|  |  |
| --- | --- |
| 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 **09.04.01/05 Современные интеллектуальные**

**программно-аппаратные комплексы**

**ОТЧЕТ**

**по лабораторной работе №3**

|  |  |
| --- | --- |
| Группа | ИУ6-23М |

|  |  |  |  |
| --- | --- | --- | --- |
| Студент |  |  | И.И. Иванов |
|  |  | (Подпись, дата) | (И.О. Фамилия) |
|  |  |  |  |
| Преподаватель |  |  | П.В. Степанов |
|  |  | (Подпись, дата) | (И.О. Фамилия) |

*Москва, 2023 г.*

**Условие**

1.9. Определить класс Квадратное уравнение. Класс должен содержать несколько конструкторов. Реализовать методы для поиска корней, экстремумов, а также интервалов убывания/возрастания. Создать массив объектов и определить наибольшие и наименьшие по значению корни.

1.10. Определить класс Булева матрица (BoolMatrix) размерности (n x m). Класс должен содержать несколько конструкторов. Реализовать методы для логического сложения (дизъюнкции), умножения и инверсии матриц. Реализовать методы для подсчета числа единиц в матрице и упорядочения строк в лексикографическом порядке

2.9. Product: id, Наименование, UPC, Производитель, Цена, Срок хранения, Количество. Создать массив объектов. Вывести: a) список товаров для заданного наименования; b) список товаров для заданного наименования, цена которых не превосходит заданную; c) список товаров, срок хранения которых больше заданного.  
2.10. Product: id, Наименование, UPC, Производитель, Цена, Срок хранения, Количество. Создать массив объектов. Вывести: a) список товаров для заданного наименования; b) список товаров для заданного наименования, цена которых не превосходит заданную; c) список товаров, срок хранения которых больше заданного.

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

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

**Програмная реализация**

1.9.

|  |
| --- |
| public class Main {  private double a;  private double b;  private double c;  public Main(double a, double b, double c) {  this.a = a;  this.b = b;  this.c = c;  }  public Main(double a, double b) {  this(a, b, 0);  }  public Main(double a) {  this(a, 0, 0);  }  public double[] findRoots() {  double[] roots = new double[2];  try {  roots = new double[2];  } catch (OutOfMemoryError e) {  // Обработка исключения при нехватке памяти  System.err.println("Ошибка: недостаточно памяти для выделения массива");  }  double discriminant = b \* b - 4 \* a \* c;  try {  if (discriminant > 0) {  roots[0] = (-b + Math.sqrt(discriminant)) / (2 \* a);  roots[1] = (-b - Math.sqrt(discriminant)) / (2 \* a);  } else if (discriminant == 0) {  roots[0] = -b / (2 \* a);  roots[1] = roots[0];  } else {  throw new Exception("Отрицательный дискриминант");  }  } catch (Exception e) {  // Обработка исключения при отрицательном значении дискриминанта  System.err.println(e.getMessage());  }  return roots;  }  public double findMinimum() {  double vertexX = -b / (2 \* a);  double vertexY = a \* vertexX \* vertexX + b \* vertexX + c;  if (a > 0) {  return vertexY;  } else {  double y1 = a \* Double.NEGATIVE\_INFINITY \* Double.NEGATIVE\_INFINITY + b \* Double.NEGATIVE\_INFINITY + c;  double y2 = a \* Double.POSITIVE\_INFINITY \* Double.POSITIVE\_INFINITY + b \* Double.POSITIVE\_INFINITY + c;  return Math.min(y1, Math.min(y2, vertexY));  }  }  public double findMaximum() {  double vertexX = -b / (2 \* a);  double vertexY = a \* vertexX \* vertexX + b \* vertexX + c;  if (a < 0) {  return vertexY;  } else {  double y1 = a \* Double.NEGATIVE\_INFINITY \* Double.NEGATIVE\_INFINITY + b \* Double.NEGATIVE\_INFINITY + c;  double y2 = a \* Double.POSITIVE\_INFINITY \* Double.POSITIVE\_INFINITY + b \* Double.POSITIVE\_INFINITY + c;  return Math.max(y1, Math.max(y2, vertexY));  }  }  public boolean isIncreasing() {  return a > 0;  }  public boolean isDecreasing() {  return a < 0;  }  public static void main(String[] args) {  Main[] equations = new Main[3];  equations[0] = new Main(1, 2, -3);  equations[1] = new Main(2, -5);  equations[2] = new Main(-3);  double maxRoot = Double.NEGATIVE\_INFINITY;  double minRoot = Double.POSITIVE\_INFINITY;  for (Main equation : equations) {  double[] roots = equation.findRoots();  for (double root : roots) {  if (!Double.isNaN(root)) {  maxRoot = Math.max(maxRoot, root);  minRoot = Math.min(minRoot, root);  }  }  }  System.out.println("Max root: " + maxRoot);  System.out.println("Min root: " + minRoot);  System.out.println("Minimum: " + equations[0].findMinimum());  }  } |

