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

ОТЧЁТ

по лабораторной работе №4

Выполнила:
студентка группы ПО-9
Кот А. А.

Проверил:
Крощенко А. А.

Брест 2024

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

Вариант 8.
Ход работы

Задание 1.

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

Работа программы:

```
Directory Music not found!
CD Contents:
Directory: The Pretty Reckless
- justtonight.mp3
- you.mp3
Directory: Muse
- hysteria.mp3

Process finished with exit code 0
```

Код программы:

CD.java

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

public class CD {
    private List<Directory> directories;

    public CD() {
        this.directories = new ArrayList<>();
    }

    private class Directory {
        private String name;
        private List<String> files;

        public Directory(String name) {
            this.name = name;
            this.files = new ArrayList<>();
        }

        public void addFile(String file) {
            this.files.add(file);
        }

        public void printDirectory() {
            System.out.println("Directory: " + name);
            for (String file : files) {
```

```
                System.out.println("- " + file);
            }
        }
    }

    public void addDirectory(String name) {
        directories.add(new Directory(name));
    }

    public void addFileToDirectory(String directoryName, String fileName) {
        for (Directory directory : directories) {
            if (directory.name.equals(directoryName)) {
                directory.addFile(fileName);
                return;
            }
        }
        System.out.println("Directory " + directoryName + " not found!");
    }

    public void printCDContents() {
        System.out.println("CD Contents:");
        for (Directory directory : directories) {
            directory.printDirectory();
        }
    }

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

        cd.addDirectory("The Pretty Reckless");
        cd.addFileToDirectory("The Pretty Reckless", "justtonight.mp3");
        cd.addFileToDirectory("The Pretty Reckless", "you.mp3");

        cd.addDirectory("Muse");
        cd.addFileToDirectory("Muse", "hysteria.mp3");
        cd.addFileToDirectory("Music", "song.mp3");

        cd.printCDContents();
    }
}
```

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.

Создать класс Текст, используя класс Абзац.

Работа программы:

```
Количество абзацев: 2
Количество слов: 6
Содержимое текста:
Это первый абзац.
Это второй абзац.

Process finished with exit code 0
```

Код программы:

Task2Main.java

```
public class Task2Main {
    public static void main(String[] args) {
        Paragraph paragraph1 = new Paragraph("Это первый абзац.");
        Paragraph paragraph2 = new Paragraph("Это второй абзац.");

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

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

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

Paragraph.java

```
public class Paragraph {
    private String content;

    public Paragraph(String content) {
        this.content = content;
    }

    public String getContent() {
        return content;
    }

    public void setContent(String content) {
        this.content = content;
    }

    public int countWords() {
        String[] words = content.split("\\s+");
        return words.length;
    }
}
```

Text.java

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

public class Text {
    private List<Paragraph> paragraphs;

    public Text() {
        this.paragraphs = new ArrayList<>();
    }

    public void addParagraph(Paragraph paragraph) {
        paragraphs.add(paragraph);
    }

    public void removeParagraph(Paragraph paragraph) {
        paragraphs.remove(paragraph);
    }
```

```
    public int countParagraphs() {
        return paragraphs.size();
    }

    public int countWords() {
        int totalWords = 0;
        for (Paragraph paragraph : paragraphs) {
            totalWords += paragraph.countWords();
        }
        return totalWords;
    }

    public void display() {
        for (Paragraph paragraph : paragraphs) {
            System.out.println(paragraph.getContent());
        }
    }
}
```

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.

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».

Работа программы:

```
Мощный ноутбук для работы и игр
Современный смартфон с хорошей камерой
Легкий и компактный планшет для чтения и просмотра видео
Before chekout (status isPaid): false
Итоговая сумма заказа: $1500.0
After chekout (status isPaid): true
Is sale registered? -- true
Иван находится в черном списке.

