|  |  |
| --- | --- |
| Gerb-BMSTU_01 | **Министерство науки и высшего образования Российской Федерации**  **Федеральное государственное бюджетное образовательное учреждение**  **высшего образования**  **«Московский государственный технический университет**  **имени Н.Э. Баумана**  **(национальный исследовательский университет)»**  **(МГТУ им. Н.Э. Баумана)** |

ФАКУЛЬТЕТ **Информатика и системы управления**

КАФЕДРА **Компьютерные системы и сети (ИУ6)**

НАПРАВЛЕНИЕ ПОДГОТОВКИ **09.04.01 Информатика и вычислительная техника**

МАГИСТЕРСКАЯ ПРОГРАММА **09.04.01/07 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 |  |  | Н.Б.Гендина | |
|  | (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.

**Задание 1 вариант 4**

Определить класс Матрица размерности (n x n). Класс должен содержать несколько конструкторов. Реализовать методы для сложения, вычитания, умножения матриц. Объявить массив объектов. Создать методы, вычисляющие первую и вторую нормы матрицы. Определить, какая из матриц имеет наименьшую первую и вторую нормы.

|  |
| --- |
| import java.util.Scanner;  public class Main {  public static void main(String[] args) {  Scanner sc = new Scanner(System.in);  System.out.println("Укажите размер матриц:");  int size = sc.nextInt();  System.out.println("Укажите количество матриц:");  int count = sc.nextInt();  Matrix[] array = new Matrix[count];  for (int i = 0; i < count; i++) {  array[i] = new Matrix(size);  System.out.println("Получившаяся матрица " + i + ":\n" + array[i]);  }  System.out.println("1 и 2 нормы матриц:");  int min1Norm = array[0].norm1(), min2Norm = array[0].norm2();  int min1NormIndex = 0, min2NormIndex = 0;  for(int i = 0; i < count; i++) {  int norm1 = array[i].norm1(), norm2 = array[i].norm2();  System.out.println(i + " матрица: 1 норма = " + norm1 + ", 2 норма = " + norm2);  if (norm1 < min1Norm) {  min1Norm = norm1;  min1NormIndex = i;  }  if (norm2 < min2Norm) {  min2Norm = norm2;  min2NormIndex = i;  }  }  System.out.println("\nНаименьшую 1 норму имеет матрица " + min1NormIndex + ":\n" + array[min1NormIndex]);  System.out.println("Наименьшую 2 норму имеет матрица " + min2NormIndex + ":\n" + array[min2NormIndex]);  }  } |

|  |
| --- |
| import java.util.Random;  public class Matrix {  private final int[][] data;  private final int size;  public Matrix(int[][] data) {  this.data = data;  this.size = data.length;  }  public Matrix(int size) {  this.size = size;  this.data = new int[size][size];  Random r = new Random();  for(int i = 0; i < size; i++) {  for(int j = 0; j < size; j++) {  this.data[i][j]=r.nextInt() % 100;  }  }  }  public String toString() {  StringBuilder result = new StringBuilder();  for(int[] arr: this.data) {  for (int el : arr) {  result.append(el).append("\t");  }  result.append("\n");  }  return String.valueOf(result);  }  private void checkSize(int[][] matrix) throws Exception {  if (this.size != matrix.length) {  throw new Exception("Матрицы должны иметь одинаковую размерность");  }  }  public int[][] add(int[][] matrix) throws Exception {  this.checkSize(matrix);  int[][] result = new int[this.size][this.size];  for(int i = 0; i < size; i++) {  for(int j = 0; j < size; j++) {  result[i][j] = this.data[i][j] + matrix[i][j];  }  }  return result;  }  public int[][] sub(int[][] matrix) throws Exception {  this.checkSize(matrix);  int[][] result = new int[this.size][this.size];  for(int i = 0; i < size; i++) {  for(int j = 0; j < size; j++) {  return result;  }  for(int j = 0; j < size; j++) {  result[i][j] = this.data[i][j] - matrix[i][j];  }  }  return result;  }  public int[][] mul(int[][] matrix) throws Exception {  this.checkSize(matrix);  int[][] result = new int[this.size][this.size];  for(int l = 0; l < size; l++) {  for(int i = 0; i < size; i++) {  for(int j = 0; j < size; j++) {  result[l][i] += this.data[l][j] \* matrix[j][i];  }  }  }  return result;  }  public int norm1() {  Integer max\_sum = null;  for(int[] arr: this.data) {  int sum = 0;  for(int el: arr) {  sum += Math.abs(el);  }  if ((max\_sum == null) || (sum > max\_sum)) {  max\_sum = sum;  }  }  return max\_sum;  }  public int norm2() {  Integer max\_sum = null;  for(int i = 0; i < size; i++) {  int sum = 0;  for(int j = 0; j < size; j++) {  sum += Math.abs(this.data[j][i]);  }  if ((max\_sum == null) || (sum > max\_sum)) {  max\_sum = sum;  }  }  return max\_sum;  }  } |

**Задание 1 вариант 5**

Определить класс Матрица размерности (m x n). Класс должен содержать несколько конструкторов. Объявить массив объектов. Передать объекты в метод, меняющий местами строки с максимальным и минимальным элементами k-го столбца. Создать метод, который изменяет i-ю матрицу путем возведения ее в квадрат.

