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

**Отчет**

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

**Название:** Классы, наследование, полиморфизм

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

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Студент | ИУ6-22М |  |  | С.В. Астахов | |
|  | (Группа) |  | (Подпись, дата) | | (И.О. Фамилия) |
|  |  |  |  | |  |
| Преподаватель |  |  |  | | П.В. Степанов |
|  |  |  | (Подпись, дата) | | (И.О. Фамилия) |

Москва, 2024

**Цель работы:** освоить базовые принципы работы с ООП на языке Java.

**Вариант: 1.**

**Задание 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.

Код решения приведен в листинге 1.

Листинг 1 — реализация решения

|  |
| --- |
| package org.example;  /\*  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.  \*/  import java.util.Arrays;  import java.util.Scanner;  import static java.lang.Math.\*;  public class Main {  public static void main(String[] args) throws Exception {  System.out.println("=== Vector class work example ===\n");  System.out.println("Enter size of vectors:");  Scanner sc = new Scanner(System.in);  int n = sc.nextInt();  System.out.println("Enter number of vectors:");  int m = sc.nextInt();  Vector[] vectors\_array = new Vector[m];  for(int i = 0; i < m; i++) {  vectors\_array[i] = new Vector(new double[n]);  for(int j = 0; j < n; j++) {  vectors\_array[i].setByIdx(j, (random() - 0.5) \* 100);  }  }  for(int i = 0; i < m; i++) {  for(int j = 0; j < m; j++) {  if (i != j) {  showVectorPair(vectors\_array[i], vectors\_array[j], i, j);  }  }  }  }  static void showVectorPair(Vector v1, Vector v2, int v1\_idx, int v2\_idx) throws Exception {  System.out.printf("=== Vectors %d-th and %d-th ===\n\n", v1\_idx, v2\_idx);  System.out.printf("v%d: %s (len:%7.2f) %nv%d: %s (len:%7.2f)%n%n", v1\_idx, v1, v1.spaceLength(), v2\_idx, v2, v2.spaceLength());  System.out.printf("v%d + v%d: %s%n%n", v1\_idx, v2\_idx, v1.add(v2));  System.out.printf("v%d - v%d: %s%n%n", v1\_idx, v2\_idx, v1.sub(v2));  System.out.printf("v%d \* v%d (scalar): %7.2f%n%n", v1\_idx, v2\_idx, v1.scalar\_mul(v2));  System.out.printf("angle(v%d, v%d)[degrees]: %7.2f%n%n", v1\_idx, v2\_idx, v1.angle(v2));  System.out.printf("v%d++: %s %nv%d--: %s%n%n", v1\_idx, v1.inc(), v1\_idx, v2.dec());  }  }  class Vector {  public double[] data;  public Vector(double[] p\_data) {  this.data = p\_data;  }  public String toString() {  StringBuilder s = new StringBuilder("[ ");  for (double elem : this.data) {  s.append(String.format("%7.2f ", elem));  }  s.append("]");  return String.valueOf(s);  }  public Vector add(Vector v) throws Exception {  Vector result = new Vector(new double[this.data.length]);  if (this.data.length != v.data.length) {  throw new Exception("Sizes don't match!");  }  for(int i = 0; i < this.data.length; i++) {  result.data[i] = this.data[i] + v.data[i];  }  return result;  }  public Vector sub(Vector v) throws Exception {  Vector result = new Vector(new double[this.data.length]);  if (this.data.length != v.data.length) {  throw new Exception("Sizes don't match!");  }  for(int i = 0; i < this.data.length; i++) {  result.data[i] = this.data[i] - v.data[i];  }  return result;  }  public double scalar\_mul(Vector v) throws Exception {  double result = 0;  if (this.data.length != v.data.length) {  throw new Exception("Sizes don't match!");  }  for(int i = 0; i < this.data.length; i++) {  result += this.data[i] \* v.data[i];  }  return result;  }  public Vector inc() {  Vector result = new Vector(new double[this.data.length]);  for (int i = 0; i < this.data.length; i++) {  result.data[i] = this.data[i] + 1;  }  return result;  }  public Vector dec() {  Vector result = new Vector(new double[this.data.length]);  for(int i = 0; i < this.data.length; i++) {  result.data[i] = this.data[i] - 1;  }  return result;  }  public double getByIdx(int i) {  return this.data[i];  }  public void setByIdx(int i, double val) {  this.data[i] = val;  }  public double spaceLength() {  double result = 0.0F;  for (double elem : this.data) {  result += pow(elem, 2);  }  return sqrt(result);  }  public double angle(Vector v) throws Exception {  return Math.toDegrees(  acos(  this.scalar\_mul(v) / (this.spaceLength() \* v.spaceLength())  )  );  }  } |

Результат исполнения программы приведен в листинге 2.

Листинг 2 — результат исполнения программы

|  |
| --- |
| === Vector class work example ===  Enter size of vectors:  2  Enter number of vectors:  3  === Vectors 0-th and 1-th ===  v0: [ -16,94 -11,30 ] (len: 20,36)  v1: [ -31,00 -48,47 ] (len: 57,54)  v0 + v1: [ -47,94 -59,77 ]  v0 - v1: [ 14,07 37,17 ]  v0 \* v1 (scalar): 1072,75  angle(v0, v1)[degrees]: 23,68  v0++: [ -15,94 -10,30 ]  v0--: [ -32,00 -49,47 ]  === Vectors 0-th and 2-th ===  v0: [ -16,94 -11,30 ] (len: 20,36)  v2: [ -18,30 -37,24 ] (len: 41,50)  v0 + v2: [ -35,24 -48,54 ]  v0 - v2: [ 1,37 25,94 ]  v0 \* v2 (scalar): 730,83  angle(v0, v2)[degrees]: 30,11  v0++: [ -15,94 -10,30 ]  v0--: [ -19,30 -38,24 ]  === Vectors 1-th and 0-th ===  v1: [ -31,00 -48,47 ] (len: 57,54)  v0: [ -16,94 -11,30 ] (len: 20,36)  v1 + v0: [ -47,94 -59,77 ]  v1 - v0: [ -14,07 -37,17 ]  v1 \* v0 (scalar): 1072,75  angle(v1, v0)[degrees]: 23,68  v1++: [ -30,00 -47,47 ]  v1--: [ -17,94 -12,30 ]  === Vectors 1-th and 2-th ===  v1: [ -31,00 -48,47 ] (len: 57,54)  v2: [ -18,30 -37,24 ] (len: 41,50)  v1 + v2: [ -49,31 -85,71 ]  v1 - v2: [ -12,70 -11,23 ]  v1 \* v2 (scalar): 2372,70  angle(v1, v2)[degrees]: 6,43  v1++: [ -30,00 -47,47 ]  v1--: [ -19,30 -38,24 ]  === Vectors 2-th and 0-th ===  v2: [ -18,30 -37,24 ] (len: 41,50)  v0: [ -16,94 -11,30 ] (len: 20,36)  v2 + v0: [ -35,24 -48,54 ]  v2 - v0: [ -1,37 -25,94 ]  v2 \* v0 (scalar): 730,83  angle(v2, v0)[degrees]: 30,11  v2++: [ -17,30 -36,24 ]  v2--: [ -17,94 -12,30 ]  === Vectors 2-th and 1-th ===  v2: [ -18,30 -37,24 ] (len: 41,50)  v1: [ -31,00 -48,47 ] (len: 57,54)  v2 + v1: [ -49,31 -85,71 ]  v2 - v1: [ 12,70 11,23 ]  v2 \* v1 (scalar): 2372,70  angle(v2, v1)[degrees]: 6,43  v2++: [ -17,30 -36,24 ]  v2--: [ -32,00 -49,47 ] |

**Задание 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.

Код решения приведен в листинге 3.

