**МИНИСТЕРСТВО ОБРАЗОВАНИЯ РЕСПУБЛИКИ БЕЛАРУСЬ**

**УЧРЕЖДЕНИЕ ОБРАЗОВАНИЯ**

**ГОМЕЛЬСКИЙ ГОСУДАРСТВЕННЫЙ ТЕХНИЧЕСКИЙ**

**УНИВЕРСИТЕТ ИМЕНИ П.О.СУХОГО**

Факультет автоматизированных и информационных систем

Кафедра «Информационные технологии»

Дисциплина «Введение в разработку программного обеспечения»

ОТЧЕТ

Лабораторная работа № 1

Развитие представлений о разработке программ. Объектно-ориентированное программирование в *Java*

Выполнил студент

Группы ИТД-21

Чайдаков И.М.

Принял преподаватель

Малиновский И. Л.

Гомель 2024

Развитие представлений о разработке программ. Объектно-ориентированное программирование в *Java*

**Цель работы:** Изучить синтаксис языка *Java* и ознакомиться с основами объектно-ориентированного программирования на этом языке.

**Задание:** Создать иерархию классов для учета самолетов в аэропорту. Создать 3 класса самолетов и минимум 3 экземпляра. Классы должны содержать поля: количество пассажиров, количество топлива, название. Создать функции для: расчета расхода топлива, вывода все самолетов начинающихся с определенной буквы, вывести все самолеты с числом пассажиров меньше заданного.

