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

Отчёт
по лабораторной работе №6

Выполнил:
студент группы ПО-9
Солышко Дмитрий Андреевич

Проверил:
Крощенко А. А.

Брест 2024

Вариант 6

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

Общее задание

• Прочитать задания, взятые из каждой группы.

• Определить паттерн проектирования, который может использоваться при реализации задания. Пояснить свой выбор.

• Реализовать фрагмент программной системы, используя выбранный паттерн. Реализовать все необходимые дополнительные классы.

Задание 1

Музыкальный магазин. Должно обеспечиваться одновременное обслуживание нескольких покупателей. Магазин должен предоставлять широкий выбор товаров.

Выбранный Паттерн: Наблюдатель. Он позволяет реализовать механизм уведомления об изменениях в состоянии объекта одному или нескольким зависимым объектам. У нас есть несколько покупателей, которые должны быть уведомлены об изменениях в ассортименте товаров (новые альбомы, специальные предложения и т. д.).

Код программы:

Main.java:

```java
public class Main {
    public static void main(String[] args) {
        MusicStore store = new MusicStore();

        Customer customer1 = new Customer("Dima", 100);
        Customer customer2 = new Customer("Zakhar", 50);

        store.addObserver(customer1);
        store.addObserver(customer2);

        store.addAlbum("Album 1", 20.0);
        store.addAlbum("Album 2", 30.0);
        store.addAlbum("Album 3", 30.0);

        store.addNews("New albums added to the store!");

        store.purchaseAlbum(customer1, "Album 1");
        store.purchaseAlbum(customer2, "Album 2");

        store.purchaseAlbum(customer1, "Album 2");
    }
}
```

Customer.java:

```java
public class Customer implements Observer {
    private String name;
    private double balance;

    public Customer(String name, double balance) {
        this.name = name;
        this.balance = balance;
    }

    @Override
    public void update(News news) {
        System.out.println(name + " learned about news: " + news.getText());
    }

    public String getName() {
        return name;
    }

    public double getBalance() {
        return balance;
    }

    public boolean hasEnoughBalance(double amount) {
        return balance >= amount;
    }

    public void deductBalance(double amount) {
        balance -= amount;
    }
}
```

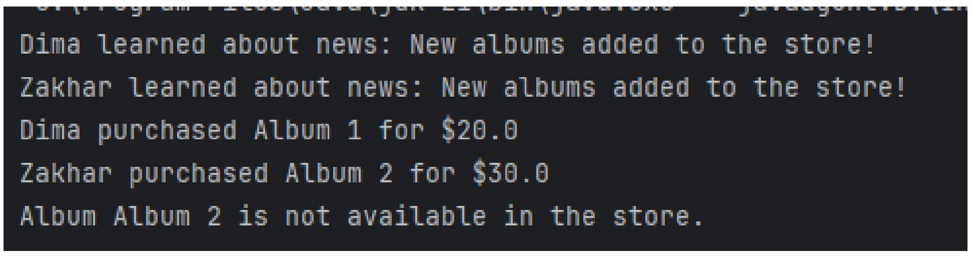
MusicStore.java:

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

public class MusicStore {
    private List<Observer> observers = new ArrayList<>();
    private Map<String, Double> availableAlbums = new HashMap<>();

    public void addObserver(Observer observer) {
        observers.add(observer);
    }

    public void notifyObservers(News news) {
        for (Observer observer : observers) {
            observer.update(news);
        }
    }

    public void addNews(String newsText) {
        News news = new News(newsText);
        notifyObservers(news);
    }

    public void addAlbum(String albumName, double price) {
        availableAlbums.put(albumName, price);
    }

    public void purchaseAlbum(Customer customer, String albumName) {
        if (availableAlbums.containsKey(albumName)) {
            double price = availableAlbums.get(albumName);
            if (customer.hasEnoughBalance(price)) {
                customer.deductBalance(price);
                availableAlbums.remove(albumName);
                System.out.println(customer.getName() + " purchased " + albumName + " for $" +
price);
            } else {
                System.out.println(customer.getName() + " does not have enough balance to
purchase " + albumName);
            }
        } else {
            System.out.println("Album " + albumName + " is not available in the store.");
        }
    }
}
```

News.java:

```java
public class News {
    private String text;

    public News(String text) {
        this.text = text;
    }

    public String getText() {
        return text;
    }
}
```

Результат работы программы:



