МИНИСТЕРСТВО ОБРАЗОВАНИЯ РЕСПУБЛИКИ БЕЛАРУСЬ

УЧРЕЖДЕНИЕ ОБРАЗОВАНИЯ
«БРЕСТСКИЙ ГОСУДАРСТВЕННЫЙ ТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ

ФАКУЛЬТЕТ ЭЛЕКТРОННО-ИНФОРМАЦИОННЫХ СИСТЕМ

Кафедра интеллектуальных информационных технологий

Отчет по лабораторной работе №6

Специальность ПО

Выполнил
Войтюк Е.О.
студент группы ПО-8
Проверил
А. А. Крощенко,
ст. преп. кафедры ИИТ,
«___k_____2024 г.

Брест 2024

Цель работы: приобрести навыки применения паттернов проектирования при решении практических задач с использованием языка Java

Задание 1. 6) Музыкальный магазин. Должно обеспечиваться одновременное обслуживание нескольких покупателей. Магазин должен предоставлять широкий выбор товаров различных музыкальных направлений.Выполнение:

Код программы

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

class MusicItem {
    private String type;
    private String genre;
    private String name;
    private double price;

    public static class Builder {
        private String type;
        private String genre;
        private String name;
        private double price;

        public Builder setType(String type) {
            this.type = type;
            return this;
        }

        public Builder setGenre(String genre) {
            this.genre = genre;
            return this;
        }

        public Builder setName(String name) {
            this.name = name;
            return this;
        }

        public Builder setPrice(double price) {
            this.price = price;
            return this;
        }

        public MusicItem build() {
            return new MusicItem(this);
```

```
        }
    }

    private MusicItem(Builder builder) {
        this.type = builder.type;
        this.genre = builder.genre;
        this.name = builder.name;
        this.price = builder.price;
    }

    @Override
    public String toString() {
        return "MusicItem{" +
                "type='" + type + '\'' +
                ", genre='" + genre + '\'' +
                ", name='" + name + '\'' +
                ", price=" + price +
                '}';
    }
}

class MusicStore {
    private static MusicStore instance;
    private List<MusicItem> items = new ArrayList<>();

    private MusicStore() {}

    public static synchronized MusicStore getInstance() {
        if (instance == null) {
            instance = new MusicStore();
        }
        return instance;
    }

    public void addItem(MusicItem item) {
        items.add(item);
    }

    public void serveCustomer(MusicItem item) {
        if (items.contains(item)) {
            System.out.println("Обслуживаем клиента с товаром: " + item);
            items.remove(item);
        } else {
            System.out.println("Извините, данный товар отсутствует на складе: " + item);
        }
    }

    public void displayItems() {
        for (MusicItem item : items) {
            System.out.println(item);
        }
    }
```

```
}

public class Main {
    public static void main(String[] args) {
        MusicStore store = MusicStore.getInstance();

        MusicItem rockAlbum = new MusicItem.Builder()
                .setType("Альбом")
                .setGenre("Рок")
                .setName("Rock Album 1")
                .setPrice(10.99)
                .build();

        MusicItem jazzAlbum = new MusicItem.Builder()
                .setType("Альбом")
                .setGenre("Джаз")
                .setName("Jazz Album 1")
                .setPrice(12.99)
                .build();

        store.addItem(rockAlbum);
        store.addItem(jazzAlbum);

        System.out.println("Товары в магазине:");
        store.displayItems();

        store.serveCustomer(rockAlbum);
        store.serveCustomer(jazzAlbum);
    }
}
```

Рисунки с результатами работы программы

```
Товары в магазине:
MusicItem{type='Альбом', genre='Рок', name='Rock Album 1', price=10.99}
MusicItem{type='Альбом', genre='Джаз', name='Jazz Album 1', price=12.99}
Обслуживаем клиента с товаром: MusicItem{type='Альбом', genre='Рок', name='Rock Album 1', price=10.99}
Обслуживаем клиента с товаром: MusicItem{type='Альбом', genre='Джаз', name='Jazz Album 1', price=12.99}