Листинг 3 — реализация решения

|  |
| --- |
| package org.example;  /\*  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.  \*/  import java.util.ArrayList;  import java.util.Arrays;  import static java.lang.Math.\*;  public class Main {  public static void main(String[] args) throws Exception {  System.out.println("=== Vector class work example ===\n");  Vector[] vectors\_array = new Vector[]{  new Vector(new double[]{1, 0, 0}),  new Vector(new Vector(new double[]{4, 0, 0})),  new Vector((ArrayList<Double>) new ArrayList<>(Arrays.asList(0.0, 1.0, 1.0)))  };  for(int i = 0; i < vectors\_array.length; i++) {  System.out.printf("Vector[%d]: %s%n%n", i, vectors\_array[i]);  System.out.printf("Vector[%d] \* %d: %s%n%n", i, i, vectors\_array[i].const\_mul(i));  }  for(int i = 0; i < vectors\_array.length; i++) {  for(int j = 0; j < vectors\_array.length; j++) {  if (i != j) {  showVectorPair(vectors\_array[i], vectors\_array[j], i, j);  }  }  }  }  static void showVectorPair(Vector v1, Vector v2, int v1\_idx, int v2\_idx) throws Exception {  System.out.printf("=== Vectors %d-th and %d-th ===\n\n", v1\_idx, v2\_idx);  System.out.printf("v%d: %s (len:%7.2f) %nv%d: %s (len:%7.2f)%n%n", v1\_idx, v1, v1.module(), v2\_idx, v2, v2.module());  System.out.printf("v%d + v%d: %s%n%n", v1\_idx, v2\_idx, v1.add(v2));  System.out.printf("v%d - v%d: %s%n%n", v1\_idx, v2\_idx, v1.sub(v2));  System.out.printf("v%d \* v%d (scalar): %7.2f%n%n", v1\_idx, v2\_idx, v1.scalar\_mul(v2));  System.out.printf("v%d and v%d are orthogonal: %b%n%n", v1\_idx, v2\_idx, v1.isOrtogonal(v2));  System.out.printf("v%d and v%d are collinear: %b%n%n", v1\_idx, v2\_idx, v1.isCollinear(v2));  }  }  class Vector {  public double[] data;  public Vector(double[] p\_data) {  this.data = p\_data;  }  public Vector(ArrayList<Double> p\_data) {  this.data = new double[p\_data.size()];  for (int i = 0; i < p\_data.size(); i++) {  this.data[i] = p\_data.get(i);  }  }  public Vector(Vector p\_data) {  this.data = p\_data.data.clone();  }  public Vector(int k) {  this.data = new double[k];  for (int i = 0; i < k; i++) {  this.data[i] = 0.0;  }  }  public String toString() {  StringBuilder s = new StringBuilder("[ ");  for (double elem : this.data) {  s.append(String.format("%7.2f ", elem));  }  s.append("]");  return String.valueOf(s);  }  public Vector add(Vector v) throws Exception {  Vector result = new Vector(new double[this.data.length]);  if (this.data.length != v.data.length) {  throw new Exception("Sizes don't match!");  }  for(int i = 0; i < this.data.length; i++) {  result.data[i] = this.data[i] + v.data[i];  }  return result;  }  public Vector sub(Vector v) throws Exception {  Vector result = new Vector(new double[this.data.length]);  if (this.data.length != v.data.length) {  throw new Exception("Sizes don't match!");  }  for(int i = 0; i < this.data.length; i++) {  result.data[i] = this.data[i] - v.data[i];  }  return result;  }  public double scalar\_mul(Vector v) throws Exception {  double result = 0;  if (this.data.length != v.data.length) {  throw new Exception("Sizes don't match!");  }  for(int i = 0; i < this.data.length; i++) {  result += this.data[i] \* v.data[i];  }  return result;  }  public Vector const\_mul(double k) throws Exception {  Vector result = new Vector(new double[this.data.length]);  for(int i = 0; i < this.data.length; i++) {  result.data[i] = this.data[i] \* k;  }  return result;  }  public double module() {  double result = 0.0F;  for (double elem : this.data) {  result += pow(elem, 2);  }  return sqrt(result);  }  public boolean isCollinear(Vector v) throws Exception {  double epsilon = 0.001;  boolean result = true;  double k = 0;  boolean k\_init = false;  if (this.data.length != v.data.length) {  throw new Exception("Sizes don't match!");  }  for(int i = 0; i < this.data.length; i++) {  if(abs(this.data[i]) > epsilon && abs(v.data[i]) > epsilon && !k\_init) {  k = this.data[i] / v.data[i];  k\_init = true;  } else if (abs(this.data[i]) > epsilon && abs(v.data[i]) > epsilon && k\_init) {  if(abs(this.data[i] / v.data[i] - k) > epsilon) {  result = false;  }  }  if(abs(this.data[i]) > epsilon ^ abs(v.data[i]) > epsilon) {  result = false;  }  }  return result;  }  public boolean isOrtogonal(Vector v) throws Exception {  double epsilon = 0.001;  return abs(this.scalar\_mul(v)) < epsilon;  }  } |

Результат исполнения программы приведен в листинге 4.

