creditCardNumber) {  
 this.creditCardNumber = creditCardNumber;  
 }  
  
 public long getBankAccountNumber() {  
 return bankAccountNumber;  
 }  
  
 public void setBankAccountNumber(int bankAccountNumber) {  
 this.bankAccountNumber = bankAccountNumber;  
 }  
 public String toString() {  
 return "Customer{" +  
 "id=" + id +  
 ", lastName='" + lastName + '\'' +  
 ", firstName='" + firstName + '\'' +  
 ", middleName='" + middleName + '\'' +  
 ", address='" + address + '\'' +  
 ", creditCardNumber=" + creditCardNumber +  
 ", bankAccountNumber=" + bankAccountNumber +  
 '}';  
 }  
}

package lr3\_task2\_v2;  
  
import java.util.ArrayList;  
import java.util.Collection;  
import java.util.Collections;  
import java.util.List;  
  
public class CustomerArray {  
 private ArrayList<Customer> CustomerArray;  
 public CustomerArray(){  
 CustomerArray = new ArrayList<Customer>();  
 }  
 public void addCustomer(Customer C){  
 CustomerArray.add(C);  
 }  
 public void ptintlAllCustomer(){  
 for (Customer customer : CustomerArray){  
 System.out.println(customer.toString());  
 }  
 }  
 public void printCustomersInAlphabeticalOrder(){  
 Collections.sort(CustomerArray, new CustomerCompare());  
 ptintlAllCustomer();  
 }  
 public void printCreidtCardNumberInRange(int n1, int n2){  
 for (Customer customer : CustomerArray){  
 if (customer.getCreditCardNumber() >= n1 && customer.getCreditCardNumber() <= n2){  
 System.out.println(customer.toString());  
 }  
 }  
 }  
}

package lr3\_task2\_v2;  
  
public class lr3\_task2\_v2 {  
 public static void main(String[] args) {  
 CustomerArray customerArray = new CustomerArray();  
 customerArray.addCustomer(new Customer(1, "Ivanov", "Ivan", "Ivanovich", "Moscow", 741243241, 12345671));  
 customerArray.addCustomer(new Customer(2, "Petrov", "Petr", "Petrovich", "Moscow", 234\_567\_821, 234567890));  
 customerArray.addCustomer(new Customer(3, "Sidorov", "Sidor", "Sidorovich", "Moscow", 545678903, 345678901));  
 customerArray.addCustomer(new Customer(4, "Kuznetsov", "Igor", "Igorevich", "Moscow", 271423415, 789673434));  
 System.out.println("Full list of Customer:");  
 customerArray.ptintlAllCustomer();  
 System.out.println();  
 System.out.println("Customers in alphabetical order:");  
 customerArray.printCustomersInAlphabeticalOrder();  
 System.out.println();  
 System.out.println("Credit number card in current range");  
 customerArray.printCreidtCardNumberInRange(200\_000\_00, 300\_000\_000);  
 }  
}

На рисунке 4 представлен результат выполнения.

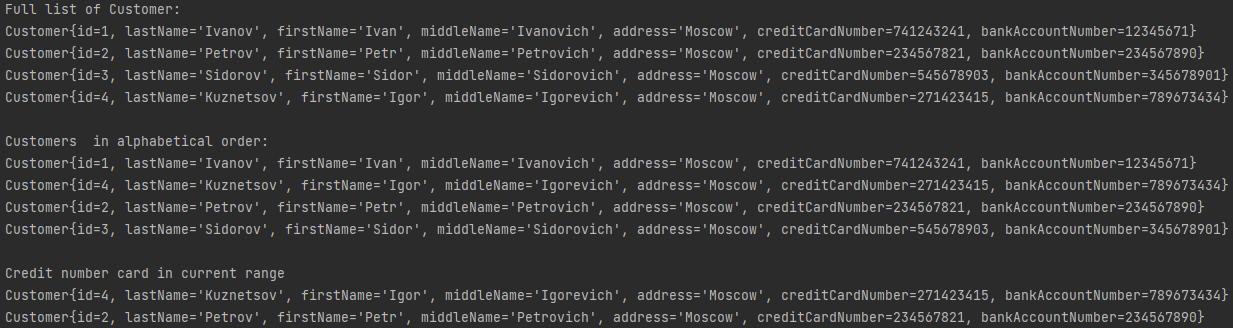


Рисунок 4 – Результат выполнения

**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 Текстовый файл, используя класс Файл. Методы: создать, переименовать, вывести на консоль содержимое, дополнить, удалить.
2. Создать объект класса Одномерный массив, используя класс Массив. Методы: создать, вывести на консоль, выполнить операции (сложить, вычесть, перемножить).

