МИНИСТЕРСТВО ОБРАЗОВАНИЯ РЕСПУБЛИКИ БЕЛАРУСЬ
УЧРЕЖДЕНИЕ ОБРАЗОВАНИЯ
“БРЕСТСКИЙ ГОСУДАРСТВЕННЫЙ ТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ”
КАФЕДРА ИНТЕЛЛЕКТУАЛЬНЫХ ИНФОРМАЦИОННЫХ ТЕХНОЛОГИЙ

ОТЧЁТ
по лабораторной работе №3

Выполнила
студентка группы ПО-9
Тупик Ю. Л.

Проверил:
Крощенко А. А.

Брест 2024

Цель работы: научиться создавать и использовать классы в программах на языке программирования Java.

Задание 1

8) Множество целых чисел переменной мощности – Предусмотреть возможность пересечения двух множеств, вывода на печать элементов множества, а так же метод, определяющий, принадлежит ли указанное значение множеству. Класс должен содержать методы, позволяющие добавлять и удалять элемент в/из множества. Конструктор должен позволить создавать объекты с начальной инициализацией. Реализацию множества осуществить на базе структуры ArrayList. Реализовать метод equals, выполняющий сравнение объектов данного типа.

Выполнение задания

Код программы

```
import java.util.ArrayList;

public class IntegerSet {
    private ArrayList<Integer> elements;

    // Конструктор без параметров
    public IntegerSet() {
        this.elements = new ArrayList<>();
    }

    // Конструктор с начальной инициализацией
    public IntegerSet(ArrayList<Integer> elements) {
        this.elements = new ArrayList<>(elements);
    }

    // Метод для добавления элемента в множество
    public void add(int element) {
        if (!contains(element)) {
            elements.add(element);
        }
    }

    // Метод для удаления элемента из множества
    public void remove(int element) {
        elements.remove(Integer.valueOf(element));
    }

    // Метод для проверки, содержится ли элемент в множестве
    public boolean contains(int element) {
        return elements.contains(element);
    }

    // Метод для вычисления пересечения двух множеств
    public IntegerSet intersect(IntegerSet otherSet) {
        ArrayList<Integer> intersection = new ArrayList<>();
        for (int element : elements) {
            if (otherSet.contains(element)) {
                intersection.add(element);
            }
        }
```

```
        return new IntegerSet(intersection);
    }

    // Переопределение метода equals для сравнения объектов IntegerSet
    @Override
    public boolean equals(Object obj) {
        if (this == obj) return true;
        if (obj == null || getClass() != obj.getClass()) return false;
        IntegerSet that = (IntegerSet) obj;
        return this.elements.equals(that.elements);
    }

    // Переопределение метода toString для вывода элементов множества
    @Override
    public String toString() {
        return "elements=" + elements;
    }
}

class Main {
    public static void main(String[] args) {
        // Примеры использования
        IntegerSet set1 = new IntegerSet();
        set1.add(1);
        set1.add(2);
        set1.add(3);

        IntegerSet set2 = new IntegerSet();
        set2.add(2);
        set2.add(3);
        set2.add(4);

        // Вывод элементов множества
        System.out.println("Множество 1: " + set1);
        System.out.println("Множество 2: " + set2);

        // Пересечение двух множеств
        IntegerSet intersection = set1.intersect(set2);
        System.out.println("Пересечение множеств: " + intersection);

        // Проверка наличия элемента в множестве
        System.out.println("Множество 1 содержит элемент 2: " +
set1.contains(2));

        // Проверка равенства множеств
        System.out.println("Множества 1 и 2 равны: " + set1.equals(set2));
    }
}
```

Результат

```
C:\Users\Юлия\.jdks\corretto-17.0.9\bin\ja
Множество 1: elements=[1, 2, 3]
Множество 2: elements=[2, 3, 4]
Пересечение множеств: elements=[2, 3]
Множество 1 содержит элемент 2: true
Множества 1 и 2 равны: false

Process finished with exit code 0
```

Задание 2

8) Автоматизированная система обработки информации об авиарейсах

Написать программу для обработки информации об авиарейсах (Airlines): Каждый рейс имеет следующие характеристики:

• Пункт назначения;
• Номер рейса;
• Тип самолета;
• Время вылета;
• Дни недели, по которым совершаются рейсы.

