МИНИСТЕРСТВО ОБРАЗОВАНИЯ РЕСПУБЛИКИ БЕЛАРУСЬ
УЧРЕЖДЕНИЕ ОБРАЗОВАНИЯ
“БРЕСТСКИЙ ГОСУДАРСТВЕННЫЙ ТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ”
КАФЕДРА ИИТ

ОТЧЁТ

по лабораторной работе №5

Выполнил:
студент 3 курса
группы ПО-9
Тусюк Т.В.

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:

9) interface Корабль ← class Грузовой Корабль ← class Танкер.

Выполнение задания:

```
interface Ship {
    void sail();
}

class CargoShip implements Ship {
    @Override
    public void sail() {
        System.out.println("Грузовой корабль плывет по воде.");
    }

    public void loadCargo() {
        System.out.println("Грузовой корабль загружает груз.");
    }
}

class Tanker implements Ship {
    @Override
    public void sail() {
        System.out.println("Танкер плывет по воде.");
    }

    public void loadOil() {
        System.out.println("Танкер загружает нефть.");
    }
}

public class Task_01 {
    public static void main(String[] args) {

        CargoShip cargoShip = new CargoShip();
        Tanker tanker = new Tanker();

        cargoShip.sail();
        cargoShip.loadCargo();

        tanker.sail();
        tanker.loadOil();
    }
}
```

Результат:

```
"C:\Program Files\Java\jdk-21\bin\
Грузовой корабль плывет по воде.
Грузовой корабль загружает груз.
Танкер плывет по воде.
Танкер загружает нефть.
```

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.
1) Создать суперкласс Транспортное средство и подклассы Автомобиль, Велосипед, Повозка. Подсчитать время и стоимость перевозки пассажиров и грузов каждым транспортным средством.
Выполнение задания:

```
abstract class Transport {
    protected String name;
    protected int passengers;
    protected double cargoCapacity;

    public Transport(String name, int passengers, double cargoCapacity) {
        this.name = name;
        this.passengers = passengers;
        this.cargoCapacity = cargoCapacity;
    }

    public abstract double calculatePassengerTime();

    public abstract double calculateCargoTime(double cargoWeight);

    public void displayInfo() {
        System.out.println("Транспортное средство: " + name);
        System.out.println("Количество пассажиров: " + passengers);
        System.out.println("Грузоподъемность: " + cargoCapacity + " кг");
    }
}

class Car extends Transport {
    private double speed;

    public Car(String name, int passengers, double cargoCapacity, double speed) {
        super(name, passengers, cargoCapacity);
        this.speed = speed;
    }

    @Override
    public double calculatePassengerTime() {
        double distance = 100;
        return distance / speed;
    }

    @Override
    public double calculateCargoTime(double cargoWeight) {
        return cargoWeight / cargoCapacity;
    }
}
```

```
class Bicycle extends Transport {
    private int speed;

    public Bicycle(String name, int passengers, double cargoCapacity, int speed) {
        super(name, passengers, cargoCapacity);
        this.speed = speed;
    }

    @Override
    public double calculatePassengerTime() {
        double distance = 100;
        return distance / speed;
    }

    @Override
    public double calculateCargoTime(double cargoWeight) {
        return 0;
    }
}

class Cart extends Transport {
    private int speed;

    public Cart(String name, int passengers, double cargoCapacity, int speed) {
        super(name, passengers, cargoCapacity);
        this.speed = speed;
    }

    @Override
    public double calculatePassengerTime() {
        double distance = 100;
        return distance / speed;
    }

    @Override
    public double calculateCargoTime(double cargoWeight) {
        return cargoWeight / cargoCapacity;
    }
}

public class Task_02 {
    public static void main(String[] args) {
        Transport car = new Car("Машина", 3, 500, 80);
        Transport bicycle = new Bicycle("Велосипед", 1, 10, 20);
        Transport cart = new Cart("Повозка", 2, 200, 10);

        car.displayInfo();
        System.out.println("Время перевозки пассажиров в машине: " +
car.calculatePassengerTime() + " ч");
        System.out.println("Время перевозки груза в машине: " +
car.calculateCargoTime(300) + " ч");

        bicycle.displayInfo();
        System.out.println("Время перевозки пассажира на велосипеде: " +
bicycle.calculatePassengerTime() + " ч");

        cart.displayInfo();
        System.out.println("Время перевозки груза на повозке: " +
cart.calculateCargoTime(150) + " ч");
    }
}
```

Результат:

```
"C:\Program Files\Java\jdk-21\bin\java.exe" "-ja
Транспортное средство: Машина
Количество пассажиров: 3
Грузоподъемность: 500.0 кг
Время перевозки пассажиров в машине: 1.25 ч
Время перевозки груза в машине: 0.6 ч
Транспортное средство: Велосипед
Количество пассажиров: 1
Грузоподъемность: 10.0 кг
Время перевозки пассажира на велосипеде: 5.0 ч
Транспортное средство: Повозка
Количество пассажиров: 2
Грузоподъемность: 200.0 кг
Время перевозки груза на повозке: 0.75 ч
```

Задание 3:

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

9) Система Железнодорожная касса. Пассажир делает Заявку на станцию назначения, время и дату поездки. Система регистрирует Заявку и осуществляет поиск подходящего Поезда.Пассажир делает выбор Поезда и получает Счет на оплату. Администратор вводит номера Поездов, промежуточные и конечные станции, цены.