**Решение:**

package lr3\_task1\_v3;  
  
public class File {  
 public String NameOfFile;  
  
 public File(String NameOfFile){  
 this.NameOfFile = NameOfFile;  
 }  
  
 public String getNameOfFile() {  
 return NameOfFile;  
 }  
  
 public void setNameOfFile(String nameOfFile) {  
 NameOfFile = nameOfFile;  
 }  
  
}

package lr3\_task1\_v3;  
  
import lr3\_task11\_v3.Day;  
  
import java.util.Objects;  
  
public class TextFile extends File {  
 public String content;  
 public TextFile(String name){  
 super(name);  
 this.content = "";  
 }  
  
 public void addContent(String content){  
 this.content += content;  
 }  
  
 public void deleteContent(){  
 this.content = "";  
 }  
  
 public void currentContent(){  
 System.out.println(this.content);  
 }  
  
 @Override  
 public int hashCode() {  
 return Objects.hash(NameOfFile, content);  
 }  
  
 @Override  
 public boolean equals(Object obj) {  
 if (this == obj) return true;  
 if (obj == null || getClass() != obj.getClass()) return false;  
 TextFile textFile = (TextFile) obj;  
 return Objects.equals(NameOfFile, textFile.NameOfFile) && Objects.equals(content, textFile.content);  
 }  
  
 @Override  
 public String toString() {  
 return "Name of file: " + NameOfFile + ", file contents:" + content;  
 }  
}

package lr3\_task1\_v3;  
  
public class lr3\_task1\_v3 {  
 public static void main(String[] args) {  
 TextFile file = new TextFile("example.txt");  
 file.addContent("Hello, world!");  
 file.currentContent();  
  
 file.setNameOfFile("new\_example.txt");  
 System.out.println(file);  
  
 file.addContent("\nThis is some additional content.");  
 file.currentContent();  
  
 file.deleteContent();  
 file.currentContent();  
 }  
}

На рисунке 5 представлен результат выполнения.

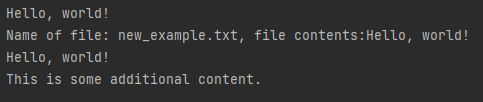


Рисунок 5 – Результат выполнения

package lr3\_task11\_v3;  
  
import java.util.Objects;  
  
public class Day {  
 Hour hour;  
 Minute minute;  
 public Day(Hour hour, Minute minute){  
 this.hour = hour;  
 this.minute = minute;  
 }  
 public Hour getHour() {  
 return hour;  
 }  
  
 public void setHour(Hour hour) {  
 this.hour = hour;  
 }  
  
 public Minute getMinute() {  
 return minute;  
 }  
  
 public void setMinute(Minute minute) {  
 this.minute = minute;  
 }  
 public void currentTime(){  
 System.*out*.println("Current time: " + hour.toString() + ":" + minute.toString());  
 }  
 public void calculateDayTime(){  
 int hour = this.hour.getHour();  
 if (hour >= 4 && hour < 12) {  
 System.*out*.println("It's morning!");  
 } else if (hour >= 12 && hour < 18) {  
 System.*out*.println("It's afternoon!");  
 } else if (hour >= 18 && hour < 22) {  
 System.*out*.println("It's evening!");  
 } else {  
 System.*out*.println("It's night!");  
 }  
 }  
  
 @Override  
 public boolean equals(Object obj) {  
 if (this == obj) return true;  
 if (obj == null || getClass() != obj.getClass()) return false;  
 Day day = (Day) obj;  
 return Objects.*equals*(hour, day.hour) && Objects.*equals*(minute, day.minute);  
 }  
  
 @Override  
 public int hashCode() {  
 return Objects.*hash*(hour, minute);  
 }  
  
