Министерство образования Республики Беларусь

Учреждение образования

«Брестский Государственный технический университет»

Кафедра ИИТ

Лабораторная работа №5

по дисциплине “Современные платформы программирования”

Выполнил:

Студент 3 курса

Группы ПО-8

Соколов С.Д.

Проверил:

Крощенко А.А.

Брест 2024

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

Ход работы

Вариант 20

Задание 1. Реализовать абстрактные классы или интерфейсы, а также наследование и полиморфизм для следующих классов:

7) interface Корабль ← abstract class Военный Корабль ← class Авианосец.Код программы:

Ship.java

```
public interface Ship {
    void move();
}
```

WarShip.java

```
public abstract class WarShip implements Ship {
    public abstract void attack();

    public abstract void move();


}
```

AircraftCarier.java

```
public class AircraftCarier extends WarShip {


    @Override
    public void move() {
        System.out.println("Авианосец движется");
    }

    @Override
    public void attack() {
        System.out.println("Авианосец атакует");
    }


}
```

Main.java

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

        AircraftCarier aircraftCarier = new AircraftCarier();


        aircraftCarier.move();
        aircraftCarier.attack();
    }
}
```

Вывод

```
C:\Users\semen\.jdks\openjdk-22.0.1
Авианосец движется
Авианосец атакует

Process finished with exit code 0
```

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 номер

Использовать объекты подклассов для моделирования реальных ситуаций и объектов.

Создать суперкласс Грузоперевозчик и подклассы Самолет, Поезд, Автомобиль. Определить время и стоимость перевозки для указанных городов и расстояний.

Код программы:

```
abstract class FreightTransporter {
    String name;
    int id;

    FreightTransporter(String name, int id) {
        this.name = name;
        this.id = id;
    }

    abstract double getTransportCost(int distance);
    abstract double getTransportTime(int distance);
}

class Plane extends FreightTransporter {
    Plane(String name, int id) {
        super(name, id);
    }

    @Override
    double getTransportCost(int distance) {

        return distance * 10.0;
    }

    @Override
    double getTransportTime(int distance) {

        return distance / 700.0;
    }
}

class Train extends FreightTransporter {
    Train(String name, int id) {
        super(name, id);
    }

    @Override
    double getTransportCost(int distance) {

        return distance * 5.0; // предположим, что стоимость составляет 5 единиц за
км
    }

    @Override
    double getTransportTime(int distance) {

        return distance / 100.0; //
    }
```

```
}

class Car extends FreightTransporter {
    Car(String name, int id) {
        super(name, id);
    }

    @Override
    double getTransportCost(int distance) {

        return distance * 7.0;
    }

    @Override
    double getTransportTime(int distance) {

        return distance / 60.0;
    }
}

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

        FreightTransporter[] transporters = new FreightTransporter[] {
                new Plane("Plane1", 1),
                new Train("Train1", 2),
                new Car("Car1", 3)
        };


        int distance = 1234;


        for (FreightTransporter transporter : transporters) {
            double cost = transporter.getTransportCost(distance);
            double time = transporter.getTransportTime(distance);

            System.out.println("Transporter: " + transporter.name);
            System.out.println("Transport Cost for " + distance + " km: " + cost);
            System.out.println("Transport Time for " + distance + " km: " + time);
            System.out.println();
        }
    }
}
```

Результаты работы программы:

```
C:\Users\semen\.jdks\openjdk-22.0.1\bin\java.exe "-j
Transporter: Plane1
Transport Cost for 1234 km: 12340.0
Transport Time for 1234 km: 1.762857142857143

Transporter: Train1
Transport Cost for 1234 km: 6170.0
Transport Time for 1234 km: 12.34

Transporter: Car1
Transport Cost for 1234 km: 8638.0
Transport Time for 1234 km: 20.566666666666666
```

Задание 3. В задании 3 ЛР №4, где возможно, заменить объявления суперклассов объявлениями абстрактных классов или интерфейсов.

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.

Main.java

```
import java.util.*;

interface Vehicle {
    String getNumber();
    String getDestination();
    String getDepartureTime();
}

abstract class Person {
    abstract String getName();
    abstract String getDestination();
    abstract String getDepartureTime();
}

class Train implements Vehicle {
    String number;
    String destination;
    String departureTime;
    double price;

    public Train(String number, String destination, String departureTime, double
price) {
        this.number = number;
        this.destination = destination;
        this.departureTime = departureTime;
        this.price = price;
    }

    public String getNumber() { return number; }
    public String getDestination() { return destination; }
    public String getDepartureTime() { return departureTime; }
}

class Passenger extends Person {
    String name;
    String destination;
    String departureTime;

    public Passenger(String name, String destination, String departureTime) {
        this.name = name;
        this.destination = destination;
        this.departureTime = departureTime;
    }

    public String getName() { return name; }
    public String getDestination() { return destination; }
    public String getDepartureTime() { return departureTime; }
}

class RailwayTicketSystem {
    List<Vehicle> vehicles = new ArrayList<>();
    List<Person> people = new ArrayList<>();

    public void addTrain(String number, String destination, String departureTime,
double price) {
```

```
        vehicles.add(new Train(number, destination, departureTime, price));
    }

    public void addPassenger(String name, String destination, String departureTime) {
        people.add(new Passenger(name, destination, departureTime));
    }

    public Vehicle searchVehicle(String destination, String departureTime) {
        for (Vehicle vehicle : vehicles) {
            if (vehicle.getDestination().equals(destination) &&
vehicle.getDepartureTime().equals(departureTime)) {
                return vehicle;
            }
        }
        return null;
    }

    public void printInvoice(Person person, Vehicle vehicle) {
        System.out.println("Invoice for " + person.getName());
        System.out.println("Vehicle Number: " + vehicle.getNumber());
        System.out.println("Destination: " + vehicle.getDestination());
        System.out.println("Departure Time: " + vehicle.getDepartureTime());
        if (vehicle instanceof Train) {
            System.out.println("Price: " + ((Train) vehicle).price);
        }
    }
}
```

MainWork.java

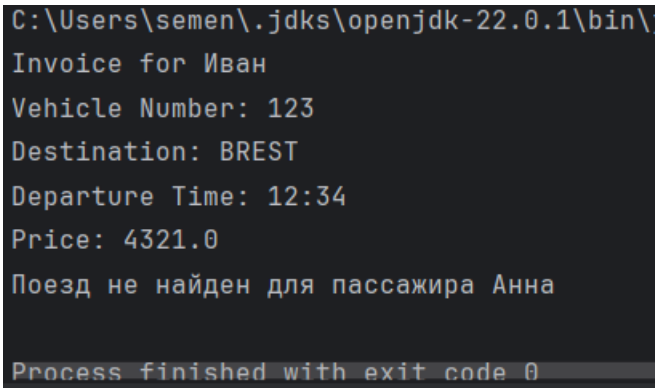
```
public class MainWork {
    public static void main(String[] args) {
        RailwayTicketSystem system = new RailwayTicketSystem();


        system.addTrain("123", "BREST", "12:34", 4321.0);
        system.addTrain("456", "MINSK", "12:12", 1234.0);


        system.addPassenger("Иван", "BREST", "12:34");
        system.addPassenger("Анна", "GRODNO", "12:12");


        for (Person person : system.people) {
            Vehicle vehicle = system.searchVehicle(person.getDestination(),
person.getDepartureTime());
            if (vehicle != null) {
                system.printInvoice(person, vehicle);
            } else {
                System.out.println("Поезд не найден для пассажира " +
person.getName());
            }
        }
    }
}
```

Результаты работы программы:



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