МИНИСТЕРСТВО ОБРАЗОВАНИЯ РЕСПУБЛИКИ БЕЛАРУСЬ
УЧРЕЖДЕНИЕ ОБРАЗОВАНИЯ
“БРЕСТСКИЙ ГОСУДАРСТВЕННЫЙ ТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ”
ФАКУЛЬТЕТ ЭЛЕКТРОННО-ИНФОРМАЦИОННЫХ СИСТЕМ
КАФЕДРА ИНТЕЛЛЕКТУАЛЬНЫХ ИНФОРМАЦИОННЫХ ТЕХНОЛОГИЙ

Отчет по лабораторной работе №4

Специальность ПО-9

Выполнил
А. С. Мисиюк,
студент группы ПО-9
Проверил
А. А. Крощенко,
ст. преп. кафедры ИИТ,
«__» _____ 2024 г.

Брест 2024

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

Вариант № 2

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 Payment (покупка) с внутренним классом, с помощью объектов которого можно сформировать покупку из нескольких товаров.

Код программы:

```
@SuppressWarnings("FieldMayBeFinal")
public class Payment {
    private List<Item> items;

    public Payment() {
        items = new ArrayList<>();
    }

    public void addItem(String name, double price) {
        Item item = new Item(name, price);
        items.add(item);
    }

    public double getTotalAmount() {
        double total = 0;
        for (Item item : items) {
            total += item.getPrice();
        }
        return total;
    }

    public void printItems() {
        for (Item item : items) {
            System.out.println(item.getName() + ": " + item.getPrice());
        }
    }

    @SuppressWarnings("InnerClassMayBeStatic")
    private class Item {
        private String name;
        private double price;

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

        public String getName() {
            return name;
        }

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

```
public class Task1
{
    public static void main(String[] args)
    {
        Payment payment = new Payment();

        payment.addItem("Товар 1", 10.0);
        payment.addItem("Товар 2", 20.0);
        payment.addItem("Товар 3", 30.0);

        payment.printItems();

        double totalAmount = payment.getTotalAmount();
        System.out.println("Общая стоимость: " + totalAmount);
    }
}
```

Пример

```
>java.exe Task1
Товар 1: 10.0
Товар 2: 20.0
Товар 3: 30.0
Общая стоимость: 60.0
```

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.

Создать класс Абзац, используя класс Строка.

Код программы:

```
@SuppressWarnings("FieldMayBeFinal")
public class CustomString
{
    private String text;

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

    public String getText()
    {
        return text;
    }
}

@SuppressWarnings("FieldMayBeFinal")
public class Paragraph
{
    // агрегация (1 парграф - 0..n строк)
    private List<CustomString> strings;

    public Paragraph()
    {
        strings = new ArrayList<>();
    }
```

```
    public void addString(CustomString customString)
    {
        strings.add(customString);
    }

    public void printParagraph()
    {
        for (CustomString customString : strings)
        {
            System.out.println(customString.getText());
        }
    }
}

public class Task2
{
    public static void main(String[] args)
    {
        Paragraph paragraph = new Paragraph();

        CustomString string1 = new CustomString("First string.");
        CustomString string2 = new CustomString("Second string.");
        CustomString string3 = new CustomString("Third string.");

        paragraph.addString(string1);
        paragraph.addString(string2);
        paragraph.addString(string3);

        paragraph.printParagraph();
    }
}
```

Пример

```
>java.exe Task2
First string.
Second string.
Third string.
```

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 **Платежи**. **Клиент** имеет **Счет** в банке и **Кредитную Карту** (**КК**). **Клиент** может оплатить **Заказ**, сделать платеж на другой **Счет**, заблокировать **КК** и аннулировать **Счет**. **Администратор** может заблокировать **КК** за превышение кредита.

Код программы:

```
public class Task3
{
    public static void main(String[] args)
    {
        Client client = new Client("Иванов", 30, "ул. Пушкина, д.10");
```

```
        Account account = new Account("1234567890", 1000.0);
        client.setAccount(account);

        CreditCard creditCard = new CreditCard("9876543210", account, 5000.0);
        client.setCreditCard(creditCard);

        Order order = new Order("Z123", 500.0);

        client.payOrder(order);
        client.payOrder(order);

        Account otherAccount = new Account("0987654321", 2000.0);
        client.makePayment(otherAccount, 300.0);
        client.makePayment(otherAccount, 300.0);

        Administrator administrator = new Administrator();
        administrator.blockCreditCard(creditCard);

        client.payOrder(order);

        client.blockCreditCard();

        client.cancelAccount();
        client.payOrder(order);
    }
}

public class Administrator
{
    public void blockCreditCard(CreditCard card)
    {
        card.setBlocked(true);
        System.out.println("Блокировка кредитной карты " + card.getCardNumber() + "
администратором");
    }
}

