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

ОТЧЁТ
по лабораторной работе №5

Выполнила
студентка группы ПО-9
Тупик Ю. Л.

Проверил:
Крощенко А. А.

Брест 2024

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

Задание 1
Реализовать абстрактные классы или интерфейсы, а также наследование и полиморфизм для следующих классов:
8) interface Врач ← class Хирург ← class Нейрохирург.

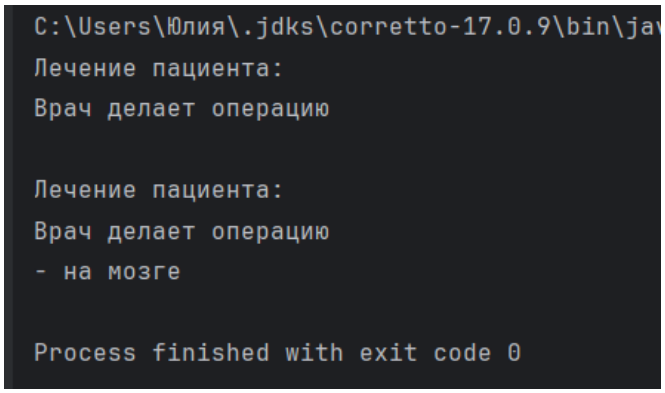
Выполнение задания

Код программы

```
// Определение интерфейса Doctor с методом treatPatient()
interface Doctor {
    void treatPatient();
}

// Класс-реализация интерфейса Doctor, представляющий Хирурга
class Surgeon implements Doctor {
    // Переопределение метода treatPatient(), который выводит сообщение о
проведении операции
    @Override
    public void treatPatient() {
        System.out.println("Врач делает операцию");
    }
}

// Класс-реализация интерфейса Doctor, представляющий Нейрохирурга
class Neurosurgeon extends Surgeon {
    // Переопределение метода treatPatient() с использованием метода
суперкласса и добавлением дополнительного сообщения
    @Override
    public void treatPatient() {
        super.treatPatient(); // Вызов метода суперкласса
        System.out.println("- на мозге"); // Дополнительное сообщение
    }
}

// Класс, представляющий пациента
class Patient {
    // Конструктор класса, принимающий объект типа Doctor и выводящий
сообщение о лечении пациента
    public Patient (Doctor attendingDoctor) {
        System.out.println("Лечение пациента:");
        attendingDoctor.treatPatient();
    }
}

public class Task1 {
    public static void main(String[] args) {
        Surgeon Alex = new Surgeon();
        Neurosurgeon Anna = new Neurosurgeon();

        new Patient(Alex); // Создание объекта Пациента с Хирургом
        System.out.println();
```

```
        new Patient(Anna); // Создание объекта Пациента с Нейрохирургом
    }
}
```

Результат



Задание 2
Создать суперкласс (абстрактный класс, интерфейс) и определить общие методы для данного класса. Создать подклассы, в которых добавить специфические свойства и методы. Часть методов переопределить. Создать массив объектов суперкласса и заполнить объектами подклассов. Объекты подклассов идентифицировать конструктором по имени или идентификационному номеру. Использовать объекты подклассов для моделирования реальных ситуаций и объектов.
8) Создать суперкласс Пассажироперевозчик и подклассы Самолет, Поезд, Автомобиль. Определить время и стоимость передвижения.

