**МИНИСТЕРСТВО ОБРАЗОВАНИЯ РЕСПУБЛИКИ БЕЛАРУСЬ**

**УЧРЕЖДЕНИЕ ОБРАЗОВАНИЯ**

**ГОМЕЛЬСКИЙ ГОСУДАРСТВЕННЫЙ ТЕХНИЧЕСКИЙ**

**УНИВЕРСИТЕТ ИМЕНИ П. О. СУХОГО**

Факультет автоматизированных и информационных систем

Кафедра «Информационные технологии»

ОТЧЁТ ПО ЛАБОРАТОРНОЙ РАБОТЕ №1

по дисциплине: «Введение в разработку программного обеспечения»

на тему: **«**Развитие представлений о разработке программ. Объектно-ориентированное программирование в Java»

Выполнил: студент гр. ИТП-21

Сеноженский В. В.

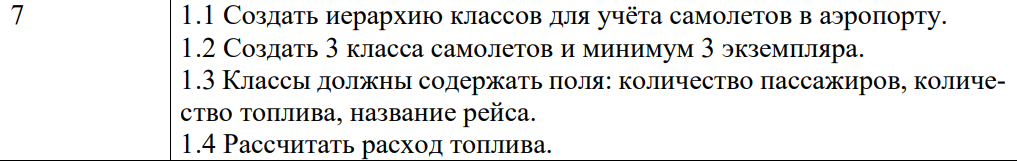
Принял: преподаватель

Башаримов Ю. С.

Гомель 2024

**Цель работы**: Разработать систему для учёта и управления самолётами в аэропорту, включающую создание иерархии классов, моделирующих различные типы самолётов, и реализацию функционала для расчёта расхода топлива и фильтрации самолётов по заданным критериям.

На рисунке 1 приведён фрагмент таблицы с индивидуальным заданием и



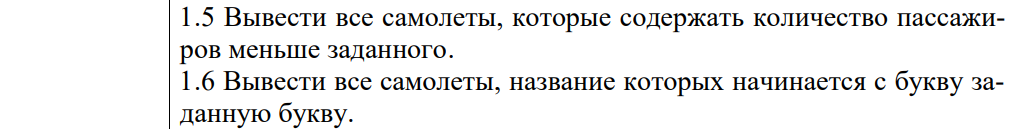


Рисунок 1 ­– Фрагмент таблицы с индивидуальными заданиями

Ход работы(Вариант 7):

Задание 1.1

Условия задания 1.1 , приведённом выше. Выполнение представлено на рисунке 1.1.

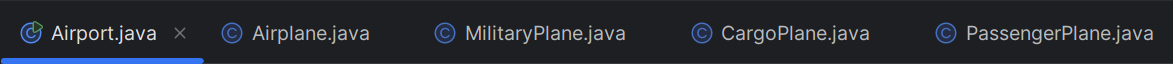


Рисунок 1.1 ­– Выполнение задания 1.1

Условия задания 1.2-1.5 на рисунке 1, приведённом выше. Код программы представлен в Приложениях А-Д.

В ходе тестирования программы можно прийти к выводу, что алгоритм работает верно. Результат выполнения программы представлен на рисунке 1.2

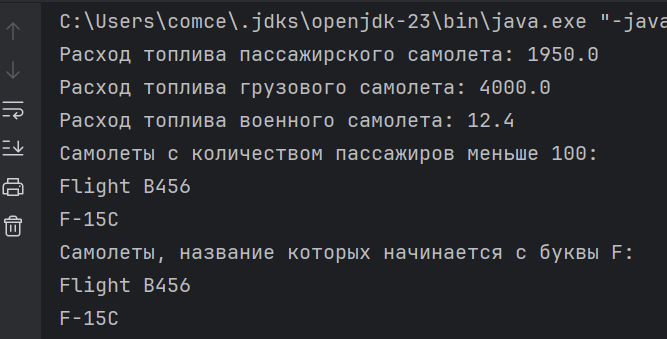


Рисунок 1.2 – Результат выполнения программы

Вывод: Разработал систему для учёта и управления самолётами. Изучил базовый синтаксис языка программирования Java. Реализовал функционал для расчёта расхода топлива.

Приложение А

(класс Airport)

import java.util.ArrayList;  
import java.util.List;  
  
public class Airport {  
 public static void main(String[] args) {  
 List<Airplane> airplanes = new ArrayList<>();  
 airplanes.add(new PassengerPlane(150, 5000, "Airbus A220-300"));  
 airplanes.add(new CargoPlane(2, 8000, "Flight B456"));  
 airplanes.add(new MilitaryPlane(2, 100, "F-15C"));  
  
 System.*out*.println("Расход топлива пассажирского самолета: " + airplanes.get(0).calculateFuelConsumption());  
 System.*out*.println("Расход топлива грузового самолета: " + airplanes.get(1).calculateFuelConsumption());  
 System.*out*.println("Расход топлива военного самолета: " + airplanes.get(2).calculateFuelConsumption());  
  
 int passengerThreshold = 100;  
 char bykva1 = 'F';  
  
 System.*out*.println("Самолеты с количеством пассажиров меньше " + passengerThreshold + ":");  
 for (Airplane airplane : airplanes) {  
 if (airplane.getPassengers() < passengerThreshold) {  
 System.*out*.println(airplane.getFlightName());  
 }  
 }  
  
 System.*out*.println("Самолеты, название которых начинается с буквы " + bykva1 + ":");  
 for (Airplane airplane : airplanes) {  
 if (airplane.getFlightName().charAt(0) == bykva1) {  
 System.*out*.println(airplane.getFlightName());  
 }  
 }  
 }  
}

Приложение Б

(класс Airplane)

public class Airplane {  
 protected int passengers;  
 protected double fuel;  
 protected String flightName;  
  
 public Airplane(int passengers, double fuel, String flightName) {  
 this.passengers = passengers;  
 this.fuel = fuel;  
 this.flightName = flightName;  
 }  
  
 public int getPassengers() {  
 return passengers;  
 }  
  
 public String getFlightName() {  
 return flightName;  
 }  
  
 public double calculateFuelConsumption() {  
 return 0;  
 }  
}

Приложение В

(класс MilitaryPlane)

public class MilitaryPlane extends Airplane {  
 public MilitaryPlane(int passengers, double fuel, String flightName) {  
 super(passengers, fuel, flightName);  
 }  
  
 @Override  
 public double calculateFuelConsumption() {  
 return fuel \* 0.124;  
 }  
}

Приложение Г

(класс CargoPlane)

public class CargoPlane extends Airplane {  
 public CargoPlane(int passengers, double fuel, String flightName) {  
 super(passengers, fuel, flightName);  
 }  
  
 @Override  
 public double calculateFuelConsumption() {  
 return fuel \* 0.5;  
 }  
}

Приложение Д

(класс PassengerPlane)

public class PassengerPlane extends Airplane {  
 public PassengerPlane(int passengers, double fuel, String flightName) {  
 super(passengers, fuel, flightName);  
 }  
  
 @Override  
 public double calculateFuelConsumption() {  
 return fuel \* 0.39;  
 }  
}