Process finished with exit code 0
```

Задание 2.

6) Учетная запись покупателя книжного интернет-магазина. Предусмотреть различные уровни учетки в зависимости от активности покупателя. Дополнительные уровни добавляют функциональные возможности и открывают доступ к уникальным предложениям.

Выполнение:

Код программы

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

interface AccountLevel {
    void applyDiscount();
}

class NewCustomer implements AccountLevel {
    public void applyDiscount() {
        System.out.println("Применяется скидка для новых клиентов");
    }
}

class RegularCustomer implements AccountLevel {
    public void applyDiscount() {
        System.out.println("Применяется скидка для постоянных клиентов");
    }
}

class PremiumCustomer implements AccountLevel {
    public void applyDiscount() {
        System.out.println("Применяется скидка для премиум-клиентов");
    }
}

class CustomerAccount {
    private AccountLevel level;
    private String name;
    private List<String> cart;

    public CustomerAccount(AccountLevel level, String name) {
        this.level = level;
        this.name = name;
        this.cart = new ArrayList<>();
    }

    public void setLevel(AccountLevel level) {
        this.level = level;
    }

    public void updateName(String name) {
        this.name = name;
```

```
    }

    public void addToCart(String item) {
        this.cart.add(item);
    }

    public void checkout() {
        System.out.println("Оформление заказа для " + name);
        for (String item : cart) {
            System.out.println("Товар: " + item);
        }
        cart.clear();
        level.applyDiscount();
    }
}

public class Main {
    public static void main(String[] args) {
        CustomerAccount account = new CustomerAccount(new NewCustomer(), "Иван");
        account.addToCart("Книга 1");
        account.addToCart("Книга 2");
        account.checkout();

        account.setLevel(new RegularCustomer());
        account.addToCart("Книга 3");
        account.checkout();

        account.setLevel(new PremiumCustomer());
        account.updateName("Алексей");
        account.addToCart("Книга 4");
        account.checkout();
    }
}
```

Рисунки с результатами работы программы

```
Оформление заказа для Иван
Товар: Книга 1
Товар: Книга 2
Применяется скидка для новых клиентов
Оформление заказа для Иван
Товар: Книга 3
Применяется скидка для постоянных клиентов
Оформление заказа для Алексей
Товар: Книга 4
Применяется скидка для премиум-клиентов

Process finished with exit code 0
```

Задание 3.

6) Проект «Принтер». Предусмотреть выполнение операций (печать, загрузка бумаги, извлечение зажатой бумаги, заправка картриджа), режимы – ожидание, печать документа, зажатие бумаги, отказ – при отсутствии бумаги или краски, атрибуты – модель, количество листов в лотке, % краски в картридже, вероятность зажатия.

Выполнение:

Код программы

```
interface PrinterStrategy {
    void execute(Printer printer);
}

class Print implements PrinterStrategy {
    public void execute(Printer printer) {
        if (printer.getPaperCount() > 0 && printer.getInkPercentage() > 0) {
            System.out.println("Печать документа...");
            printer.setPaperCount(printer.getPaperCount() - 1);
            printer.setInkPercentage(printer.getInkPercentage() - 0.1);
        } else {
            printer.setState(new FailureState(printer));
            printer.executeStrategy();
        }
    }
}

class LoadPaper implements PrinterStrategy {
```