 @Override  
 public String toString() {  
 return hour + ":" + minute;  
 }  
}

package lr3\_task11\_v3;  
  
public class Hour {  
 private int hour;  
  
 public Hour(int hour) {  
 this.hour = hour;  
 }  
  
 public int getHour() {  
 return hour;  
 }  
  
 public void setHour(int hour) {  
 this.hour = hour;  
 }  
  
 @Override  
 public String toString() {  
 return String.format("%d", hour);  
 }  
}

package lr3\_task11\_v3;  
  
public class Minute {  
 private int minute;  
  
 public Minute(int minute) {  
 this.minute = minute;  
 }  
  
 public int getMinute() {  
 return minute;  
 }  
  
 public void setMinute(int minute) {  
 this.minute = minute;  
 }  
  
 @Override  
 public String toString() {  
 return String.*format*("%d", minute);  
 }  
}

package lr3\_task11\_v3;  
  
import java.lang.reflect.Type;  
import java.time.LocalTime;  
  
public class lr3\_task11\_v3 {  
 public static void main(String[] args) {  
 LocalTime localTime = LocalTime.*now*();  
 Minute minute = new Minute(localTime.getMinute());  
 Hour hour = new Hour(localTime.getHour());  
 Day day = new Day(hour, minute);  
 day.currentTime();  
 day.calculateDayTime();  
 }  
}

На рисунке 6 представлен результат выполнения.



Рисунок 6 – Результат выполнения

**Вариант 4:**

Построить модель программной системы.

1. Система Факультатив. Преподаватель объявляет запись на Курс. Студент записывается на Курс, обучается и по окончании Преподаватель выставляет Оценку, которая сохраняется в Архиве. Студентов, Преподавателей и Курсов при обучении может быть несколько.

2. Система Платежи. Клиент имеет Счет в банке и Кредитную Карту (КК). Клиент может оплатить Заказ, сделать платеж на другой Счет, заблокировать КК и аннулировать Счет. Администратор может заблокировать КК за превышение кредита.

**Решение:**

package lr3\_task1\_v4;  
  
import java.util.ArrayList;  
  
public class Archive {  
 private ArrayList<Grade> ListOfGrage;  
 public Archive(){  
 this.ListOfGrage = new ArrayList<Grade>();  
 }  
 public void addGrade(Grade grade){  
 ListOfGrage.add(grade);  
 }  
  
 public void printArchiveGrade(){  
 for(Grade grade : ListOfGrage){  
 System.*out*.println(grade.toString());  
 System.*out*.println();  
 }  
 }  
}

package lr3\_task1\_v4;  
  
import java.util.ArrayList;  
  
public class Course {  
 private int id;  
 private String NameOfCourse;  
 private Teacher teacher;  
 private ArrayList<Student> students;  
 public Course(int id, String nameOfCourse, Teacher teacher){  
 this.id = id;  
 this.NameOfCourse = nameOfCourse;  
 this.teacher = teacher;  
 this.students = new ArrayList<Student>();  
 }  
  
 public void addStudent(Student student){  
 students.add(student);  
 }  
  
 public Teacher getTeacher() {  
 return teacher;  
 }  
  
 public ArrayList<Student> getStudents() {  
 return students;  
 }  
  
 public String getNameOfCourse() {  
 return NameOfCourse;  
 }  
  
 @Override  
 public String toString() {  
 return "id: " + id + " Name of course: " + NameOfCourse;  
 }  
}

package lr3\_task1\_v4;  
  
public class Grade {  
 private Teacher teacher;  
 private Course course;  
 private Student student;  
 private int grade;  
 public Grade(Teacher teacher, Course course, Student student, int grade){  
 this.course = course;  
 this.teacher = teacher;  
 this.student = student;  
 this.grade = grade;  
 }  
  
 @Override  
 public String toString() {  
 return "Lastname of Teacher: " + teacher.getLastName() +  
 "\nName of Course: " + course.getNameOfCourse() +  
 "\nLastname of student: " + student.getLastName() +  
 "\nGrade: " + grade;  
 }  
  
 public int getGrade() {  
 return grade;  
 }  
  
 public void setGrade(byte grade) {  
 this.grade = grade;  
 }  
  
 public Course getCourse() {  
 return course;  
 }  
  
 public Student getStudent() {  
 return student;  
 }  
  
 public Teacher getTeacher() {  
 return teacher;  
 }  
}

package lr3\_task1\_v4;  
  
import java.util.ArrayList;  
import java.util.Scanner;  
  