|  |
| --- |
| import java.util.Scanner;  public class Main {  public static void main(String[] args) {  Scanner sc = new Scanner(System.in);  System.out.println("Укажите количество строк в матрицах:");  int m = sc.nextInt();  System.out.println("Укажите количество столбцов в матрицах:");  int n = sc.nextInt();  System.out.println("Укажите количество матриц:");  int count = sc.nextInt();  Matrix[] array = new Matrix[count];  for (int i = 0; i < count; i++) {  array[i] = new Matrix(m, n);  System.out.println("Получившаяся матрица " + i + ":\n" + array[i]);  }  System.out.println("Укажите относительно какого столбца менять строки:");  int k = sc.nextInt();  if (k > n - 1) {  System.out.println("Данного столбца не существует.");  return;  }  array = swapRowsInMatrix(array, k);  System.out.println("Объекты с поменяными местами строками с максимальным и минимальным элементами " + k + "-го столбца:");  printArray(array);  System.out.println("Укажите какую матрицу надо возвести в квадрат:");  int i = sc.nextInt();  if (i > count - 1) {  System.out.println("Данной матрицы не существует.");  return;  }  try {  array = squaringCertainMatrix(array, i);  } catch (Exception e) {  throw new RuntimeException(e);  }  System.out.println("Объекты с возведенной в квадрат " + i + " матрицей:");  printArray(array);  }  public static Matrix[] swapRowsInMatrix(Matrix[] array, int k) {  Matrix[] result = new Matrix[array.length];  for (int i = 0; i < array.length; i++) {  result[i] = array[i].swapRowsInMatrix(k);  }  return result;  }  public static Matrix[] squaringCertainMatrix(Matrix[] array, int i) throws Exception {  Matrix[] result = new Matrix[array.length];  for (int j = 0; j < array.length; j++) {  if (j == i) {  result[j] = array[j].squaring();  continue;  }  result[j] = array[j];  }  return result;  }  public static void printArray(Matrix[] array) {  for (Matrix matrix: array){  System.out.println(matrix);  }  }  } |

|  |
| --- |
| import java.util.Random;  public class Matrix {  private final int[][] data;  private final int m;  private final int n;  public Matrix(int[][] data) {  this.data = data;  this.m = data.length;  this.n = data[0].length;  }  public Matrix(int m, int n) {  this.m = m;  this.n = n;  this.data = new int[m][n];  Random r = new Random();  for(int i = 0; i < m; i++) {  for(int j = 0; j < n; j++) {  this.data[i][j]=r.nextInt() % 100;  }  }  }  public String toString() {  StringBuilder result = new StringBuilder();  for(int[] arr: this.data) {  for (int el : arr) {  result.append(el).append("\t\t");  }  result.append("\n");  }  return String.valueOf(result);  }  public Matrix squaring() throws Exception {  if (this.m != this.n) {  throw new Exception("Возводить в квадрат можно только квадратные матрицы.");  }  int[][] result = new int[this.m][this.n];  for(int l = 0; l < m; l++) {  for(int i = 0; i < n; i++) {  for(int j = 0; j < n; j++) {  result[l][i] += this.data[l][j] \* this.data[j][i];  }  }  }  return new Matrix(result);  }  public Matrix swapRowsInMatrix(int k) {  int[][] result = new int[this.m][this.n];  int min = this.data[0][k], max = this.data[0][k];  int indexMin = 0, indexMax = 0;  for (int i = 0; i < this.m; i++){  if (this.data[i][k] < min) {  min = this.data[i][k];  indexMin = i;  }  if (this.data[i][k] > max) {  max = this.data[i][k];  indexMax = i;  }  }  for (int i = 0; i < this.m; i++){  for (int j = 0; j < this.n; j++) {  if (i == indexMin) {  result[i][j] = this.data[indexMax][j];  continue;  }  if (i == indexMax) {  result[i][j] = this.data[indexMin][j];  continue;  }  result[i][j] = this.data[i][j];  }  }  return new Matrix(result);  }  } |

**Задание 2 вариант 4**

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. Abiturient: id, Фамилия, Имя, Отчество, Адрес, Телефон, Оценки. Создать массив объектов. Вывести: a) список абитуриентов, имеющих неудовлетворительные оценки; b) список абитуриентов, средний балл у которых выше заданного; c) выбрать заданное число n абитуриентов, имеющих самый высокий средний балл (вывести также полный список абитуриентов, имеющих полупроходной балл).

|  |
| --- |
| import java.util.Scanner;  public class Main {  public static void main(String[] args) {  Scanner sc = new Scanner(System.in);  Collection collection = new Collection(new Student[]{  new Student(1,"Гендина", "Нина", "Борисовна",  "ул. 3 Гражданская", "79999999999", new int[]{4, 5, 5, 5, 5}),  new Student(2,"Баканов", "Роман", "Викторович",  "ул. Островского", "78888888888", new int[]{4, 4, 4, 5, 4}),  new Student(3,"Иванов", "Иван", "Иванович",  "ул. Краснобогатырская", "76666666666", new int[]{2, 3, 4, 5, 3}),  new Student(4,"Петров", "Петр", "Петрович",  "ул. Арбат", "73333333333", new int[]{2, 3, 4, 4, 4}),  new Student(5,"Сергеев", "Сергей", "Сергеевич",  "ул. Миллионная", "74444444444", new int[]{2, 3, 2, 2, 3})  });  System.out.println("Список абитуриентов, имеющих неудовлетворительные оценки:");  System.out.println(collection.getStudentsWithBadMarks());  System.out.println("Укажите средний балл:");  int averageMark = sc.nextInt();  System.out.println("Список абитуриентов, средний балл у которых выше " + averageMark + ":");  System.out.println(collection.getStudentsWithAverageMarkAboveGiven(averageMark));  System.out.println("Укажите количество абитуриентов:");  int n = sc.nextInt();  System.out.println(n + " абитуриентов, имеющих самый высокий средний балл:");  System.out.println(collection.getStudentsWithMaxAverageMark(n));  System.out.println("Абитуриентов, имеющие полупроходной балл:");  System.out.println(collection.getStudentsWithSemiPassingMark(n));  }  } |

