|  |  |
| --- | --- |
|  | **Министерство образования и науки 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

НАПРАВЛЕНИЕ ПОДГОТОВКИ: 09.04.01 Информатика и вычислительная техника

**О т ч е т**

**по лабораторной работе №3**

**Дисциплина: Языки программирование при работе с большими данными**

**Название: Наследование и полиморфизм**

студент группы ИУ6-21М  **\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ Коваленко М.В.**

(Подпись, дата) (Фамилия И.О.)

Преподаватель **\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ Степанов П.В.**

(Подпись, дата) (Фамилия И.О.)

Москва, 2020

**Задание 1**

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.

**Программа**

**Класс Square\_equation**

|  |
| --- |
| import static java.lang.Math.pow;  import static java.lang.Math.sqrt;  public class Square\_equation {  private double a;  private double b;  private double c;  public Square\_equation(double a, double b, double c) {  checkQuadratic(a);  this.a = a;  this.b = b;  this.c = c;  }  public Square\_equation(double a, double b) {  this.a = a;  this.b = b;  }  public static void checkQuadratic(double a) {  if (a == 0) {  System.out.println("Введите квадратное уравнение");  }  }  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);  }  // Интервал убывания  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;  }  // Интервал возрастания  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;  }  } |

**Основной файл**

|  |
| --- |
| import java.util.ArrayList;  import java.util.Arrays;  import java.util.List;  // + класс Square\_equation  public class Lab3\_1 {  public static void main(String[] args) {  List<Square\_equation> quadratics = new ArrayList<>();  quadratics.add(new Square\_equation(3., 2., 5.));  quadratics.add(new Square\_equation(1.5, 8.5, 3.));  quadratics.add(new Square\_equation(2.5, 10., 2.));  quadratics.add(new Square\_equation(1., 0, 0));  quadratics.add(new Square\_equation(1., 0));  for (Square\_equation quadratic : quadratics) {  System.out.print("Discriminant = ");  System.out.println(quadratic.getDiscriminant());  System.out.print("Root(s) = ");  System.out.println(Arrays.toString(quadratic.getX()));  System.out.print("Decreasing interval = ");  System.out.println(Arrays.toString(quadratic.getDecreasingInterval()));  System.out.print("Increasing interval = ");  System.out.println(Arrays.toString(quadratic.getIncreasingInterval()));  System.out.println();  }  }  } |

**Задание 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.

Book: id, Название, Автор(ы), Издательство, Год издания, Количество страниц, Цена, Переплет. Создать массив объектов. Вывести: a) список книг заданного автора; b) список книг, выпущенных заданным издательством; c) список книг, выпущенных после заданного года.

**Программа**

**Основной файл**

|  |
| --- |
| import java.util.ArrayList;  import java.util.List;  // + класс Book  public class Lab3\_2 {  public static void main(String[] args) {  String[] authorsBook1 = {"Author1"};  Book book1 = new Book("nameBook1", authorsBook1, "publisher1", 2019, 300, 500, "hard");  String[] authorBook2 = {"Author2"};  Book book2 = new Book("nameBook2", authorBook2, "publisher2", 2018, 100, 1000, "limp");  String[] authorBook3 = {"Author3", "Author4", "Author2"};  Book book3 = new Book("nameBook3", authorBook3, "publisher2", 2000, 400, 100, "limp");  Book[] books = {book1, book2, book3};  System.out.println("Book by Author2: " + booksByAuthor("Author2", books));  System.out.println("Book by publisher1: " + booksByPublisher("publisher1", books));  System.out.println("Book after 2018: " + booksAfterYear(2018, books));  System.out.println("Book with limp binding type: " + booksWithBindingType("limp", books));  }  public static List<Book> booksByAuthor(String author, Book[] books) {  List<Book> result = new ArrayList<>();  for (Book book : books) {  for (String authorBook : book.getAuthors()) {  if (author.equals(authorBook)) {  result.add(book);  }  }  }  return result;  }  public static List<Book> booksByPublisher(String publisher, Book[] books) {  List<Book> result = new ArrayList<>();  for (Book book : books) {  if (publisher.equals(book.getPublisher())) {  result.add(book);  }  }  return result;  }  public static List<Book> booksAfterYear(int year, Book[] books) {  List<Book> result = new ArrayList<>();  for (Book book : books) {  if (year < book.getYear()) {  result.add(book);  }  }  return result;  }  public static List<Book> booksWithBindingType(String bindingType, Book[] books) {  List<Book> result = new ArrayList<>();  for (Book book : books) {  if (bindingType.equals(book.getBindingType())) {  result.add(book);  }  }  return result;  }  } |

**Класс Book**

|  |
| --- |
| import java.util.Arrays;  public class Book {  private String name;  private String[] authors;  private String publisher;  private int year;  private int pages;  private int price;  private String bindingType;  public Book(String name, String[] authors, String publisher, int year, int pages, int price, String bindingType) {  this.name = name;  this.authors = authors;  this.publisher = publisher;  this.year = year;  this.pages = pages;  this.price = price;  this.bindingType = bindingType;  }  public int getPrice() {  return price;  }  public void setPrice(int price) {  this.price = price;  }  public String getName() {  return name;  }  public String[] getAuthors() {  return (String[]) authors.clone();  }  public String getPublisher() {  return publisher;  }  public int getYear() {  return year;  }  public int getPages() {  return pages;  }  public String getBindingType() {  return bindingType;  }  @Override  public String toString() {  return "Book{" +  "name='" + name + '\'' +  ", authors=" + Arrays.toString(authors) +  ", publisher='" + publisher + '\'' +  ", year=" + year +  ", pages=" + pages +  ", price=" + price +  ", bindingType='" + bindingType + '\'' +  '}';  }  } |

**Задание 3**

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 Пианино, используя класс Клавиша. Методы: настроить, играть на пианино, нажимать клавишу.

