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

ОТЧЁТ

по лабораторной работе №4

Выполнила:
студентка 3 курса
группы ПО-9
Бердникова В.А.

Проверил:
Крощенко А.А.

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.

Входные данные:

```
payment.addProduct( name: "Milk",  price: 2.5);
payment.addProduct( name: "Bread",  price: 1.0);
payment.addProduct( name: "Oranges",  price: 3.0);
```

```
payment.addProduct( name: "Ice-cream",  price: 5.0);
payment.removeProduct( name: "Milk");
```

Выходные данные:

```
D:\ForJava\Java\j
Payment list:
Milk: 2.5
Bread: 1.0
Oranges: 3.0
Full Cost: 6.5

Payment list:
Bread: 1.0
Oranges: 3.0
Ice-cream: 5.0
Full Cost: 9.0
```

Код программы:

Main.java

```
public class Main {
    public static void main(String[] args) {
        Payment payment = new Payment();
        payment.addProduct("Milk", 2.5);
        payment.addProduct("Bread", 1.0);
        payment.addProduct("Oranges", 3.0);

        payment.printList();
        System.out.println("Full Cost: " + payment.getFullCost()+"\n");

        payment.addProduct("Ice-cream", 5.0);
        payment.removeProduct("Milk");
```

```
        payment.printList();
        System.out.println("Full Cost: " + payment.getFullCost());

    }
}
```

Payment.java

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

public class Payment {
    private List<Product> products;

    public Payment() {
        this.products = new ArrayList<>();
    }

    public void addProduct(String name, double price) {
        Product product = new Product(name, price);
        products.add(product);
    }

    public double getFullCost() {
        double cost = 0;
        for (Product pr : products) {
            cost += pr.getPrice();
        }
        return cost;
    }

    public void printList() {
        System.out.println("Payment list:");
        for (Product pr : products) {
            System.out.println(pr.name + ": " + pr.price);
        }
    }

    public void removeProduct(String name) {
        products.removeIf(product -> product.name.equals(name));
    }

    private static class Product {
        private String name;
        private double price;

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

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

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.

Входные данные:

```
MyString line1 = new MyString( text: "The first string");
MyString line2 = new MyString( text: "This is the second string");
MyString line3 = new MyString( text: "And here is another one");
```

Выходные данные:

```
The first string
This is the second string
And here is another one
Total length of strings: 3
```

Код программы:

Task2.java

```
public class Task2 {
    public static void main(java.lang.String[] args) {
        MyString line1 = new MyString("The first string");
        MyString line2 = new MyString("This is the second string");
        MyString line3 = new MyString("And here is another one");

        Paragraph paragraph = new Paragraph();
        paragraph.addString(line1);
        paragraph.addString(line2);
        paragraph.addString(line3);

        paragraph.printParagraph();
        System.out.println("Total length of strings: " +
paragraph.getStringCount());
    }
}
```

Paragraph.java

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

public class Paragraph {
    private List<MyString> lines;

    public Paragraph() {
        this.lines = new ArrayList<>();
    }

    public void addString(MyString line) {
        lines.add(line); }

    public void printParagraph() {
        for (MyString line : lines) {
            line.printString();   }
    }

    public int getStringCount() {
        return lines.size();
    }
}
```

MyString.java

```
public class MyString {
    String text;

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

    public void printString() {
        System.out.println(text);
    }
}
```

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.

Входные данные:

```
Account account = new Account( balance: 1000);
CreditCard creditCard = new CreditCard( creditLimit: 2000);
Client client = new Client( name: "John", account, creditCard);
Order order = new Order( totalPrice: 500);
```

```
Account otherAcc = new Account( balance: 0);
client.transferMoney( amount: 200,otherAcc);
```

Выходные данные:

```
Оплата заказа прошла успешно
Текущий долг на карте: 500.0
Кредитная карта заблокирована Администратором
Карта разблокирована
Снято со счета: 200.0
Перевод денег прошел успешно: 200.0
Счет получателя: 200.0
Счет отправителя: 800.0
```

Код программы:

Client.java

```
public class Client {
    private String name;
    private Account account;
    private CreditCard creditCard;
```