|  |
| --- |
| import java.util.Arrays;  import java.util.Comparator;  import java.util.stream.Stream;  public class Collection {  private final Student[] students;  public Collection(Student[] students) {  this.students = students;  }  public String toString() {  StringBuilder result = new StringBuilder();  for(Student student: this.students) {  result.append(student);  }  return String.valueOf(result);  }  public Collection getStudentsWithBadMarks() {  Stream<Student> result = Arrays.stream(this.students).filter(Student::hasBadMarks);  return new Collection(result.toArray(Student[]::new));  }  public Collection getStudentsWithAverageMarkAboveGiven(float mark) {  Stream<Student> result = Arrays.stream(this.students).filter((s) -> s.getAverageMark() > mark);  return new Collection(result.toArray(Student[]::new));  }  public Collection getStudentsWithMaxAverageMark(int count) {  Stream<Student> result = Arrays.stream(this.students).sorted(Comparator.comparing(Student::getAverageMark).reversed()).limit(count);  return new Collection(result.toArray(Student[]::new));  }  public Collection getStudentsWithSemiPassingMark(int count) {  Student[] sortedStudents = Arrays.stream(this.students).sorted(  Comparator.comparing(Student::getAverageMark).reversed()  ).toArray(Student[]::new);  float semiPassingMark = sortedStudents[count].getAverageMark();  Stream<Student> result = Arrays.stream(sortedStudents).filter((s) -> s.getAverageMark() == semiPassingMark);  return new Collection(result.toArray(Student[]::new));  }  } |

|  |
| --- |
| import java.util.Arrays;  public class Student {  private int id;  private String lastName;  private String firstName;  private String patronymic;  private String address;  private String phone;  private int[] marks;  public Student() {}  public Student(int id, String lastName, String firstName, String patronymic, String address, String phone, int[] marks) {  this.id = id;  this.lastName = lastName;  this.firstName = firstName;  this.patronymic = patronymic;  this.address = address;  this.phone = phone;  this.marks = marks;  }  public String toString() {  return String.format(  "Student(%d) %s %s %s.\tAddress: %s.\tPhone: %s.\tMarks: %s\n",  this.id, this.lastName, this.firstName, this.patronymic, this.address, this.phone,  Arrays.toString(this.marks)  );  }  public void setId(int id) {  this.id = id;  }  public int getId() {  return this.id;  }  public void setLastName(String lastName) {  this.lastName = lastName;  }  public String getLastName() {  return this.lastName;  }  public void setFirstName(String firstName) {  this.firstName = firstName;  }  public String getFirstName() {  return this.firstName;  }  public void setPatronymic(String patronymic) {  this.patronymic = patronymic;  }  public String getPatronymic() {  return this.patronymic;  }  public void setAddress(String address) {  this.address = address;  }  public String getAddress() {  return this.address;  }  public void setPhone(String phone) {  this.phone = phone;  }  public String getPhone() {  return this.phone;  }  public void setMarks(int[] marks) {  this.marks = marks;  }  public int[] getMarks() {  return this.marks;  }  public boolean hasBadMarks(){  for(int mark: this.marks) {  if (mark < 4) {  return true;  }  }  return false;  }  public float getAverageMark() {  return (float) Arrays.stream(this.marks).sum() / this.marks.length;  }  } |

**Задание 3 вариант 4**

Создать приложение, удовлетворяющее требованиям, приведенным в задании. Аргументировать принадлежность классу каждого создаваемого метода и корректно переопределить для каждого класса методы equals(), hashCode(), toString(). Создать объект класса Простая дробь, используя класс Число. Методы: вывод на экран, сложение, вычитание, умножение, деление.

|  |
| --- |
| public class Main {  public static void main(String[] args) {  Fraction fraction = new Fraction(1,2);  Fraction fraction2 = new Fraction(3,4);  Fraction fraction3 = new Fraction(6,12);  Fraction fraction4 = new Fraction(42,56);  Fraction fraction5 = new Fraction(3,4);  System.out.println("Имеющиеся дроби:");  System.out.println(fraction);  System.out.println(fraction2);  System.out.println(fraction3);  System.out.println(fraction4);  System.out.println(fraction5);  System.out.println();  System.out.println(fraction + " + " + fraction2 + " = " + fraction.sum(fraction2));  System.out.println(fraction + " + " + fraction4 + " = " + fraction.sum(fraction4));  System.out.println(fraction + " - " + fraction4 + " = " + fraction.sub(fraction4));  System.out.println(fraction + " - " + fraction3 + " = " + fraction.sub(fraction3));  System.out.println(fraction + " \* " + fraction4 + " = " + fraction.mul(fraction4));  System.out.println(fraction + " \* " + fraction3 + " = " + fraction.mul(fraction3));  System.out.println(fraction + " : " + fraction4 + " = " + fraction.division(fraction4));  System.out.println(fraction + " : " + fraction3 + " = " + fraction.division(fraction3));  System.out.println("hashCode для " + fraction + ": " + fraction.hashCode());  System.out.println("hashCode для " + fraction2 + ": " + fraction2.hashCode());  System.out.println("hashCode для " + fraction3 + ": " + fraction3.hashCode());  System.out.println("hashCode для " + fraction4 + ": " + fraction4.hashCode());  System.out.println("hashCode для " + fraction5 + ": " + fraction5.hashCode());  System.out.println(fraction + " == " + fraction4 + " - " + fraction.equals(fraction4));  System.out.println(fraction + " == " + fraction3 + " - " + fraction.equals(fraction3));  System.out.println(fraction2 + " == " + fraction4 + " - " + fraction2.equals(fraction4));  System.out.println(fraction2 + " == " + fraction5 + " - " + fraction2.equals(fraction5));  }  } |

