|  |  |
| --- | --- |
| 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**

**Отчет**

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

**Название:** Классы. Наследование. Полиморфизм

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

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Студент | ИУ6-21М |  |  | Ю.А. Вишневская |
|  | (Группа) |  | (Подпись, дата) | (И.О. Фамилия) |
|  |  |  |  |  |
| Преподаватель |  |  |  | П. В. Степанов |
|  |  |  | (Подпись, дата) | (И.О. Фамилия) |

Москва, 2020

**Вариант 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.

Square\_equation

|  |
| --- |
| import static java.lang.Math.pow;  import static java.lang.Math.sqrt;  public class Square\_equation {  private double a; // параметр при x^2  private double b; // параметр при x  private double c; // параметр при x^0  public Square\_equation (double a, double b, double c) {  checkQuadratic(a);  this.a = a;  this.b = b;  this.c = c;  }  public Square\_equation (double b, double c) {  this.a = 1;  this.b = b;  this.c = c;  }  public double getA() {  return a;  }  public void setA(double a) {  this.a = a;  }  public double getB() {  return b;  }  public void setB(double b) {  this.b = b;  }  public double getC() {  return c;  }  public void setC(double c) {  this.c = c;  }  public double getDiscriminant() {  return pow(b, 2) - 4 \* a \* c;  }  public double[] getX() {  double d = getDiscriminant();  double[] x;  if (d > 0) {  x = new double[2];  x[0] = (-b + sqrt(d)) / (2 \* a);  x[1] = (-b - sqrt(d)) / (2 \* a);  return x;  } else if (d == 0) {  x = new double[1];  x[0] = -b / (2 \* a);  return x;  } else {  x = new double[0];  }  return x;  }  public double getExtremum() {  return -b / (2 \* a);  }  public double[] getDecreasingInterval() {  double[] interval = new double[2];  if (a > 0) {  interval[0] = Double.NEGATIVE\_INFINITY;  interval[1] = getExtremum();  } else {  interval[0] = getExtremum();  interval[1] = Double.POSITIVE\_INFINITY;  }  return interval;  }  public double[] getIncreasingInterval() {  double[] interval = new double[2];  if (a > 0) {  interval[0] = getExtremum();  interval[1] = Double.POSITIVE\_INFINITY;  } else {  interval[0] = Double.NEGATIVE\_INFINITY;  interval[1] = getExtremum();  }  return interval;  }  public static void checkQuadratic(double a) {  if (a == 0) {  System.out.println("Это уравнение не является квадратным!");  }  }  } |

|  |
| --- |
| import java.util.ArrayList;  import java.util.Arrays;  import java.util.List;  public class LR3\_V1\_9 {  public static void main(String[] args) {  List<Square\_equation> quadratics = new ArrayList<>();  quadratics.add(new Square\_equation(2., 1., 5.));  quadratics.add(new Square\_equation(1., 10.5, 3.));  quadratics.add(new Square\_equation(2.5, 20., 3.));  quadratics.add(new Square\_equation(2, 3));  quadratics.add(new Square\_equation(0, -5.));  for (Square\_equation quadratic : quadratics) {  System.out.println(quadratic);  System.out.print("Discriminant = "); // Дискриминант  System.out.println(quadratic.getDiscriminant());  System.out.print("Root(s) = "); // Корни  System.out.println(Arrays.toString(quadratic.getX()));  System.out.print("Decreasing interval = "); // Интервалы возрастания и убывания  System.out.println(Arrays.toString(quadratic.getDecreasingInterval()));  System.out.print("Increasing interval = ");  System.out.println(Arrays.toString(quadratic.getIncreasingInterval()));  System.out.println();  }  }  } |

**Вариант 2**

Создать классы, спецификации которых приведены ниже. Определить конструкторы и методы setТип(), getТип(), toString(). Определить дополнительно методы в классе, создающем массив объектов. Задать критерий выбора данных и вывести эти данные на консоль.