**Ход выполнения задания:**

**1.** Описываем нашу иерархию классов с помощью *UML*-диаграммы (Рисунок 1).

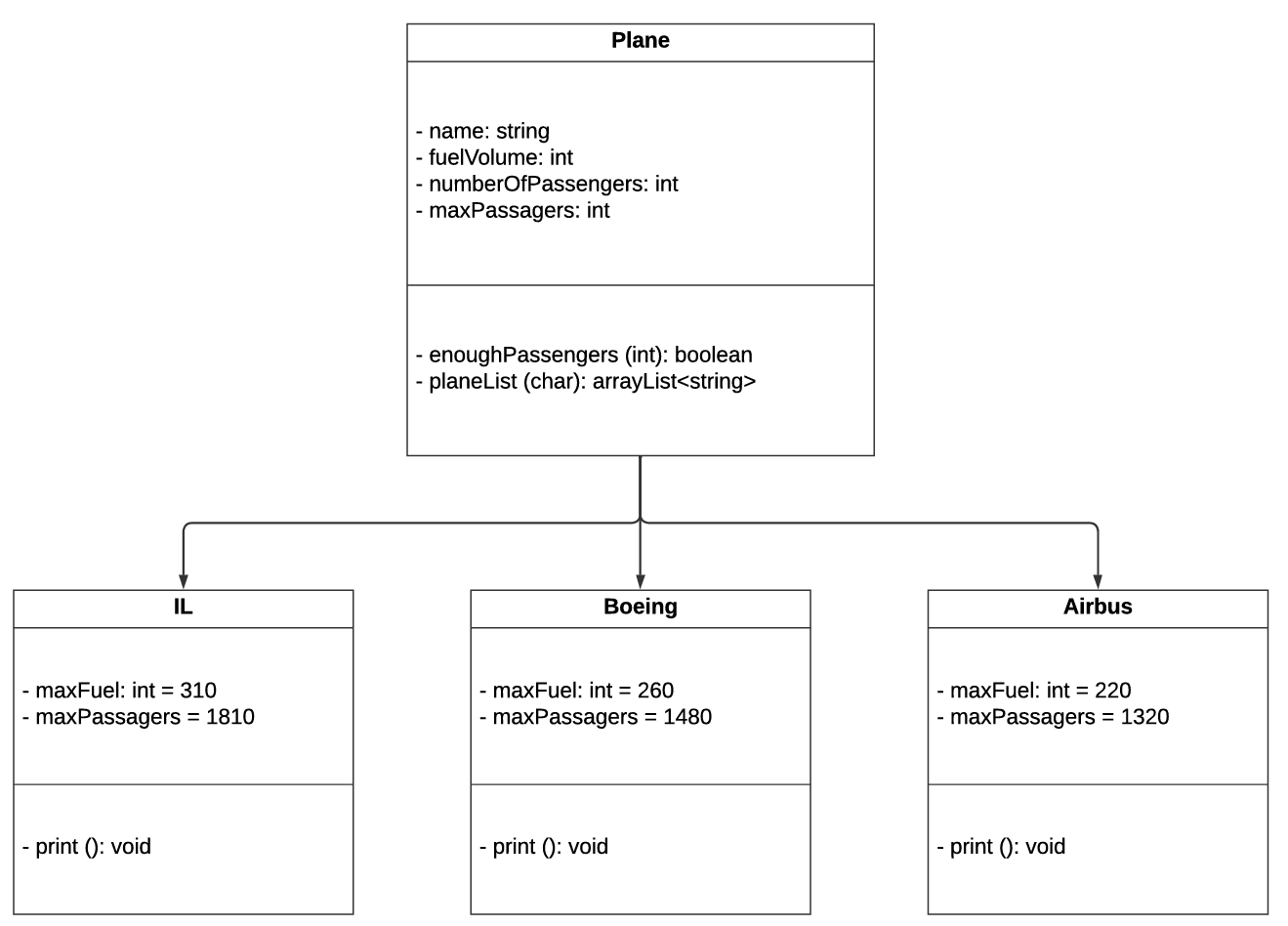


Рисунок 1 — Иерархия классов в программе.

**2.** Описываем класс Planes представляющий собой самолет и его параметры (см. Приложение А).

**3.** Описываем классы: Airbus, Boeing и IL добавляющих к родительскому классу Plane дополнительные параметры (см. Приложение A).

**4.** Описываем главный файл *Program* в котором создаем экземпляры классов и взаимодействуем с ними при помощи консольного интерфейса (Рисунок 2).

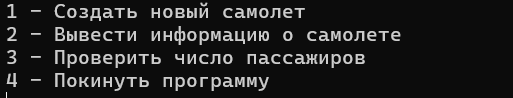


Рисунок 2 — Интерфейс программы.

**5.** Создаем экземпляр самолета модели Boeing с запасом топлива в 175 единиц в количеством пассажиров в 1100 единиц (Рисунок 3).

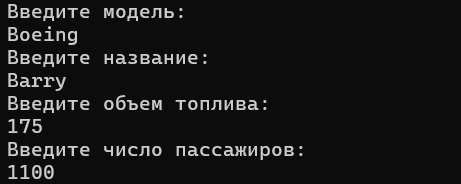


Рисунок 3 — Создание нового объекта типа *Plane*.

**6.** Теперь выведем данные о созданном самолете, открываем список самолетов и находим созданный нами объект (Рисунок 4).

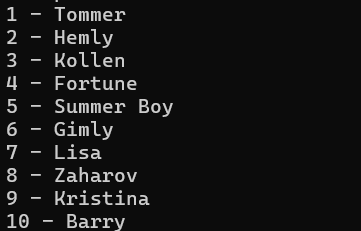


Рисунок 4 — Список всех самолетов в программе.

Откроет информацию о созданном нами самолете и просмотрим ее (Рисунок 5).

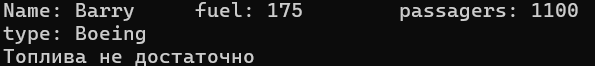


Рисунок 5 — Информация о созданном самолете.

**6.** Попробуем вывести все самолеты имеющие менее 1200 пассажиров, как мы можем наблюдать, созданный нами самолет также попадает в этот список (Рисунок 6).

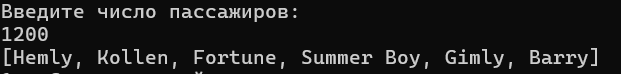


Рисунок 6 — Список самолетов с менее чем 1200 пассажирами на борту.

**Вывод:** *Java* предоставляет удобный инструментарий для объектно-ориентированного программирования позволяя создавать легко читаемые и полезные приложения. Удобная система сборки проекта, разные ценовые категории компилятора и группировка классов в пакеты позволяют сделать разработку на языке *Java* удобной для любого пользователя.

**ПРИЛОЖЕНИЕ А**

**Код используемых классов**

**Plane**

package Planes;

/\*\*

 \* Класс Plane представляет самолет с его характеристиками:

 \* название, количество топлива, пассажиры и максимальная емкость топлива.

 \*/

public class Plane {

    // Название самолета

    private String name = "";

    /\*\*

     \* Возвращает название самолета.