|  |
| --- |
| public class Fraction extends Number {  private final int numerator;  private final int denominator;  public Fraction(int numerator, int denominator) {  this.numerator = numerator;  this.denominator = denominator;  }  @Override  public String toString() {  return String.format("%d/%d", this.numerator, this.denominator);  }  @Override  public int intValue() {  return this.numerator / this.denominator;  }  @Override  public long longValue() {  return (long) this.numerator / this.denominator;  }  @Override  public float floatValue() {  return (float) this.numerator / this.denominator;  }  @Override  public double doubleValue() {  return (double) this.numerator / this.denominator;  }  public int[] getNumeratorAndDenominator() {  return new int[] {this.numerator, this.denominator};  }  public boolean equals(Object o) {  if (o instanceof Fraction fraction) {  int nod = NOD(this.numerator, this.denominator);  int objectNumerator = fraction.getNumeratorAndDenominator()[0];  int objectDenominator = fraction.getNumeratorAndDenominator()[1];  int objectNod = NOD(objectNumerator, objectDenominator);  return (this.numerator / nod == objectNumerator / objectNod)  && (this.denominator / nod == objectDenominator / objectNod);  }  return false;  }  public int hashCode() {  int nod = NOD(this.numerator, this.denominator);  int numerator = this.numerator / nod;  int denominator = this.denominator / nod;  return Float.hashCode((float) numerator / denominator);  }  public void print() {  System.out.println(this);  }  public Fraction sum(Fraction other) {  int resultNumerator = this.numerator \* other.denominator + other.numerator \* this.denominator ;  int resultDenominator = this.denominator \* other.denominator;  int nod = NOD(resultNumerator, resultDenominator);  return new Fraction(resultNumerator / nod, resultDenominator / nod);  }  public Fraction sub(Fraction other) {  int resultNumerator = this.numerator \* other.denominator - other.numerator \* this.denominator ;  int resultDenominator = this.denominator \* other.denominator;  int nod = NOD(resultNumerator, resultDenominator);  return new Fraction(resultNumerator / nod, resultDenominator / nod);  }  public Fraction mul(Fraction other) {  int resultNumerator = this.numerator \* other.numerator;  int resultDenominator = this.denominator \* other.denominator;  int nod = NOD(resultNumerator, resultDenominator);  return new Fraction(resultNumerator / nod, resultDenominator / nod);  }  public Fraction division(Fraction other) {  int resultNumerator = this.numerator \* other.denominator;  int resultDenominator = this.denominator \* other.numerator;  int nod = NOD(resultNumerator, resultDenominator);  return new Fraction(resultNumerator / nod, resultDenominator / nod);  }  private int NOD(int a, int b) {  a = Math.abs(a);  b = Math.abs(b);  if (a == 0) {  return b;  }  while (a != b) {  if (a > b) {  a -= b;  } else {  b -= a;  }  }  return a;  }  } |

**Задание 3 вариант 5**

Создать приложение, удовлетворяющее требованиям, приведенным в задании. Аргументировать принадлежность классу каждого создаваемого метода и корректно переопределить для каждого класса методы equals(), hashCode(), toString(). Создать объект класса Дом, используя классы Окно, Дверь. Методы: закрыть на ключ, вывести на консоль количество окон, дверей.

|  |
| --- |
| public class Main {  public static void main(String[] args) {  House house = new House(10, "3 гражданская",  new Door[]{new Door(), new Door()},  new Window[] {  new Window(), new Window(),  new Window(10, 30), new Window(10, 30)  });  House house2 = new House(10, "3 гражданская",  new Door[]{new Door(), new Door()},  new Window[] {  new Window(), new Window(),  new Window(10, 30), new Window(10, 30)  });  House house3 = new House(15, "Краснобогатырская",  new Door[]{new Door()},  new Window[]{new Window(), new Window()}  );  System.out.println(house);  System.out.println("Количество окон: " + house.getNumberOfWindows() + ", количество дверей: " + house.getNumberOfDoor());  System.out.println("hashCode = " + house.hashCode());  System.out.println();  System.out.println(house2);  System.out.println("Количество окон: " + house2.getNumberOfWindows() + ", количество дверей: " + house2.getNumberOfDoor());  System.out.println("hashCode = " + house2.hashCode());  System.out.println();  System.out.println(house3);  System.out.println("Количество окон: " + house3.getNumberOfWindows() + ", количество дверей: " + house3.getNumberOfDoor());  System.out.println("hashCode = " + house3.hashCode());  System.out.println();  System.out.println("Закрываем дом на ключ.");  house.lockWithKey();  System.out.println("Дом закрыт: " + house.isСlosed());  System.out.println();  System.out.println("Открываем дом ключом.");  house.openWithKey();  System.out.println("Дом закрыт: " + house.isСlosed());  }  } |