@SuppressWarnings("FieldMayBeFinal")
public class CreditCard
{
    private String cardNumber;
    private double creditLimit;
    private Account account;
    private boolean isBlocked;

    public CreditCard(String cardNumber, Account account, double creditLimit)
    {
        this.cardNumber = cardNumber;
        this.account = account;
        this.creditLimit = creditLimit;
        this.isBlocked = false;
    }

    public String getCardNumber()
    {
        return cardNumber;
    }

    @SuppressWarnings("unused")
    public double getCreditLimit()
    {
        return creditLimit;
    }

    public Account getAccount()
    {
        return (!this.isBlocked) ? account : null;
```

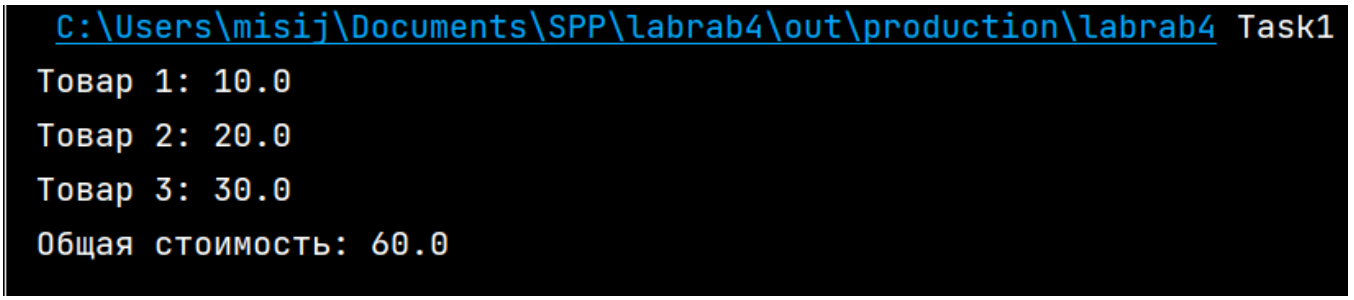
```
    }

    public void setBlocked(boolean blocked)
    {
        isBlocked = blocked;
    }
}

@SuppressWarnings({"FieldMayBeFinal", "unused"})
public class Account
{
    private String accountNumber;
    private double balance;

    public Account(String accountNumber, double balance)
    {
        this.accountNumber = accountNumber;
        this.balance = balance;
    }

    public String getAccountNumber()
    {
        return accountNumber;
    }

    public double getBalance()
    {
        return balance;
    }

    public boolean pay(double amount)
    {
        if (this.balance - amount < 0) {
            return false;
        }

        this.balance -= amount;
        return true;
    }
}

@SuppressWarnings({"FieldMayBeFinal"})
public class Client
{
   // { ... }

    /** Клиент может оплатить Заказ
     * @param order заказ, который клиент хочет оплатить
     */
    public void payOrder(Order order)
    {
        Account acc;
        if (this.creditCard != null) {
            acc = this.creditCard.getAccount();
        }
        else if (this.account != null) {
            acc = this.account;
        }
        else {
            System.out.println("У клиента " + name + " нет счета");
            return;
        }

        if (acc == null) {
            System.out.println("Карточка клиента " + this.name + " заблокирована!");
            return;
        }
```

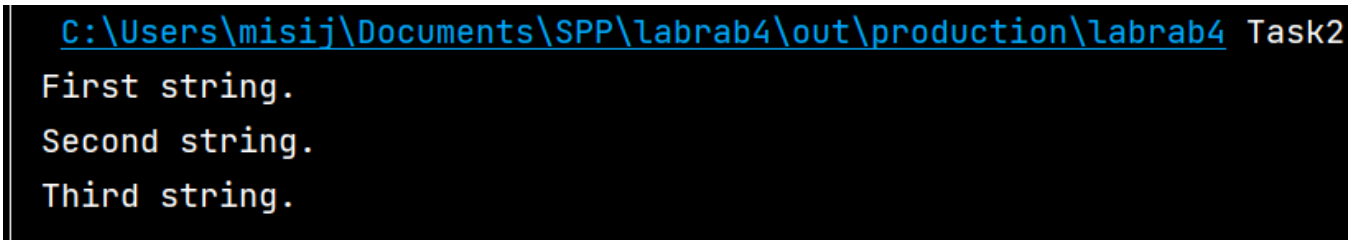
```
        if (order.isPayed()) {
            System.out.println("Заказ №" + order.getOrderNumber() + " уже оплачен");
            return;
        }

        if (!acc.pay(order.getOrderAmount())) {
            System.out.println("Нехватает баланса");
            return;
        }
        order.ConfirmOrder();