     \*

     \* @return название самолета

     \*/

    public String getName() {

        return name;

    }

    // Количество топлива в самолете

    private int fuel = 0;

    /\*\*

     \* Возвращает текущее количество топлива в самолете.

     \*

     \* @return количество топлива

     \*/

    public int getFuel() {

        return fuel;

    }

    // Количество пассажиров

    private int passengers = 0;

    /\*\*

     \* Возвращает количество пассажиров на борту.

     \*

     \* @return количество пассажиров

     \*/

    public int getPassengers() {

        return passengers;

    }

    // Максимальное количество топлива, которое может вместить самолет

    private int maxFuel = 0;

    /\*\*

     \* Возвращает максимальное количество топлива, которое может вместить самолет.

     \*

     \* @return максимальное количество топлива

     \*/

    public int getMaxFuel() {

        return maxFuel;

    }

    /\*\*

     \* Конструктор класса Plane, который инициализирует основные параметры самолета.

     \*

     \* @param name название самолета

     \* @param fuel количество топлива

     \* @param passengers количество пассажиров

     \* @param maxFuel максимальная емкость топлива

     \*/

    public Plane(String name, int fuel, int passengers, int maxFuel) {

        this.name = name;

        this.fuel = fuel;

        this.passengers = passengers;

        this.maxFuel = maxFuel;

    }

    /\*\*

     \* Проверяет, достаточно ли топлива для полета. Условие полета —

     \* должно быть больше 70% от максимальной емкости топлива.

     \*

     \* @return true, если топлива достаточно, иначе false

     \*/

    public boolean enoughFuel() {

        return this.fuel - maxFuel \* 0.7 > 0;

    }

    /\*\*

     \* Печатает информацию о самолете: название, количество топлива, пассажиров и максимальную емкость.

     \*/

    public void print() {

        System.out.printf("Name: %s \tfuel: %d\t passengers: %d\t maxFuel: %d\n", name, fuel, passengers, maxFuel);

    }

}

**Airbus**

package Planes;

/\*\*

 \* Класс Airbus представляет самолет типа Airbus, наследующий основные характеристики от класса Plane.

 \* Ограничения: максимальная вместимость — 1320 пассажиров, максимальная емкость топлива — 220.

 \*/

public class Airbus extends Plane {

    // Максимальное количество пассажиров для Airbus

    public static int maxPassengers = 1320;

    // Максимальное количество топлива для Airbus

    public static int maxFuel = 220;

    /\*\*

     \* Конструктор класса Airbus.

     \* Инициализирует самолет с заданными названием, количеством топлива и пассажиров,

     \* проверяя их на соответствие ограничениям по максимальному топливу и пассажирам.

     \*

     \* @param name название самолета

     \* @param fuel количество топлива

     \* @param passengers количество пассажиров

     \*/

    public Airbus(String name, int fuel, int passengers) {

        // Используем Math.min для предотвращения превышения максимальных значений топлива и пассажиров

        super(name, Math.min(fuel, maxFuel), Math.min(passengers, maxPassengers), maxFuel);

        // Выводим сообщение об ошибке, если заданные параметры превышают допустимые пределы

        if (fuel > maxFuel || passengers > maxPassengers) {

            System.out.println("Creation error: Exceeded max limits");

        }

    }

    /\*\*

     \* Переопределенный метод для вывода информации о самолете.

     \* Выводит название, количество топлива и пассажиров, а также указывает, что это Airbus.

     \*/

    @Override

    public void print() {

        System.out.printf("Name: %s \tfuel: %d\t passengers: %d\ntype - Airbus\n",

                        super.getName(), super.getFuel(), super.getPassengers(), super.getMaxFuel());

    }

}

**Boeing**

package Planes;

/\*\*

 \* Класс Boeing представляет самолет типа Boeing, наследующий основные характеристики от класса Plane.

 \* Ограничения: максимальная вместимость — 1480 пассажиров, максимальная емкость топлива — 260.

 \*/

public class Boeing extends Plane {

    // Максимальное количество пассажиров для Boeing

    public static int maxPassengers = 1480;

    // Максимальное количество топлива для Boeing

    public static int maxFuel = 260;

    /\*\*

     \* Конструктор класса Boeing.

     \* Инициализирует самолет с заданными названием, количеством топлива и пассажиров,

     \* проверяя их на соответствие ограничениям по максимальному топливу и пассажирам.