```
Dima learned about news: New albums added to the store!
Zakhar learned about news: New albums added to the store!
Dima purchased Album 1 for $20.0
Zakhar purchased Album 2 for $30.0
Album Album 2 is not available in the store.
```

Задание 2

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.

Я здесь выбрал паттерн "Стратегия". Он позволяет изменять поведение объекта в зависимости от ситуации или требований клиента. В данном случае, различные уровни активности покупателя - это различные стратегии, которые определяют функциональные возможности и доступ к уникальным предложениям.

Код программы

ActivityLevel.java(interface):

```java
package Task_02;

public interface ActivityLevel {
    void purchaseBook(Book book, CustomerAccount customer);
    void participateInContest(CustomerAccount customer);
}
```

BasicLevel.java:

```java
package Task_02;

public class BasicLevel implements ActivityLevel {
    @Override
    public void purchaseBook(Book book, CustomerAccount customer) {
        System.out.println("Purchasing book: " + book.getTitle());
        customer.incrementPoints(book.getPoints());
        checkUpgrade(customer);
    }

    @Override
    public void participateInContest(CustomerAccount customer) {
        System.out.println("Contest participation available for premium and elite levels.");
    }

    private void checkUpgrade(CustomerAccount customer) {
        if (customer.getPoints() >= 5) {
            customer.setActivityLevel(new PremiumLevel());
            System.out.println("Congratulations! You have been upgraded to Premium Level.");
        }
    }
}
```

PremiumLevel.java:

```java
package Task_02;

import java.util.Random;

public class PremiumLevel implements ActivityLevel {
    @Override
    public void purchaseBook(Book book, CustomerAccount customer) {
        System.out.println("Purchasing book: " + book.getTitle());
        customer.incrementPoints(book.getPoints());
        checkUpgrade(customer);
    }

    @Override
    public void participateInContest(CustomerAccount customer) {
        Random random = new Random();
        int result = random.nextInt(100);
        if (result < 30) {
            System.out.println("You won the contest");
            customer.addBalance(20);
        } else {
            System.out.println("Better luck next time! No prize this time.");
        }
    }

    private void checkUpgrade(CustomerAccount customer) {
        if (customer.getPoints() >= 10) {
            customer.setActivityLevel(new EliteLevel());
            System.out.println("Congratulations! You have been upgraded to Elite Level.");
        }
    }
}
```

EliteLevel.java:

```java
package Task_02;

import java.util.Random;

public class EliteLevel implements ActivityLevel {
    @Override
    public void purchaseBook(Book book, CustomerAccount customer) {
        System.out.println("Purchasing book: " + book.getTitle());
        customer.incrementPoints(book.getPoints());
    }

    @Override
    public void participateInContest(CustomerAccount customer) {
        Random random = new Random();
        int result = random.nextInt(100);
        if (result < 50) {
            System.out.println("You won the contest");
            customer.addBalance(50);
        } else {
            System.out.println("Better luck next time! No prize this time.");
        }
    }
}
```

Book.java:

```java
package Task_02;

public class Book {
    private String title;
    private int points;
    private double price;

    public Book(String title, int points, double price) {
        this.title = title;
        this.points = points;
        this.price = price;
    }