        System.out.println("Оплата заказа " + order.getOrderNumber() + " клиентом " + this.name);
    }

    /** Сделать платеж на другой Счет
     * @param targetAccount другой Счет
     * @param amount сумма платежа
     */
    public void makePayment(Account targetAccount, double amount)
    {
        Account acc;
        if (this.creditCard != null) {
            acc = this.creditCard.getAccount();
        }
        else if (this.account != null) {
            acc = this.account;
        }
        else {
            System.out.println("У клиента " + name + " нет счета");
            return;
        }

        if (acc == null) {
            System.out.println("Карточка клиента " + this.name + " заблокирована!");
            return;
        }

        if (!acc.pay(amount)) {
            System.out.println("Нехватает баланса");
            return;
        }
        targetAccount.pay(-amount);

        System.out.println("Выполнение платежа на счет " + targetAccount.getAccountNumber() + " в
размере " + amount);
    }

    /** Заблокировать КК
     */
    public void blockCreditCard()
    {
        this.creditCard.setBlocked(true);
        System.out.println("Блокировка кредитной карты клиента " + name);
    }

    /** Аннулировать Счет
     */
    public void cancelAccount()
    {
        this.account = null;
        this.creditCard = null;

        System.out.println("Аннулирование счета клиента " + name);
    }
}
```

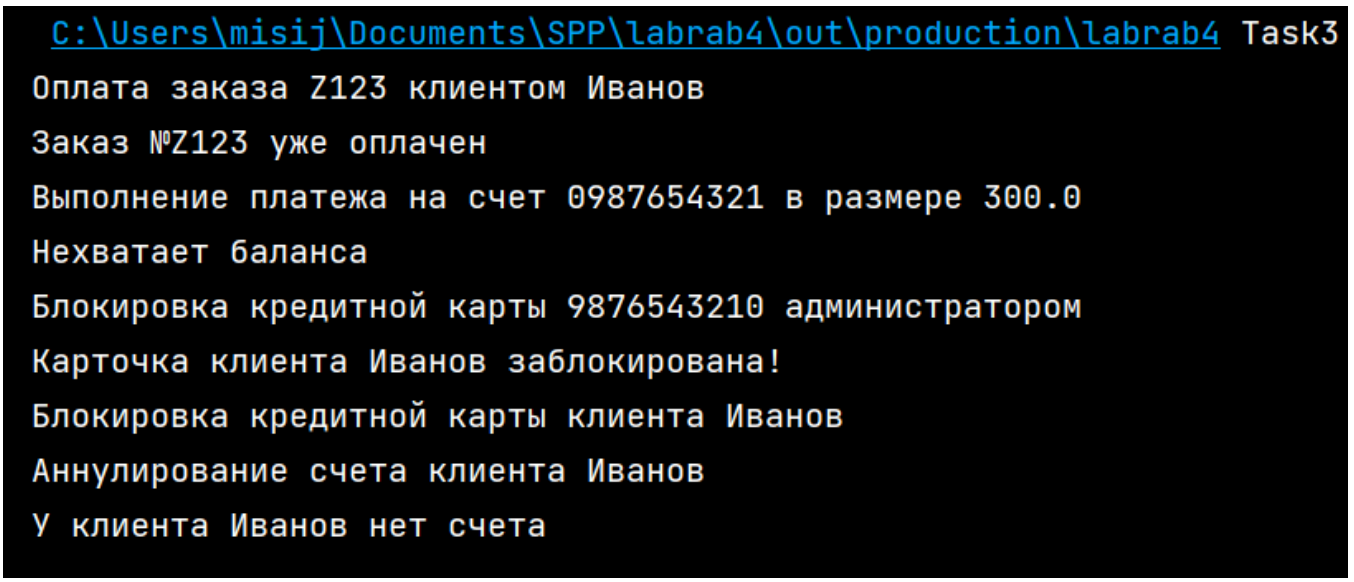
Пример

```
>java.exe Task3
Оплата заказа Z123 клиентом Иванов
Заказ №Z123 уже оплачен
Выполнение платежа на счет 0987654321 в размере 300.0
Нехватает баланса
Блокировка кредитной карты 9876543210 администратором
Карточка клиента Иванов заблокирована!
Блокировка кредитной карты клиента Иванов
Аннулирование счета клиента Иванов
У клиента Иванов нет счета
```

Рисунки с результатами работы программы







Вывод: исследовал создание классов и объектное-ориентированное программирование на языке программирования Java.