|  |  |
| --- | --- |
| Gerb-BMSTU_01 | **Министерство науки и высшего образования Российской Федерации**  **Федеральное государственное бюджетное образовательное учреждение**  **высшего образования**  **«Московский государственный технический университет**  **имени Н.Э. Баумана**  **(национальный исследовательский университет)»**  **(МГТУ им. Н.Э. Баумана)** |

ФАКУЛЬТЕТ **Информатика и системы управления**

КАФЕДРА **Компьютерные системы и сети (ИУ6)**

НАПРАВЛЕНИЕ ПОДГОТОВКИ **09.04.01 Информатика и вычислительная техника**

МАГИСТЕРСКАЯ ПРОГРАММА **09.04.01/12 Интеллектуальный анализ больших**

**данных в системах поддержки принятия решений**

**Отчет**

|  |
| --- |
| **по лабораторной работе № 4** |

Вариант 6

**Название:** Внутренние классы. Интерфейсы.

**Дисциплина:** Языки программирования для работы с большими данными

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Студент | ИУ6-23М |  |  | В.А. Гордеев |
|  | (Группа) |  | (Подпись, дата) | (И.О. Фамилия) |
|  |  |  |  |  |
| Преподаватель |  |  |  | П.В. Степанов |
|  |  |  | (Подпись, дата) | (И.О. Фамилия) |

Москва, 2023

**ВВЕДЕНИЕ**

**Задание:**

* 1. Создать класс Shop (магазин) с внутренним классом, с помощью объектов которого можно хранить информацию об отделах, товарах и услуг.
  2. Создать класс Cправочная Cлужба Oбщественного Tранспорта с внутренним классом, с помощью объектов которого можно хранить информацию о времени, линиях маршрутов и стоимости проезда.
  3. Реализовать абстрактные классы или интерфейсы, а также наследование и полиморфизм для следующих классов. interface Корабль <- abstract class Военный Корабль <- class Авианосец.
  4. Реализовать абстрактные классы или интерфейсы, а также наследование и полиморфизм для следующих классов. interface Врач <- class Хирург <- class Нейрохирург

**ОСНОВНАЯ ЧАСТЬ**

**Задание 1.1**

Создан класс Shop (магазин) с внутренним классом, с помощью объектов которого можно хранить информацию об отделах, товарах и услуг.

Код программы представлен в листингах 1, 2. Результат работы программы показан на рисунке 1.

Листинг 1 – Код класса Shop

import java.util.ArrayList;  
import java.util.List;  
  
public class Shop {  
 private String name;  
 private List<Department> departments;  
 public Shop(String name) {  
 this.name = name;  
 departments = new ArrayList<>();  
 }  
 public void addDepartment(String name) {  
 departments.add(new Department(name));  
 }  
 public void addProduct(String departmentName, String productName, double price) throws Exception {  
 Department department = findDepartment(departmentName);  
 department.addProduct(new Product(productName, price));  
 }  
 public void addService(String departmentName, String serviceName, double price) throws Exception {  
 Department department = findDepartment(departmentName);  
 department.addService(new Service(serviceName, price));  
 }  
 public void printAllDepartmentsInShop() {  
 for (Department department : departments) {  
 System.*out*.println(department.getName());  
 }  
 }  
 public void printAllProductsFromDepartment(String departmentName) throws Exception {  
 Department department = findDepartment(departmentName);  
 department.printAllProductsFromDepartment();  
 }  
 public void printAllServicesInDepartment(String departmentName) throws Exception {  
 Department department = findDepartment(departmentName);  
 department.printAllServicesInDepartment();  
 }  
 private Department findDepartment(String departmentName) throws Exception {  
 for (Department department : departments) {  
 if (department.getName().equals(departmentName)) {  
 return department;  
 }  
 }  
 throw new Exception("Отдел с именем: " + name + " - не найден");  
 }  
 private class Department {  
 private String name;  
 private List<Product> products;  
 private List<Service> services;  
 public Department(String name) {  
 this.name = name;  
 products = new ArrayList<>();  
 services = new ArrayList<>();  
 }  
 public void addProduct(Product product) {  
 products.add(product);  
 }  
 public void addService(Service service) {  
 services.add(service);  
 }  
 public String getName() {  
 return name;  
 }  
 public void printAllProductsFromDepartment() {  
 System.*out*.println("Продукты в " + name + ":");  
 for (Product product : products) {  
 System.*out*.println(product.getName() + " - " + product.getPrice());  
 }  
 }  
 public void printAllServicesInDepartment() {  
 System.*out*.println("Услуги в " + name + ":");  
 for (Service service : services) {  
 System.*out*.println(service.getName() + " - " + service.getPrice());  
 }  
 }  
 }  
 private class Product {  
 private String name;  
 private double price;  
 public Product(String name, double price) {  
 this.name = name;  
 this.price = price;  
 }  
 public String getName() {  
 return name;  
 }  
 public double getPrice() {  
 return price;  
 }  
 }  
 private class Service {  
 private String name;  
 private double price;  
 public Service(String name, double price) {  
 this.name = name;  
 this.price = price;  
 }  
 public String getName() {  
 return name;  
 }  
 public double getPrice() {  
 return price;  
 }  
 }  
}