public class lr3\_task1\_v4 {  
 public static void main(String[] args) {  
 ArrayList<Teacher> teachers = new ArrayList<Teacher>();  
 ArrayList<Student> students = new ArrayList<Student>();  
 ArrayList<Course> courses = new ArrayList<Course>();  
 Archive archive = new Archive();  
 Scanner in = new Scanner(System.*in*);  
  
 teachers.add(new Teacher(1,"Sidr","Sidorov"));  
 teachers.add(new Teacher(2,"Petr", "Petrov"));  
 teachers.add(new Teacher(3,"Ivan", "Ivanov"));  
  
 students.add(new Student(1,"Adam", "Adamov"));  
 students.add(new Student(2, "Viktor", "Viktorov"));  
 students.add(new Student(3, "Danila", "Karasyov"));  
  
 while (true){  
 System.*out*.println("Select role:\n1.Teacher\n2.Student");  
 int selected = in.nextInt();  
 switch (selected){  
 case 1:  
 System.*out*.println("Choose yourself:");  
 for (Teacher teacher : teachers){  
 System.out.println(teacher.toString());  
 }  
 int SelectedTeacher = in.nextInt();  
 System.out.println("Select task:\n1.Create course registration\n2.Set grade");  
 int SelectedTeacherTask = in.nextInt();  
 switch (SelectedTeacherTask){  
 case 1:  
 System.out.println("Enter name of course:");  
 String NameOfCourse = in.next();  
 Course newCourse = new Course(courses.size(),NameOfCourse, teachers.get(SelectedTeacher-1));  
 courses.add(newCourse);  
 teachers.get(SelectedTeacher-1).createCourseRegistration(newCourse);  
 break;  
 case 2:  
 System.out.println("Select course:");  
 for (Course course : teachers.get(SelectedTeacher-1).getCourses()){  
 System.out.println(course.toString());  
 }  
 int SelectedTeacherCourse = in.nextInt();  
 System.out.println("Select student:");  
 for (Student student : courses.get(SelectedTeacherCourse).getStudents()){  
 System.out.println(student.toString());  
 }  
 int SelectedTeacherStudent = in.nextInt();  
 System.out.println("Grade of student:");  
 int GradeStudent = in.nextInt();  
 teachers.get(SelectedTeacher-1).setGrade(archive, students.get(SelectedTeacherStudent-1), courses.get(SelectedTeacherCourse), GradeStudent);  
 System.out.println("Student's grade is saved");  
 System.out.println("Archive:");  
 archive.printArchiveGrade();  
 }  
 break;  
 case 2:  
 System.out.println("Choose yourself:");  
 for (Student student : students){  
 System.out.println(student.toString());  
 }  
 int SelectedStudent = in.nextInt();  
 System.out.println("Select course for registration:");  
 for (Course course: courses){  
 System.out.println(course.toString());  
 }  
 int SelectedStudentCourse = in.nextInt();  
 courses.get(SelectedStudentCourse).addStudent(students.get(SelectedStudent-1));  
 break;  
 }  
 }  
 }  
}

package lr3\_task1\_v4;  
  
import java.util.ArrayList;  
  
public class Student {  
 private int id;  
 private String firstName, lastName;  
 private ArrayList<Course> courses;  
 private ArrayList<Grade> grades;  
 public Student(int id, String firstName, String lastName){  
 this.id = id;  
 this.firstName = firstName;  
 this.lastName = lastName;  
 this.courses = new ArrayList<Course>();  
 this.grades = new ArrayList<Grade>();  
 }  
  
 public void getAvailableCourses(ArrayList<Course> Courses){  
 for (Course course : Courses){  
 System.*out*.println(course.toString());  
 }  
 }  
  
 public void registrationOnCourse(Course currentCourse){  
 currentCourse.addStudent(this);  
 }  
  
 public void getListOfGrades(){  
 for (Grade grade : grades){  
 System.*out*.println(grade.toString());  
 }  
 }  
  
 public String getLastName() {  
 return lastName;  
 }  
  
 public String getFirstName() {  
 return firstName;  
 }  
  
 public ArrayList<Course> getCourses() {  
 return courses;  
 }  
  
 public ArrayList<Grade> getGrades() {  
 return grades;  
 }  
  
 @Override  
 public String toString() {  
 return "Id: " + id + " Lastname: " + lastName + " Firstname: " + firstName;  
 }  
}