    public String getTitle() {
        return title;
    }

    public int getPoints() {
        return points;
    }

    public double getPrice() {
        return price;
    }
}
```

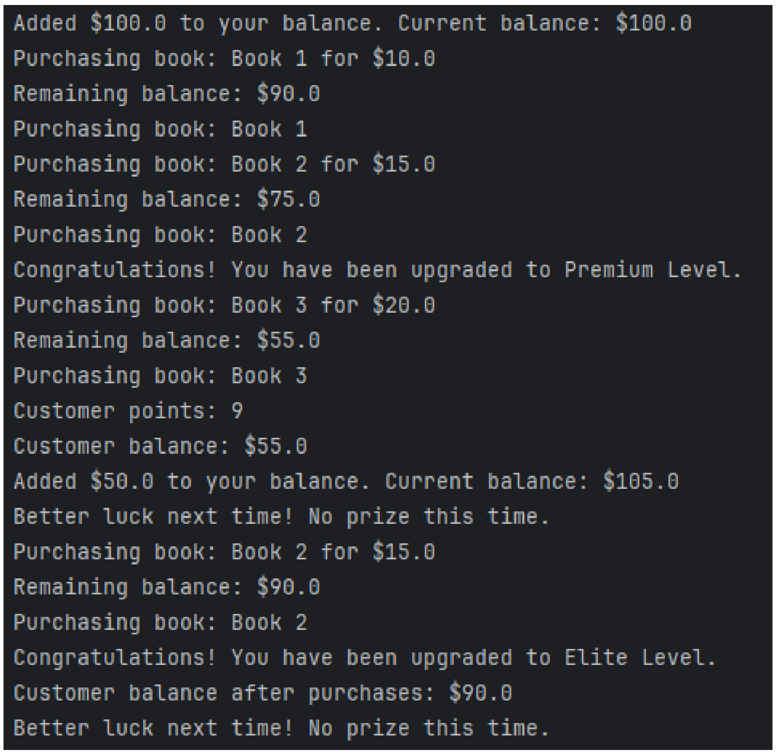
CustomerAccount:

```java
package Task_02;
public class CustomerAccount {
    private ActivityLevel activityLevel;
    private int points;
    private double balance;

    public CustomerAccount() {
        this.activityLevel = new BasicLevel();
        this.points = 0;
        this.balance = 0;
    }

    public void setActivityLevel(ActivityLevel activityLevel) {
        this.activityLevel = activityLevel;
    }

    public void incrementPoints(int points) {
        this.points += points;
    }

    public int getPoints() {
        return points;
    }

    public double getBalance() {
        return balance;
    }

    public void purchaseBook(Book book) {
        if (book.getPrice() <= balance) {
            System.out.println("Purchasing book: " + book.getTitle() + " for $" +
book.getPrice());
            balance -= book.getPrice();
            System.out.println("Remaining balance: $" + balance);
        } else {
            System.out.println("Not enough balance to purchase " + book.getTitle());
        }
        activityLevel.purchaseBook(book, this);
    }

    public void participateInContest() {
        activityLevel.participateInContest(this);
    }

    public void addBalance(double amount) {
        balance += amount;
        System.out.println("Added $" + amount + " to your balance. Current balance: $" +
balance);
    }
}
```

Main.java:

```java
package Task_02;

public class Main {
    public static void main(String[] args) {
        CustomerAccount customer1 = new CustomerAccount();
        Book book1 = new Book("Book 1", 2, 10.0);
        Book book2 = new Book("Book 2", 3, 15.0);
        Book book3 = new Book("Book 3", 4, 20.0);

        customer1.addBalance(100.0);

        customer1.purchaseBook(book1);
        customer1.purchaseBook(book2);
        customer1.purchaseBook(book3);

        System.out.println("Customer points: " + customer1.getPoints());
        System.out.println("Customer balance: $" + customer1.getBalance());

        customer1.addBalance(50.0);

        customer1.participateInContest();

        customer1.purchaseBook(book2);

        System.out.println("Customer balance after purchases: $" + customer1.getBalance());

        customer1.participateInContest();
    }
}
```

Результат работы программы:



```
Added $100.0 to your balance. Current balance: $100.0
Purchasing book: Book 1 for $10.0
Remaining balance: $90.0
Purchasing book: Book 1
Purchasing book: Book 2 for $15.0
Remaining balance: $75.0
Purchasing book: Book 2
Congratulations! You have been upgraded to Premium Level.
Purchasing book: Book 3 for $20.0
Remaining balance: $55.0
Purchasing book: Book 3
Customer points: 9
Customer balance: $55.0
Added $50.0 to your balance. Current balance: $105.0
Better luck next time! No prize this time.
Purchasing book: Book 2 for $15.0
Remaining balance: $90.0
Purchasing book: Book 2
Congratulations! You have been upgraded to Elite Level.
Customer balance after purchases: $90.0
Better luck next time! No prize this time.
```

Задание 3

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.

Я выбрал паттерн состояний (State) для реализации проекта "Принтер". Паттерн состояний позволяет объекту изменять свое поведение в зависимости от внутреннего состояния, причем выглядит так, что объект меняет свой класс.

Код программы

PrinterState:

```java
package Task_03;

public interface PrinterState {
    void printDocument(Printer printer);
    void removeJam(Printer printer);
    void loadPaper(Printer printer, int count);
    void refillInk(Printer printer);
    String getStateDescription();
}
```

IdleState.java:

```java
package Task_03;
import java.util.Random;

public class IdleState implements PrinterState {
    @Override
    public void printDocument(Printer printer) {
        System.out.println("Switching to Printing state...");
        printer.setCurrentState(new PrintingState());
        System.out.println(printer.getCurrentState().getStateDescription());
        printer.printDocument();
    }

    @Override
    public void removeJam(Printer printer) {
        System.out.println("No paper jam to remove.");
    }

    @Override
    public void loadPaper(Printer printer, int count) {
        printer.setPaperCount(printer.getPaperCount() + count);
        System.out.println("Loaded " + count + " sheets of paper.");
    }

    @Override
    public void refillInk(Printer printer) {
        printer.setInkLevel(100);
        System.out.println("Ink refilled to 100%.");
    }

    @Override
    public String getStateDescription() {
        return "IdleState";
    }
}
```

OutOfPaperOrInkState:

```java
package Task_03;

public class OutOfPaperOrInkState implements PrinterState {
    @Override
    public void printDocument(Printer printer) {
        System.out.println("Cannot print. Out of paper or ink.");
    }

    @Override
    public void removeJam(Printer printer) {
        System.out.println("No paper jam to remove.");
    }

    @Override
    public void loadPaper(Printer printer, int count) {
        printer.setPaperCount(printer.getPaperCount() + count);
        printer.setCurrentState(new IdleState());
        System.out.println("Paper and ink refilled. Printer ready.");
        System.out.println(printer.getCurrentState().getStateDescription());
    }

    @Override
    public void refillInk(Printer printer) {
        printer.setInkLevel(100);
        printer.setCurrentState(new IdleState());
        System.out.println("Paper and ink refilled. Printer ready.");
        System.out.println(printer.getCurrentState().getStateDescription());
    }

    @Override
    public String getStateDescription() {
        return "OutOfPaperOrInkState";
    }
}
```

PaperJamState:

```java
package Task_03;

public class PaperJamState implements PrinterState {
    @Override
    public void printDocument(Printer printer) {
        System.out.println("Cannot print. Paper jammed.");
    }

    @Override
    public void removeJam(Printer printer) {
        System.out.println("Removing paper jam...");
        printer.setCurrentState(new IdleState());
        System.out.println(printer.getCurrentState().getStateDescription());
    }

    @Override
    public void loadPaper(Printer printer, int count) {
        printer.setPaperCount(printer.getPaperCount() + count);
        System.out.println("Loaded " + count + " sheets of paper.");
    }

    @Override
    public void refillInk(Printer printer) {
        printer.setInkLevel(100);
        System.out.println("Ink refilled to 100%.");
    }

    @Override
    public String getStateDescription() {
        return "PaperJamState";
    }
}
```

PrintingState:

```java
package Task_03;
import java.util.Random;
public class PrintingState implements PrinterState {
    @Override
    public void printDocument(Printer printer) {
        int jamProbability = new Random().nextInt(100) + 1;
        if (jamProbability <= printer.getJamProbability()) {
            System.out.println("Paper jammed!");
            printer.setCurrentState(new PaperJamState());
            System.out.println(printer.getCurrentState().getStateDescription());
        } else if (printer.getPaperCount() <= 0 || printer.getInkLevel() <= 0) {
            System.out.println("Out of paper or ink. Printing stopped.");
            printer.setCurrentState(new OutOfPaperOrInkState());
            System.out.println(printer.getCurrentState().getStateDescription());
        }
        else {
            System.out.println("Already printing...");
            printer.setPaperCount(printer.getPaperCount() - 1);
            printer.setInkLevel(printer.getInkLevel() - 10);
            printer.setCurrentState(new IdleState());
            System.out.println(printer.getCurrentState().getStateDescription());
        }
    }

    @Override
    public void removeJam(Printer printer) {
        System.out.println("Cannot remove paper jam while printing.");
    }

    @Override
    public void loadPaper(Printer printer, int count) {
        System.out.println("Cannot load paper while printing.");
    }

    @Override
    public void refillInk(Printer printer) {
        System.out.println("Cannot refill ink while printing.");
    }