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

Student

|  |
| --- |
| public class Student {  private int id; // идентификатор  private String surname; // фамилия  private String name; // имя  private String patronymic; // отчество  private String birthday; // дата рождения  private String address; // адрес  private String phone; // телефон  private String faculty; // факультет  private int course; // курс  private String group; // группа  public Student(){  }  public Student(int id, String surname, String name, String patronymic, String birthday, String address, String phone, String faculty, int course, String group) {  this.id = id;  this.surname = surname;  this.name = name;  this.patronymic = patronymic;  this.birthday = birthday;  this.address = address;  this.phone = phone;  this.faculty = faculty;  this.course = course;  this.group = group;  }  public Student(int id, String surname, String name, String birthday, String address, String phone, String faculty,  int course, String group){  this.id = id;  this.surname = surname;  this.name = name;  this.birthday = birthday;  this.address = address;  this.phone = phone;  this.faculty = faculty;  this.course = course;  this.group = group;  }  // get'ры  public int getId() {  return id;  }  public String getSurname() {  return surname;  }  public String getName() {  return name;  }  public String getPatronymic() {  return patronymic;  }  public String getBirthday() {  return birthday;  }  public String getAddress() {  return address;  }  public String getPhone() {  return phone;  }  public String getFaculty() {  return faculty;  }  public int getCourse() {  return course;  }  public String getGroup() {  return group;  }  // set'ры  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 setPatronymic(String patronymic) {  this.patronymic = patronymic;  }  public void setBirthday(String birthday) {  this.birthday = birthday;  }  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(String group) {  this.group = group;  }  @Override  public String toString() {  return "Student{" +  "id=" + id +  ", surname='" + surname + '\'' +  ", name='" + name + '\'' +  ", patronymic='" + patronymic + '\'' +  ", birthday='" + birthday + '\'' +  ", address='" + address + '\'' +  ", phone='" + phone + '\'' +  ", faculty='" + faculty + '\'' +  ", course=" + course +  ", group='" + group + '\'' +  '}';  }  public void listFaculty(String faculty){  if(this.faculty == faculty){  System.out.println(getId() + " " + getSurname() + " " + getName() + " " + getPatronymic());  }  }  public void listCourse(String faculty, int course){  if(this.faculty == faculty && this.course == course){  System.out.println(getId() + " " + getSurname() + " " + getName() + " " + getPatronymic() + " " + getCourse() + " " + getGroup());  }  }  public void age(int years){  String[] strBirthday = getBirthday().split("\\.");  //System.out.print(strBirthday[1] + " " + strBirthday[2]);  int year = Integer.parseInt(strBirthday[2]);  if(year >= years){  System.out.println(getId() + " " + getSurname() + " " + getName() + " " + getPatronymic() + " " + getCourse() + " " + getGroup());  }  }  public void listGroup(String group){  if(this.group == group){  System.out.println(getId() + " " + getSurname() + " " + getName() + " " + getPatronymic() + " " + getCourse() + " " + getGroup());  }  }  } |

|  |
| --- |
| public class LR3\_V2\_1 {  public static void main(String[] args) {  Student ivanov = new Student(1, "Ivanov", "Ivan", "Ivanovich", "01.02.1996", "Moscow, Russia", "8(916)555-55-55", "IU", 2, "IU6-42");  Student petrov = new Student(2, "Petrov", "Petr", "Petrovich", "02.03.1997", "Moscow, Russia","8(916)666-66-66", "IU", 2, "IU6-42");  Student sidorov = new Student(3, "Sidorov", "Sergey", "Sergeevich", "03.04.1998", "Tula, Russia","8(916)777-77-77", "IBM", 1, "IBM2-11");  Student alexeeva = new Student(4, "Alexeeva", "Alexandra", "Alexandrovna", "04.05.1999", "Omsk, Russia","8(916)888-88-88", "IU", 1, "IU2-11");  Student[] students = {ivanov, petrov, sidorov, alexeeva};  //список студентов заданного факультета  System.out.println("Список студентов факультета IU: ");  for(Student student : students){  student.listFaculty("IU");  }  System.out.println();  //список студентов заданного курса и факультета  System.out.println("Список студентов 2 курса факультета IU: ");  for(Student student : students){  student.listCourse("IU", 2);  }  System.out.println();  //список студентов, родившихся после заданного года  System.out.println("Список студентов, родившихся после 1997 года: ");  for(Student student : students){  student.age(1997);  }  System.out.println();  //) список учебной группы  System.out.println("Список студентов группы IU6-42: ");  for(Student student : students){  student.listGroup("IU6-42");  }  }  } |

**Вариант 3**

Создать приложение, удовлетворяющее требованиям, приведенным в задании. Аргументировать принадлежность классу каждого создаваемого метода и корректно переопределить для каждого класса методы equals(), hashCode(), toString().