|  |
| --- |
| import java.util.Objects;  public class House {  private final int number;  private final String street;  private final Door[] doors;  private final Window[] windows;  public House(int number, String street, Door[] doors, Window[] windows) {  this.number = number;  this.street = street;  this.doors = doors;  this.windows = windows;  }  public String toString() {  return String.format("House number %d, street: %s.", this.number, this.street);  }  public boolean equals(Object o) {  if (o instanceof House house) {  return (Objects.equals(this.street, house.street)) && (this.number == house.number);  }  return false;  }  public int hashCode() {  return String.format("%sh%d", this.street, this.number).hashCode();  }  public void print() {  System.out.println(this);  }  public void lockWithKey() {  for(Door door: this.doors) {  door.lockWithKey();  }  }  public void openWithKey() {  for(Door door: this.doors) {  door.openWithKey();  }  }  public int getNumberOfWindows() {  return this.windows.length;  }  public int getNumberOfDoor() {  return this.doors.length;  }  public boolean isСlosed() {  for (Door door: this.doors) {  if (door.getIsOpen()) {  return false;  }  }  return true;  }  } |

|  |
| --- |
| import java.util.Objects;  public class Door {  private final int width;  private final int height;  private final String material;  private final String color;  private boolean isOpen = true;  public Door() {  this.width = 10;  this.height = 100;  this.material = "wood";  this.color = "black";  }  public Door(int width, int height, String material, String color) {  this.width = width;  this.height = height;  this.material = material;  this.color = color;  }  public void lockWithKey() {  this.isOpen = false;  }  public void openWithKey() {  this.isOpen = true;  }  public String toString() {  return String.format("Door (%dx%d), color: %s, material: %s.\n", this.width, this.height, this.color, this.material);  }  public boolean equals(Object o) {  if (o instanceof Door door) {  return (this.width == door.width)  && (this.height == door.height)  && (Objects.equals(this.material, door.material))  && (Objects.equals(this.color, door.color));  }  return false;  }  public int hashCode() {  return String.format("%s%dx%d%s", this.material, this.width, this.height, this.color).hashCode();  }  public boolean getIsOpen() {  return this.isOpen;  }  } |

|  |
| --- |
| import java.util.Random;  public class Window {  private final int width;  private final int height;  public Window(int width, int height) {  this.width = width;  this.height = height;  }  public Window() {  Random r = new Random();  this.width = r.nextInt() % 100;  this.height = r.nextInt() % 100;  }  public String toString() {  return String.format("Window (%dx%d).\n", this.width, this.height);  }  public boolean equals(Object o) {  if (o instanceof Window window) {  return (this.width == window.width) && (this.height == window.height);  }  return false;  }  public int hashCode() {  return String.format("%dx%d", this.width, this.height).hashCode();  }  } |

**Задание 4 вариант 4**

Построить модель программной системы. Система Вступительные экзамены. Абитуриент регистрируется на Факультет, сдает Экзамены. Преподаватель выставляет Оценку. Система подсчитывает средний балл и определяет Абитуриентов, зачисленных в учебное заведение.