Программа должна обеспечить:

• Генерацию списка рейсов;
• Вывод списка рейсов для заданного пункта назначения;
• Вывод списка рейсов для заданного дня недели;
• Вывод списка рейсов для заданного дня недели, время вылета для которых больше заданного;
• Все рейсы самолетов некоторого типа;
• Группировка рейсов по числу пассажиров (маломестные - 1-100 чел, средместные (100-200), крупные рейсы (200-350))

Выполнение задания

Код программы

```
import java.io.BufferedReader;
import java.io.FileReader;
import java.io.IOException;
import java.time.LocalDateTime;
import java.util.ArrayList;
import java.util.List;
import java.time.DayOfWeek;
import java.time.format.TextStyle;
import java.util.HashMap;
import java.util.Locale;
import java.util.Map;
import java.util.stream.Collectors;

public class Flight {
    private String destination;
    private String flightNumber;
    private String aircraftType;
    private LocalDateTime departureTime;
    private List<String> daysOfWeek;

    public Flight(String destination, String flightNumber, String aircraftType,
  LocalDateTime departureTime, List<String> daysOfWeek) {
        this.destination = destination;
        this.flightNumber = flightNumber;
        this.aircraftType = aircraftType;
        this.departureTime = departureTime;
        this.daysOfWeek = daysOfWeek;
    }

    // Геттеры дял доступа к полям класса Flights
    public String getDestination() {
        return destination;
    }

    public LocalDateTime getDepartureTime() {
```

```
        return departureTime;
    }

    public List<String> getDaysOfWeek() {
        return daysOfWeek;
    }

    public String getFlightNumber() {
        return flightNumber;
    }

    public String getAircraftType() {
        return aircraftType;
    }

    // Пример использования классов
    public static void main(String[] args) {
        List<Flight> flights = readFlightsFromFile("src/flights.txt");
        FlightManager manager = new FlightManager(flights);

        // Вывод списка всех рейсов
        System.out.println("Список всех рейсов:");
        manager.printFlights(flights);

        // Вывод списка рейсов для заданного пункта назначения
        String destination = "Москву";
        System.out.println("Рейсы в " + destination + ":");
        List<Flight> flightsToDestination =
manager.getFlightsByDestination(destination);
        manager.printFlights(flightsToDestination);

        // Вывод списка рейсов для заданного дня недели
        String dayOfWeek = "Пн";
        System.out.println("Рейсы на " + dayOfWeek + ":");
        List<Flight> flightsOnDay = manager.getFlightsByDayOfWeek(dayOfWeek);
        manager.printFlights(flightsOnDay);

        // Вывод списка рейсов для заданного дня недели, время вылета для которых
больше заданного
        int hour = 10;
        int minute = 0;
        System.out.println("Рейсы на " + dayOfWeek + ", вылет после " + hour + ":"
+ minute + ":");
        List<Flight> flightsOnDayAfterTime =
manager.getFlightsByDayAndTimeAfter(dayOfWeek, hour, minute);
        manager.printFlights(flightsOnDayAfterTime);

        // Все рейсы самолетов некоторого типа
        String aircraftType = "Boeing 747";
        System.out.println("Рейсы на самолете типа " + aircraftType + ":");
        List<Flight> flightsByAircraftType =
manager.getFlightsByAircraftType(aircraftType);
        manager.printFlights(flightsByAircraftType);

        // Группировка рейсов по числу пассажиров
        System.out.println("Группировка рейсов по числу пассажиров:");
        manager.printGroupedFlightsByPassengerCapacity();
    }

    // Метод для чтения рейсов из файла
    private static List<Flight> readFlightsFromFile(String filename) {
        List<Flight> flights = new ArrayList<>();
```

```
        try (BufferedReader br = new BufferedReader(new FileReader(filename))) {
            String line;
            while ((line = br.readLine()) != null) {
                String[] parts = line.split(",");
                String destination = parts[0];
                String flightNumber = parts[1];
                String aircraftType = parts[2];
                LocalDateTime departureTime = LocalDateTime.parse(parts[3]);
                List<String> daysOfWeek = new ArrayList<>();
                for (int i = 4; i < parts.length; i++) {
                    daysOfWeek.add(parts[i]);
                }
                flights.add(new Flight(destination, flightNumber, aircraftType,
  departureTime, daysOfWeek));
            }
        } catch (IOException e) {
            e.printStackTrace();
        }
        return flights;
    }
}

    class FlightManager {
        private List<Flight> flights;

        public FlightManager(List<Flight> flights) {
            this.flights = flights;
        }

        // Метод для получения списка рейсов для заданного пункта назначения
        public List<Flight> getFlightsByDestination(String destination) {
            return flights.stream()
                    .filter(flight ->
  flight.getDestination().equalsIgnoreCase(destination))
                    .collect(Collectors.toList());
        }

        // Метод для получения списка рейсов для заданного дня недели