package lr3\_task1\_v4;  
  
import lr3\_task11\_v3.Day;  
  
import java.util.ArrayList;  
import java.util.Objects;  
  
public class Teacher {  
 private int id;  
 private String firstName;  
 private String lastName;  
 private ArrayList<Course> courses;  
 public Teacher(int id, String firstName, String lastName){  
 this.id = id;  
 this.firstName = firstName;  
 this.lastName = lastName;  
 this.courses = new ArrayList<Course>();  
 }  
 public void createCourseRegistration(Course course){  
 courses.add(course);  
 }  
 public void setGrade(Archive archive, Student student, Course course, int grade){  
 Grade newGrade = new Grade(this, course, student, grade);  
 student.getGrades().add(newGrade);  
 archive.addGrade(newGrade);  
 }  
  
 public String getFirstName() {  
 return firstName;  
 }  
  
 public String getLastName() {  
 return lastName;  
 }  
  
 public ArrayList<Course> getCourses() {  
 return courses;  
 }  
  
 @Override  
 public String toString() {  
 return "Id: " + id + " Lastname: " + lastName + " Firstname: " + firstName;  
 }  
  
 @Override  
 public boolean equals(Object obj) {  
 if (this == obj) return true;  
 if (obj == null || getClass() != obj.getClass()) return false;  
 Teacher teacher = (Teacher) obj;  
 return firstName.equals(teacher.firstName) && lastName.equals(teacher.lastName);  
 }  
}

На рисунке 7 представлен результат выполнения.

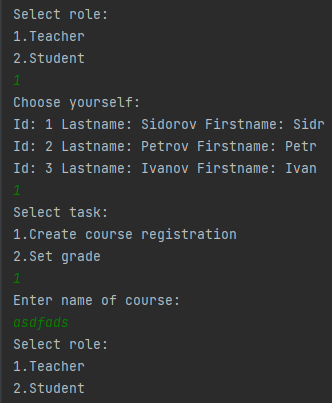


Рисунок 7 – Результат выполнения

package lr3\_task2\_v4;  
  
public class Account {  
 private int numberOfAccount, balance;  
 private Client client;  
 public Account(int numberOfAccount, Client client){  
 this.numberOfAccount= numberOfAccount;  
 this.balance = 0;  
 this.client = client;  
 }  
  
 public boolean makePayment(int deposit){  
 if(deposit < balance){  
 balance -= deposit;  
 System.out.println("The operation was successfully carried out");  
 return true;  
 }else {  
 System.out.println("Not enough money.");  
 return false;  
 }  
 }  
  
 public void takePayment(int deposit){  
 balance += deposit;  
 }  
  
 public void cancelBalance(){  
 balance = 0;  
 }  
 public int getNumberOfAccount() {  
 return numberOfAccount;  
 }  
  
 public Client getClient() {  
 return client;  
 }  
  
  
  
 public int getBalance() {  
 return balance;  
 }  
  
 @Override  
 public String toString() {  
 return "Account{" +  
 "numberOfAccount=" + numberOfAccount +  
 ", balance=" + balance +  
 '}';  
 }  
}

package lr3\_task2\_v4;  
  
public class Administrator {  
 private String NameOfAdministrator;  
 public Administrator(String nameOfAdministrator){  
 this.NameOfAdministrator = nameOfAdministrator;  
 }  
 public void blockedCreditCard(CreditCard creditCard){  
 if(creditCard.getBalance() > creditCard.getCreditLimit()){  
 System.*out*.println("Limit has been exceeded. CreditCard will be blocked...");  
 creditCard.block();  
 }else {  
 System.*out*.println("Limit has not been exceeded");  
 }  
 }  
}

package lr3\_task2\_v4;  
  
import java.util.ArrayList;  
  
public class Client {  
 private int id;  
 private String Name;  
 private CreditCard creditCard;  
 private Account account;  
 private ArrayList<Order> orders;  
 public Client(int id, String name){  
 this.id = id;  
 this.Name = name;  
 this.orders = new ArrayList<Order>();  
 }  
  
 public void setAccount(Account account) {  
 this.account = account;  
 }  
  
 public void setCreditCard(CreditCard creditCard) {  
 this.creditCard = creditCard;  
 }  
  