    @Override
    public String getStateDescription() {
        return "PrintingState";
    }
}
```

Printer:

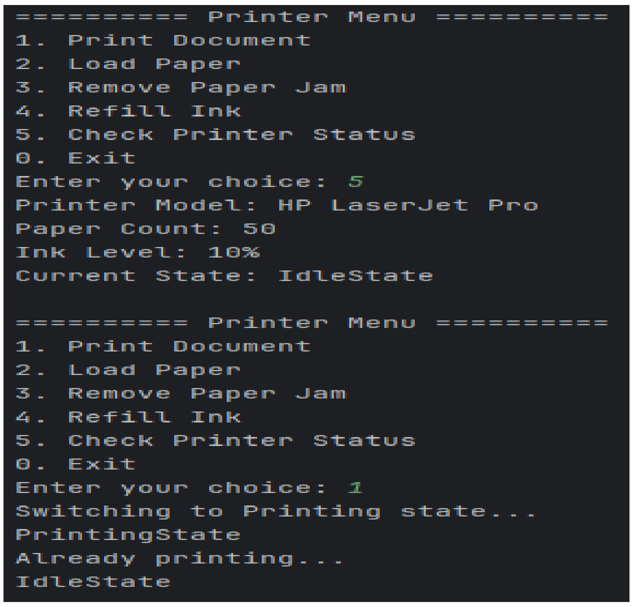
```java
package Task_03;

public class Printer {
    private String model;
    private int paperCount;
    private int inkLevel;
    private int jamProbability;
    private PrinterState currentState;

    public Printer(String model, int paperCount, int inkLevel, int jamProbability) {
        this.model = model;
        this.paperCount = paperCount;
        this.inkLevel = inkLevel;
        this.jamProbability = jamProbability;
        this.currentState = new IdleState();
    }

    public void printDocument() {
        currentState.printDocument(this);
    }

    public void loadPaper(int count) {
        currentState.loadPaper(this, count);
    }
    public void refillInk() {
        currentState.refillInk(this);
    }

    public void removeJam() {
        currentState.removeJam(this);
    }

    public String getModel() {
        return model;
    }

    public int getPaperCount() {
        return paperCount;
    }

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

    public int getInkLevel() {
        return inkLevel;
    }

    public void setInkLevel(int inkLevel) {
        this.inkLevel = inkLevel;
    }

    public int getJamProbability() {
        return jamProbability;
    }

    public void setCurrentState(PrinterState currentState) {
        this.currentState = currentState;
    }

    public PrinterState getCurrentState() {
        return currentState;
    }

    public void printStatus() {
        System.out.println("Printer Model: " + model);
        System.out.println("Paper Count: " + paperCount);
        System.out.println("Ink Level: " + inkLevel + "%");
        System.out.println("Current State: " + currentState.getStateDescription());
    }
}
```

Main:

```java
package Task_03;

import java.util.Scanner;

public class Main {
    public static void main(String[] args) {
        Scanner scanner = new Scanner(System.in);
        Printer printer = new Printer("HP LaserJet Pro", 50, 10, 5);

        boolean isRunning = true;
        while (isRunning) {
            System.out.println("========== Printer Menu ==========");
            System.out.println("1. Print Document");
            System.out.println("2. Load Paper");
            System.out.println("3. Remove Paper Jam");
            System.out.println("4. Refill Ink");
            System.out.println("5. Check Printer Status");
            System.out.println("0. Exit");
            System.out.print("Enter your choice: ");