Листинг 4 — результат исполнения программы

|  |
| --- |
| === Vector class work example ===  Vector[0]: [ 1,00 0,00 0,00 ]  Vector[0] \* 0: [ 0,00 0,00 0,00 ]  Vector[1]: [ 4,00 0,00 0,00 ]  Vector[1] \* 1: [ 4,00 0,00 0,00 ]  Vector[2]: [ 0,00 1,00 1,00 ]  Vector[2] \* 2: [ 0,00 2,00 2,00 ]  === Vectors 0-th and 1-th ===  v0: [ 1,00 0,00 0,00 ] (len: 1,00)  v1: [ 4,00 0,00 0,00 ] (len: 4,00)  v0 + v1: [ 5,00 0,00 0,00 ]  v0 - v1: [ -3,00 0,00 0,00 ]  v0 \* v1 (scalar): 4,00  v0 and v1 are orthogonal: false  v0 and v1 are collinear: true  === Vectors 0-th and 2-th ===  v0: [ 1,00 0,00 0,00 ] (len: 1,00)  v2: [ 0,00 1,00 1,00 ] (len: 1,41)  v0 + v2: [ 1,00 1,00 1,00 ]  v0 - v2: [ 1,00 -1,00 -1,00 ]  v0 \* v2 (scalar): 0,00  v0 and v2 are orthogonal: true  v0 and v2 are collinear: false  === Vectors 1-th and 0-th ===  v1: [ 4,00 0,00 0,00 ] (len: 4,00)  v0: [ 1,00 0,00 0,00 ] (len: 1,00)  v1 + v0: [ 5,00 0,00 0,00 ]  v1 - v0: [ 3,00 0,00 0,00 ]  v1 \* v0 (scalar): 4,00  v1 and v0 are orthogonal: false  v1 and v0 are collinear: true  === Vectors 1-th and 2-th ===  v1: [ 4,00 0,00 0,00 ] (len: 4,00)  v2: [ 0,00 1,00 1,00 ] (len: 1,41)  v1 + v2: [ 4,00 1,00 1,00 ]  v1 - v2: [ 4,00 -1,00 -1,00 ]  v1 \* v2 (scalar): 0,00  v1 and v2 are orthogonal: true  v1 and v2 are collinear: false  === Vectors 2-th and 0-th ===  v2: [ 0,00 1,00 1,00 ] (len: 1,41)  v0: [ 1,00 0,00 0,00 ] (len: 1,00)  v2 + v0: [ 1,00 1,00 1,00 ]  v2 - v0: [ -1,00 1,00 1,00 ]  v2 \* v0 (scalar): 0,00  v2 and v0 are orthogonal: true  v2 and v0 are collinear: false  === Vectors 2-th and 1-th ===  v2: [ 0,00 1,00 1,00 ] (len: 1,41)  v1: [ 4,00 0,00 0,00 ] (len: 4,00)  v2 + v1: [ 4,00 1,00 1,00 ]  v2 - v1: [ -4,00 1,00 1,00 ]  v2 \* v1 (scalar): 0,00  v2 and v1 are orthogonal: true  v2 and v1 are collinear: false |

**Задание 3:** 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.

Код решения приведен в листинге 5.

Листинг 5 — реализация решения

|  |
| --- |
| package org.example;  /\*  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.  \*/  import java.text.ParseException;  import java.text.SimpleDateFormat;  import java.util.\*;  public class Main {  public static void main(String[] args) throws ParseException {  System.out.println("=== StudentCollection class example work ===");  SimpleDateFormat dateFormat = new SimpleDateFormat("yyyy-MM-dd");  StudentCollection st\_collection = new StudentCollection(new Student[]{  new Student(1,"Sergey", "Astakhov", "Victorovich", dateFormat.parse("2001-01-01"), "Balashiha" , "+7-900-000-00-00", "ICS", 5, 2),  new Student(2,"Mark", "Khabarov", "X.", dateFormat.parse("2001-05-01"), "Fryazino" , "+7-900-000-00-01", "ICS", 5, 2),  new Student(3,"Nina", "Gendina", "X.", dateFormat.parse("2000-07-01"), "Moscow" , "+7-900-000-00-02", "ICS", 5, 2),  new Student(4,"Ivan", "Ivanov", "X.", dateFormat.parse("1999-05-01"), "Moscow" , "+7-900-000-00-03", "SM", 3, 1),  new Student(5,"Boris", "Ivanov", "X.", dateFormat.parse("2001-05-02"), "Moscow" , "+7-900-000-00-04", "SM", 3, 1)  });  System.out.println("=== Filter by faculty (SM) ===");  System.out.println(st\_collection.filterByFaculty("SM"));  System.out.println("=== Filter by birthyear (1999) ===");  System.out.println(st\_collection.filterByBirthYear(1999));  System.out.println("=== Filter by group (ICS, 5, 2) ===");  System.out.println(st\_collection.filterByGroup("ICS", 5, 2));  System.out.println("=== All by faculty and course ===");  System.out.println(st\_collection.toStringByFacultyAndCourse());  }  }  class StudentCollection {  private Student[] data;  public StudentCollection(Student[] data) {  this.data = data;  }  public String toString() {  String result = "";  for(Student elem: this.data) {  result = String.format("%s%n%n%s", result, elem.toString());  }  return result;  }  public StudentCollection filterByFaculty(String p\_faculty) {  Student[] result;  Student[] filtered = Arrays.stream(this.data)  .filter(x -> Objects.equals(x.getFaculty(), p\_faculty))  .toArray(Student[]::new);  return new StudentCollection(filtered);  }  public Student[] getData() {  return this.data;  }  public Map<String, Map<Integer, StudentCollection>> getByFacultyAndCourse() {  Map<String, Map<Integer, StudentCollection>> result = new HashMap<>();  for (Student elem : this.data) {  Map<Integer, StudentCollection> default\_inner\_map = new HashMap<>();  Student[] tmp\_data = result  .getOrDefault(  elem.getFaculty(),  default\_inner\_map  )  .getOrDefault(  elem.getCourse(),  new StudentCollection(new Student[]{})  )  .getData();  Student[] new\_data = Arrays.copyOf(tmp\_data, tmp\_data.length + 1);  new\_data[tmp\_data.length] = elem;  Map<Integer, StudentCollection> tmp\_inner\_map;  tmp\_inner\_map = result.getOrDefault(  elem.getFaculty(),  default\_inner\_map  );  tmp\_inner\_map.put(elem.getCourse(), new StudentCollection(new\_data));  result.put(elem.getFaculty(), tmp\_inner\_map);  }  return result;  }  public String toStringByFacultyAndCourse() {  String result = "";  Map<String, Map<Integer, StudentCollection>> tmp = this.getByFacultyAndCourse();  for(String faculty\_key : tmp.keySet()){  for(int course\_key : tmp.get(faculty\_key).keySet()){  result = String.format("%s%n%n=== faculty: %s course: %d ===%n%n%s",  result, faculty\_key, course\_key, tmp.get(faculty\_key).get(course\_key).toString());  }  }  return result;  }  public StudentCollection filterByBirthYear(int year) throws ParseException {  Student[] result;  SimpleDateFormat dateFormat = new SimpleDateFormat("yyyy");  Date date = dateFormat.parse(String.format("%d", year + 1)); // Because we need end of year  Student[] filtered = Arrays.stream(this.data)  .filter(x -> x.getBirthdate().after(date))  .toArray(Student[]::new);  return new StudentCollection(filtered);  }  public StudentCollection filterByGroup(String faculty, int course, int group) throws ParseException {  Student[] result;  Student[] filtered = Arrays.stream(this.data)  .filter(x -> x.getFaculty().equals(faculty)  && x.getCourse() == course  && x.getGroup() == group  )  .toArray(Student[]::new);  return new StudentCollection(filtered);  }  }  class Student {  private int id;  private String name;  private String surname;  private String thirdname;  private Date birthdate;  private String address;  private String phone;  private String faculty;  private int course;  private int group;  public Student(int id, String name, String surname, String thirdname, Date birthdate, String address, String phone, String faculty, int course, int group) {  this.id = id;  this.name = name;  this.surname = surname;  this.thirdname = thirdname;  this.birthdate = birthdate;  this.address = address;  this.phone = phone;  this.faculty = faculty;  this.course = course;  this.group = group;  }  public String toString() {  SimpleDateFormat sdf = new SimpleDateFormat("yyyy-MM-dd");  return String.format("Student(id=%d): %s %s %s (%s course: %d groud: %d)\n", this.id, this.name, this.surname, this.thirdname, this.faculty, this.course, this.group) +  String.format("Birthdate: %s Phone: %s\n", sdf.format(this.birthdate), this.phone) +  String.format("Address: %s\n", this.address);  }  public int getId() {  return id;  }  public String getName() {  return name;  }  public String getSurname() {  return surname;  }  public String getThirdname() {  return thirdname;  }  public Date getBirthdate() {  return birthdate;  }  public String getAddress() {  return address;  }  public String getPhone() {  return phone;  }  public String getFaculty() {  return faculty;  }  public int getCourse() {  return course;  }  public int getGroup() {  return group;  }  public void setId(int id) {  this.id = id;  }  public void setName(String name) {  this.name = name;  }  public void setSurname(String surname) {  this.surname = surname;  }  public void setThirdname(String thirdname) {  this.thirdname = thirdname;  }  public void setBirthdate(Date birthdate) {  this.birthdate = birthdate;  }  public void setAddress(String address) {  this.address = address;  }  public void setPhone(String phone) {  this.phone = phone;  }  public void setFaculty(String faculty) {  this.faculty = faculty;  }  public void setCourse(int course) {  this.course = course;  }  public void setGroup(int group) {  this.group = group;  }  } |