|  |
| --- |
| import java.util.Arrays;  public class Main {  public static void main(String[] args) {  // Создали два факультета  Faculty facultyOfComputerScience = new Faculty(  "Computer Science", "CS1",  new Subject[] {Subject.INFORMATICS, Subject.MATHEMATICS}, 2,  new Teacher[]{  new Teacher("Иванов", "Иван", "Иванович"),  new Teacher("Петров", "Петр", "Петрович"),  new Teacher("Сидоров", "Семен", "Семенович")  }  );  Faculty facultyOfPhysics = new Faculty(  "Physics", "Ph1",  new Subject[] {Subject.PHYSICS, Subject.MATHEMATICS}, 1,  new Teacher[]{  new Teacher("Иванова", "Анна", "Ивановна"),  new Teacher("Петрова", "Алина", "Андреевна")  }  );  Applicant applicant1 = new Applicant("Третьяков", "Иван", "Игоревич");  Applicant applicant2 = new Applicant("Алексеева", "Анна", "Романовна");  Applicant applicant3 = new Applicant("Касаткин", "Никита", "Максимович");  Applicant applicant4 = new Applicant("Новикова", "Алиса", "Тимофеевна");  Applicant applicant5 = new Applicant("Иванов", "Ян", "Максимович");  // Зарегистрировались на факультеты  facultyOfComputerScience.register(applicant1);  facultyOfComputerScience.register(applicant2);  facultyOfComputerScience.register(applicant3);  facultyOfPhysics.register(applicant3);  facultyOfPhysics.register(applicant4);  facultyOfPhysics.register(applicant5);  System.out.println("Студенты на факультетах:");  System.out.println(facultyOfComputerScience.getInformationAboutApplicants());  System.out.println(facultyOfPhysics.getInformationAboutApplicants());  // Сдача экзаменов  facultyOfComputerScience.passExam(Subject.INFORMATICS, applicant1);  facultyOfComputerScience.passExam(Subject.MATHEMATICS, applicant1);  facultyOfComputerScience.passExam(Subject.INFORMATICS, applicant2);  facultyOfComputerScience.passExam(Subject.MATHEMATICS, applicant2);  facultyOfComputerScience.passExam(Subject.INFORMATICS, applicant3);  facultyOfComputerScience.passExam(Subject.MATHEMATICS, applicant3);  facultyOfPhysics.passExam(Subject.PHYSICS, applicant3);  facultyOfPhysics.passExam(Subject.MATHEMATICS, applicant3);  facultyOfPhysics.passExam(Subject.PHYSICS, applicant4);  facultyOfPhysics.passExam(Subject.MATHEMATICS, applicant4);  facultyOfPhysics.passExam(Subject.PHYSICS, applicant5);  facultyOfPhysics.passExam(Subject.MATHEMATICS, applicant5);  // Вывод результатов экзаменов для студентов по факультетам  System.out.println("Результаты экзаменов:");  System.out.println(applicant1 + ": " + applicant1.getExamResults(facultyOfComputerScience.getCode()));  System.out.println(applicant2 + ": " + applicant2.getExamResults(facultyOfComputerScience.getCode()));  System.out.println(applicant3 + ": " + applicant3.getExamResults(facultyOfComputerScience.getCode()));  System.out.println(applicant3 + ": " + applicant3.getExamResults(facultyOfPhysics.getCode()));  System.out.println(applicant4 + ": " + applicant4.getExamResults(facultyOfPhysics.getCode()));  System.out.println(applicant5 + ": " + applicant5.getExamResults(facultyOfPhysics.getCode()));  // Определение зачисленных абитуриентов  System.out.println("\nЗачисленные абитуриенты:");  System.out.println(facultyOfComputerScience + ": " + Arrays.toString(facultyOfComputerScience.determinationOfAdmittedApplicants()));  System.out.println(facultyOfPhysics + ": " + Arrays.toString(facultyOfPhysics.determinationOfAdmittedApplicants()));  }} |
| import java.util.HashMap;  public class Applicant {  private final String lastName;  private final String name;  private final String patronymic;  HashMap<String, HashMap<Subject, Integer>> examResults;  public Applicant(String lastName, String name, String patronymic) {  this.lastName = lastName;  this.name = name;  this.patronymic = patronymic;  this.examResults = new HashMap<>();  }  public String toString() {  return String.format("Applicant %s %s %s", this.lastName, this.name, this.patronymic);  }  public String getExamResults(String facultyCode) {  if (examResults.containsKey(facultyCode)) {  StringBuilder result = new StringBuilder(String.format("Для факультета %s оценки: ", facultyCode));  for (Subject subject: this.examResults.get(facultyCode).keySet()) {  result.append(String.format("%s - %d, ", subject, this.examResults.get(facultyCode).get(subject)));  }  return String.valueOf(result);  }  return String.format("Для факультета %s нет оценок", facultyCode);  }  public void setExamResults(String facultyCode, Subject subject, int mark) {  if (!examResults.containsKey(facultyCode)) {  this.examResults.put(facultyCode, new HashMap<>());  }  this.examResults.get(facultyCode).put(subject, mark);  }  public float getAverageMark(String facultyCode) {  if (examResults.containsKey(facultyCode)) {  int sum = 0, count = 0;  for (Subject subject: this.examResults.get(facultyCode).keySet()) {  sum += this.examResults.get(facultyCode).get(subject);  count +=1;  }  return (float) sum / count;  }  return 0f;  }  } |

|  |
| --- |
| import java.util.\*;  public class Faculty {  private final String name;  private final String code;  private final Subject[] subjects;  private final int totalNumberOfStudents;  private final Teacher[] teachers;  private List<Applicant> applicants;  public Faculty(String name, String code, Subject[] subjects, int totalNumberOfStudents, Teacher[] teachers) {  this.name = name;  this.code = code;  this.subjects = subjects;  this.totalNumberOfStudents = totalNumberOfStudents;  this.teachers = teachers;  this.applicants = new ArrayList<>();  }  public String toString() {  return String.format("Faculty %s (%s)", this.name, this.code);  }  public String getName() {  return this.name;  }  public String getCode() {  return this.code;  }  public Subject[] getSubjects() {  return this.subjects;  }  public List<Applicant> getApplicants() {  return this.applicants;  }  public int getTotalNumberOfStudents() {  return this.totalNumberOfStudents;  }  public Teacher[] getTeachers() {  return this.teachers;  }  public void register(Applicant applicant) {  this.applicants.add(applicant);  }  public void passExam(Subject subject, Applicant applicant) {  int randomTeacher = new Random().nextInt(this.teachers.length);  this.teachers[randomTeacher].gradeExam(subject, applicant, this.code);  }  public Applicant[] determinationOfAdmittedApplicants() {  return this.applicants.stream().sorted(new Comparator<Applicant>() {  public int compare(Applicant a1, Applicant a2) {  return Float.compare(a1.getAverageMark(code), a2.getAverageMark(code));  }  }.reversed()  ).limit(this.totalNumberOfStudents).toArray(Applicant[]::new);  }  public String getInformationAboutApplicants() {  StringBuilder result = new StringBuilder(String.format("Students by %s: \n", this));  for (Applicant applicant: this.applicants) {  result.append(applicant).append("\n");  }  return String.valueOf(result);  }  } |

|  |
| --- |
| public enum Subject {  MATHEMATICS,  INFORMATICS,  PHYSICS,  CHEMISTRY,  RUSSIAN\_LANGUAGE  } |

|  |
| --- |
| import java.util.Random;  public class Teacher {  private final String lastName;  private final String name;  private final String patronymic;  public Teacher(String lastName, String name, String patronymic) {  this.lastName = lastName;  this.name = name;  this.patronymic = patronymic;  }  public String toString() {  return String.format("Teacher %s %s %s", this.lastName, this.name, this.patronymic);  }  public void gradeExam(Subject subject, Applicant applicant, String facultyCode) {  int mark = new Random().nextInt() % 5;  applicant.setExamResults(facultyCode, subject, Math.abs(mark) + 1);  }  } |

