МИНИСТЕРСТВО ОБРАЗОВАНИЯ РЕСПУБЛИКИ БЕЛАРУСЬ
УЧРЕЖДЕНИЕ ОБРАЗОВАНИЯ
“БРЕСТСКИЙ ГОСУДАРСТВЕННЫЙ ТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ”
КАФЕДРА ИНТЕЛЛЕКТУАЛЬНЫХ ИНФОРМАЦИОННЫХ ТЕХНОЛОГИЙ

ОТЧЁТ
по лабораторной работе №4

Выполнила
студентка группы ПО-9
Тупик Ю. Л.

Проверил:
Крощенко А. А.

Брест 2024

Цель работы: приобрести практические навыки в области объектно-ориентированного проектирования.

Задание 1
Реализовать указанный класс, включив в него вспомогательный внутренний класс или классы. Реализовать 2-3 метода (на выбор). Продемонстрировать использование реализованных классов.
8) Создать класс CD (mp3-диск) с внутренним классом, с помощью объектов которого можно хранить информацию о каталогах, подкаталогах и записях.

Выполнение задания

Код программы

```
import java.util.ArrayList;
import java.util.List;

public class CD {
    // Список каталогов на диске CD
    private List<Directory> directories;

    // Конструктор класса CD, инициализирует список каталогов
    public CD() {
        this.directories = new ArrayList<>();
    }

    // Внутренний класс, представляющий каталог на диске CD
    private class Directory {
        private String name;
        private List<String> files;

        // Конструктор класса Directory, инициализирует имя каталога и
список файлов
        public Directory(String name) {
            this.name = name;
            this.files = new ArrayList<>();
        }

        // Метод для добавления файла в каталог
        public void addFile(String file) {
            this.files.add(file);
        }

        // Метод для вывода содержимого каталога на экран
        public void printDirectory() {
            System.out.println("Directory: " + name);
            for (String file : files) {
                System.out.println("- " + file);
            }
        }
    }

    // Метод для добавления нового каталога на диск
    public void addDirectory(String name) {
        directories.add(new Directory(name));
```

```
    }

    // Метод для добавления файла в указанный каталог
    public void addFileToDirectory(String directoryName, String fileName) {
        // Поиск каталога по имени
        for (Directory directory : directories) {
            if (directory.name.equals(directoryName)) {
                directory.addFile(fileName);
                return;
            }
        }
        System.out.println("Directory " + directoryName + " not found!");
    }

    // Метод для вывода содержимого диска на экран
    public void printCD() {
        System.out.println("CD:");
        for (Directory directory : directories) {
            directory.printDirectory();
        }
    }

    public static void main(String[] args) {
        CD cd = new CD();

        cd.addDirectory("Rihanna");
        cd.addFileToDirectory("Rihanna", "diamond.mp3");
        cd.addFileToDirectory("Rihanna", "umbrella.mp3");
        cd.addFileToDirectory("Rihanna", "work.mp3");


        cd.addDirectory("Ariana Grande");
        cd.addFileToDirectory("Ariana Grande", "7_rings.mp3");
        cd.addFileToDirectory("Ariana Grande", "dangerous_woman.mp3");

        cd.printCD();
    }
}
```

Результат

```
C:\Users\Юлия\.jdks\corretto-17.0.9\bin
CD:
Directory: Rihanna
- diamond.mp3
- umbrella.mp3
- work.mp3
Directory: Ariana Grande
- 7_rings.mp3
- dangerous_woman.mp3

Process finished with exit code 0
```

Задание 2
Реализовать агрегирование. При создании класса агрегируемый класс объявляется как атрибут (локальная переменная, параметр метода). Включить в каждый класс 2-3 метода на выбор. Продемонстрировать использование разработанных классов.
8) Создать класс Текст, используя класс Абзац.

Выполнение задания

Код программы

```
import java.util.ArrayList;
import java.util.List;

class Paragraph {
    // Содержание абзаца
    private String content;

    // Конструктор для инициализации содержания абзаца
    public Paragraph(String content) {
        this.content = content;
    }

    // Метод для получения содержания абзаца
    public String getContent() {
        return content;
    }

    //Метод для установки содержания абзаца
    public void setContent(String content) {
        this.content = content;
    }

    // Метод для подсчета количества слов в абзаце
    public int countWords() {
        String[] words = content.split("\\s+");
        return words.length;
    }
}

class Text {
    //Список абзацев
    private List<Paragraph> paragraphs;

    //Конструктор для инициализации списка абзацев
    public Text() {
        this.paragraphs = new ArrayList<>();
    }

    // Метод для добавления абзаца в текст
    public void addParagraph(Paragraph paragraph) {
        paragraphs.add(paragraph);
    }

    // Метод для удаления абзаца из текста
    public void removeParagraph(Paragraph paragraph) {
```

```
        paragraphs.remove(paragraph);
    }

    // Метод для подсчета количества абзацев в тексте
    public int countParagraphs() {
        return paragraphs.size();
    }

    // Метод для подсчета общего количества слов в тексте
    public int countWords() {
        int totalWords = 0;
        for (Paragraph paragraph : paragraphs) {
            totalWords += paragraph.countWords();
        }
        return totalWords;
    }

    // Метод для отображения содержимого текста
    public void display() {
        for (Paragraph paragraph : paragraphs) {
            System.out.println(paragraph.getContent());
        }
    }
}

public class Task2 {
    public static void main(String[] args) {
        Paragraph paragraph1 = new Paragraph("Я помню чудное мгновенье:");
        Paragraph paragraph2 = new Paragraph("Передо мной явилась ты,");
        Paragraph paragraph3 = new Paragraph("Как мимолетное виденье,");
        Paragraph paragraph4 = new Paragraph("Как гений чистой красоты.");

        Text text = new Text();
        text.addParagraph(paragraph1);
        text.addParagraph(paragraph2);
        text.addParagraph(paragraph3);
        text.addParagraph(paragraph4);

        System.out.println("Количество абзацев: " +
text.countParagraphs());
        System.out.println("Количество слов: " + text.countWords());

        System.out.println("Содержимое текста:");
        text.display();
    }
}
```