            int choice = scanner.nextInt();
            switch (choice) {
                case 1:
                    printer.printDocument();
                    break;
                case 2:
                    System.out.print("Enter number of sheets to load: ");
                    int sheets = scanner.nextInt();
                    printer.loadPaper(sheets);
                    break;
                case 3:
                    printer.removeJam();
                    break;
                case 4:
                    printer.refillInk();
                    break;
                case 5:
                    printer.printStatus();
                    break;
                case 0:
                    isRunning = false;
                    System.out.println("Exit...");
                    break;
                default:
                    System.out.println("Invalid choice. Please enter a valid option.");
            }
            System.out.println();
        }
        scanner.close();
    }
}
```

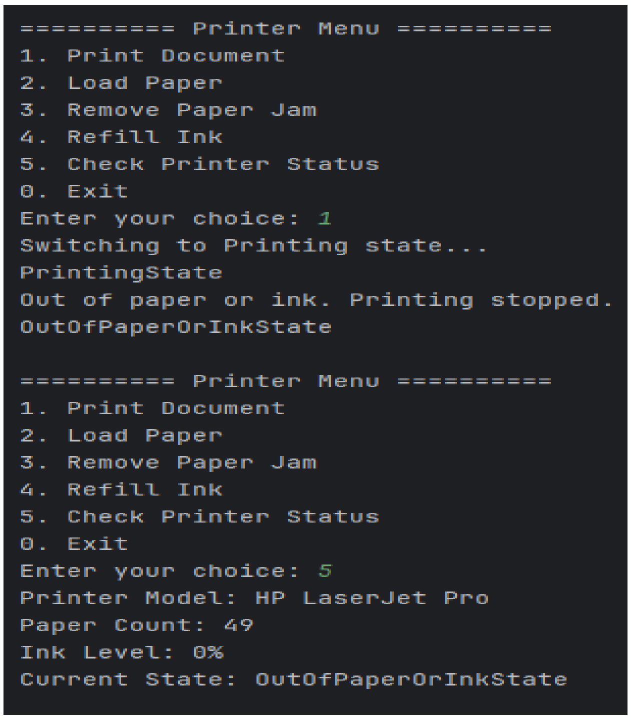
Результат работы программы:

Проверяем состояние принтера, а после печатаем безошибочно:



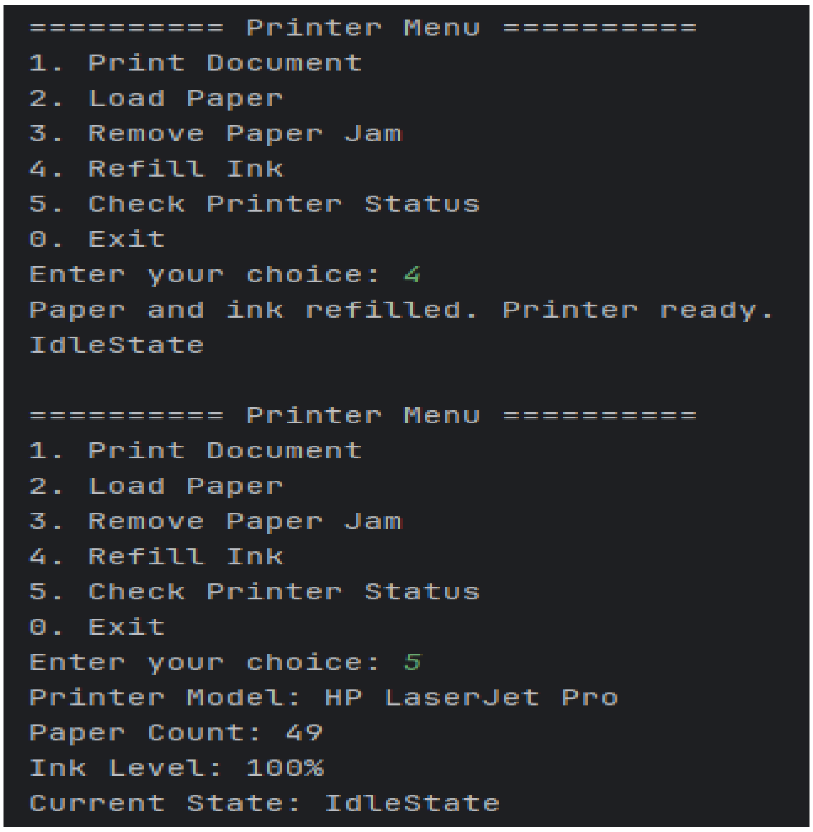
```
========== Printer Menu ==========
1. Print Document
2. Load Paper
3. Remove Paper Jam
4. Refill Ink
5. Check Printer Status
0. Exit
Enter your choice: 5
Printer Model: HP LaserJet Pro
Paper Count: 50
Ink Level: 10%
Current State: IdleState