**Задание 4 вариант 5**

Построить модель программной системы. Система Библиотека. Читатель оформляет Заказ на Книгу. Система осуществляет поиск в Каталоге. Библиотекарь выдает Читателю Книгу на абонемент или в читальный зал. При невозвращении Книги Читателем он может быть занесен Администратором в «черный список».

|  |
| --- |
| public class Main {  public static void main(String[] args) {  Library library = new Library(  "Ромашка",  "ул. 3 гражданская, д.13",  new Librarian[]{  new Librarian("Иванов", "Иван", "Иванович", false),  new Librarian("Петров", "Петр", "Петрович", false),  },  new Librarian("Зайцева", "Татьяна", "Ивановна", true),  7,  12  );  library.setCatalog(new Book[]{  new Book("Руслан и Людмила", "Пушкин"),  new Book("Война и мир", "Толстой"),  new Book("Преступление и наказание", "Достоевский"),  new Book("Мастер и Маргарита", "Булгаков"),  new Book("Собачье сердце", "Булгаков"),  });  System.out.printf("Информация о библиотеке:\n" + library);  Reader reader1 = new Reader(1, "Гендина", "Нина", "Борисовна");  Reader reader2 = new Reader(2, "Баканов", "Роман", "Викторович");  Reader reader3 = new Reader(3, "Шишкин", "Сегей", "Андреевич");  // Читатели присоединяются к библиотеке  library.addReaderToLibrary(reader1);  library.addReaderToLibrary(reader2);  library.addReaderToLibrary(reader3);  System.out.println("\n\nВсе читатели библиотеки:\n" + library.getReaders());  // Оформление заказов на книги  System.out.println(library.orderBook(reader1, "Руслан и Людмила", BookIssueType.SUBSCRIPTION));  System.out.println(library.orderBook(reader2, "Война и мир", BookIssueType.READING\_ROOM));  System.out.println(library.orderBook(reader3, "Руслан и Людмила", BookIssueType.READING\_ROOM));  System.out.println(library.orderBook(reader3, "Собачье сердце", BookIssueType.SUBSCRIPTION));  System.out.println();  // Проверка просроченного владения книгой и занесение в черный список  System.out.println("Пользователи, просрочившие сдачу книги: " + library.checkReaders());  // Возврат книги  System.out.println();  System.out.println(library.returnBook(reader1, "Руслан и Людмила"));  System.out.println(library.returnBook(reader1, "Война и мир"));  System.out.println(library.returnBook(reader2, "Война и мир"));  System.out.println(library.returnBook(reader3, "Руслан и Людмила"));  System.out.println(library.returnBook(reader3, "Собачье сердце"));  }  } |

|  |
| --- |
| import java.util.ArrayList;  import java.util.HashMap;  import java.util.List;  import java.util.Random;  public class Library {  private final String name;  private final String address;  private final Librarian[] librarians;  private final Librarian administrator;  private final int permissibleNumberOfDaysForStoringBook;  private final int permissibleNumberOfHoursOfUsingBook;  private HashMap<String, Book> catalog;  private HashMap<Integer, Reader> readers;  public Library(String name, String address, Librarian[] librarians, Librarian administrator, int permissibleNumberOfDaysForStoringBook, int permissibleNumberOfHoursOfUsingBook) {  this.name = name;  this.address = address;  this.librarians = librarians;  this.administrator = administrator;  this.permissibleNumberOfDaysForStoringBook = permissibleNumberOfDaysForStoringBook;  this.permissibleNumberOfHoursOfUsingBook = permissibleNumberOfHoursOfUsingBook;  this.readers = new HashMap<>();  }  public String toString() {  return String.format("""  Библиотека %s по адресу: %s. Администратор: %s.  Допустимое количество дней аренды книги: %d.  Допустимое количество часов пользования книгой в библиотеке: %d.""",  this.name, this.address, this.administrator,  this.permissibleNumberOfDaysForStoringBook, this.permissibleNumberOfHoursOfUsingBook  );  }  public void setCatalog(Book[] books) {  this.catalog = new HashMap<>();  for (Book book: books) {  this.catalog.put(book.getName(), book);  }  }  public String getReaders() {  StringBuilder result = new StringBuilder();  for (Reader reader: this.readers.values()) {  result.append(reader).append("\n");  }  return String.valueOf(result);  }  public String orderBook(Reader reader, String bookName, BookIssueType bookIssueType) {  if (!this.readers.containsKey(reader.getId())) {  return String.format("Читатель %s не принадлежит библиотеке", reader);  }  if (reader.getIsBanned()) {  return String.format("Читатель %s забанен", reader);  }  if (!this.catalog.containsKey(bookName)) {  return String.format("Книги %s нет в библиотеке", bookName);  }  if (!this.catalog.get(bookName).getIsFree()) {  return String.format("Книга %s в данный момент занята", bookName);  }  int randomLibrarian = new Random().nextInt(this.librarians.length);  this.librarians[randomLibrarian].makeOrder(reader, this.catalog.get(bookName), bookIssueType);  return String.format("Книга %s выдана: %s", bookName, reader);  }  public String returnBook(Reader reader, String bookName) {  if (!this.readers.containsKey(reader.getId())) {  return String.format("Читатель %s не принадлежит библиотеке", reader);  }  if (!this.catalog.containsKey(bookName)) {  return String.format("Книги %s нет в библиотеке", bookName);  }  if (!reader.haveBook(this.catalog.get(bookName))) {  return String.format("Книги %s нет у %s", bookName, reader);  }  int randomLibrarian = new Random().nextInt(this.librarians.length);  this.librarians[randomLibrarian].returnBook(reader, this.catalog.get(bookName));  if (!reader.booksWithExpiredTimeExists(permissibleNumberOfDaysForStoringBook, permissibleNumberOfHoursOfUsingBook)) {  reader.setIsBanned(false);  return String.format("Книга %s возвращена, %s нет в черном списке", bookName, reader);  }  return String.format("Книга %s возвращена от %s", bookName, reader);  }  public void addReaderToLibrary(Reader reader) {  this.readers.put(reader.getId(), reader);  }  public List<Reader> checkReaders() {  List<Reader> bannedReaders = new ArrayList<>();  for (Reader reader: this.readers.values()) {  if (reader.booksWithExpiredTimeExists(permissibleNumberOfDaysForStoringBook, permissibleNumberOfHoursOfUsingBook)) {  reader.setIsBanned(true);  bannedReaders.add(reader);  }  }  return bannedReaders;  }  } |

