

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

ОТЧЁТ
по лабораторной работе №6

Выполнил
студент 3 курса
группы ПО-9
Ничингер Кирилл Александрович

Проверил:
Крощенко А. А.

Брест 2024

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

Вариант 6

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

Код программы:

```
public class MusicStore {
    private static MusicStore instance;
    private List<Product> products;

    private MusicStore() {
        products = new ArrayList<>();
    }

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

    public synchronized void serveCustomer(Customer customer) {
        if (products.isEmpty())
            return;
        Product product = products.get((int) Math.round(Math.random() *
products.size()));
        if (customer.isProductObtainable(product)) {
            customer.buyProduct(product);
        }
    }
}

public class Customer {
    private String name;
    private float budget;
    private List<Product> inventory;

    public Customer(String name, float budget) {
        this.name = name;
        this.budget = budget;
        inventory = new ArrayList<>(List.of());
    }

    public void buyProduct(Product product) {
        if (isProductObtainable(product)) {
            budget -= product.getPrice();
            inventory.add(new Product(product));
        }
    }
}
```

Входные данные:

```
public class MusicStoreTest {
    public static void main(String[] args) {
        MusicStore musicStore = MusicStore.getInstance();
        musicStore.addProducts(List.of(
            new Product("Krutaya pesnya", 50),
            new Product("Spichki", 80)
        ));

        System.out.println("Music store info:");
        System.out.println(musicStore);

        Customer customer = new Customer("Kirill", 100);

        System.out.println("\nCustomer info before purchase:");
        System.out.println(customer);

        musicStore.serveCustomer(customer);

        System.out.println("\nCustomer info after purchase:");
        System.out.println(customer);
    }
}
```

Результат работы программы:

```
D:\SDK\JDK\bin\java.exe "-javaagent:D:\JetBrains\IntelliJ IDEA 2023.3.4\lib\idea_rt.jar=50267:D:\JetBr
Music store info:
MusicStore{products=[Product{name='Krutaya pesnya', price=50.0}, Product{name='Spichki', price=80.0}]}

Customer info before purchase:
Customer{name='Kirill', budget=100.0, inventory=[]}

Customer info after purchase:
Customer{name='Kirill', budget=20.0, inventory=[Product{name='Spichki', price=80.0}]}

Process finished with exit code 0
```

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. (Decorator)

Код программы:

```
public interface AccountLevel {
    public void buyBook(Book book);
    public void addToShoppingCart(Book book);
    public List<Book> getSpecialOffer();
}

public class BaseAccountLevel implements AccountLevel {
    @Override
    public void buyBook(Book book) {
        System.out.println("Покупка книги " + book.getTitle());
    }

    @Override
    public void addToShoppingCart(Book book) {
        System.out.println("Корзина недоступна");
    }

    @Override
    public List<Book> getSpecialOffer() {
        System.out.println("Специальные предложения недоступны");
        return new ArrayList<>();
    }
}

public class BaseDecorator implements AccountLevel{
    private AccountLevel wrappee;

    public BaseDecorator(AccountLevel accountLevel) {
        this.wrappee = accountLevel;
    }

    @Override
    public void buyBook(Book book) {
        wrappee.buyBook(book);
    }

    @Override
    public void addToShoppingCart(Book book) {
        wrappee.addToShoppingCart(book);
    }

    @Override
    public List<Book> getSpecialOffer() {
        return wrappee.getSpecialOffer();
    }
}
```

```

public class DeluxeAccountLevel extends BaseDecorator {
    public DeluxeAccountLevel(AccountLevel accountLevel){
        super(accountLevel);
    }
    @Override
    public void buyBook(Book book) {
        System.out.println("Покупка книги " + book.getTitle());
    }

    @Override
    public void addToShoppingCart(Book book) {
        System.out.println("Книга " + book.getTitle() + " добавлена в корзину");
    }

    @Override
    public List<Book> getSpecialOffer() {
        System.out.println("Специальные предложения отправлены");
        return new ArrayList<>();
    }
}

public class Customer implements AccountLevel{
    private AccountLevel accountLevel;

    public Customer(){
        accountLevel = new BaseAccountLevel();
    }

    @Override
    public void buyBook(Book book) {
        accountLevel.buyBook(book);
    }

    @Override
    public void addToShoppingCart(Book book) {
        accountLevel.addToShoppingCart(book);
    }

    @Override
    public List<Book> getSpecialOffer() {
        return accountLevel.getSpecialOffer();
    }

    public void toDeluxeAccount(){
        accountLevel = new DeluxeAccountLevel(accountLevel);
    }

    public void toPremiumAccount(){
        accountLevel = new PremiumAccountLevel(accountLevel);
    }
}

```

Входные данные:

```
public class BookStoreTest {  
    public static void main(String[] args) {  
        Customer customer = new Customer();  
        Book book = new Book("Voina i mir", "Privet", 50);  
  
        System.out.println("Базовый аккаунт:");  
        customer.buyBook(book);  
        customer.addToShoppingCart(book);  
        customer.getSpecialOffer();  
  
        System.out.println("\nПремиум аккаунт:");  
        customer.toPremiumAccount();  
        customer.buyBook(book);  
        customer.addToShoppingCart(book);  
        customer.getSpecialOffer();  
  
        System.out.println("\nДелюкс аккаунт:");  
        customer.toDeluxeAccount();  
        customer.buyBook(book);  
        customer.addToShoppingCart(book);  
        customer.getSpecialOffer();  
    }  
}
```

