

**Министерство образования Республики Беларусь
Учреждение Образования
«Брестский Государственный Технический Университет»
Кафедра ИИТ**

**Лабораторная работа №5
По дисциплине СПП за 5 семестр
Тема: «Java»**

Выполнил:
Студент 3-го курса
Группы ПО-5
Крощук В.В.
Проверил:
Крощенко А.А.

Брест 2021

Лабораторная работа №5

Цель работы:

приобрести практические навыки в области объектно-ориентированного проектирования

Вариант: 8

Задания и выполненные решения:

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

interface Врач ← class Хирург ← class Нейрохирург.

Код:

Main.java

```
package com.company;

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

public class Main {

    public static void main(String[] args) {
        List<Doctor> doctorList = new ArrayList<>();
        doctorList.add(new Surgeon());
        doctorList.add(new Neurosurgeon());
        doctorList.add(new Surgeon());
        doctorList.add(new Neurosurgeon());
        doctorList.add(new Surgeon());
        doctorList.add(new Surgeon());
        doctorList.forEach(Doctor::printPost);
    }
}
```

Doctor.java

```
package com.company;

public interface Doctor {
    void printPost();
}
```

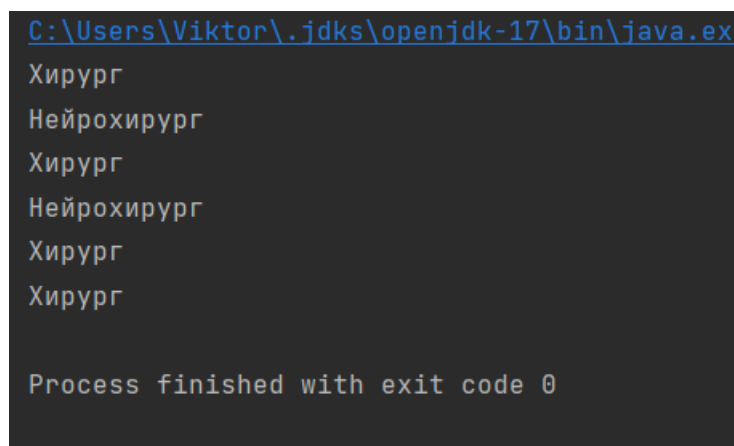
Surgeon.java

```
package com.company;  
public class Surgeon implements Doctor {  
    @Override  
    public void printPost() {  
        System.out.println("Хирург");  
    }  
}
```

Neurosurgeon.java

```
package com.company;  
public class Neurosurgeon extends Surgeon {  
    @Override  
    public void printPost() {  
        System.out.println("Нейрохирург");  
    }  
}
```

Результат:



C:\Users\Viktor\.jdk\openjdk-17\bin\java.exe
Хирург
Нейрохирург
Хирург
Нейрохирург
Хирург
Хирург

Process finished with exit code 0

Выполнение происходит согласно поставленной задачи. Работает корректно!

- 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.

Код:

Main.java

```
package com.company;

import java.math.BigDecimal;
import java.util.ArrayList;
import java.util.List;

public class Main {

    public static void main(String[] args) {
        List<PassengerCarrier> list = new ArrayList<>();
        list.add(new Airplane(980, new BigDecimal("0.2")));
        list.add(new Car(130, new BigDecimal("0.1")));
        list.add(new Train(90, new BigDecimal("0.07")));
        System.out.println("Вывод скорости и стоимости транспортных:");
        list.forEach(transport -> {
            transport.printTypeOfTransport();
            transport.printPrice();
            transport.printSpeed();
        });
        int distance = 1500;
        System.out.println("\nРасчет для каждого транспорта, расстояние = "+
            distance + "km");
        list.forEach(transport -> transport.calculateMovement(distance));
    }
}
```