     \*

     \* @param name название самолета

     \* @param fuel количество топлива

     \* @param passengers количество пассажиров

     \*/

    public Boeing(String name, int fuel, int passengers) {

        // Используем Math.min для предотвращения превышения максимальных значений топлива и пассажиров

        super(name, Math.min(fuel, maxFuel), Math.min(passengers, maxPassengers), maxFuel);

        // Выводим сообщение об ошибке, если заданные параметры превышают допустимые пределы

        if (fuel > maxFuel || passengers > maxPassengers) {

            System.out.println("Creation error: Exceeded max limits");

        }

    }

    /\*\*

     \* Переопределенный метод для вывода информации о самолете.

     \* Выводит название, количество топлива и пассажиров, а также указывает, что это Boeing.

     \*/

    @Override

    public void print() {

        System.out.printf("Name: %s \tfuel: %d\t passagers: %d\ntype: Boeing\n",

                        super.getName(), super.getFuel(), super.getPassengers(), super.getMaxFuel());

    }

}

**IL**

package Planes;

/\*\*

 \* Класс IL представляет самолет типа IL, наследующий основные характеристики от класса Plane.

 \* Ограничения: максимальная вместимость — 1810 пассажиров, максимальная емкость топлива — 310.

 \*/

public class IL extends Plane {

    // Максимальное количество пассажиров для IL

    public static int maxPassengers = 1810;

    // Максимальное количество топлива для IL

    public static int maxFuel = 310;

    /\*\*

     \* Конструктор класса IL.

     \* Инициализирует самолет с заданными названием, количеством топлива и пассажиров,

     \* проверяя их на соответствие ограничениям по максимальному топливу и пассажирам.

     \*

     \* @param name название самолета

     \* @param fuel количество топлива

     \* @param passengers количество пассажиров

     \*/

    public IL(String name, int fuel, int passengers) {

        // Используем Math.min для предотвращения превышения максимальных значений топлива и пассажиров

        super(name, Math.min(fuel, maxFuel), Math.min(passengers, maxPassengers), maxFuel);

        // Выводим сообщение об ошибке, если заданные параметры превышают допустимые пределы

        if (fuel > maxFuel || passengers > maxPassengers) {

            System.out.println("Creation error: Exceeded max limits");

        }

    }

    /\*\*

     \* Переопределенный метод для вывода информации о самолете.

     \* Выводит название, количество топлива и пассажиров, а также указывает, что это IL.

     \*/

    @Override

    public void print() {

        System.out.printf("Name: %s \tfuel: %d\t passagers: %d\ntype: IL\n",

                        super.getName(), super.getFuel(), super.getPassengers(), super.getMaxFuel());

    }

}

**ПРИЛОЖЕНИЕ B**

**Код класса Program**

import java.util.ArrayList;

import java.util.Scanner;

import Planes.Boeing;

import Planes.Airbus;

import Planes.IL;

import Planes.Plane;

/\*\*

 \* Основной класс программы, который предоставляет функции для создания и управления объектами самолетов.

 \* Поддерживает модели Boeing, Airbus, IL.

 \*/

public class Program {

    /\*\*

     \* Возвращает список самолетов, название которых начинается на заданную букву.

     \*

     \* @param list список самолетов

     \* @param letter начальная буква имени самолета

     \* @return список самолетов, название которых начинается на заданную букву

     \*/

    public static ArrayList<Plane> planeList(ArrayList<Plane> list, String letter) {

        ArrayList<Plane> newList = new ArrayList<Plane>();

        for (Plane plane : list) {

            if (plane.getName().startsWith(letter)) {

                newList.add(plane);

            }

        }

        return newList;

    }

    /\*\*

     \* Создает новый самолет в зависимости от модели и добавляет его в список.

     \* Проверяет корректность введенных данных, включая топливо и пассажиров.