1.10.

|  |
| --- |
| public class Main {  public static void main(String[] args) {  boolean[][] matrix1 = {{true, false}, {false, true}};  boolean[][] matrix2 = {{false, true}, {true, false}};  BoolMatrix boolMatrix1 = new BoolMatrix(matrix1);  BoolMatrix boolMatrix2 = new BoolMatrix(matrix2);  System.out.println("Initial matrices:");  System.out.println(boolMatrix1);  System.out.println(boolMatrix2);  BoolMatrix orResult = boolMatrix1.or(boolMatrix2);  BoolMatrix andResult = boolMatrix1.and(boolMatrix2);  BoolMatrix notResult = boolMatrix1.not();  System.out.println("OR result:");  System.out.println(orResult);  System.out.println("AND result:");  System.out.println(andResult);  System.out.println("NOT result:");  System.out.println(notResult);  int onesCount = boolMatrix1.countOnes();  System.out.println("Number of ones in matrix 1: " + onesCount);  boolMatrix1.sortLexicographically();  System.out.println("Sorted lexicographically matrix 1:");  System.out.println(boolMatrix1);  }  } |

2.9.

|  |
| --- |
| import java.util.ArrayList;  public class Main {  public static void main(String[] args) {  ProductArray products = new ProductArray();  products.addProduct(new Product(1, "Banana", "123456789", "Fruit Corp.", 0.5, 7, 10));  products.addProduct(new Product(2, "Apple", "987654321", "Fruit Corp.", 0.8, 14, 5));  products.addProduct(new Product(3, "Orange", "456789123", "Fruit Corp.", 0.6, 10, 8));  // Список товаров для заданного наименования  ArrayList<Product> result1 = products.getProductsByName("Apple");  for (Product product : result1) {  System.out.println(product.toString());  }  // Список товаров для заданного наименования, цена которых не превосходит заданную  ArrayList<Product> result2 = products.getProductsByPriceAndName("Banana", 1.0);  for (Product product : result2) {  System.out.println(product.toString());  }  // Список товаров, срок хранения которых больше заданного  ArrayList<Product> result3 = products.getProductsByShelfLife(9);  for (Product product : result3) {  System.out.println(product.toString());  }  }  } |

2.10.

|  |
| --- |
| import java.util.ArrayList;  public class Main {  public static void main(String[] args) {  Train[] trains = new Train[10];  trains[0] = new Train("Москва", 123, "10:00", 50, 20, 100, 5);  trains[1] = new Train("Санкт-Петербург", 456, "11:30", 80, 35, 150, 10);  trains[2] = new Train("Казань", 789, "13:15", 60, 25, 120, 8);  trains[3] = new Train("Владивосток", 321, "14:45", 70, 30, 130, 7);  trains[4] = new Train("Екатеринбург", 654, "17:00", 55, 22, 110, 6);  trains[5] = new Train("Ростов-на-Дону", 987, "19:15", 75, 28, 140, 9);  trains[6] = new Train("Нижний Новгород", 246, "21:30", 65, 26, 125, 7);  trains[7] = new Train("Красноярск", 135, "22:45", 40, 18, 90, 3);  trains[8] = new Train("Пермь", 864, "23:30", 45, 20, 100, 4);  trains[9] = new Train("Самара", 975, "01:00", 85, 40, 160, 12);  System.out.println("Список поездов, следующих до Москвы:");  for(Train train : trains) {  if(train.getDestination().equals("Москва")) {  System.out.println(train.toString());  }  }  System.out.println("Список поездов, следующих до Казани и отправляющихся после 13:00:");  for(Train train : trains) {  if(train.getDestination().equals("Казань") && train.getDepartureTime().compareTo("13:00") > 0) {  System.out.println(train.toString());  }  }  System.out.println("Список поездов, отправляющихся до Екатеринбурга и имеющих общие места:");  for(Train train : trains) {  if(train.getDestination().equals("Екатеринбург") && train.getCommonSeats() > 0) {  System.out.println(train.toString());  }  }  }  } |

3.8.

|  |
| --- |
| public class Main {  public static void main(String[] args) {  Piano fata = new Piano();  fata.keys = new Key[3];  fata.keys[0] = new Key("PUM!");  fata.keys[1] = new Key("PAM!");  fata.keys[2] = new Key("POM!");  fata.play();  }  } |

3.9.

|  |
| --- |
| public class Main {  public static void main(String[] args) {  Fotografia foto1 = new Fotografia();  foto1.setNazvanie("Пейзаж");  Fotografia foto2 = new Fotografia();  foto2.setNazvanie("Портрет");  Fotoalbum album = new Fotoalbum();  album.addFoto(foto1);  album.addFoto(foto2);  System.out.println(album.getCount()); // выводит 2  }  } |