        public List<Flight> getFlightsByDayOfWeek(String dayOfWeek) {
            return flights.stream()
                    .filter(flight -> flight.getDaysOfWeek().contains(dayOfWeek))
                    .collect(Collectors.toList());
        }

        // Метод для получения списка рейсов для заданного дня недели и времени
  вылета после указанного
        public List<Flight> getFlightsByDayAndTimeAfter(String dayOfWeek, int
  hour, int minute) {
            return flights.stream()
                    .filter(flight -> flight.getDaysOfWeek().contains(dayOfWeek))
                    .filter(flight -> flight.getDepartureTime().getHour() > hour
  ||
                            (flight.getDepartureTime().getHour() == hour &&
                                    flight.getDepartureTime().getMinute() >
  minute))
                    .collect(Collectors.toList());
        }

        // Метод для получения списка рейсов для заданного типа самолета
        public List<Flight> getFlightsByAircraftType(String aircraftType) {
            return flights.stream()
```

```
                .filter(flight ->
flight.getAircraftType().equalsIgnoreCase(aircraftType))
                .collect(Collectors.toList());
    }

    // Метод для группировки рейсов по числу пассажиров
    public Map<String, List<Flight>> groupFlightsByPassengerCapacity() {
        Map<String, List<Flight>> groupedFlights = new HashMap<>();
        List<Flight> smallFlights = new ArrayList<>();
        List<Flight> mediumFlights = new ArrayList<>();
        List<Flight> largeFlights = new ArrayList<>();

        for (Flight flight : flights) {
            int capacity =
calculatePassengerCapacity(flight.getAircraftType());
            if (capacity <= 100) {
                smallFlights.add(flight);
            } else if (capacity <= 200) {
                mediumFlights.add(flight);
            } else {
                largeFlights.add(flight);
            }
        }

        groupedFlights.put("Маломестные рейсы (1-100 чел)", smallFlights);
        groupedFlights.put("Среднеместные рейсы (100-200 чел)",
mediumFlights);
        groupedFlights.put("Крупные рейсы (200-350 чел)", largeFlights);

        return groupedFlights;
    }

    // Метод для расчета вместимости самолета по типу
    private int calculatePassengerCapacity(String aircraftType) {
        if (aircraftType.equalsIgnoreCase("Boeing 747")) {
            return 150;
        } else if (aircraftType.equalsIgnoreCase("Airbus A340")) {
            return 295;
        } else {
            return 90;
        }
    }

    // Метод для вывода группированных рейсов по числу пассажиров
    public void printGroupedFlightsByPassengerCapacity() {
        Map<String, List<Flight>> groupedFlights =
groupFlightsByPassengerCapacity();
        for (Map.Entry<String, List<Flight>> entry :
groupedFlights.entrySet()) {
            System.out.println(entry.getKey());
            for (Flight flight : entry.getValue()) {
                System.out.println(flight.getFlightNumber() + " - " +
flight.getAircraftType());
            }
        }
    }

    // Метод для вывода списка рейсов
    public void printFlights(List<Flight> flights) {
        for (Flight flight : flights) {
            StringBuilder days = new StringBuilder();
            for (String dayAbbreviation : flight.getDaysOfWeek()) {
```

```
                    // Получаем объект DayOfWeek на основе целочисленного значения
                    DayOfWeek dayOfWeek =
DayOfWeek.of(getDayOfWeekValue(dayAbbreviation));
                    // Получаем полное название дня недели
                    String fullDayName = dayOfWeek.getDisplayName(TextStyle.FULL,
Locale.getDefault());
                    days.append(fullDayName).append(", ");
                }
                if (days.length() > 0) {
                    days.delete(days.length() - 2, days.length()); // Убираем
лишнюю запятую и пробел в конце
                }
                System.out.println("Номер рейса: " + flight.getFlightNumber());
                System.out.println("Пункт назначения: " +
flight.getDestination());
                System.out.println("Тип самолета: " + flight.getAircraftType());
                System.out.println("Время вылета: " + flight.getDepartureTime());
                System.out.println("Дни недели: " + days);
                System.out.println();
            }
        }

        // Метод для преобразования сокращенного названия дня недели в
целочисленное значение
        private int getDayOfWeekValue(String abbreviation) {
            switch (abbreviation.toUpperCase()) {
                case "ПН":
                    return 1;
                case "ВТ":
                    return 2;
                case "СР":
                    return 3;
                case "ЧТ":
                    return 4;
                case "ПТ":
                    return 5;
                case "СБ":
                    return 6;
                case "ВС":
                    return 7;
                default:
                    throw new IllegalArgumentException("Неверное сокращенное
название дня недели: " + abbreviation);
            }
        }
    }
```