     \*

     \* @param list список самолетов, в который будет добавлен новый самолет

     \* @param scanner объект для ввода данных с консоли

     \* @return обновленный список самолетов

     \*/

    private static ArrayList<Plane> createPlane(ArrayList<Plane> list, Scanner scanner) {

        System.out.println("Введите модель: ");

        scanner.nextLine();

        String model = scanner.nextLine();

        System.out.println("Введите название: ");

        String name = scanner.nextLine();

        System.out.println("Введите объем топлива: ");

        int fuel = scanner.nextInt();

        System.out.println("Введите число пассажиров: ");

        int passengers = scanner.nextInt();

        switch(model) {

            case "Boeing": {

                if (fuel <= Boeing.maxFuel && passengers <= Boeing.maxPassengers) {

                    Boeing plane = new Boeing(name, fuel, passengers);

                    list.add(plane);

                } else {

                    System.out.println("Input data error. Please, try again");

                }

                break;

            }

            case "Airbus": {

                if (fuel <= Airbus.maxFuel && passengers <= Airbus.maxPassengers) {

                    Airbus plane = new Airbus(name, fuel, passengers);

                    list.add(plane);

                } else {

                    System.out.println("Input data error. Please, try again");

                }

                break;

            }

            case "IL": {

                if (fuel <= IL.maxFuel && passengers <= IL.maxPassengers) {

                    IL plane = new IL(name, fuel, passengers);

                    list.add(plane);

                } else {

                    System.out.println("Input data error. Please, try again");

                }

                break;

            }

            default: {

                System.out.println("model error. Please, try again");

                break;

            }

        }

        return list;

    }

    /\*\*

     \* Основной метод программы. Позволяет пользователю добавлять самолеты, просматривать информацию

     \* о них, проверять количество пассажиров и выходить из программы.

     \*

     \* @param args аргументы командной строки (не используются)

     \*/

    public static void main(String[] args) {

        ArrayList<Plane> planeList = new ArrayList<Plane>();

        ArrayList<String> nameList = new ArrayList<String>();

        // Создание начальных объектов самолетов

        Boeing boeing\_1 = new Boeing("Tommer", 190, 1275);

        Boeing boeing\_2 = new Boeing("Hemly", 179, 1138);

        Boeing boeing\_3 = new Boeing("Kollen", 192, 1056);

        Airbus airbus\_1 = new Airbus("Fortune", 168, 689);

        Airbus airbus\_2 = new Airbus("Summer Boy", 112, 1007);

        Airbus airbus\_3 = new Airbus("Gimly", 202, 1159);

        IL iL\_1 = new IL("Lisa", 266, 1734);

        IL iL\_2 = new IL("Zaharov", 241, 1612);

        IL iL\_3 = new IL("Kristina", 247, 1483);

        // Добавление самолетов в список

        planeList.add(boeing\_1);

        planeList.add(boeing\_2);

        planeList.add(boeing\_3);

        planeList.add(airbus\_1);

        planeList.add(airbus\_2);

        planeList.add(airbus\_3);

        planeList.add(iL\_1);

        planeList.add(iL\_2);

        planeList.add(iL\_3);

        Scanner in = new Scanner(System.in);

        boolean endIsNear = false;

        int choice;

        int i;

        // Основной цикл программы

        while (!endIsNear) {

            System.out.printf("1 - Создать новый самолет\n" +

                              "2 - Вывести информацию о самолете\n" +

                              "3 - Проверить число пассажиров\n" +

                              "4 - Покинуть программу\n");

            choice = in.nextInt();

            switch(choice) {

                case 1: {

                    // Создание нового самолета

                    createPlane(planeList, in);

                    break;

                }

                case 2: {

                    // Вывод информации о самолете

                    i = 1;

                    System.out.printf("Выберите самолет из списка: \n");

                    for (Plane plane : planeList) {

                        System.out.printf("%d - %s\n", i, plane.getName());

                        i++;

                    }

                    choice = in.nextInt();

                    if (choice > planeList.size()) {

                        System.out.println("input error. Please, try again");

                        break;

                    }

                    planeList.get(choice-1).print();

                    if (planeList.get(choice-1).enoughFuel()) {

                        System.out.println("Топлива достаточно");

                    } else {

                        System.out.println("Топлива не достаточно");

                    }

                    break;

                }

                case 3: {

                    // Проверка на количество пассажиров

                    System.out.println("Введите число пассажиров: ");

                    int num = in.nextInt();

                    for (Plane plane : planeList) {

                        if (plane.getPassengers() < num) {

                            nameList.add(plane.getName());

                        }

                    }

                    System.out.println(nameList);

                    break;

                }

                case 4: {

                    // Завершение программы

                    endIsNear = true;

                    break;

                }

                default: {

                    System.out.println("input error. Please, try again");

                    break;

                }

            }

        }

        in.close();

    }

}