Airplane.java

```
package com.company;

import java.math.BigDecimal;

public class Airplane extends PassengerCarrier {
    public Airplane(int speed, BigDecimal pricePerKm) {
        super(speed, pricePerKm);
    }
    @Override
    void printTypeOfTransport() {
        System.out.println("Самолет:");
    }
    @Override
    void calculateMovement(int distance) {
        System.out.println("Поездка на самолете займет " + distance /
this.getSpeed() +
            " час(ов) и будет стоить " +
this.getPricePerKm().multiply(BigDecimal.valueOf(distance)) + "$");
    }
}
```

Car.java

```
package com.company;
import java.math.BigDecimal;

public class Car extends PassengerCarrier {
    public Car(int speed, BigDecimal pricePerKm) {
        super(speed, pricePerKm);
    }
    @Override
    void printTypeOfTransport() {
        System.out.println("Машина:");
    }
    @Override
    void calculateMovement(int distance) {
        System.out.println("Поездка на машине займет " +
distance/this.getSpeed()
        + " час(ов) и будет стоить " +
this.getPricePerKm().multiply(BigDecimal.valueOf(distance)) + "$");
    }
}
```

Train.java

```
package com.company;
import java.math.BigDecimal;

public class Train extends PassengerCarrier {
    public Train(int speed, BigDecimal pricePerKm) {
        super(speed, pricePerKm);
    }
    @Override
    void printTypeOfTransport() {
        System.out.println("Поезд:");
    }
    @Override
    void calculateMovement(int distance) {
        System.out.println("Поездка на поезде займет " +
distance/this.getSpeed()
        + " час(ов) и будет стоить " +
this.getPricePerKm().multiply(BigDecimal.valueOf(distance)) +
"$");
    }
}
```

PassengerCarrier.java

```
package com.company;
import java.math.BigDecimal;

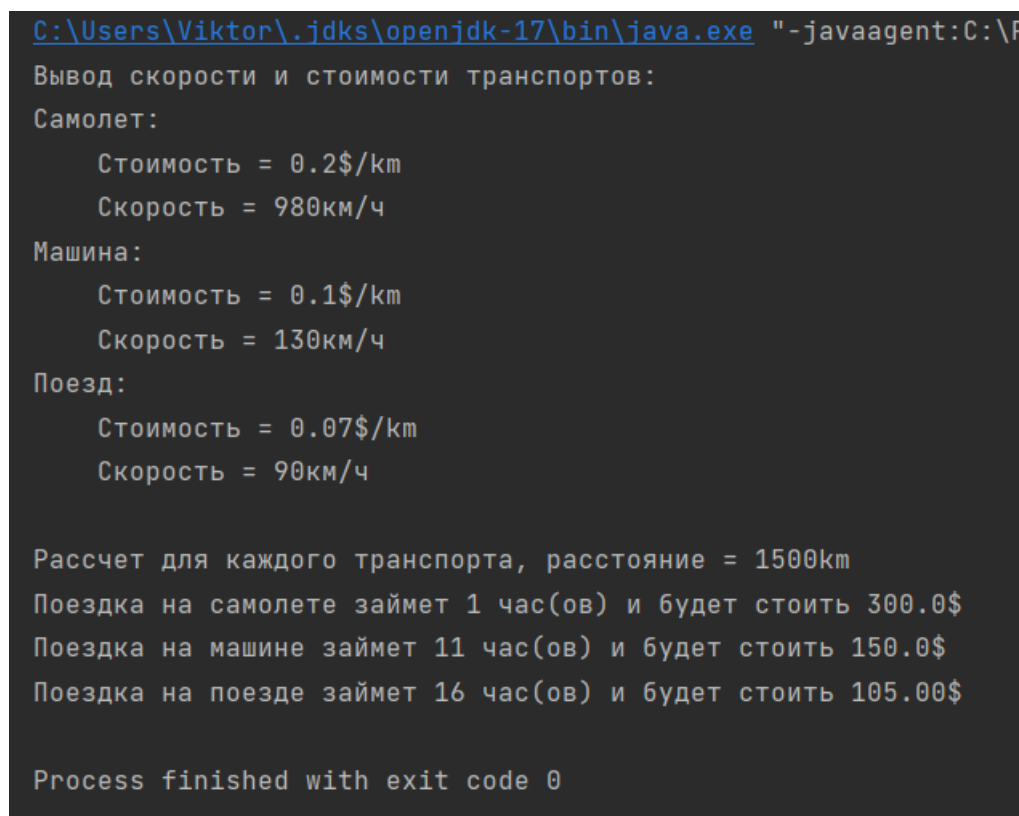
public abstract class PassengerCarrier {
    private int speed;
```

```

private BigDecimal pricePerKm;
public PassengerCarrier(int speed, BigDecimal pricePerKm) {
    this.speed = speed;
    this.pricePerKm = pricePerKm;
}
abstract void printTypeOfTransport();
abstract void calculateMovement(int distance);
public void printSpeed() {
    System.out.println("\tСкорость = " + speed + "км/ч");
}
public void printPrice() {
    System.out.println("\tСтоимость = " + pricePerKm + "$/km");
}
public int getSpeed() {
    return speed;
}
public void setSpeed(int speed) {
    this.speed = speed;
}
public BigDecimal getPricePerKm() {
    return pricePerKm;
}
public void setPricePerKm(BigDecimal pricePerKm) {
    this.pricePerKm = pricePerKm;
}
}