Листинг 2 – Код класса Main

// Создать класс Shop (магазин) с внутренним классом, с помощью объектов  
// которого можно хранить информацию об отделах, товарах и услуг.  
public class Main {  
 public static void main(String[] args) {  
 Shop bearShop = new Shop("Вкусно - и точка");  
  
 bearShop.addDepartment("Пиво");  
 bearShop.addDepartment("Гренки");  
 try {  
 bearShop.addProduct("Пиво", "Козел темный", 100);  
 bearShop.addProduct("Пиво", "Козел светлый", 90);  
 bearShop.addService("Пиво", "Налить пиво", 10);  
 bearShop.printAllProductsFromDepartment("Пиво");  
 } catch (Exception e) {  
 throw new RuntimeException(e);  
 }  
  
 }  
}

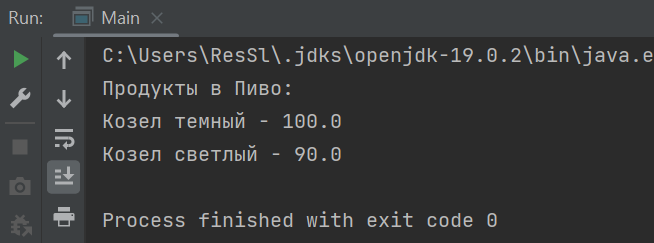


Рисунок 1 – Результат работы программы 1.1

**Задание 1.2**

Создан класс Cправочная Cлужба Oбщественного Tранспорта с внутренним классом, с помощью объектов которого можно хранить информацию о времени, линиях маршрутов и стоимости проезда.

Код программы представлен в листингах 3, 4. Результат работы программы показан на рисунке 1.

Листинг 3 – Код класса InformationServicePublicTransport

import java.util.ArrayList;  
  
public class InformationServicePublicTransport {  
 private ArrayList<TransportationInfo> transportations;  
  
 public InformationServicePublicTransport() {  
 transportations = new ArrayList<>();  
 }  
  
 public void addTransportationInfo(  
 int timeHours,  
 int timeMinutes,  
 String routeLineFrom,  
 String routeLineTo,  
 double cost) {  
 TransportationInfo info = new TransportationInfo(timeHours, timeMinutes, routeLineFrom, routeLineTo, cost);  
 transportations.add(info);  
 }  
  
 public TransportationInfo searchTransportationInfoByTime(int timeHours, int timeMinutes) {  
 for (TransportationInfo transportation : transportations) {  
 if (transportation.getTimeHours() > timeHours) {  
 System.*out*.println("Информация по заданному времени: \n" + transportation.toString());  
 return transportation;  
 } else if (transportation.getTimeMinutes() >= timeMinutes && transportation.getTimeHours() == timeHours) {  
 System.*out*.println("Информация по заданному времени: \n" + transportation.toString());  
 return transportation;  
 }  
 }  
 return null;  
 }  
  
 public TransportationInfo searchTransportationInfoByRoute(String routeLineFrom, String routeLineTo) {  
 for (TransportationInfo transportation : transportations) {  
 if ((transportation.getRouteLineFrom().equals(routeLineFrom)) && (transportation.getRouteLineTo().equals(routeLineTo))) {  
 System.*out*.println("Информация по заданному маршруту: \n" + transportation.toString());  
 return transportation;  
 }  
 }  
 return null;  
 }  
  
 private class TransportationInfo {  
 private int timeHours;  
 private int timeMinutes;  
  
 private String routeLineFrom;  
 private String routeLineTo;  
 private double cost;  
  