Результат работы программы:

```
D:\SDK\JDK\bin\java.exe "-javaagent:D:\
```

Базовый аккаунт:

Покупка книги Voina i mir

Корзина недоступна

Специальные предложения недоступны

Премиум аккаунт:

Покупка книги Voina i mir

Книга Voina i mir добавлена в корзину

Специальные предложения недоступны

Делюкс аккаунт:

Покупка книги Voina i mir

Книга Voina i mir добавлена в корзину

Специальные предложения отправлены

```
Process finished with exit code 0
```

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. (State)

Код программы:

```
public interface PrinterFunctions {
    void print(int numPaper);
    void loadPaper(int numPaper);
    void extractJammingPaper();
    void refillCartridge();
}

public class Printer implements PrinterFunctions{
    private String model;
    private int pageCount;
    private double paintPercentage; // [0..100]
    private double jammingProbability; // [0..100]
    private State state;

    public Printer(String model, int pageCount, double paintPercentage, double
jammingProbability) {
        this.model = model;
        this.pageCount = pageCount;
        this.paintPercentage = paintPercentage;
        this.jammingProbability = jammingProbability;
        state = new WaitingState(this);
    }

    public void changeState(State state){
        this.state = state;
    }

    @Override
    public void print(int numPaper) {
        state.print(numPaper);
    }

    @Override
    public void loadPaper(int numPaper) {
        state.loadPaper(numPaper);
    }

    @Override
    public void extractJammingPaper() {
        state.extractJammingPaper();
    }

    @Override
    public void refillCartridge() {
        state.refillCartridge();
    }
}
```

```

public abstract class State implements PrinterFunctions{
    protected Printer printer;
    protected String stateName = "None";

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

public class PrintingState extends State {
    private final double paintConsumption = 5;
    public PrintingState(Printer printer) {
        super(printer);
        stateName = "PrintingState";
    }

    @Override
    public void print(int numPaper) {
        for (int i = 0; i < numPaper; i++){
            if (isPrintingAvailable()){
                double randValue = Math.random() * 100;
                if (randValue <= printer.getJammingProbability()){
                    printer.changeState(new JammingState(printer));
                    return;
                }

                printer.setPagesCount(printer.getPagesCount() - 1);
                printer.setPaintPercentage(printer.getPaintPercentage() -
paintConsumption);
            }
            else {
                printer.changeState(new FailureState(printer));
                return;
            }
        }
        printer.changeState(new WaitingState(printer));
    }

    private boolean isPrintingAvailable() {
        return printer.getPaintPercentage() - paintConsumption >= 0
            && printer.getPagesCount() - 1 >= 0;
    }
}

```



```

public class WaitingState extends State {
    public WaitingState(Printer printer) {
        super(printer);
        stateName = "WaitingState";
    }

    @Override
    public void print(int numPaper) {
        printer.changeState(new PrintingState(printer));
        printer.print(numPaper);
    }

    @Override
    public void loadPaper(int numPaper) {
        printer.setPagesCount(printer.getPagesCount() + numPaper);
    }

    @Override
    public void refillCartridge() {
        printer.setPaintPercentage(100);
    }
}

public class FailureState extends State {
    public FailureState(Printer printer) {
        super(printer);
        stateName = "FailureState";
    }

    @Override
    public void loadPaper(int numPaper) {
        printer.setPagesCount(printer.getPagesCount() + numPaper);
        if (isPrintingAvailable())
            printer.changeState(new WaitingState(printer));
    }

    @Override
    public void refillCartridge() {
        printer.setPaintPercentage(100);
        if (isPrintingAvailable())
            printer.changeState(new WaitingState(printer));
    }

    private boolean isPrintingAvailable() {
        return printer.getPaintPercentage() > 0
            && printer.getPagesCount() > 0;
    }
}

public class JammingState extends State {
    public JammingState(Printer printer) {
        super(printer);
        stateName = "JammingState";
    }

    @Override
    public void extractJammingPaper() {
        printer.changeState(new WaitingState(printer));
    }
}

```

Входные данные:

```
public class PrinterTest {
    public static void main(String[] args) {
        Printer printer = new Printer("HP", 5, 100, 10);
        System.out.println("New printer:");
        System.out.println(printer);

        System.out.println("\nPrinted 8 papers:");
        printer.print(8);
        System.out.println(printer);

        System.out.println("\nLoaded 10 papers:");
        printer.loadPaper(10);
        System.out.println(printer);

        System.out.println("\nPrinted 5 papers:");
        printer.print(5);
        System.out.println(printer);
    }
}
```

Результат работы программы:

```
D:\SDK\JDK\bin\java.exe "-javaagent:D:\JetBrains\IntelliJ IDEA 2023.3.4\lib\idea_rt.jar=50674:D:
New printer:
Printer{model='HP', pageCount=5, paintPercentage=100.0, state=State{stateName='WaitingState'}}

Printed 8 papers:
Printer{model='HP', pageCount=0, paintPercentage=75.0, state=State{stateName='FailureState'}}

Loaded 10 papers:
Printer{model='HP', pageCount=10, paintPercentage=75.0, state=State{stateName='WaitingState'}}

Printed 5 papers:
Printer{model='HP', pageCount=6, paintPercentage=55.0, state=State{stateName='JammingState'}}

Process finished with exit code 0
```

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