Результат исполнения программы приведен в листинге 6.

Листинг 6 — результат исполнения программы

|  |
| --- |
| === StudentCollection class example work ===  === Filter by faculty (SM) ===  Student(id=4): Ivan Ivanov X. (SM course: 3 groud: 1)  Birthdate: 1999-05-01 Phone: +7-900-000-00-03  Address: Moscow  Student(id=5): Boris Ivanov X. (SM course: 3 groud: 1)  Birthdate: 2001-05-02 Phone: +7-900-000-00-04  Address: Moscow  === Filter by birthyear (1999) ===  Student(id=1): Sergey Astakhov Victorovich (ICS course: 5 groud: 2)  Birthdate: 2001-01-01 Phone: +7-900-000-00-00  Address: Balashiha  Student(id=2): Mark Khabarov X. (ICS course: 5 groud: 2)  Birthdate: 2001-05-01 Phone: +7-900-000-00-01  Address: Fryazino  Student(id=3): Nina Gendina X. (ICS course: 5 groud: 2)  Birthdate: 2000-07-01 Phone: +7-900-000-00-02  Address: Moscow  Student(id=5): Boris Ivanov X. (SM course: 3 groud: 1)  Birthdate: 2001-05-02 Phone: +7-900-000-00-04  Address: Moscow  === Filter by group (ICS, 5, 2) ===  Student(id=1): Sergey Astakhov Victorovich (ICS course: 5 groud: 2)  Birthdate: 2001-01-01 Phone: +7-900-000-00-00  Address: Balashiha  Student(id=2): Mark Khabarov X. (ICS course: 5 groud: 2)  Birthdate: 2001-05-01 Phone: +7-900-000-00-01  Address: Fryazino  Student(id=3): Nina Gendina X. (ICS course: 5 groud: 2)  Birthdate: 2000-07-01 Phone: +7-900-000-00-02  Address: Moscow  === All by faculty and course ===  === faculty: ICS course: 5 ===  Student(id=1): Sergey Astakhov Victorovich (ICS course: 5 groud: 2)  Birthdate: 2001-01-01 Phone: +7-900-000-00-00  Address: Balashiha  Student(id=2): Mark Khabarov X. (ICS course: 5 groud: 2)  Birthdate: 2001-05-01 Phone: +7-900-000-00-01  Address: Fryazino  Student(id=3): Nina Gendina X. (ICS course: 5 groud: 2)  Birthdate: 2000-07-01 Phone: +7-900-000-00-02  Address: Moscow  === faculty: SM course: 3 ===  Student(id=4): Ivan Ivanov X. (SM course: 3 groud: 1)  Birthdate: 1999-05-01 Phone: +7-900-000-00-03  Address: Moscow  Student(id=5): Boris Ivanov X. (SM course: 3 groud: 1)  Birthdate: 2001-05-02 Phone: +7-900-000-00-04  Address: Moscow |

**Задание 4:** 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.

Код решения приведен в листинге 7.

Листинг 7 — реализация решения

|  |
| --- |
| package org.example;  /\*  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.  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.  \*/  import java.util.Arrays;  import java.util.Collections;  import java.util.Comparator;  import java.util.List;  public class Main {  public static void main(String[] args) {  System.out.println("=== Customer class work example ===\n\n");  CustomerCollection cs\_collection = new CustomerCollection(  new Customer[]{  new Customer(1, "Astakhov", "Sergey", "Victorovich", "Balashiha", "1111-2222-0000", "1111-2222-8000"),  new Customer(2, "Zakharov", "Pavel", "X.", "Moscow", "1111-2222-0001", "1111-2222-8001"),  new Customer(3, "Loktev", "Michael", "X.", "Kemerovo", "1111-2222-0002", "1111-2222-8002"),  new Customer(4, "Petrov", "Ivan", "X.", "Balashiha", "1111-2222-0003", "1111-2222-8003"),  new Customer(5, "Anisin", "Petr", "X.", "Ivanovo", "1111-2222-0004", "1111-2222-8004")  }  );  System.out.println("=== Customers sorted by alphabet ===\n\n");  System.out.println(cs\_collection.orderByAlphabet());  System.out.println("=== Customers filtered by cardnums(1111-2222-0001, 1111-2222-0003) ===\n\n");  System.out.println(cs\_collection.filterByCardnums("1111-2222-0001", "1111-2222-0003"));  }  }  class CustomerCollection {  Customer[] data;  public CustomerCollection(Customer[] p\_data) {  this.data = p\_data;  }  public String toString() {  String result = "";  for(Customer elem: data){  result = String.format("%s %n%n%s", result, elem);  }  return result;  }  public CustomerCollection orderByAlphabet() {  List<Customer> data\_list = Arrays.asList(this.data);  data\_list.sort(Comparator.comparing(x -> x.getFio()));  return new CustomerCollection(data\_list.toArray(new Customer[0]));  }  public CustomerCollection filterByCardnums(String num1, String num2) {  Customer[] filtered = Arrays.stream(this.data)  .filter(x ->  Long.parseLong(x.getCardNumber().replace("-", "")) >=  Long.parseLong(num1.replace("-", ""))  && Long.parseLong(x.getCardNumber().replace("-", "")) <=  Long.parseLong(num2.replace("-", ""))  )  .toArray(Customer[]::new);  return new CustomerCollection(filtered);  }  }  class Customer {  int id;  String surname;  String name;  String thirdname;  String address;  String card\_number;  String account\_number;  public Customer(int id, String surname, String name, String thirdname, String address, String card\_number, String account\_number) {  this.id = id;  this.surname = surname;  this.name = name;  this.thirdname = thirdname;  this.address = address;  this.card\_number = card\_number;  this.account\_number = account\_number;  }  public String toString() {  return String.format("Client(id=%d) %s %s %s%n", this.id, this.surname, this.name, this.thirdname) +  String.format("Address: %s%n", this.address) +  String.format("Card: %s Account:%s%n", this.card\_number, this.account\_number);  }  public String getFio() {  return String.format("%s %s %s", this.surname, this.name, this.thirdname);  }  public int getId() {  return id;  }  public String getSurname() {  return surname;  }  public String getName() {  return name;  }  public String getThirdname() {  return thirdname;  }  public String getAddress() {  return address;  }  public String getCardNumber() {  return card\_number;  }  public String getAccountNumber() {  return account\_number;  }  public void setId(int id) {  this.id = id;  }  public void setSurname(String surname) {  this.surname = surname;  }  public void setName(String name) {  this.name = name;  }  public void setThirdname(String thirdname) {  this.thirdname = thirdname;  }  public void setAddress(String address) {  this.address = address;  }  public void setCard\_number(String card\_number) {  this.card\_number = card\_number;  }  public void setAccount\_number(String account\_number) {  this.account\_number = account\_number;  }  } |