========== Printer Menu ==========
1. Print Document
2. Load Paper
3. Remove Paper Jam
4. Refill Ink
5. Check Printer Status
0. Exit
Enter your choice: 1
Switching to Printing state...
PrintingState
Already printing...
IdleState
```

Далее уже краска заканчивается, так как она была заполнена на 10%, и при печати у нас происходит сбой:



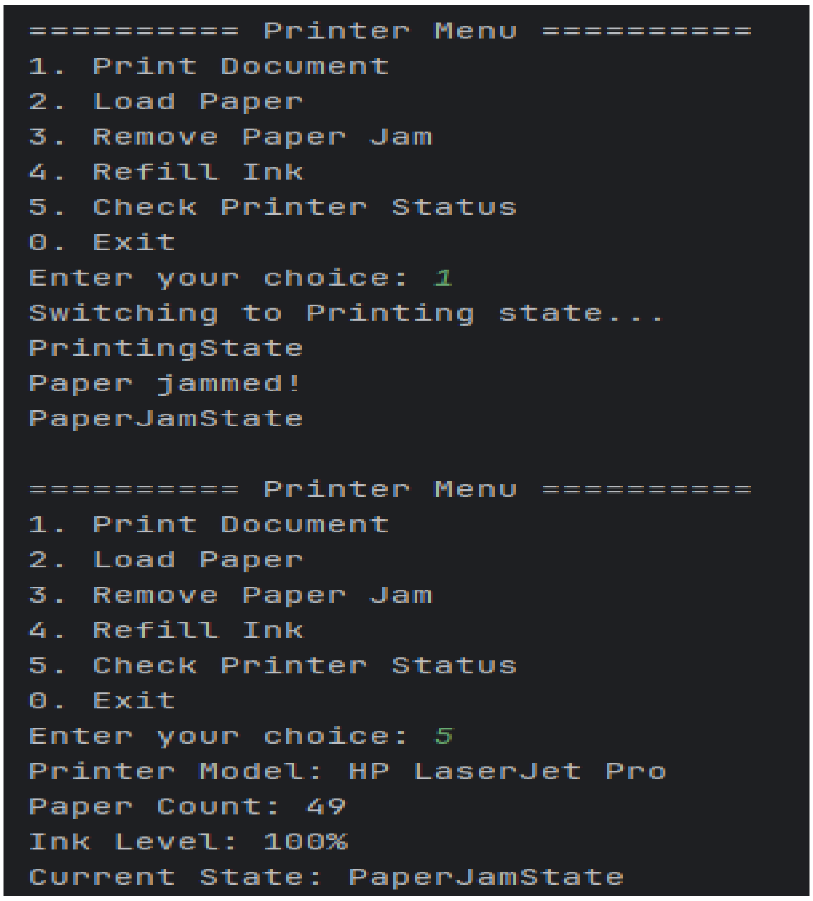
```
========== Printer Menu ==========
1. Print Document
2. Load Paper
3. Remove Paper Jam
4. Refill Ink
5. Check Printer Status
0. Exit
Enter your choice: 1
Switching to Printing state...
PrintingState
Out of paper or ink. Printing stopped.
OutOfPaperOrInkState

========== Printer Menu ==========
1. Print Document
2. Load Paper
3. Remove Paper Jam
4. Refill Ink
5. Check Printer Status
0. Exit
Enter your choice: 5
Printer Model: HP LaserJet Pro
Paper Count: 49
Ink Level: 0%
Current State: OutOfPaperOrInkState
```

Теперь заполняем краску, и выводим состояние принтера



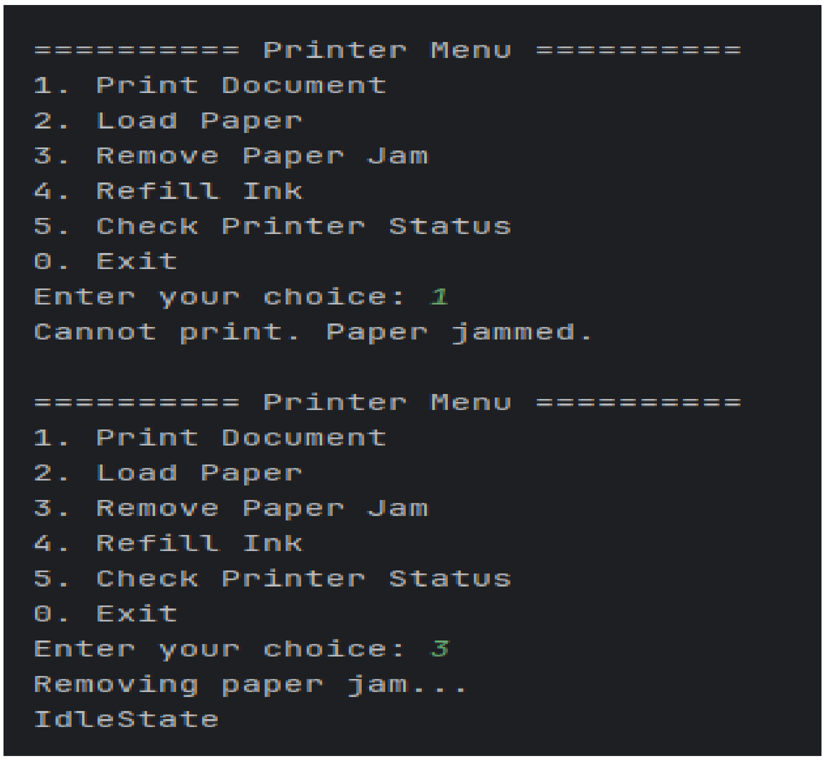
```
========== Printer Menu ==========
1. Print Document
2. Load Paper
3. Remove Paper Jam
4. Refill Ink
5. Check Printer Status
0. Exit
Enter your choice: 4
Paper and ink refilled. Printer ready.
IdleState

