Министерство образования Республики Беларусь
Учреждение образования
«Брестский государственный технический университет»
Кафедра ИИТ

Лабораторная работа №5
По дисциплине «СПП»

Выполнил
студент 3 курса
группы ПО-5:
Смекалов К.Р.
Проверил:
Крощенко А.А.

Брест, 2021

Вариант 8

Цель: приобрести практическкие навыки в области объектно-ориентированного проектирования.

Задание 1:

Реализовать абстрактные классы или интерфейсы, а также наследование и полиморфизм для следующего класса - interface Врач ← class Хирург ← class Нейрохирург.

Код программы:

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 {
    public void printPost() {
        System.out.println("Surgeon");
    }
}
```

Neurosurgen.java

```
package com.company;

public class Neurosurgeon extends Surgeon {
    public void printPost() {
        System.out.println("Neurosurgeon");
    }
}
```

Результат выполнения:

```
Surgeon
Neurosurgeon
Surgeon
Neurosurgeon
Surgeon
Surgeon
```

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.

Код программы:

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("Print speed and price of transport:");
        list.forEach(transport -> {transport.printTypeOfTransport();
            transport.printPrice(); transport.printSpeed();});
        int distance = 1500;
        System.out.println("\nCalculate movement for each transport, distance = "+
                distance + "km");
        list.forEach(transport -> transport.calculateMovement(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("\tSpeed = " + speed + "km per hour");
    }
    public void printPrice() {
        System.out.println("\tPrice = " + pricePerKm + "$ per 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;
    }
}
```

Train.java

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

public class Train extends PassengerCarrier {
    public Train(int speed, BigDecimal pricePerKm) {
        super(speed, pricePerKm);
    }
    void printTypeOfTransport() {
        System.out.println("Train:");
    }
    void calculateMovement(int distance) {
        System.out.println("A trip by train will take " + distance/this.getSpeed()
                + " hours and will cost " +
                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);
    }
    void printTypeOfTransport() {
        System.out.println("Car:");
    }
    void calculateMovement(int distance) {
        System.out.println("A trip by car will take " + distance/this.getSpeed()
                + " hours and will cost " +
                this.getPricePerKm().multiply(BigDecimal.valueOf(distance)));
    }
}
```

Airplane.java

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

public class Airplane extends PassengerCarrier {
    public Airplane(int speed, BigDecimal pricePerKm) {
        super(speed, pricePerKm);
    }
    void printTypeOfTransport() {
        System.out.println("Airplane");
    }
    void calculateMovement(int distance) {
        System.out.println("Flight by plane will take " + distance / this.getSpeed()
                + " hours and will cost " +
                this.getPricePerKm().multiply(BigDecimal.valueOf(distance)));
    }
}
```

Результат выполнения:

```
Print speed and price of transport:
Airplane
    Price = 0.2$ per km
    Speed = 980km per hour
Car:
    Price = 0.1$ per km
    Speed = 130km per hour
Train:
    Price = 0.07$ per km
    Speed = 90km per hour

Calculate movement for each transport, distance = 1500km
Flight by plane will take 1 hours and will cost 300.0
A trip by car will take 11 hours and will cost 150.0
A trip by train will take 16 hours and will cost 105.00
```

Задание 3:

В задании 3 ЛР №4, где возможно, заменить объявления суперклассов объявлениями абстрактных классов или интерфейсов. Возьмем 3 задание 4-ой лабораторной работы.

Код программы:

Main.java

```java
package com.company;

public class Main {
    public static void main(String[] args) {
        OnlineStore onlineStore = new OnlineStore();
        Administrator administrator = new Administrator(onlineStore);
        administrator.addProduct("Sofa");
        administrator.addProduct("Laptop");
        administrator.addProduct("Chair");
        administrator.addProduct("Lamp");
        administrator.addProduct("Phone");
        administrator.addProduct("Table");
        Client client1 = new Client(1, onlineStore);
        System.out.println("All products:");
        client1.printProducts();
        client1.addOrder(2).pay();
        client1.addOrder(3);
        client1.addOrder(1).pay();
        System.out.println("\nUser's orders:");
        client1.printOrders();
        Client client2 = new Client(2, onlineStore);
        client2.addOrder(5).pay();
        client2.addOrder(1).pay();
        administrator.addToBlackList();
        System.out.println("\nUser is trying to add order:");
        client1.addOrder(5);
        client2.addOrder(6).pay();
    }
}
```

Product.java

```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 "Product №" + productId +
                ": '" + productName + '\'';
    }
}
```

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 +
                ", isPaid=" + isPaid;
    }
}
```

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) {
        if (!isInBlackList(userId)) {
            Order order = new Order(userId, productList.get(--productId));
            orderList.add(order);
            return order;
        } else {
            System.out.println("Sorry, but you are in the blacklist");
            return null;
        }
    }
}
```

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);
    }
}
```

User.java

```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 выполнения:

```
All products:
Product №1: 'Sofa'
Product №2: 'Laptop'
Product №3: 'Chair'
Product №4: 'Lamp'
Product №5: 'Phone'
Product №6: 'Table'

User's orders:
Product №2: 'Laptop', isPaid=true
Product №3: 'Chair', isPaid=false
Product №1: 'Sofa', isPaid=true

User is trying to add order:
Sorry, but you are in the blacklist
```

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