Process finished with exit code 0
```

Код программы:

Task3.java

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

// Интерфейс для обработки оплаты заказа
interface PaymentProcess {
    void pay();
}

// Абстрактный класс, представляющий товар
abstract class Product {
    protected String name;
    protected double price;
```

```
    public Product(String name, double price) {
        this.name = name;
        this.price = price;
    }

    public abstract String getDescription();
}

// Класс, представляющий товар
class Item extends Product {
    private String description;

    public Item(String name, double price, String description) {
        super(name, price);
        this.description = description;
    }

    @Override
    public String getDescription() {
        return description;
    }
}

// Класс, представляющий заказ на товары
class Order implements PaymentProcess {
    private List<Product> products;
    private boolean paid = false;

    public Order() {
        products = new ArrayList<>();
    }

    public void addProduct(Product product) {
        products.add(product);
    }

    public void removeProduct(Product product) {
        products.remove(product);
    }

    public double calculateTotalPrice() {
        double total = 0;
        for (Product product : products) {
            total += product.price;
        }
        return total;
    }

    @Override
    public void pay() {
        paid = true;
    }

    public boolean isPaid() {
        return paid;
    }
}

// Класс, представляющий продажу товаров
class Sale {
    private Order order;
    private boolean registered;

    public Sale(Order order) {
        this.order = order;
```

```
    }

    public void registerSale() {
        registered = true;
    }

    public boolean isRegistered() {
        return registered;
    }
}

// Класс, представляющий администратора магазина
class Administrator {
    private List<String> blackList;
    private List<Product> products;

    public void addProductInfo(Product product) {
        products.add(product);
    }

    public List<Product> getProducts() {
        return products;
    }

    public Administrator() {
        blackList = new ArrayList<>();
        products = new ArrayList<>();
    }

    public void addToBlackList(String clientName) {
        blackList.add(clientName);
    }

    public boolean isBlackListed(String clientName) {
        return blackList.contains(clientName);
    }

    public void registerSale(Sale sale) {
        sale.registerSale();
    }
}

// Класс, представляющий клиента магазина
class Client {
    private String name;
    private Order currentOrder;

    public Client(String name) {
        this.name = name;
    }

    public void createOrder() {
        currentOrder = new Order();
    }

    public Order getCurrentOrder() {
        return currentOrder;
    }

    public void addToOrder(Product product) {
        currentOrder.addProduct(product);
    }

    public void removeFromOrder(Product product) {
        currentOrder.removeProduct(product);
    }
```

```
    public double checkout() {
        double totalPrice = currentOrder.calculateTotalPrice();
        currentOrder.pay();
        return totalPrice;
    }

    public String getName() {
        return this.name;
    }
}

public class Task3 {
    public static void main(String[] args) {
        Administrator admin = new Administrator();

        Product item1 = new Item("Ноутбук", 1000.0, "Мощный ноутбук для
работы и игр");
        Product item2 = new Item("Смартфон", 500.0, "Современный смартфон с
хорошей камерой");
        Product item3 = new Item("Планшет", 300.0, "Легкий и компактный
планшет для чтения и просмотра видео");

        admin.addProductInfo(item1);
        admin.addProductInfo(item2);
        admin.addProductInfo(item3);

        // Проверяем, что информация о товарах добавлена успешно
        List<Product> productList = admin.getProducts();
        for (Product product : productList) {
            System.out.println(product.getDescription());
        }

        Client client = new Client("Иван");
        client.createOrder();
        client.addToOrder(item1);
        client.addToOrder(item2);
        client.addToOrder(item3);
        client.removeFromOrder(item3);

        System.out.println("Before chekout (status isPaid): " +
client.getCurrentOrder().isPaid());
        double totalPrice = client.checkout();
        System.out.println("Итоговая сумма заказа: $" + totalPrice);
        System.out.println("After chekout (status isPaid): " +
client.getCurrentOrder().isPaid());

        Sale sale = new Sale(client.getCurrentOrder());
        admin.registerSale(sale);
        System.out.println("Is sale registered? -- " + sale.isRegistered());

        admin.addToBlackList(client.getName());

        boolean blackListed = admin.isBlackListed(client.getName());
        if (blackListed) {
            System.out.println(client.getName() + " находится в черном
списке.");
        }
    }
}
```

Вывод: в ходе лабораторной работы мы приобрели практические навыки в области объектно-ориентированного проектирования.