 public TransportationInfo(  
 int timeHours,  
 int timeMinutes,  
 String routeLineFrom,  
 String routeLineTo,  
 double cost) {  
 this.timeHours = timeHours;  
 this.timeMinutes = timeMinutes;  
 this.routeLineFrom = routeLineFrom;  
 this.routeLineTo = routeLineTo;  
 this.cost = cost;  
 }  
  
 public int getTimeHours() {  
 return timeHours;  
 }  
  
 public int getTimeMinutes() {  
 return timeMinutes;  
 }  
  
 public String getRouteLineFrom() {  
 return routeLineFrom;  
 }  
  
 public String getRouteLineTo() {  
 return routeLineTo;  
 }  
  
 public String getTime() {  
 return timeHours + ":" + timeMinutes;  
 }  
  
 public String getRouteLine() {  
 return routeLineFrom + "-" + routeLineTo;  
 }  
  
 public double getCost() {  
 return cost;  
 }  
  
 @Override  
 public String toString() {  
 return "Информация о маршруте: " + "\n" +  
 "Время отъезда: " + timeHours + ":" + timeMinutes + "\n" +  
 "Откуда: " + routeLineFrom + '\n' +  
 "Куда: " + routeLineTo + '\n' +  
 "Стоимость: " + cost + '\n';  
 }  
 }  
}

Листинг 4 – Код класса Main

// Создать класс Cправочная Cлужба Oбщественного Tранспорта с внутренним классом,  
// с помощью объектов которого можно хранить информацию о времени, линиях маршрутов и стоимости проезда.  
public class Main {  
 public static void main(String[] args) {  
 InformationServicePublicTransport depo = new InformationServicePublicTransport();  
  
 depo.addTransportationInfo(10,0,"Kiev", "Moscow", 50);  
 depo.addTransportationInfo(11,0,"Kiev", "Minsk", 70);  
 depo.addTransportationInfo(10,0,"Moscow", "Bataysk", 100);  
 depo.addTransportationInfo(11,0,"Moscow", "Minsk", 100);  
  
 depo.searchTransportationInfoByRoute("Moscow","Bataysk");  
 depo.searchTransportationInfoByTime(10,50);  
 }  
}

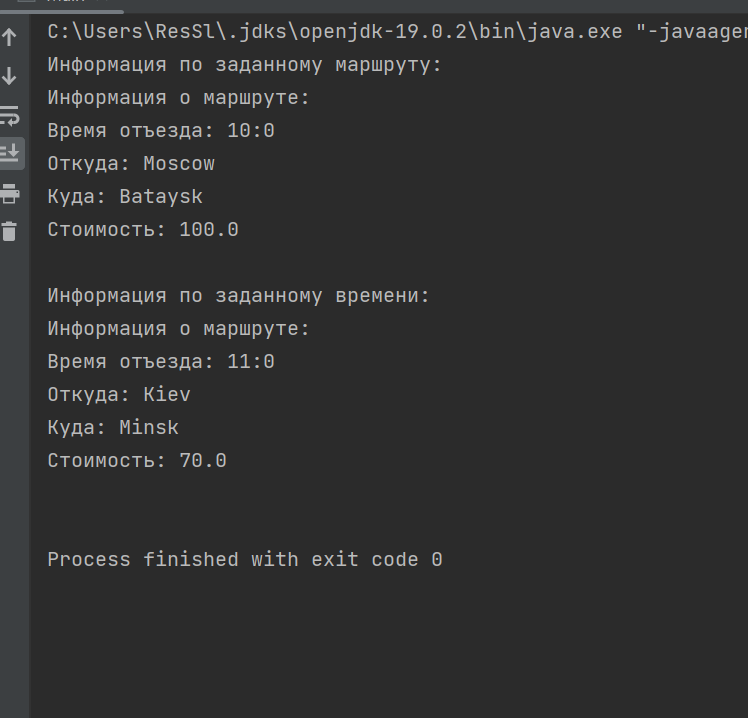


Рисунок 2 – Результат работы программы 1.2

**Задание 1.3**

Реализованы абстрактные классы или интерфейсы, а также наследование и полиморфизм для следующих классов. interface Корабль <- abstract class Военный Корабль <- class Авианосец.

Код программы представлен в листинге 5. Результат работы программы показан на рисунке 1.