Выполнение задания

Код программы

```
abstract class PassengerCarrier {
    protected int travelTimeInMin;
    protected int costOfTravel;

    // Конструктор суперкласса
    public PassengerCarrier(int travelTimeInSec, int costOfTravel) {
        this.travelTimeInMin = travelTimeInSec;
        this.costOfTravel = costOfTravel;
    }

    // Общий метод для путешествия
    public void travel() {
        System.out.println("Пассажироперевозчик доберется до места
назначения за "
                + travelTimeInMin
                + " минут, поездка обойдется вам в "
                + costOfTravel + "$");
    }

    // Общий метод для старта
    public void start() {
        System.out.println("Пассажироперевозчик отправляется");
    }

    // Общий метод для остановки
    public void stop() {
        System.out.println("Пассажироперевозчик прибыл в пункт назначения.
" +
                "Благодарим вас за то, что выбрали нас.");
    }
}

class Plane extends PassengerCarrier {
    String airline;

    // Конструктор подкласса Самолет
    public Plane (int travelTimeInSec, int costOfTravel, String airline) {
        super(travelTimeInSec, costOfTravel);
        this.airline = airline;
    }
```

```
    // Переопределение метода для путешествия на самолете
    @Override
    public void travel() {
        System.out.println("Самолет долетит до места назначения за "
                + travelTimeInMin
                + " минут, перелет обойдется вам в "
                + costOfTravel + "$");
    }

    // Дополнительный метод для вывода информации об авиакомпании
    public void fly() {
        System.out.println("Вы летите рейсом авиакомпании \"" + airline +
"\"");
    }
}

class Train extends PassengerCarrier {
    boolean isElectric;

    // Конструктор подкласса Поезд
    public Train (int travelTimeInSec, int costOfTravel, boolean
isElectric) {
        super(travelTimeInSec, costOfTravel);
        this.isElectric = isElectric;
    }

    // Переопределение метода для путешествия на поезде
    @Override
    public void travel() {
        System.out.println("Поезд прибывает в пункт назначения за "
                + travelTimeInMin
                + " минут, проезд обойдется вам в "
                + costOfTravel + "$");
    }

    // Дополнительный метод для вывода информации о типе поезда
    public void showType () {
        if (isElectric) System.out.println("Вы будете ехать на
электропоезде");
        else System.out.println("Вы будете ехать на старом поезде");
    }
}

class Car extends PassengerCarrier {
    String brand;

    // Конструктор подкласса Автомобиль
    public Car (int travelTimeInSec, int costOfTravel, String brand) {
        super(travelTimeInSec, costOfTravel);
        this.brand = brand;
    }

    // Переопределение метода для путешествия на автомобиле
    @Override
```

```
    public void travel() {
        System.out.println("Автомобиль доедет до места назначения за "
                + travelTimeInMin
                + " минут, поездка обойдется вам в "
                + costOfTravel + "$");
    }

    // Дополнительный метод для вывода информации об автомобиле
    public void showOffModel () {
        System.out.println("Вы будете ехать на автомобиле " + brand);
    }
}

public class Task2 {
    public static void main(String[] args) {
        PassengerCarrier[] carriers = new PassengerCarrier[3];

        carriers[0] = new Plane(300, 999, "Aviasales");
        carriers[1] = new Train(880, 299, true);
        carriers[2] = new Car(1500, 99, "BMW");

        // Итерация по массиву объектов для моделирования ситуаций
        for (PassengerCarrier carrier : carriers) {
            carrier.start();
            carrier.travel();
            // Проверка типа объекта и вызов соответствующего метода
            if (carrier instanceof Plane) {
                ((Plane) carrier).fly();
            } else if (carrier instanceof Train) {
                ((Train) carrier).showType();
            } else if (carrier instanceof Car) {
                ((Car) carrier).showOffModel();
            }
            carrier.stop();
            System.out.println();
        }
    }
}
```

Результат

```
C:\Users\Юлия\.jdks\corretto-17.0.9\bin\java.exe "-javaagent:C:\Program Files\JetBrains\Int
Пассажироперевозчик отправляется
Самолет долетит до места назначения за 300 минут, перелет обойдется вам в 999$
Вы летите рейсом авиакомпании "Aviasales"
Пассажироперевозчик прибыл в пункт назначения. Благодарим вас за то, что выбрали нас.

Пассажироперевозчик отправляется
Поезд прибывает в пункт назначения за 880 минут, проезд обойдется вам в 299$
Вы будете ехать на электропоезде
Пассажироперевозчик прибыл в пункт назначения. Благодарим вас за то, что выбрали нас.

Пассажироперевозчик отправляется
Автомобиль доедет до места назначения за 1500 минут, поездка обойдется вам в 99$
Вы будете ехать на автомобиле BMW
Пассажироперевозчик прибыл в пункт назначения. Благодарим вас за то, что выбрали нас.


Process finished with exit code 0
```

Задание 3
В задании 3 ЛР No4, где возможно, заменить объявления суперклассов объявлениями абстрактных классов или интерфейсов.