```

Результат:



```

C:\Users\Viktor\.jdk\openjdk-17\bin\java.exe "-javaagent:C:\P
Вывод скорости и стоимости транспортов:
Самолет:
    Стоимость = 0.2$/km
    Скорость = 980км/ч
Машина:
    Стоимость = 0.1$/km
    Скорость = 130км/ч
Поезд:
    Стоимость = 0.07$/km
    Скорость = 90км/ч

Расчет для каждого транспорта, расстояние = 1500km
Поездка на самолете займет 1 час(ов) и будет стоить 300.0$
Поездка на машине займет 11 час(ов) и будет стоить 150.0$
Поездка на поезде займет 16 час(ов) и будет стоить 105.00$

Process finished with exit code 0

```

Исходя из полученных результатов, все работает корректно, согласно поставленному заданию.

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

Код:

Main.java

```
package com.company;

public class Main {

    public static void main(String[] args) {
        OnlineStore onlineStore = new OnlineStore();
        Administrator administrator = new Administrator(onlineStore);
        administrator.addProduct("Диван");
        administrator.addProduct("Ноутбук");
        administrator.addProduct("Стул");
        administrator.addProduct("Лампа");
        administrator.addProduct("Телефон");
        administrator.addProduct("Стол");
        Client client1 = new Client(1, onlineStore);
        System.out.println("Все товары: ");
        client1.printProducts();
        client1.addOrder(2).pay();
        client1.addOrder(3);
        client1.addOrder(1).pay();
        System.out.println("\nЗаказы 1 клиента: ");
        client1.printOrders();
        Client client2 = new Client(2, onlineStore);
        client2.addOrder(5).pay();
        client2.addOrder(1).pay();
        System.out.println("\nЗаказы 2 клиента: ");
        client2.printOrders();
        administrator.addToBlackList();
        System.out.println("\nПользователь пытается добавить товар: ");
        client1.addOrder(5).pay();
        client2.addOrder(6).pay();
        System.out.println("\nЗаказы 1 клиента: ");
        client1.printOrders();
        System.out.println("\nЗаказы 2 клиента: ");
        client2.printOrders();
    }
}
```

Administrator.java

```
package com.company;

public class Administrator extends User {
    public Administrator(OnlineStore onlineStore) {
        super(onlineStore);
    }
}
```

```

    public void addProduct(String productName) {
        getOnlineStore().addProduct(productName);
    }
    public void addToBlackList() {
        getOnlineStore().addToBlackList();
    }
}

```

Client.java

```

package com.company;
public class Client extends User {
    private int clientId;
    public Client(int clientId, OnlineStore onlineStore) {
        super(onlineStore);
        this.clientId = clientId;
    }
    public void printProducts() {
        getOnlineStore().printProducts();
    }
    public void printOrders() {
        getOnlineStore().printUserOrders(clientId);
    }
    public Order addOrder(int productId) {
        return getOnlineStore().addOrder(clientId, productId);
    }
}