Результат исполнения программы приведен в листинге 8.

Листинг 8 — результат исполнения программы

|  |
| --- |
| === Customer class work example ===  === Customers sorted by alphabet ===    Client(id=5) Anisin Petr X.  Address: Ivanovo  Card: 1111-2222-0004 Account:1111-2222-8004    Client(id=1) Astakhov Sergey Victorovich  Address: Balashiha  Card: 1111-2222-0000 Account:1111-2222-8000    Client(id=3) Loktev Michael X.  Address: Kemerovo  Card: 1111-2222-0002 Account:1111-2222-8002    Client(id=4) Petrov Ivan X.  Address: Balashiha  Card: 1111-2222-0003 Account:1111-2222-8003    Client(id=2) Zakharov Pavel X.  Address: Moscow  Card: 1111-2222-0001 Account:1111-2222-8001  === Customers filtered by cardnums(1111-2222-0001, 1111-2222-0003) ===    Client(id=3) Loktev Michael X.  Address: Kemerovo  Card: 1111-2222-0002 Account:1111-2222-8002    Client(id=4) Petrov Ivan X.  Address: Balashiha  Card: 1111-2222-0003 Account:1111-2222-8003    Client(id=2) Zakharov Pavel X.  Address: Moscow  Card: 1111-2222-0001 Account:1111-2222-8001 |

**Задание 5:** Создать объект класса Текстовый файл, используя класс Файл. Методы: создать, переименовать, вывести на консоль содержимое, дополнить, удалить.

Код решения приведен в листинге 9.

Листинг 9 — реализация решения

|  |
| --- |
| package org.example;  /\*  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.  About equals() and hashCode(): https://stackoverflow.com/questions/24446763/difference-between-equals-and-hashcode  If you have two objects that are equal then they must have the same hashCode, however the reverse is not true  \*/  import java.io.File;  import java.io.IOException;  import java.lang.annotation.Documented;  import java.nio.file.Files;  import java.nio.file.Path;  import java.nio.file.StandardOpenOption;  import java.util.Objects;  import java.util.Scanner;  public class Main {  public static void main(String[] args) throws IOException {  System.out.println("=== Text file work example ===\n\n");  TextFile a = new TextFile("a.txt");  a.createOnDisk();  a.append("aaaaaa");  TextFile a\_copy = new TextFile("a\_copy.txt");  a\_copy.createOnDisk();  a\_copy.append("aaaaaa");  TextFile a1 = new TextFile("a1.txt");  a1.createOnDisk();  a1.append("a1a1a1a1a1a");  TextFile b = new TextFile("b.txt");  b.createOnDisk();  b.append("bbbbbbbbbbbbbbbbb");  b.append("\nb1b1b1");  System.out.println("=== Files ===\n\n");  System.out.println(a);  System.out.println(a\_copy);  System.out.println(a1);  System.out.println(b);  System.out.println("=== Rename a1 -> a2 ===\n");  System.out.println(a1.rename("a2.txt"));  System.out.println(a1);  System.out.println("=== Equals ===\n");  System.out.printf("a.equals(a): %b%n", a.equals(a));  System.out.printf("a.equals(a\_copy): %b%n", a.equals(a\_copy));  System.out.printf("a.equals(b): %b%n", a.equals(b));  System.out.println("\n\nEnter a string to delete files and finish: \n");  Scanner scanner = new Scanner(System.in);  scanner.nextLine();  System.out.println(a.delete());  System.out.println(a\_copy.delete());  System.out.println(a1.delete());  System.out.println(b.delete());  }  }  class TextFile extends File {  public TextFile(String filename) {  super(filename);  }  public boolean createOnDisk() throws IOException {  return createNewFile();  }  public String getName() {  return super.getName();  }  public boolean rename(String new\_name) {  return renameTo(new File(new\_name));  }  public String getText() throws IOException {  return Files.readString(toPath());  }  public void printText() throws IOException {  System.out.println(getText());  }  public void append(String s) throws IOException {  Files.writeString(toPath(), s, StandardOpenOption.APPEND);  }  public boolean delete() throws SecurityException {  return super.delete();  }  @Override  public String toString() {  String content = "";  try {  content = getText();  } catch (IOException e) {  throw new RuntimeException(e);  }  return String.format(  "Filepath: %s (rights r-w-x: %b-%b-%b) %nHash: %h %nText: %s%n",  getAbsolutePath(), canRead(), canWrite(), canExecute(), hashCode(), content);  }  // based only on content and rights  @Override  public int hashCode() {  String content = "";  try {  content = getText();  } catch (IOException e) {  throw new RuntimeException(e);  }  return String.format(  "r-w-x: %b-%b-%b content: %s",  canRead(), canWrite(), canExecute(), content  ).hashCode();  }  // compare also path  @Override  public boolean equals(Object obj) {  return Objects.equals(toString(), obj.toString())  && getClass().equals(obj.getClass());  }  } |

Результат исполнения программы приведен в листинге 10.