```
    public Client(String name, Account account, CreditCard creditCard) {
        this.name = name;
        this.account = account;
        this.creditCard = creditCard;
    }

    public void makePayment(Order order) {
        double totalPrice = order.getTotalPrice();
        if (creditCard.isBlocked()){
            System.out.println("Карта заблокирована. Невозможно провести
операцию");
            return;
        }
        if( creditCard.getAvailableCredit() >= totalPrice) {
            creditCard.increaseDebt(totalPrice);
            order.setPaid(true);
            System.out.println("Оплата заказа прошла успешно");
        } else {
            System.out.println("Недостаточно средств на кредитной карте.");
        }
    }

    public void transferMoney(double amount, Account recipientAccount) {
        if (creditCard.isBlocked()){
            System.out.println("Карта заблокирована. Невозможно провести
операцию");
            return;
        }
        if (account.getBalance() >= amount) {
            account.withdraw(amount);
            recipientAccount.deposit(amount);
            System.out.println("Перевод денег прошел успешно: " + amount);
        } else {
            System.out.println("Недостаточно средств на счете.");
        }
    }

    public void blockCreditCard() {
        creditCard.block();
        System.out.println("Кредитная карта заблокирована Клиентом");
    }

    public void cancelAccount() {
        account.cancel();
    }}
```

CreditCard.java

```
public class CreditCard {
    private double creditLimit;
    private double currentDebt;
    private boolean blocked;

    public CreditCard(double creditLimit) {
        this.creditLimit = creditLimit;
        this.currentDebt = 0;
        this.blocked = false;
    }

    public void block() {
        this.blocked = true;
    }

    public void unblock() {
```

```
        this.blocked = false;
        System.out.println("Карта разблокирована");
    }

    public double getAvailableCredit() {
        return creditLimit - currentDebt;
    }

    public void increaseDebt(double amount) {
        currentDebt += amount;
    }


    public boolean isBlocked() {
        return blocked;
    }

    public double getCurrentDebt(){
        return currentDebt;
    }

    public double getCreditLimit(){
        return creditLimit;
    }
}
```

Account.java

```
public class Account {
    private double balance;
    private boolean active;

    public Account(double balance) {
        this.balance = balance;
        this.active = true;
    }

    public void deposit(double amount) {
        balance += amount;
    }

    public void withdraw(double amount) {
        if (balance >= amount) {
            balance -= amount;
            System.out.println("Снято со счета: " + amount);
        } else {
            System.out.println("Недостаточно средств на счете.");
        }
    }

    public void cancel() {
        this.active = false;
        System.out.println("Счет аннулирован");
    }

    public double getBalance() {
        return balance;
    }
}
```

Admin.java

```
public class Admin {
    public void blockCreditCardForExceedingCredit(CreditCard creditCard) {
            creditCard.block();
```

```
            System.out.println("Кредитная карта заблокирована
Администратором");

    }}
```

Order.java

```
public class Order {
    private double totalPrice;
    private boolean paid;

    public Order(double totalPrice) {
        this.totalPrice = totalPrice;
        this.paid = false;
    }

    public double getTotalPrice() {
        return totalPrice;
    }

    public boolean isPaid() {
        return paid;
    }

    public void setPaid(boolean paid) {
        this.paid = paid;
    }
}
```

Task3.java

```
public class Task3 {
    public static void main(String[] args) {
        Account account = new Account(1000);
        CreditCard creditCard = new CreditCard(2000);
        Client client = new Client("John", account, creditCard);
        Order order = new Order(500);
        client.makePayment(order);

        double debt = creditCard.getCurrentDebt();
        System.out.println("Текущий долг на карте: " + debt);

        Admin admin = new Admin();
        admin.blockCreditCardForExceedingCredit(creditCard);

        creditCard.unblock();

        Account otherAcc = new Account(0);
        client.transferMoney(200,otherAcc);
        double otherBalance = otherAcc.getBalance();
        System.out.println("Счет получателя: " + otherBalance);
        double balance = account.getBalance();
        System.out.println("Счет отправителя: " + balance);

    }}
```

Вывод: приобрела практические навыки в области объектно-ориентированного проектирования.