**Программа**

**Основной файл**

|  |
| --- |
| public class Lab3\_3 {  public static void main(String[] args)  {  Piano piano1 = new Piano("Model1", "Note1", "Расстроено");  Piano piano2 = new Piano("Model2", "Note2", "Настроено");  System.out.println("Модель пианино1: "+ piano1.getPianoname());  System.out.println("Модель пианино2: "+ piano2.getPianoname());  System.out.println("Настроить: " + piano2.configure());  System.out.println("Нажать клавишу: " + piano1.k.pressKey());  System.out.println(piano1.equals(piano2));  System.out.println(piano1.k.toString());  System.out.println(piano2.toString());  }  } |

**Класс Piano**

|  |
| --- |
| public class Piano {  private String model = "Модель пианино не указана";  private String note = "Клавиши не нажаты";  private String settings = "Пианино не настроено";  public Keys k = new Keys();  Piano(String model,String note, String settings)  {  this.model = model;  this.note = note;  this.settings = settings;  }  public void setPianoname(String model)  {  model = model;  }  public String getPianoname()  {  return model;  }  // Метод "Настроить"  public String configure()  {  return Piano.this.settings="Пианино настроено";  }  // Метод "Играть на пианино"  public String playPiano()  {  return Piano.this.note="Пианино играет";  }  @Override  public String toString() {  return "Piano{" +  "model='" + model + '\'' +  ", note='" + note + '\'' +  ", settings='" + settings + '\'' +  ", k=" + k +  '}';  }  } |

**Класс Keys**

|  |
| --- |
| import java.util.Objects;  public class Keys {  public String key = "Клавиша";  public String pressKey()  {  return Keys.this.key="Клавиша нажата";  }  @Override  public boolean equals(Object o) {  if (this == o) return true;  if (o == null || getClass() != o.getClass()) return false;  Keys keys = (Keys) o;  return Objects.equals(key, keys.key);  }  @Override  public int hashCode() {  return Objects.hash(key);  }  @Override  public String toString() {  return "Keys{" +  "key='" + key + '\'' +  '}';  }  } |

**Задание 4**

Построить модель программной системы.

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».

**Программа**

**Основной файл**

|  |
| --- |
| public class Lab3\_4 {  public static void main(String[] args)  {  Administrator admin = new Administrator();  Product product = new Product("Product1", 350);  Order order = new Order();  Client client = new Client();  admin.createNewProduct("Product1", 350);  order.addProdToOrder(product);  order.showOrder();  System.out.println(product.getName());  }  } |

**Класс Product**

|  |
| --- |
| public class Product {  private String name;  private int price;  public Product (String name, int price) {  this.name = name;  this.price = price;  }  public String getName() {  return name;  }  public int getPrice() {  return price;  }  public void setName(String name) {  this.name = name;  }  public void setPrice (int price) {  this.price = price;  }  @Override  public String toString() {  return "Название = "+name+" Стоимость = "+price;  }  } |

**Класс Order**

|  |
| --- |
| import java.util.ArrayList;  public class Order {  private ArrayList<Product> orderlist = new ArrayList<Product>(); //продукты в заказе  private boolean payment = false; //оплачен ли заказ  private boolean register = false; //обработан ли заказ Администратором  //добавить товар к Заказу  public void addProdToOrder(Product product) {  orderlist.add(product);  }  //показать Заказ  public void showOrder() {  for (Product e: orderlist) {  System.out.println (e);  }  }  //оплачен или нет  public boolean isPayment() {  return payment;  }  //оплачен (да/нет)  public void setPayment(boolean s) {  payment = s;  }  public boolean isRegister() {  return register;  }  public void setRegister(boolean s) {  register = s;  }  } |

**Класс Client**

|  |
| --- |
| public class Client {  private Order order;  //заказать  public void book(Order order) {  this.order = order;  }  public Order getOrder() {  return order;  }  //показать Заказ  public void showOrder() {  System.out.println("Ваш заказ: ");  order.showOrder();  }  //оплатить Заказ  public void pay() {  if(order.isPayment() == true) {  System.out.println("Вы уже оплатили заказ");  } else {  order.setPayment(true);  }  }  //Забрать товар  public void take() {  if(order.isPayment() == false) {  System.out.println("Вы ещё не оплатили товар");  } else if(order.isRegister() == false) {  System.out.println("Ваша заявка ещё не обработана");  } else {  System.out.println("Спасибо за покупку!");  }  }  } |

**Вывод**

В результате выполнения данной лабораторной работы разобралась с наследованием и полиморфизмом на языке Java.