Листинг 10 — результат исполнения программы

|  |
| --- |
| === Text file work example ===  === Files ===  Filepath: C:\Users\sergey.astakhov\Desktop\BMSTU-JAVA-BIGDATA\task\_3\_3\_1\a.txt (rights r-w-x: true-true-true)  Hash: fb58e8ba  Text: aaaaaa  Filepath: C:\Users\sergey.astakhov\Desktop\BMSTU-JAVA-BIGDATA\task\_3\_3\_1\a\_copy.txt (rights r-w-x: true-true-true)  Hash: fb58e8ba  Text: aaaaaa  Filepath: C:\Users\sergey.astakhov\Desktop\BMSTU-JAVA-BIGDATA\task\_3\_3\_1\a1.txt (rights r-w-x: true-true-true)  Hash: e8e5de97  Text: a1a1a1a1a1a  Filepath: C:\Users\sergey.astakhov\Desktop\BMSTU-JAVA-BIGDATA\task\_3\_3\_1\b.txt (rights r-w-x: true-true-true)  Hash: 974696af  Text: bbbbbbbbbbbbbbbbb  b1b1b1  === Rename a1 -> a2 ===  false  Filepath: C:\Users\sergey.astakhov\Desktop\BMSTU-JAVA-BIGDATA\task\_3\_3\_1\a1.txt (rights r-w-x: true-true-true)  Hash: e8e5de97  Text: a1a1a1a1a1a  === Equals ===  a.equals(a): true  a.equals(a\_copy): false  a.equals(b): false  Enter a string to delete files and finish:  exit  true  true  true  true |

**Задание 6:** Создать объект класса Одномерный массив, используя класс Массив. Методы: создать, вывести на консоль, выполнить операции (сложить, вычесть, перемножить).

Код решения приведен в листинге 11.

Листинг 11 — реализация решения

|  |
| --- |
| package org.example;  /\*  Создать приложение, удовлетворяющее требованиям, приведенным в задании.  Аргументировать принадлежность классу каждого создаваемого метода и корректно переопределить  для каждого класса методы equals(), hashCode(), toString().  2. Создать объект класса Одномерный массив, используя класс Массив.  Методы: создать, вывести на консоль, выполнить операции (сложить, вычесть, перемножить).  \*/  import java.util.ArrayList;  import java.util.Objects;  public class Main {  public static void main(String[] args) throws Exception {  System.out.println("=== MyArray class work example ===\n");  MyArray a = new MyArray();  a.add(1);  a.add(2);  a.add(3);  MyArray a1 = new MyArray();  a1.add(1);  a1.add(2);  a1.add(3);  MyArray a2 = a;  MyArray b = new MyArray();  b.add(6);  b.add(5);  b.add(4);  MyArray c = new MyArray();  c.add(0);  System.out.println("=== Arrays ===\n");  System.out.printf("a: %s %n%n", a);  System.out.printf("a1: %s %n%n", a1);  System.out.printf("a2: %s %n%n", a2);  System.out.printf("b: %s %n%n", b);  System.out.printf("c: %s %n%n", c);  System.out.println("=== Math ===\n");  System.out.printf("a + b: %s %n%n", a.sum(b));  System.out.printf("a - b: %s %n%n", a.sub(b));  System.out.printf("a \* b: %s %n%n", a.mul(b));  System.out.println("=== Equals ===\n");  System.out.printf("a.equals(a): %b %n", a.equals(a));  System.out.printf("a.equals(a1): %b %n", a.equals(a1));  System.out.printf("a.equals(a2): %b %n", a.equals(a2));  System.out.printf("a.equals(b): %b %n%n", a.equals(b));  System.out.println("=== Exception catch ===\n");  try {  System.out.printf("a + c: %s %n%n", a.sum(c));  }  catch (Exception e) {  System.out.printf("Exception (a + c): %s", e);  }  }  }  class MyArray extends ArrayList<Integer> {  public MyArray() {  super();  }  public boolean add(Integer x) {  return super.add(x);  }  public void print() {  System.out.println(this);  }  @Override  public String toString() {  String result = "MyArray[]{";  for(Integer elem: this){  result = String.format("%s %5d", result, elem);  }  result = String.format("%s } (hash: %h)", result, this.hashCode());  return result;  }  public MyArray sum(MyArray v) throws Exception {  MyArray result = new MyArray();  if(this.size() != v.size()) {  throw new Exception("Sizes don't match!");  }  for (int i = 0; i < this.size(); i++) {  result.add(this.get(i) + v.get(i));  }  return result;  }  public MyArray sub(MyArray v) throws Exception {  MyArray result = new MyArray();  if(this.size() != v.size()) {  throw new Exception("Sizes don't match!");  }  for (int i = 0; i < this.size(); i++) {  result.add(this.get(i) - v.get(i));  }  return result;  }  public MyArray mul(MyArray v) throws Exception {  MyArray result = new MyArray();  if(this.size() != v.size()) {  throw new Exception("Sizes don't match!");  }  for (int i = 0; i < this.size(); i++) {  result.add(this.get(i) \* v.get(i));  }  return result;  }  @Override  public int hashCode() {  int hash = 0;  for(Integer elem: this){  hash = Objects.hash(hash, elem);  }  return hash;  }  @Override  public boolean equals(Object o) {  return this.toString().equals(o.toString())  && this.getClass().equals(o.getClass());  }  } |

Результат исполнения программы приведен в листинге 12.

Листинг 12 — результат исполнения программы

|  |
| --- |
| === MyArray class work example ===  === Arrays ===  a: MyArray[]{ 1 2 3 } (hash: e93a3)  a1: MyArray[]{ 1 2 3 } (hash: e93a3)  a2: MyArray[]{ 1 2 3 } (hash: e93a3)  b: MyArray[]{ 6 5 4 } (hash: ea6c6)  c: MyArray[]{ 0 } (hash: 3c1)  === Math ===  a + b: MyArray[]{ 7 7 7 } (hash: eaac8)  a - b: MyArray[]{ -5 -3 -1 } (hash: e7c7e)  a \* b: MyArray[]{ 6 10 12 } (hash: ea769)  === Equals ===  a.equals(a): true  a.equals(a1): true  a.equals(a2): true  a.equals(b): false  === Exception catch ===  Exception (a + c): java.lang.Exception: Sizes don't match! |

**Задание 7:** Построить модель программной системы. 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.

Код решения приведен в листинге 13.