 public void makeOrderWithCreditCard(Order order){  
 if(this.creditCard.makePayment(order.getCost())){  
 orders.add(order);  
 }  
 }  
  
 public void makePaymentWithCreditCard(int deposit, Account account){  
 if(this.creditCard.makePayment(deposit)){  
 account.takePayment(deposit);  
 }  
 }  
  
 public void makeOrderWithAccount(Order order){  
 if(this.account.makePayment(order.getCost())){  
 orders.add(order);  
 }  
 }  
  
 public void makePaymentWithAccount(int deposit, Account account){  
 if(this.account.makePayment(deposit)){  
 account.takePayment(deposit);  
 }  
 }  
  
 public void blockCreditCard(){  
 creditCard.block();  
 }  
  
 public void cancelBankAccount(){  
 account.cancelBalance();  
 }  
  
 public Account getAccount() {  
 return account;  
 }  
  
 public ArrayList<Order> getOrders() {  
 return orders;  
 }  
  
 public CreditCard getCreditCard() {  
 return creditCard;  
 }  
  
 public String getName() {  
 return Name;  
 }  
  
 @Override  
 public String toString() {  
 return "Client{" +  
 "id=" + id +  
 ", Name='" + Name + '\'' +  
 '}';  
 }  
}

package lr3\_task2\_v4;  
  
public class CreditCard {  
 private int numberOfCard;  
 private Client client;  
 private int creditLimit;  
 private int balance;  
 private boolean isBlocked;  
  
 public CreditCard(int numberOfCard, Client client) {  
 this.numberOfCard = numberOfCard;  
 this.creditLimit = 1\_000\_000;  
 this.balance = 0;  
 this.isBlocked = false;  
 }  
  
 public int getNumberOfCard() {  
 return numberOfCard;  
 }  
  
 public Client getClient() {  
 return client;  
 }  
  
 public int getCreditLimit() {  
 return creditLimit;  
 }  
  
 public int getBalance() {  
 return balance;  
 }  
  
 public boolean isBlocked() {  
 return isBlocked;  
 }  
  
 public boolean makePayment(int amount) {  
 if (!isBlocked) {  
 if (balance + amount <= creditLimit) {  
 System.*out*.println("The operation was successfully carried out");  
 balance += amount;  
 return true;  
 } else {  
 System.*out*.println("Credit limit exceeded. Transaction declined.");  
 return false;  
 }  
 } else {  
 System.*out*.println("Credit card is blocked. Transaction declined.");  
 return false;  
 }  
 }  
 public void block() {  
 isBlocked = true;  
 }  
  
 public void unblock() {  
 isBlocked = false;  
 }  
  
 @Override  
 public String toString() {  
 return "Number of card: " + numberOfCard + ", balance: " + balance;  
 }  
}

package lr3\_task2\_v4;  
  
import lr3\_task1\_v4.Teacher;  
  
import java.util.ArrayList;  
import java.util.Scanner;  
  
public class lr3\_task2\_v4 {  
 public static void main(String[] args) {  
 Scanner in = new Scanner(System.*in*);  
 ArrayList<Account> accounts = new ArrayList<Account>();  
 ArrayList<CreditCard> creditCards = new ArrayList<CreditCard>();  
 ArrayList<Client> clients = new ArrayList<Client>();  
 ArrayList<Order> orders = new ArrayList<Order>();  
  
 Administrator administrator = new Administrator("Admin Adminov");  
  
 clients.add(new Client(1, "Ivan Ivanov"));  
 clients.add(new Client(2, "Sidr Sidorov"));  
 clients.add(new Client(3,"Petr Petrov"));  
  
 orders.add(new Order(100000, "Computer"));  
 orders.add(new Order(5000, "Sweater"));  
 orders.add(new Order(3000,"Jeans"));  
  