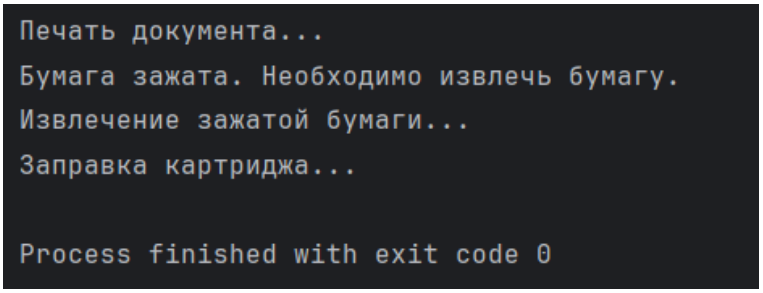
```
    public void execute(Printer printer) {
        System.out.println("Загрузка бумаги...");
        printer.setPaperCount(100);
    }
}

class ExtractPaper implements PrinterStrategy {
    public void execute(Printer printer) {
        System.out.println("Извлечение зажатой бумаги...");
        printer.setJamProbability(0.01);
    }
}

class RefillCartridge implements PrinterStrategy {
    public void execute(Printer printer) {
        System.out.println("Заправка картриджа...");
        printer.setInkPercentage(100.0);
    }
}

interface PrinterState {
    void handleRequest(Printer printer);
}

class IdleState implements PrinterState {
    private Printer printer;

    public IdleState(Printer printer) {
        this.printer = printer;
    }

    public void handleRequest(Printer printer) {
        if (Math.random() < printer.getJamProbability()) {
            printer.setState(new JamState(printer));
        } else {
            printer.executeStrategy();
        }
    }
}

class JamState implements PrinterState {
    private Printer printer;

    public JamState(Printer printer) {
        this.printer = printer;
    }

    public void handleRequest(Printer printer) {
        System.out.println("Бумага зажата. Необходимо извлечь бумагу.");
        printer.setStrategy(new ExtractPaper());
        printer.executeStrategy();
        printer.setState(new IdleState(printer));
```

```
    }
}

class FailureState implements PrinterState {
    private Printer printer;

    public FailureState(Printer printer) {
        this.printer = printer;
    }

    public void handleRequest(Printer printer) {
        if (printer.getPaperCount() == 0) {
            System.out.println("Отсутствует бумага. Необходимо загрузить бумагу.");
            printer.setStrategy(new LoadPaper());
        } else if (printer.getInkPercentage() == 0) {
            System.out.println("Отсутствует краска. Необходимо заправить картридж.");
            printer.setStrategy(new RefillCartridge());
        }
        printer.executeStrategy();
        printer.setState(new IdleState(printer));
    }
}

class Printer {
    private PrinterStrategy strategy;
    private PrinterState state;
    private String model;
    private int paperCount;
    private double inkPercentage;
    private double jamProbability;

    public Printer(String model, int paperCount, double inkPercentage, double jamProbability) {
        this.model = model;
        this.paperCount = paperCount;
        this.inkPercentage = inkPercentage;
        this.jamProbability = jamProbability;
        this.strategy = new Print();
        this.state = new IdleState(this);
    }

    public String getModel() {
        return model;
    }

    public void setModel(String model) {
        this.model = model;
    }

    public int getPaperCount() {
        return paperCount;
    }
```

```
    public void setPaperCount(int paperCount) {
        this.paperCount = paperCount;
    }

    public double getInkPercentage() {
        return inkPercentage;
    }

    public void setInkPercentage(double inkPercentage) {
        this.inkPercentage = inkPercentage;
    }

    public double getJamProbability() {
        return jamProbability;
    }

    public void setJamProbability(double jamProbability) {
        this.jamProbability = jamProbability;
    }

    public void setStrategy(PrinterStrategy strategy) {
        this.strategy = strategy;
    }

    public void executeStrategy() {
        strategy.execute(this);
    }

    public void setState(PrinterState state) {
        this.state = state;
    }

    public void handleRequest() {
        state.handleRequest(this);
    }
}

public class Main {
    public static void main(String[] args) {
        Printer printer = new Printer("HP LaserJet", 100, 100.0, 0.8);

        printer.handleRequest();

        printer.setStrategy(new LoadPaper());
        printer.handleRequest();

        printer.setStrategy(new ExtractPaper());
        printer.handleRequest();

        printer.setStrategy(new RefillCartridge());
        printer.handleRequest();
    }
```

```
}
```

Рисунки с результатами работы программы



Вывод: приобрел навыки применения паттернов проектирования при решении практических задач с использованием языка Java