Листинг 13 — реализация решения

|  |
| --- |
| package org.example;  /\*  Cистема 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.  \*/  import java.util.\*;  public class Main {  public static void main(String[] args) throws Exception {  System.out.println("=== Facultative class work example ===\n");  Facultative facultative = new Facultative();  Professor popov = new Professor("Popov");  Professor gurenko = new Professor("Gurenko");  Student mark = new Student("Mark");  Student danil = new Student("Danil");  Student vlad = new Student("Vlad");  Student pavel = new Student("Pavel");  Course math = new Course("Math");  Course logic = new Course("Logic");  Course electronics = new Course("Electronics");  facultative.registerCourse(popov, electronics);  facultative.registerCourse(gurenko, math);  facultative.registerCourse(gurenko, logic);  facultative.assignToCourse(electronics, mark);  facultative.assignToCourse(electronics, danil);  facultative.assignToCourse(math, mark);  facultative.assignToCourse(math, danil);  facultative.assignToCourse(math, vlad);  facultative.assignToCourse(math, pavel);  facultative.assignToCourse(logic, vlad);  facultative.assignToCourse(logic, pavel);  facultative.setMark(gurenko, math, mark, 5);  facultative.setMark(popov, electronics, danil, 4);  System.out.println("=== Facultative class state ===\n");  System.out.println(facultative);  }  }  class Facultative {  private HashMap<Professor, CourseSet> professorToCoursesLink;  private HashMap<Course, StudentSet> courseToStudentsLink;  private HashMap<Course, HashMap<Student, Integer>> archive;  public Facultative() {  this.professorToCoursesLink = new HashMap<>();  this.courseToStudentsLink = new HashMap<>();  this.archive = new HashMap<>();  }  public String toString(){  String result = "=== Active ===\n";  for(Professor pf: this.professorToCoursesLink.keySet()) {  result = String.format("%s %n====> %s", result, pf);  CourseSet cs\_set = this.professorToCoursesLink.get(pf);  for (NamedEntity cs: cs\_set) {  result = String.format("%s %n==> %s", result, (Course) cs);  StudentSet ss = this.courseToStudentsLink.get((Course) cs);  result = String.format("%s %s%n", result, ss);  }  }  result = String.format("%s %n%n %s", result, "=== Archive ===");  for(Course cs: this.archive.keySet()) {  result = String.format("%s %n%n==> %s", result, cs);  for (Student ss: this.archive.get(cs).keySet()) {  result = String.format("%s %n%s [%d]", result, ss, this.archive.get(cs).get(ss));  }  }  return result;  }  public void registerCourse(Professor professor, Course course) throws Exception {  CourseSet tmp\_cs = this.professorToCoursesLink.getOrDefault(professor, new CourseSet());  tmp\_cs.add(course);  this.professorToCoursesLink.put(professor, tmp\_cs);  this.courseToStudentsLink.put(course, new StudentSet());  }  public void assignToCourse(Course course, Student student) throws Exception {  if (!this.courseToStudentsLink.containsKey(course)) {  throw new Exception("Course doesn't exists");  }  StudentSet tmp\_ss = this.courseToStudentsLink.getOrDefault(course, new StudentSet());  tmp\_ss.addToSet(student);  this.courseToStudentsLink.put(course, tmp\_ss);  }  public void setMark(Professor professor, Course course, Student student, Integer mark) throws Exception {  if (!this.professorToCoursesLink.getOrDefault(professor, new CourseSet()).contains(course)) {  throw new Exception("No such course assigned to professor!");  }  if (!this.courseToStudentsLink.getOrDefault(course, new StudentSet()).contains(student)) {  throw new Exception("No such student assigned to course!");  }  StudentSet tmp\_ss = this.courseToStudentsLink.getOrDefault(course, new StudentSet());  tmp\_ss.remove(student);  this.courseToStudentsLink.put(course, tmp\_ss);  HashMap<Student, Integer> tmp\_marks = this.archive.getOrDefault(course, new HashMap<>());  tmp\_marks.put(student, mark);  this.archive.put(course, tmp\_marks);  }  }  class StudentSet extends NamedEntitySet {  public StudentSet() {  super();  }  }  class ProfessorSet extends NamedEntitySet {  public ProfessorSet() {  super();  }  }  class CourseSet extends NamedEntitySet {  public CourseSet() {  super();  }  }  class NamedEntitySet extends HashSet<NamedEntity> {  public NamedEntitySet() {  }  @Override  public String toString() {  String result = "";  for(NamedEntity elem: this){  result = String.format("%s %n%s", result, elem);  }  return result;  }  public boolean contains(NamedEntity entity) {  return super.contains(entity);  }  public void addToSet(NamedEntity entity) throws Exception {  if (this.contains(entity)) {  throw new Exception("Object is already in collection");  }  super.add(entity);  }  public void remove(NamedEntity entity) throws Exception {  if (!this.contains(entity)) {  throw new Exception("Object not in collection");  }  super.remove(entity);  }  }  class NamedEntity {  private static Integer globalIdCounter = 0;  protected final Integer id;  protected String name;  public NamedEntity(String name) {  this.id = globalIdCounter;  globalIdCounter++;  this.name = name;  }  public String toString(){  return String.format("id: %d name: %s", this.id, this.name);  }  public Integer getId() {  return id;  }  public String getName() {  return name;  }  public void setName(String name) {  this.name = name;  }  }  class Student extends NamedEntity {  public Student(String name) {  super(name);  }  @Override  public String toString(){  return String.format("<Student> id: %d name: %s", this.id, this.name);  }  }  class Professor extends NamedEntity {  public Professor(String name) {  super(name);  }  @Override  public String toString(){  return String.format("<Professor> id: %d name: %s", this.id, this.name);  }  }  class Course extends NamedEntity {  public Course(String name) {  super(name);  }  @Override  public String toString(){  return String.format("<Course> id: %d name: %s", this.id, this.name);  }  } |

Результат исполнения программы приведен в листинге 14.

Листинг 14 — результат исполнения программы

|  |
| --- |
| === Facultative class work example ===  === Facultative class state ===  === Active ===    ====> <Professor> id: 0 name: Popov  ==> <Course> id: 8 name: Electronics  <Student> id: 2 name: Mark    ====> <Professor> id: 1 name: Gurenko  ==> <Course> id: 7 name: Logic  <Student> id: 4 name: Vlad  <Student> id: 5 name: Pavel    ==> <Course> id: 6 name: Math  <Student> id: 4 name: Vlad  <Student> id: 5 name: Pavel  <Student> id: 3 name: Danil    === Archive ===  ==> <Course> id: 6 name: Math  <Student> id: 2 name: Mark [5]  ==> <Course> id: 8 name: Electronics  <Student> id: 3 name: Danil [4] |

**Задание 8:** Построить модель программной системы. 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.

Код решения приведен в листинге 15.