Выполнение задания

Код программы

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

// Интерфейс, описывающий товар
interface ProductDescription {
    String getDescription();
}

// Класс, представляющий товар
class Product implements ProductDescription {
    private String name;
    private double price;
    private String description;

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

    public String getName() {
        return name;
    }

    public double getPrice() {
        return price;
    }

    public String getDescription() {
        return description;
    }

    public void setDescription(String description) {
        this.description = description;
    }
}

// Класс, представляющий корзину с товарами
class Cart {
    private List<Product> products;

    public Cart() {
        products = new ArrayList<>();
    }

    public void addProduct(Product product) {
        products.add(product);
    }

    public double calculateTotalPrice() {
        double total = 0;
        for (Product product : products) {
            total += product.getPrice();
        }
        return total;
    }
```

```
}

// Абстрактный класс, обобщающий атрибуты и методы администратора и клиента
abstract class User {
    // Общие атрибуты и методы
}

// Класс, представляющий администратора
class Administrator extends User {
    static private List<String> paymentRegistry = new ArrayList<>();
    static private List<String> blacklist = new ArrayList<>();

    // Метод для регистрации оплаты товара клиентом
    public static void registerPayment(String clientName, double amount) {
        paymentRegistry.add("Клиент: " + clientName + "; Сумма: " + amount +
"руб");
        System.out.println("Клиент: " + clientName + "; Сумма: " + amount +
"руб");
    }

    // Метод для добавления клиента в "черный список"
    public static void addToBlacklist(String clientName) {
        blacklist.add(clientName);
    }

    // Метод для проверки, находится ли клиент в "черном списке"
    public static boolean isBlacklisted(String clientName) {
        return blacklist.contains(clientName);
    }

    // Метод для добавления описания товара
    public static void addProductDescription(Product product, String
description) {
        product.setDescription(description);
    }
}

// Класс, представляющий клиента
class Client extends User {
    private String name;
    private double money;
    private Cart cart;

    public Client(String name, double money) {
        this.name = name;
        this.money = money;
        this.cart = new Cart();
    }

    // Метод для добавления товара в корзину клиента
    public void addProductToCart(Product product) {
        cart.addProduct(product);
    }

    // Метод для оплаты товаров в корзине
    public void pay() {
        double totalPrice = cart.calculateTotalPrice();
        if (totalPrice <= money) {
            money -= totalPrice;
            Administrator.registerPayment(name, totalPrice);
        } else {
            Administrator.addToBlacklist(name);
```

```
            System.out.println("Клиент " + name + " не имеет достаточно денежных
средств " +
                    "и будет добавлена в \"Черный список\".");
        }
    }
}

public class Task3 {
    public static void main(String[] args) {
        Client client1 = new Client("Даша", 40);
        Client client2 = new Client("Оля", 20);

        Product product1 = new Product("Тушь для ресниц", 27);
        Product product2 = new Product("Помада для губ", 13);

        Administrator.addProductDescription(product1,
                "Супер объем со сценическим эффектом, тон 01");
        Administrator.addProductDescription(product2,
                "Жидкая стойкая матовая помада для губ, тон 15");

        client1.addProductToCart(product1);
        client1.addProductToCart(product2);

        client2.addProductToCart(product1);

        client1.pay();
        client2.pay();

        System.out.println("Оля есть в \"Черном списке\"? " +
                Administrator.isBlacklisted("Оля"));

        System.out.println(product1.getName() + " (" + product1.getDescription()
+ ")");
        System.out.println(product2.getName() + " (" + product2.getDescription()
+ ")");
    }
}
```

Результат

```
C:\Users\Юлия\.jdks\corretto-17.0.9\bin\java.exe "-javaagent:C:\Program Files\JetBrains\
Клиент: Даша; Сумма: 40.0руб
Клиент Оля не имеет достаточно денежных средств и будет добавлена в "Черный список".
Оля есть в "Черном списке"? true
Тушь для ресниц (Супер объем со сценическим эффектом, тон 01)
Помада для губ (Жидкая стойкая матовая помада для губ, тон 15)

Process finished with exit code 0
```

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