========== Printer Menu ==========
1. Print Document
2. Load Paper
3. Remove Paper Jam
4. Refill Ink
5. Check Printer Status
0. Exit
Enter your choice: 5
Printer Model: HP LaserJet Pro
Paper Count: 49
Ink Level: 100%
Current State: IdleState
```

Спустя несколько попыток, мне не везет, и Бумага зажимается:



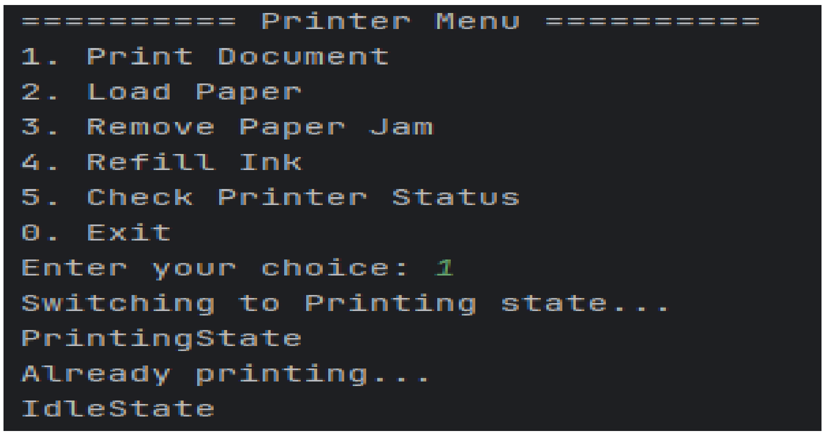
```
========== Printer Menu ==========
1. Print Document
2. Load Paper
3. Remove Paper Jam
4. Refill Ink
5. Check Printer Status
0. Exit
Enter your choice: 1
Switching to Printing state...
PrintingState
Paper jammed!
PaperJamState

========== Printer Menu ==========
1. Print Document
2. Load Paper
3. Remove Paper Jam
4. Refill Ink
5. Check Printer Status
0. Exit
Enter your choice: 5
Printer Model: HP LaserJet Pro
Paper Count: 49
Ink Level: 100%
Current State: PaperJamState
```

Далее пытаемся еще раз распечатать с зажатой бумагой, и видим, что из-за состояния зажатой бумаги печать невозможна. Поэтому достаём зажатую бумагу.



```
========== Printer Menu ==========
1. Print Document
2. Load Paper
3. Remove Paper Jam
4. Refill Ink
5. Check Printer Status
0. Exit
Enter your choice: 1
Cannot print. Paper jammed.

========== Printer Menu ==========
1. Print Document
2. Load Paper
3. Remove Paper Jam
4. Refill Ink
5. Check Printer Status
0. Exit
Enter your choice: 3
Removing paper jam...
IdleState
```

Всё хорошо печатается



```
========== Printer Menu ==========
1. Print Document
2. Load Paper
3. Remove Paper Jam
4. Refill Ink
5. Check Printer Status
0. Exit
Enter your choice: 1
Switching to Printing state...
PrintingState
Already printing...
IdleState
```

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