Flights.txt

Москва,SU101,Airbus A340,2024-04-21T08:00:00,Пн,Ср,Пт
Санкт-Петербург,S7345,Boeing 747,2024-04-22T10:00:00,Вт,Чт,Сб
Сочи,U6279,Boeing 747,2024-04-23T12:00:00,Ср,Пт
Екатеринбург,U2345,Airbus A320,2024-04-24T14:00:00,Пн,Ср,Сб
Казань,RU456,Boeing 747,2024-04-25T16:00:00,Вт,Чт

Результат

Список всех рейсов

```
C:\Users\Юлия\.jdks\corretto-17.0.9\bin\java.exe
Список всех рейсов:
Номер рейса: SU101
Пункт назначения: Москва
Тип самолета: Airbus A340
Время вылета: 2024-04-21T08:00
Дни недели: понедельник, среда, пятница

Номер рейса: S7345
Пункт назначения: Санкт-Петербург
Тип самолета: Boeing 747
Время вылета: 2024-04-22T10:00
Дни недели: вторник, четверг, суббота

Номер рейса: U6279
Пункт назначения: Сочи
Тип самолета: Boeing 747
Время вылета: 2024-04-23T12:00
Дни недели: среда, пятница

Номер рейса: U2345
Пункт назначения: Екатеринбург
Тип самолета: Airbus A320
Время вылета: 2024-04-24T14:00
Дни недели: понедельник, среда, суббота
```

```
Номер рейса: RU456
Пункт назначения: Казань
Тип самолета: Boeing 747
Время вылета: 2024-04-25T16:00
Дни недели: вторник, четверг
```

Рейсы в Москву

```
Рейсы в Москва:
Номер рейса: SU101
Пункт назначения: Москва
Тип самолета: Airbus A340
Время вылета: 2024-04-21T08:00
Дни недели: понедельник, среда, пятница
```

Рейсы на Пн

```
Рейсы на Пн:
Номер рейса: SU101
Пункт назначения: Москва
Тип самолета: Airbus A340
Время вылета: 2024-04-21T08:00
Дни недели: понедельник, среда, пятница

Номер рейса: U2345
Пункт назначения: Екатеринбург
Тип самолета: Airbus A320
Время вылета: 2024-04-24T14:00
Дни недели: понедельник, среда, суббота
```

Рейсы на Пн, вылет после 10:0

```
Рейсы на Пн, вылет после 10:0:
Номер рейса: U2345
Пункт назначения: Екатеринбург
Тип самолета: Airbus A320
Время вылета: 2024-04-24T14:00
Дни недели: понедельник, среда, суббота
```

Рейсы на самолете типа Boeing 747

```
Рейсы на самолете типа Boeing 747:
Номер рейса: S7345
Пункт назначения: Санкт-Петербург
Тип самолета: Boeing 747
Время вылета: 2024-04-22T10:00
Дни недели: вторник, четверг, суббота

Номер рейса: U6279
Пункт назначения: Сочи
Тип самолета: Boeing 747
Время вылета: 2024-04-23T12:00
Дни недели: среда, пятница

Номер рейса: RU456
Пункт назначения: Казань
Тип самолета: Boeing 747
Время вылета: 2024-04-25T16:00
Дни недели: вторник, четверг
```

Группировка рейсов по числу пассажиров

```
Группировка рейсов по числу пассажиров:
Маломестные рейсы (1-100 чел)
U2345 - Airbus A320
Среднеместные рейсы (100-200 чел)
S7345 - Boeing 747
U6279 - Boeing 747
RU456 - Boeing 747
Крупные рейсы (200-350 чел)
SU101 - Airbus A340

Process finished with exit code 0
```

Вывод: научилась создавать и использовать классы в программах на языке программирования Java.