```

OnlineStore.java

```

package com.company;
import java.util.ArrayList;
import java.util.List;
import java.util.stream.Collectors;

public class OnlineStore {
    private List<Product> productList;
    private List<Order> orderList;
    private List<Integer> blackList;
    public OnlineStore() {
        this.productList = new ArrayList<>();
        this.orderList = new ArrayList<>();
        this.blackList = new ArrayList<>();
    }
    private boolean isInBlackList(int userId) {
        return blackList.contains(userId);
    }
    public void printProducts() {
        productList.forEach(System.out::println);
    }
}

```



```

    public void printUserOrders(int userId) {
        orderList.stream().filter(order -> order.getUserId() ==
            userId).forEach(System.out::println);
    }
    public void addToBlackList() {
        blackList = orderList.stream().filter(order ->
!order.isPaid()).map(Order::getUserId).collect(Collectors.toList());
    }
    public void addProduct(String productName) {
        productList.add(new Product(productList.size() + 1, productName));
    }
    public Order addOrder(int userId, int productId) {
        Order order = new Order(userId, productList.get(--productId));
        if(isInBlackList(userId)) {
            System.out.println("Простите но вы в черном списке");
        } else{
            orderList.add(order);
        }
        return order;
    }
}

```

Order.java

```

package com.company;
public class Order {
    private int userId;
    private Product product;
    private boolean isPaid;
    public Order(int userId, Product product) {
        this.userId = userId;
        this.product = product;
        this.isPaid = false;
    }
    public void pay() {
        this.isPaid = true;
    }
    public int getUserId() {
        return userId;
    }
    public boolean isPaid() {
        return isPaid;
    }
    @Override
    public String toString() {
        return product +
            ", Куплено=" + isPaid;
    }
}

```

Product.java

```
package com.company;
public class Product {
    private int productId;
    private String productName;
    public Product() {
    }
    public Product(int productId, String productName) {
        this.productId = productId;
        this.productName = productName;
    }
    public int getProductId() {
        return productId;
    }
    public void setProductId(int productId) {
        this.productId = productId;
    }
    public String getProductName() {
        return productName;
    }
    public void setProductName(String productName) {
        this.productName = productName;
    }
    @Override
    public String toString() {
        return "Товар №" + productId +
            ": '" + productName + '\'';
    }
}
```

User.java

```
package com.company;
public class User {
    private OnlineStore onlineStore;
    public User(OnlineStore onlineStore) {
        this.onlineStore = onlineStore;
    }
    public OnlineStore getOnlineStore() {
        return onlineStore;
    }
    public void setOnlineStore(OnlineStore onlineStore) {
        this.onlineStore = onlineStore;
    }
}
```

Результат:

```
C:\Users\Viktor\.jdk\openjdk-17\bin\java
Все товары:
Товар №1: 'Диван'
Товар №2: 'Ноутбук'
Товар №3: 'Стул'
Товар №4: 'Лампа'
Товар №5: 'Телефон'
Товар №6: 'Стол'

Заказы 1 клиента:
Товар №2: 'Ноутбук', Куплено=true
Товар №3: 'Стул', Куплено=false
Товар №1: 'Диван', Куплено=true

Заказы 2 клиента:
Товар №5: 'Телефон', Куплено=true
Товар №1: 'Диван', Куплено=true

Пользователь пытается добавить товар:
Простите но вы в черном списке

Заказы 1 клиента:
Товар №2: 'Ноутбук', Куплено=true
Товар №3: 'Стул', Куплено=false
Товар №1: 'Диван', Куплено=true

Заказы 2 клиента:
Товар №5: 'Телефон', Куплено=true
Товар №1: 'Диван', Куплено=true
Товар №6: 'Стол', Куплено=true

Process finished with exit code 0
|
```

Исходя из полученных результатов, все работает корректно, согласно поставленному заданию.

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