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

ОТЧЁТ

по лабораторной работе №5

Выполнила:
студентка группы ПО-9
Кот А. А.

Проверил:
Крощенко А. А.

Брест 2024

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

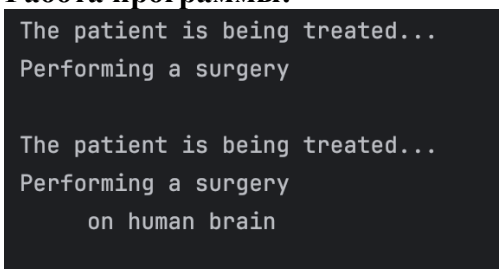
Вариант 8.

Ход работы

Задание 1.

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

Работа программы:



Код программы:

Task1.java

```
interface Doctor {
    void treatPatient();
}

class Surgeon implements Doctor {
    @Override
    public void treatPatient() {
        System.out.println("Performing a surgery");
    }
}

class Neurosurgeon extends Surgeon {
    @Override
    public void treatPatient() {
        super.treatPatient();
        System.out.println("\t on human brain");
    }
}

class Patient {
    public Patient (Doctor attendingDoctor) {
        System.out.println("The patient is being treated...");
        attendingDoctor.treatPatient();
    }
}

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

        new Patient(SallyMulligan);
        System.out.println();
        new Patient(KattyPerry);
    }
```

```
}
```

Задание 2.

В следующих заданиях требуется 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.

Создать суперкласс Пассажироперевозчик и подклассы Самолет, Поезд, Автомобиль. Определить время и стоимость передвижения.

Работа программы:

```
Carrier departs...
The plane will fly to its destination in 3600 minutes. Travel will cost you 500
You're flying by "Belavia" airline!
Carrier has reached its destination. Thank you for using our carrier.

Carrier departs...
The train will choo-choo to its destination in 7200 minutes. Travel will cost you 200
You're travelling by electric train.
Carrier has reached its destination. Thank you for using our carrier.

Carrier departs...
The car will drive to its destination in 18000 minutes. Travel will cost you 100
My car is of Toyota Corolla brand!
Carrier has reached its destination. Thank you for using our carrier.
```

Код программы:

Task2.java

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

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

    public void travel() {
        System.out.println("The carrier will reach its destination in "
                + travelTimeInMin
                + " minutes. Travel will cost you "
                + costOfTravel);
    }

    public void start() {
        System.out.println("Carrier departs...");
    }

    public void stop() {
        System.out.println("Carrier has reached its destination. " +
                "Thank you for using our carrier.");
    }
}
```

```
class Plane extends PassengerCarrierAbstract {
    String airline;
    public Plane (int travelTimeInSec, int costOfTravel, String airline) {
        super(travelTimeInSec, costOfTravel);
        this.airline = airline;
    }
    @Override
    public void travel() {
        System.out.println("The plane will fly to its destination in "
                + travelTimeInMin
                + " minutes. Travel will cost you "
                + costOfTravel);
    }

    public void fly() {
        System.out.println("You're flying by \"" + airline + "\" airline!");
    }
}

class Train extends PassengerCarrierAbstract {
    boolean isElectric;
    public Train (int travelTimeInSec, int costOfTravel, boolean isElectric)
{
        super(travelTimeInSec, costOfTravel);
        this.isElectric = isElectric;
    }

    @Override
    public void travel() {
        System.out.println("The train will choo-choo to its destination in "
                + travelTimeInMin
                + " minutes. Travel will cost you "
                + costOfTravel);
    }

    public void showType () {
        if (isElectric) System.out.println("You're travelling by electric
train.");
        else System.out.println("You're travelling by an old type of
train.");
    }
}

class Car extends PassengerCarrierAbstract {
    String brand;

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

    @Override
    public void travel() {
        System.out.println("The car will drive to its destination in "
                + travelTimeInMin
                + " minutes. Travel will cost you "
                + costOfTravel);
    }

    public void showOffModel () {
        System.out.println("My car is of " + brand + " brand!");
    }
}

public class Task2 {
    public static void main(String[] args) {
```

```
        PassengerCarrierAbstract[] carriers = new
PassengerCarrierAbstract[3];

        carriers[0] = new PlaneAbstract(3600, 500, "Belavia");
        carriers[1] = new TrainAbstract(7200, 200, true);
        carriers[2] = new CarAbstract(18000, 100, "Toyota Corolla");

        for (PassengerCarrierAbstract carrier : carriers) {
            carrier.start();
            carrier.travel();
            if (carrier instanceof PlaneAbstract) {
                ((PlaneAbstract) carrier).fly();
            } else if (carrier instanceof TrainAbstract) {
                ((TrainAbstract) carrier).showType();
            } else if (carrier instanceof CarAbstract) {
                ((CarAbstract) carrier).showOffModel();
            }
            carrier.stop();
            System.out.println();
        }
    }
}
```

Задание 3.

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

В задании целесообразно заменить только базовый класс PassengerCarrier на абстрактный. Поменяется лишь этот класс.

Работа программы: идентична заданию 2.

Код программы:
Task3.java (класс PassengerCarrier, остальные классы не изменились)

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

    public PassengerCarrierAbstract(int travelTimeInSec, int costOfTravel) {
        this.travelTimeInMin = travelTimeInSec;
        this.costOfTravel = costOfTravel;
    }

    public abstract void travel();

    public void start() {
        System.out.println("Carrier departs...");
    }

    public void stop() {
        System.out.println("Carrier has reached its destination. " +
                "Thank you for using our carrier.");
    }
}
```

Вывод: в ходе лабораторной работы мы приобрели практические навыки в области объектно-ориентированного проектирования.