Листинг 5 – Код класса Main

// Реализовать абстрактные классы или интерфейсы, а также наследование и полиморфизм для следующих классов  
// interface Корабль <- abstract class Военный Корабль <- class Авианосец.  
public class Main {  
 public static void main(String[] args) {  
 AircraftCarrier aircraft = new AircraftCarrier( 30, 5);  
  
 aircraft.sail();  
 aircraft.attack();  
 }  
}  
  
interface Ship {  
 void sail();  
}  
  
abstract class WarShip implements Ship {  
 private int speed;  
 public WarShip(int speed) {  
 this.speed = speed;  
 }  
  
 public int getSpeed() {  
 return speed;  
 }  
  
 public void setSpeed(int speed) {  
 this.speed = speed;  
 }  
  
 abstract void attack();  
  
 @Override  
 public void sail() {  
 System.*out*.println("Военный корабль поплыл!");  
 }  
}  
  
class AircraftCarrier extends WarShip{  
 private int numberOfPlanes;  
 public AircraftCarrier(int speed, int numberOfPlanes) {  
 super(speed);  
 this.numberOfPlanes = numberOfPlanes;  
 }  
 public int getNumberOfPlanes() {  
 return numberOfPlanes;  
 }  
 public void setNumberOfPlanes(int numberOfPlanes) {  
 this.numberOfPlanes = numberOfPlanes;  
 }  
 @Override  
 public void sail() {  
 System.*out*.println("Авианосец поплыл!");  
 }  
 @Override  
 public void attack() {  
 System.*out*.println("Самолеты полетели!");  
 }  
}

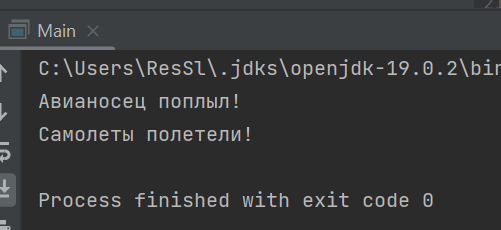


Рисунок 3 – Результат работы программы 1.3

**Задание 1.4**

Реализованы абстрактные классы или интерфейсы, а также наследование и полиморфизм для следующих классов. interface Корабль <- abstract class Военный Корабль <- class Авианосец.

Код программы представлен в листинге 6. Результат работы программы показан на рисунке 4.

Листинг 6 – Код класса Main

// Реализовать абстрактные классы или интерфейсы, а также наследование и полиморфизм для следующих классов  
// interface Врач <- class Хирург <- class Нейрохирург.  
public class Main {  
 public static void main(String[] args) {  
 Surgeon surgeon = new Surgeon(14);  
 Neurosurgeon neurosurgeon = new Neurosurgeon(10);  
  
 surgeon.treat();  
 neurosurgeon.treat();  
 }  
}  
  
interface Doctor {  
 void treat();  
}  
  
class Surgeon implements Doctor {  
 private int numberOfOperation;  
  
 public Surgeon(int numberOfOperation) {  
 this.numberOfOperation = numberOfOperation;  
 }  
  
 public int getNumberOfOperation() {  
 return numberOfOperation;  
 }  
  
 public void setNumberOfOperation(int numberOfOperation) {  
 this.numberOfOperation = numberOfOperation;  
 }  
  
 @Override  
 public void treat() {  
 System.*out*.println("Хирург лечит пациента");  
 }  
}  
  
class Neurosurgeon extends Surgeon{  
 public Neurosurgeon(int numberOfOperation) {  
 super(numberOfOperation);  
 }  
 @Override  
 public void treat() {  
 System.*out*.println("Нейрохирург лечит пациента");  
 }  
}

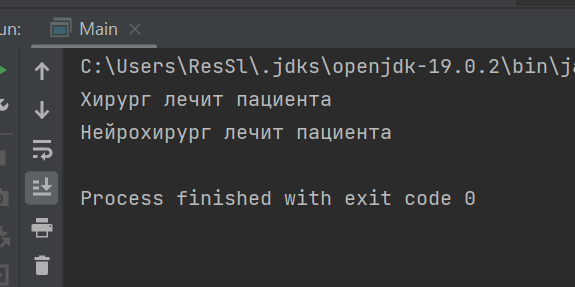


Рисунок 4 – Результат работы программы 1.4

**Вывод**: В результате выполнения лабораторной работы были получены практические навыки для работы с вложенными класса, интерфейсами и абстрактными классам.