Выполнение задания:

```
import java.util.*;

interface Transport {
    double calculatePassengerTime();
    double calculateCargoTime(double cargoWeight);
    void displayInfo();
}

interface Passenger {
    void createRequest(String destinationStation, Date dateTime);
    void chooseTrain(ArrayList<Train> trains);
    Invoice getInvoice(double price);
}

interface Transportation {
    void addStation(Station station);
    double getPriceForStation(Station station);
}

interface Priceable {
    void setPrice(Station station, double price);
}

class PassengerImpl implements Passenger {
```

```
    private String name;
    private String passportNumber;

    public PassengerImpl(String name, String passportNumber) {
        this.name = name;
        this.passportNumber = passportNumber;
    }

    @Override
    public String toString() {
        return "Пассажир: " + name + ", Паспорт: " + passportNumber;
    }

    public void createRequest(String destinationStation, Date dateTime) {
        System.out.println("Заявка создана на станцию: " + destinationStation + " на
" + dateTime);
    }

    public void chooseTrain(ArrayList<Train> trains) {
        Train chosenTrain = trains.get(0);
        System.out.println("Выбран поезд: " + chosenTrain.getTrainNumber());
    }

    public Invoice getInvoice(double price) {
        System.out.println("Счет выставлен на сумму: " + price);
        return new Invoice(price);
    }
}

class Train implements Transportation, Priceable {
    private String trainNumber;
    private ArrayList<Station> stations;
    private HashMap<Station, Double> prices;

    public Train(String trainNumber) {
        this.trainNumber = trainNumber;
        this.stations = new ArrayList<>();
        this.prices = new HashMap<>();
    }

    public void addStation(Station station) {
        stations.add(station);
    }

    public void setPrice(Station station, double price) {
        prices.put(station, price);
        System.out.println("Установлена цену для поезда " + trainNumber + " на
станции " + station + ": " + price);
    }

    public String getTrainNumber() {
        return trainNumber;
    }

    public double getPriceForStation(Station station) {
        return prices.get(station);
    }
}

class Station {
    private String name;
```

```
    private Date arrivalTime;
    private Date departureTime;

    public Station(String name, Date arrivalTime, Date departureTime) {
        this.name = name;
        this.arrivalTime = arrivalTime;
        this.departureTime = departureTime;
    }

    @Override
    public String toString() {
        return name;
    }
}

class Ticket {
    private Passenger passenger;
    private Train train;
    private Station destinationStation;
    private Date dateTime;
    private double price;

    public Ticket(Passenger passenger, Train train, Station destinationStation, Date
dateTime, double price) {
        this.passenger = passenger;
        this.train = train;
        this.destinationStation = destinationStation;
        this.dateTime = dateTime;
        this.price = price;
    }

    public void printTicket() {
        System.out.println("Детали билета:");
        System.out.println("Пассажир: " + passenger);
        System.out.println("Поезд: " + train.getTrainNumber());
        System.out.println("Станция назначения: " + destinationStation);
        System.out.println("Дата и время: " + dateTime);
        System.out.println("Стоимость: " + price);
    }
}

class Invoice {
    private double amount;

    public Invoice(double amount) {
        this.amount = amount;
    }

    public void printInvoice() {
        System.out.println("Счет на сумму: " + amount);
    }
}

class Administrator {
    public void addTrain(Train train) {
        System.out.println("Поезд добавлен: " + train.getTrainNumber());
    }

    public void setPrice(Train train, Station station, double price) {
        train.setPrice(station, price);
    }
```

```
}

class Main {
    public static void main(String[] args) {
        Passenger passenger = new PassengerImpl("Энакин Скайуокер", "ABC12345");
        Train train = new Train("Экспресс");
        Station station = new Station("ЗвездаСмерти", new Date(), new Date());

        train.addStation(station);

        Administrator admin = new Administrator();
        admin.setPrice(train, station, 50.0);

        passenger.createRequest("ЗвездаСмерти", new Date());

        ArrayList<Train> trains = new ArrayList<>();
        trains.add(train);

        passenger.chooseTrain(trains);

        double price = train.getPriceForStation(station);
        Invoice invoice = passenger.getInvoice(price);
        invoice.printInvoice();
        Ticket ticket = new Ticket(passenger, train, station, new Date(), price);
        ticket.printTicket();
    }
}
```

Результат:

```
"C:\Program Files\Java\jdk-21\bin\java.exe" "-javaagent:D:\Новая папка\I
Установлена цену для поезда Экспресс на станции ЗвездаСмерти: 50.0
Заявка создана на станцию: ЗвездаСмерти на Tue May 14 11:53:49 MSK 2024
Выбран поезд: Экспресс
Счет выставлен на сумму: 50.0
Счет на сумму: 50.0
Детали билета:
Пассажир: Пассажир: Энакин Скайуокер, Паспорт: ABC12345
Поезд: Экспресс
Станция назначения: ЗвездаСмерти
Дата и время: Tue May 14 11:53:49 MSK 2024
Стоимость: 50.0
```

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