|  |
| --- |
| public class Librarian {  private final String lastName;  private final String name;  private final String patronymic;  private final boolean isAdministrator;  public Librarian(String lastName, String name, String patronymic, boolean isAdministrator) {  this.lastName = lastName;  this.name = name;  this.patronymic = patronymic;  this.isAdministrator = isAdministrator;  }  public String toString() {  String result = String.format("Библиотекарь %s %s %s", this.lastName, this.name, this.patronymic);  if (this.isAdministrator) {  result += " (администратор)";  }  return result;  }  public void makeOrder(Reader reader, Book book, BookIssueType bookIssueType) {  reader.makeOrder(book, bookIssueType);  book.setFree(false);  }  public void returnBook(Reader reader, Book book) {  reader.returnBook(book);  book.setFree(true); }} |
| public enum BookIssueType {  READING\_ROOM,  SUBSCRIPTION  }  public class Book {  private final String name;  private final String author;  private boolean isFree = true;  public Book(String name, String author) {  this.name = name;  this.author = author;  }  public String toString() {  return String.format("Книга %s (автор: %s)", this.name, this.author);  }  public void setFree(boolean isFree) {  this.isFree = isFree;  }  public boolean getIsFree() {  return this.isFree;  }  public String getName() {  return this.name;  }  public String getAuthor() {  return this.author;  }  } |

|  |
| --- |
| import java.time.Duration;  import java.time.LocalDateTime;  import java.util.HashMap;  public class Reader {  private final int id;  private final String lastName;  private final String name;  private final String patronymic;  private boolean isBanned;  private HashMap<Book, BookIssueType> booksWithIssueType;  private HashMap<Book, LocalDateTime> booksWithOrderDates;  public Reader(int id, String lastName, String name, String patronymic) {  this.id = id;  this.lastName = lastName;  this.name = name;  this.patronymic = patronymic;  this.booksWithIssueType = new HashMap<>();  this.booksWithOrderDates = new HashMap<>();  this.isBanned = false;  }  public String toString() {  return String.format("Читатель %s %s %s (%d)", this.lastName, this.name, this.patronymic, this.id);  }  public void makeOrder(Book book, BookIssueType bookIssueType) {  this.booksWithIssueType.put(book, bookIssueType);  this.booksWithOrderDates.put(book, LocalDateTime.now());  }  public void returnBook(Book book) {  this.booksWithIssueType.remove(book);  this.booksWithOrderDates.remove(book);  }  public int getId() {  return this.id;  }  public boolean getIsBanned() {  return this.isBanned;  }  public void setIsBanned(boolean isBanned) {  this.isBanned = isBanned;  }  public boolean booksWithExpiredTimeExists(int permissibleNumberOfDaysForStoringBook, int permissibleNumberOfHoursOfUsingBook) {  for (Book book: booksWithOrderDates.keySet()) {  BookIssueType bookIssueType = this.booksWithIssueType.get(book);  if ((bookIssueType == BookIssueType.SUBSCRIPTION) && (Duration.between(LocalDateTime.now(), this.booksWithOrderDates.get(book)).toDays() > permissibleNumberOfDaysForStoringBook)) {  return true;  }  if ((bookIssueType == BookIssueType.READING\_ROOM) && (Duration.between(LocalDateTime.now(), this.booksWithOrderDates.get(book)).toHours() > permissibleNumberOfHoursOfUsingBook)) {  return true;  }  }  return false;  }  public boolean haveBook(Book book) {  return this.booksWithIssueType.containsKey(book) && this.booksWithOrderDates.containsKey(book);  }  } |

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