Листинг 15 — реализация решения

|  |
| --- |
| package org.example;  /\*  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.  \*/  import java.util.HashMap;  import java.util.HashSet;  public class Main {  public static void main(String[] args) throws Exception {  System.out.println("=== Bank class work example ===\n\n");  Person clientSergey = new Person("Sergey");  Person clientMax = new Person("Max");  Person clientDmitry = new Person("Dmitry");  Person adminAlex = new Person("Alex");  Bank bank = new Bank();  bank.registerClient(clientSergey, -5000, 10000);  bank.registerClient(clientMax, -5000, 10000);  bank.registerClient(clientDmitry, -5000, 10000);  bank.hireAdmin(adminAlex);  bank.purchaseByCard(clientSergey, 100, bank.getAccountIdByPerson(clientMax)); // card -> account; Sergey -100; 10000; Max 0; 10100  bank.purchaseByCard(clientDmitry, 15000, bank.getAccountIdByPerson(clientMax)); // card -> account; Dmitry -15000; 10000; Max 0; 25100  bank.transferFromAccount(clientDmitry, 100, bank.getAccountIdByPerson(clientMax)); // account -> account; Dmitry -15100; 10000; Max 0; 25200  bank.blockCardById(clientSergey, bank.getCardIdByPerson(clientSergey)); // card.isBlocked  bank.closeAccount(clientDmitry); // card -> account; Dmitry -5100; null  bank.blockCardById(adminAlex, bank.getCardIdByPerson(clientDmitry)); // card.isBlocked  System.out.println("=== System state after all operations ===\n\n");  System.out.println(bank);  }  }  class Bank {  private HashMap<Person, Integer> cardsIdByClient;  private HashMap<Person, Integer> accountsIdByClient;  private HashMap<Integer, Card> cardsById;  private HashMap<Integer, Account> accountsById;  private HashSet<Person> admins;  private HashSet<Person> clients;  public Bank() {  cardsIdByClient = new HashMap<>();  accountsIdByClient = new HashMap<>();  cardsById = new HashMap<>();  accountsById = new HashMap<>();  admins = new HashSet<>();  clients = new HashSet<>();  }  public String toString() {  String result = "=== Clients ===\n";  for(Person client : clients) {  result = String.format("%s %n%n %s", result, client);  if(cardsIdByClient.containsKey(client)) {  Card card = cardsById.get(cardsIdByClient.get(client));  result = String.format("%s %n %s", result, card);  }  if(accountsIdByClient.containsKey(client)) {  Account account = accountsById.get(accountsIdByClient.get(client));  result = String.format("%s %n %s", result, account);  }  }  result += "\n\n=== Admins ===\n";  for(Person admin : admins) {  result = String.format("%s %n %s", result, admin);  }  return result;  }  public void hireAdmin(Person admin) {  admins.add(admin);  }  public void registerClient(Person client, Integer cardLimit, Integer accountBalance) throws Exception {  Card card = new Card(cardLimit);  Account account = new Account(accountBalance);  if(clients.contains(client)) {  throw new Exception("Client is already registered!");  }  cardsIdByClient.put(client, card.getId());  accountsIdByClient.put(client, account.getId());  cardsById.put(card.getId(), card);  accountsById.put(account.getId(), account);  clients.add(client);  }  public void purchaseByCard(Person client, Integer amount, Integer targetAccountId) throws Exception {  if(!cardsIdByClient.containsKey(client)) {  throw new Exception("Client has no card!");  }  if(!accountsById.containsKey(targetAccountId)) {  throw new Exception("Target account doesn't exists!");  }  if(amount <= 0) {  throw new Exception("Incorrect amount!");  }  Integer cardId = cardsIdByClient.get(client);  Card card = cardsById.get(cardId);  if(card.getIsBlocked()) {  throw new Exception("Card is blocked!");  }  Account targetAccount = accountsById.get(targetAccountId);  card.changeBalance(-amount);  targetAccount.changeBalance(+amount);  cardsById.put(cardId, card);  accountsById.put(targetAccountId, targetAccount);  }  public void transferFromAccount(Person client, Integer amount, Integer targetAccountId) throws Exception {  if(!accountsIdByClient.containsKey(client)) {  throw new Exception("Client has no account!");  }  if(!accountsById.containsKey(targetAccountId)) {  throw new Exception("Target account doesn't exists!");  }  if(amount <= 0) {  throw new Exception("Incorrect amount!");  }  Integer accountId = accountsIdByClient.get(client);  Account account = accountsById.get(accountId);  if(amount > account.getBalance()) {  throw new Exception("Incorrect amount!");  }  Account targetAccount = accountsById.get(targetAccountId);  account.changeBalance(-amount);  targetAccount.changeBalance(+amount);  accountsById.put(accountId, account);  accountsById.put(targetAccountId, targetAccount);  }  public Integer getCardIdByPerson(Person client) throws Exception {  if(!cardsIdByClient.containsKey(client)){  throw new Exception("Client has no card!");  }  return cardsIdByClient.get(client);  }  public Integer getAccountIdByPerson(Person client) throws Exception {  if(!accountsIdByClient.containsKey(client)){  throw new Exception("Client has no account!");  }  return accountsIdByClient.get(client);  }  public void blockCardById(Person person, Integer targetCardId) throws Exception {  if(!cardsIdByClient.containsKey(person) && !admins.contains(person)) {  throw new Exception("Client has no card!");  }  Integer personCardId = cardsIdByClient.getOrDefault(person, -1);  if(!personCardId.equals(targetCardId) && !admins.contains(person)) {  throw new Exception("Client is not owner of card and is not admin!");  }  Card card = cardsById.get(targetCardId);  if(!personCardId.equals(targetCardId) && admins.contains(person) && !card.limitReached()) {  throw new Exception("Card limit is not reached!");  }  card.block();  cardsById.put(targetCardId, card);  }  public void closeAccount(Person client) throws Exception {  if(!accountsIdByClient.containsKey(client)) {  throw new Exception("Client has no account!");  }  if(!cardsIdByClient.containsKey(client)) {  throw new Exception("Client has no card!");  }  Integer accountId = accountsIdByClient.get(client);  Integer cardId = cardsIdByClient.get(client);  Card card = cardsById.get(cardId);  Account account = accountsById.get(accountId);  card.changeBalance(+account.getBalance());  account.changeBalance(-account.getBalance());  cardsById.put(cardId, card);  accountsById.remove(accountId);  accountsIdByClient.remove(client);  }  private class Card {  private static Integer cardIdCounter = 1000;  private Integer id;  private Integer balance;  private Integer limit;  private Boolean isBlocked;  public Card(Integer limit) {  this.id = cardIdCounter;  this.balance = 0;  this.limit = limit;  this.isBlocked = false;  cardIdCounter++;  }  public String toString() {  return String.format("<Card> id: %d balance: %d limit: %d isBlocked: %b", id, balance, limit, isBlocked);  }  public Boolean limitReached() {  return balance < limit;  }  public void block(){  this.isBlocked = true;  }  public Boolean getIsBlocked() {  return isBlocked;  }  public void changeBalance(Integer amount){  this.balance += amount;  }  public Integer getId() {  return id;  }  public Integer getBalance() {  return balance;  }  }  private class Account {  private static Integer accountIdCounter = 1000;  private Integer id;  private Integer balance;  public Account(Integer balance) {  this.id = accountIdCounter;  this.balance = balance;  accountIdCounter++;  }  public String toString() {  return String.format("<Account> id: %d balance: %d", id, balance);  }  public void changeBalance(Integer amount){  this.balance += amount;  }  public Integer getId() {  return id;  }  public Integer getBalance() {  return balance;  }  }  }  class Person {  private static Integer idCounter = 0;  private String name;  private Integer id;  public Person(String name) {  this.name = name;  this.id = idCounter;  idCounter++;  }  public String toString() {  return String.format("<Person> id: %s name: %s", this.id, this.name);  }  } |

Результат исполнения программы приведен в листинге 16.

Листинг 16 — результат исполнения программы

|  |
| --- |
| === Bank class work example ===  === System state after all operations ===  === Clients ===    <Person> id: 0 name: Sergey  <Card> id: 1000 balance: -100 limit: -5000 isBlocked: true  <Account> id: 1000 balance: 10000  <Person> id: 2 name: Dmitry  <Card> id: 1002 balance: -5100 limit: -5000 isBlocked: true  <Person> id: 1 name: Max  <Card> id: 1001 balance: 0 limit: -5000 isBlocked: false  <Account> id: 1001 balance: 25200  === Admins ===    <Person> id: 3 name: Alex |

**Вывод:** в ходе лабораторной работы были освоены базовые принципы работы с ООП на языке Java.