7. Создать объект класса Дерево, используя классы Лист. Методы: зацвести, опасть листьям, покрыться инеем, пожелтеть листьям.

Tree

|  |
| --- |
| public class Tree {  private String typeTree = "Тип дерева не указан";  private String bloomsTree = "Дерево не цветет";  private String frostTree = "Дерево не покрыто инеем";  public Leaf leafs = new Leaf();  public Tree(String typeTree, String bloomsTree, String frostTree) {  this.typeTree = typeTree;  this.bloomsTree = bloomsTree;  this.frostTree = frostTree;  }  public Tree() {  }  // set'ры  public void setTypeTree(String typeTree) {  this.typeTree = typeTree;  }  public void setBloomsTree(String bloomsTree) {  this.bloomsTree = bloomsTree;  }  public void setFrostTree(String frostTree) {  this.frostTree = frostTree;  }  // get'ры  public String getTypeTree() {  return typeTree;  }  public String getBloomsTree() {  return bloomsTree;  }  public String getFrostTree() {  return frostTree;  }  // Метод "Зацвести"  public String bloom()  {  return Tree.this.bloomsTree = "Дерево цветет";  }  // Метод "Покрыться инеем"  public String frost()  {  return Tree.this.frostTree = "Дерево покрылось инеем";  }  @Override  public String toString() {  return "Tree{" +  "typeTree='" + typeTree + '\'' +  ", bloomsTree='" + bloomsTree + '\'' +  ", frostTree='" + frostTree + '\'' +  '}';  }  } |

Leaf

|  |
| --- |
| import java.util.Objects;  public class Leaf {  public String leaf = "Лист на дереве";  public String color = "Зеленый";  public Leaf(String leaf, String color) {  this.leaf = leaf;  this.color = color;  }  public Leaf() {  }  // get'ры  public String getLeaf() {  return leaf;  }  public String getColor() {  return color;  }  // set'ры  public void setLeaf(String leaf) {  this.leaf = leaf;  }  public void setColor(String color) {  this.color = color;  }  // Метод "Опасть листьям"  public String fallLeaf() {  return Leaf.this.leaf = "Листья опали";  }  // Метод "Пожелтеть листьям"  public String yellowLeaf() {  return Leaf.this.color = "Желтый";  }  @Override  public boolean equals(Object o) {  if (this == o) return true;  if (o == null || getClass() != o.getClass()) return false;  Leaf leaf1 = (Leaf) o;  return Objects.equals(leaf, leaf1.leaf) &&  Objects.equals(color, leaf1.color);  }  @Override  public int hashCode() {  return Objects.hash(leaf, color);  }  @Override  public String toString() {  return "Leaf{" +  "leaf='" + leaf + '\'' +  ", color='" + color + '\'' +  '}';  }  } |

|  |
| --- |
| public class LR3\_V3\_7 {  public static void main(String[] args)  {  Tree lipa = new Tree("Липа", "Липа не цветет", "Липа не замерзла");  Tree klen = new Tree("Клен", "Клен не цветет", "Клен не замерз");  System.out.println("Тип первого дерева: "+ lipa.getTypeTree());  System.out.println("Тип второго дерева: "+ klen.getTypeTree());  System.out.println("Зацвести: " + lipa.bloom());  System.out.println("Покрыться инеем: " + lipa.frost());  System.out.println("Опасть листьям: " + lipa.leafs.fallLeaf());  System.out.println("Пожелтеть листьям: Цвет листьев -" + lipa.leafs.yellowLeaf());  System.out.println(lipa.equals(klen));  System.out.println(lipa.leafs.toString());  System.out.println(klen.toString());  }  } |

**Вариант 4**

Построить модель программной системы.

9. Система Интернет-магазин. Администратор добавляет информацию о Товаре. Клиент делает и оплачивает Заказ на Товары. Администратор регистрирует Продажу и может занести неплательщиков в «черный список».