 while (true) {  
 System.*out*.println("Select role:\n1.User\n2.Administrator");  
 int selected = in.nextInt();  
 switch (selected) {  
 case 1:  
 System.*out*.println("Choose yourself:");  
 for (Client client : clients){  
 System.*out*.println(client.toString());  
 }  
 //int SelectedClient = in.nextInt();  
 Client client = clients.get(in.nextInt()-1);  
 System.*out*.println("Select task:\n1.Create credit card\n2.Create bank account\n3.Top up your account" +  
 "\n4.Make a purchase\n5.Blocked credit card\n6.Cancel the account\n7.Transfer between accounts.\n8.Balance of account/credit card");  
 int SelectedClientTask = in.nextInt();  
 switch (SelectedClientTask){  
 case 1:  
 CreditCard newCreditCard = new CreditCard((int)(Math.*random*()\*1000000000), client);  
 creditCards.add(newCreditCard);  
 client.setCreditCard(newCreditCard);  
 System.*out*.println("The credit card was successfully created");  
 System.*out*.println(client.getCreditCard().toString());  
 break;  
 case 2:  
 Account newAccount = new Account((int)(Math.*random*()\*1000000000), client);  
 accounts.add(newAccount);  
 client.setAccount(newAccount);  
 System.*out*.println("The bank account was successfully created");  
 System.*out*.println(client.getAccount().toString());  
 break;  
 case 3:  
 System.*out*.println("Enter a deposit to top up");  
 client.getAccount().takePayment(in.nextInt());  
 System.*out*.println(client.getAccount().toString());  
 break;  
 case 4:  
 System.*out*.println("Select a purchase:");  
 for (Order order : orders){  
 System.*out*.println(order.toString());  
 }  
 Order order = orders.get(in.nextInt()-1);  
 System.*out*.println("Choose a payment method:\n1.Bank Account\n2.Credit card");  
 switch (in.nextInt()){  
 case 1:  
 client.makeOrderWithAccount(order);  
 break;  
 case 2:  
 client.makeOrderWithCreditCard(order);  
 break;  
 }  
 break;  
 case 5:  
 System.*out*.println("Credit card was clsoed");  
 client.blockCreditCard();  
 break;  
 case 6:  
 System.*out*.println("Bank account was canceled");  
 client.cancelBankAccount();  
 break;  
 case 7:  
 System.*out*.println("Select other account:");  
 for (Account account : accounts){  
 System.*out*.println(account.toString());  
 }  
 Account account = accounts.get(in.nextInt()-1);  
 System.*out*.println("Enter the transfer amount:");  
 int amount = in.nextInt();  
 System.*out*.println("Choose a payment method:\n1.Bank Account\n2.Credit card");  
 switch (in.nextInt()){  
 case 1:  
 client.makePaymentWithAccount(amount,account);  
 break;  
 case 2:  
 client.makePaymentWithCreditCard(amount,account);  
 break;  
 }  
 break;  
 case 8:  
 System.*out*.println("Balance of Account: " + client.getAccount().getBalance());  
 System.*out*.println("Balance of Credit Card: " + client.getCreditCard().getBalance());  
 }  
 break;  
 case 2:  
 System.*out*.println("Select a credit card for block:");  
 for (CreditCard creditCard: creditCards){  
 System.*out*.println(creditCard.toString());  
 }  
 CreditCard creditCardForBlock = creditCards.get(in.nextInt()-1);  
 administrator.blockedCreditCard(creditCardForBlock);  
 break;  
 }  
 }  
  
 }  
}

package lr3\_task2\_v4;  
  
public class Order {  
 private String descriptionOfOrder;  
 private int cost;  
 public Order(int cost, String descriptionOfOrder){  
 this.cost = cost;  
 this.descriptionOfOrder = descriptionOfOrder;  
 }  
  
 public int getCost() {  
 return cost;  
 }  
  
 public String getDescriptionOfOrder() {  
 return descriptionOfOrder;  
 }  
  
 @Override  
 public String toString() {  
 return "Description: " + descriptionOfOrder + ", Cost: " + cost;  
 }  
}

На рисунке 8 представлен результат выполнения.

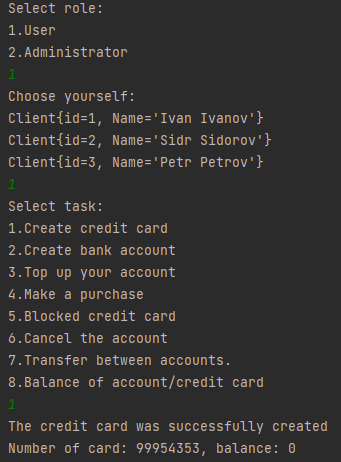


Рисунок 8 – Результат выполнения

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