Результат

```
C:\Users\Юлия\.jdks\corretto-17.0.9\bi
Количество абзацев: 4
Количество слов: 15
Содержимое текста:
Я помню чудное мгновенье:
Передо мной явилась ты,
Как мимолетное виденье,
Как гений чистой красоты.

Process finished with exit code 0
```

Задание 3

Построить модель программной системы с применением отношений (обобщения, агрегации, ассоциации, реализации) между классами. Задать атрибуты и методы классов. Реализовать (если необходимо) дополнительные классы. Продемонстрировать работу разработанной системы.

8) 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».

Выполнение задания

Код программы

```
import java.util.ArrayList;
import java.util.List;

class Product {
    private String name;
    private double price;
    private String description;

    // Конструктор для создания нового товара с заданным  названием и ценой
    public Product(String name, double price) {
        this.name = name;
        this.price = price;
    }

    // Геттер для получения названия товара
    public String getName() {
        return name;
    }

    // Геттер для получения цены товара
    public double getPrice() {
        return price;
    }

    // Геттер для получения описания товара
    public String getDescription() {
        return description;
    }
```

```
    // Сеттер для задания описания товара
    public void setDescription(String description) {
        this.description = description;
    }
}

class Cart {
    private List<Product> products; // Список товаров в корзине

    // Конструктор для создания новой корзины
    public Cart() {
        products = new ArrayList<>();
    }

    // Метод для добавления товара в корзину
    public void addProduct(Product product) {
        products.add(product);
    }

    // Метод для вычисления общей стоимости товаров в корзине
    public double calculateTotalPrice() {
        double total = 0;
        for (Product product : products) {
            total += product.getPrice();
        }
        return total;
    }
}

class Administrator {
    static private List<String> paymentRegistry = new ArrayList<>(); //
Реестр оплат
    static private List<String> blacklist = new ArrayList<>(); // "Черный
список" клиентов

    // Метод для регистрации оплаты товара клиентом
    public static void registerPayment(String clientName, double amount) {
        paymentRegistry.add("Клиент: " + clientName + "; Сумма: $" +
amount);
        System.out.println("Клиент: " + clientName + "; Сумма: $" +
amount);
    }

    // Метод для добавления клиента в "черный список"
    public static void addToBlacklist(String clientName) {
        blacklist.add(clientName);
    }

    // Метод для проверки, находится ли клиент в "черном списке"
    public static boolean isBlacklisted(String clientName) {
        return blacklist.contains(clientName);
    }
```

```
    // Метод для добавления описания товара
    public static void addProductDescription(Product product, String
description) {
        product.setDescription(description);
    }

}

class Client {
    private String name;
    private double money;
    private Cart cart;

    // Конструктор для создания нового клиента с заданным именем и
денежными средствами
    public Client(String name, double money) {
        this.name = name;
        this.money = money;
        this.cart = new Cart();
    }

    // Метод для добавления товара в корзину клиента
    public void addProductToCart(Product product) {
        cart.addProduct(product);
    }

    // Метод для оплаты товаров в корзине
    public void pay() {
        double totalPrice = cart.calculateTotalPrice();
        if (totalPrice <= money) {
            money -= totalPrice;
            Administrator.registerPayment(name, totalPrice);
        } else {
            Administrator.addToBlacklist(name);
            System.out.println("Клиент " + name + " не имеет достаточно
денежных средств " +
                    "и будет добавлена в \"Черный список\".");
        }
    }
}

public class Task3{
    public static void main(String[] args) {
        Client client1 = new Client("Даша", 40);
        Client client2 = new Client("Оля", 20);

        Product product1 = new Product("Тушь для ресниц", 27);
        Product product2 = new Product("Помада для губ", 13);

        Administrator.addProductDescription(product1,
                "Супер объем со сценическим эфффектом, тон 01");
        Administrator.addProductDescription(product2,
                "Жидкая стойкая матовая помада для губ, тон 15");
```

```
        client1.addProductToCart(product1);
        client1.addProductToCart(product2);

        client2.addProductToCart(product1);

        client1.pay(); // Даша имеет достаточно денег для оплаты товаров
        client2.pay(); // Оле не имеет достаточно денег и будет добавлена в
"черный список"

        // Проверка нахождения Оли в "черном списке"
        System.out.println("Оля есть в \"Черном списке\"? " +
                Administrator.isBlacklisted("Оля"));

        // Получение описания товаров
        System.out.println(product1.getName() + " (" +
product1.getDescription() + ")");
        System.out.println(product2.getName() + " (" +
product2.getDescription() + ")");
    }
}
```

Результат

```
C:\Users\Юлия\.jdks\corretto-17.0.9\bin\java.exe "-javaagent:C:\Program Files\JetBrai
Клиент: Даша; Сумма: 40.0руб
Клиент Оля не имеет достаточно денежных средств и будет добавлена в "Черный список".
Оля есть в "Черном списке"? true
Тушь для ресниц (Супер объем со сценическим эфффектом, тон 01)
Помада для губ (Жидкая стойкая матовая помада для губ, тон 15)

Process finished with exit code 0
```

Вывод: приобрела практические навыки в области объектно-ориентированного проектирования.