Administrator

|  |
| --- |
| import java.util.ArrayList;  public class Administrator {  private ArrayList<Product> products = new ArrayList<Product>();  private ArrayList<Client> blackСlients = new ArrayList<Client>();  // регистрация товара и добавление неплательщиков в черный список  public void registerOrder(Client client) {  if (client.getOrder().isPayment()) {  client.getOrder().setRegister(true);  } else {  blackСlients.add(client);  }  }  // создание новых продуктов  public void createNewProduct(String name, int price) {  products.add(new Product(name,price));  }  } |

Client

|  |
| --- |
| public class Client {  private Order order; // Заказ  public Client(Order order) {  this.order = order;  }  public Client() {  }  // сделать Заказ  public void basket(Order order) {  this.order = order;  }  // get'ры  public Order getOrder() {  return order;  }  // set'ры  public void setOrder(Order order) {  this.order = order;  }  // демонстрация Заказа  public void showOrderCl() {  System.out.println("Вы заказали: ");  order.showOrder();  }  // оплата Заказа  public void pay() {  if(order.isPayment()) {  System.out.println("Вы уже оплатили заказ");  } else {  order.setPayment(true);  }  }  // забрать Заказ  public void take() {  if(!order.isPayment()) {  System.out.println("Оплатите, пожалуйста, заказ");  } else if(!order.isRegister()) {  System.out.println("Ваша заявка, к сожалению, не обработана Администратором");  } else {  System.out.println("Спасибо за покупку! Ждем Вас ещё!");  }  }  } |

Order

|  |
| --- |
| import java.util.ArrayList;  public class Order {  private ArrayList<Product> listOrder = new ArrayList<Product>(); // список товаров в заказе  private boolean paymentOrder = false; // оплата заказа  private boolean registerOrder = false; // обработка заказа Администратором  public Order(ArrayList<Product> listOrder, boolean paymentOrder, boolean registerOrder) {  this.listOrder = listOrder;  this.paymentOrder = paymentOrder;  this.registerOrder = registerOrder;  }  public Order() {  }  // get'ры  public ArrayList<Product> getListOrder() {  return listOrder;  }  public boolean isPaymentOrder() {  return paymentOrder;  }  public boolean isRegisterOrder() {  return registerOrder;  }  // set'ры  public void setListOrder(ArrayList<Product> listOrder) {  this.listOrder = listOrder;  }  public void setPaymentOrder(boolean paymentOrder) {  this.paymentOrder = paymentOrder;  }  public void setRegisterOrder(boolean registerOrder) {  this.registerOrder = registerOrder;  }  // добавление списка товаров в заказ  public void addProdToOrder(ArrayList<Product> listOrder) {  this.listOrder = listOrder;  }  // добавление товара в заказ  public void addProdToOrder(Product product) {  listOrder.add(product);  }  // вывести заказ  public void showOrder() {  for (Product elem: listOrder) {  System.out.println (elem);  }  }  // проверка оплаты  public boolean isPayment() {  return paymentOrder;  }  // оплачен ли? Ответы: да/нет  public void setPayment(boolean state) {  paymentOrder = state;  }  // проверка обработки Администратором  public boolean isRegister() {  return registerOrder;  }  // обработан ли? Ответы: да/нет  public void setRegister(boolean state) {  registerOrder = state;  }  } |

Product

|  |
| --- |
| public class Product {  private String nameProduct; // Имя товара  private int priceProduct; // Стоимость товара  public Product(String nameProduct, int priceProduct) {  this.nameProduct = nameProduct;  this.priceProduct = priceProduct;  }  public Product() {  }  // get'ры  public String getNameProduct() {  return nameProduct;  }  public int getPriceProduct() {  return priceProduct;  }  // set'ры  public void setNameProduct(String nameProduct) {  this.nameProduct = nameProduct;  }  public void setPriceProduct(int priceProduct) {  this.priceProduct = priceProduct;  }  @Override  public String toString() {  return "Product{" +  "nameProduct='" + nameProduct + '\'' +  ", priceProduct=" + priceProduct +  '}';  }  } |

|  |
| --- |
| public class LR3\_V4\_9 {  public static void main(String[] args)  {  Administrator admin = new Administrator();  Product bulochka = new Product("Булочка", 150);  Order order = new Order();  Client client = new Client();  admin.createNewProduct("Булочка", 150);  order.addProdToOrder(bulochka);  client.basket(order);  client.showOrderCl();  System.out.println(bulochka.getNameProduct());  }  } |