

**Министерство образования Республики Беларусь**

**Учреждение образования**

**«Брестский государственный технический университет»**

Кафедра ИИТ

**ОТЧЕТ**

**ЛАБОРАТОРНАЯ РАБОТА №3**

ССП

Выполнил  
студент группы  
АС-50:  
Протасевич А.В.  
Проверил:  
Крощенко А.А.

Брест 2020

Цель работы:

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

## Вариант 8

### Задание 1

Реализовать простой класс.

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.

Код программы:

Main.java:

```
import java.util.ArrayList;
```

```
public class main {
    public static void main(String[] args) {
        IntegerSet integerSet1 = new IntegerSet();
        integerSet1.addItem(15);
        integerSet1.addItem(135);
        integerSet1.addItem(20);
        ArrayList<Integer> integerList = new ArrayList() {{
            add(10);
            add(25);
            add(135);
        }};
        IntegerSet integerSet2 = new IntegerSet(integerList);
        System.out.println("First set: " + integerSet1);
        System.out.println("Second set: " + integerSet2);
        System.out.println("integerSet1 == integerSet2: " +
            (integerSet1.equals(integerSet2)));
        integerSet2.addItem(123);
        integerSet1.deleteItemById(2);
        System.out.println("Second item of integerSet2 = " +
            integerSet2.getItemById(2));
        System.out.println("integerSet2 contains 25: " + integerSet2.contains(25));
        System.out.println("intersections: " + integerSet2.intersections(integerSet1));
    }
}
```

IntegerSet:

```
import java.util.ArrayList;
```

```
public class IntegerSet {
    private ArrayList<Integer> setOfIntegers;

    public IntegerSet() {
        this.setOfIntegers = new ArrayList();
    }

    public IntegerSet(ArrayList<Integer> setOfIntegers) {
        this.setOfIntegers = new ArrayList<>();
        for(Integer el: setOfIntegers) {
            if (!this.setOfIntegers.contains(el))
                this.setOfIntegers.add(el);
        }
    }

    public ArrayList<Integer> intersections(IntegerSet set) {
        ArrayList<Integer> list = set.getSetOfIntegers();
        list.retainAll(this.setOfIntegers);
        return list;
    }

    public boolean contains(int item) {
        return this.setOfIntegers.contains(item);
    }

    public int getItemById(int id) {
        --id;
        return (Integer)this.setOfIntegers.get(id);
    }

    public void addItem(int item) {
        this.setOfIntegers.add(item);
    }

    public void deleteItemById(int id) {
        this.setOfIntegers.remove(id);
    }

    public ArrayList<Integer> getSetOfIntegers() {
        return this.setOfIntegers;
    }

    public void setSetOfIntegers(ArrayList<Integer> setOfIntegers) {
        this.setOfIntegers = setOfIntegers;
    }

    public String toString() {
        return "IntegerSet = " + this.setOfIntegers;
    }
}
```

```

public boolean equals(Object o) {
    if (this == o) {
        return true;
    } else if (o != null && this.getClass() == o.getClass()) {
        IntegerSet that = (IntegerSet)o;
        return this.setOfIntegers.equals(that.setOfIntegers);
    } else {
        return false;
    }
}
}

```

Вывод программы:

```

First set: IntegerSet = [15, 135, 20]
Second set: IntegerSet = [10, 25, 135]
integerSet1 == integerSet2: false
Second item of integerSet2 = 25
integerSet2 contains 25: true
intersections: [135]

```

## Задание 2

Разработать автоматизированную систему на основе некоторой структуры данных, манипулирующей объектами пользовательского класса. Реализовать требуемые функции обработки данных.

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));
- Все рейсы самолетов туда-обратно.

Код программы:

Main.java

```
import java.io.*;
import java.text.ParseException;
import java.text.SimpleDateFormat;
import java.util.ArrayList;

public class main {
    private static Airline airline;
    private static final SimpleDateFormat dateFormat = new SimpleDateFormat("dd.MM.yyyy
HH:mm:ss");

    public static void main(String[] args) throws ParseException {
        airline = new Airline(new ArrayList<>());
        getAirline();
        System.out.println("All flights:");
        airline.printListOfFlights();
        System.out.println("\nFlights for selected destination:");
        airline.printListOfFlightsForSelectedDestination("New York");
        System.out.println("\nFlights for selected day:");
        airline.printListOfFlightsForSelectedDay("TUESDAY");
        System.out.println("\nFlights for selected day and time:");
        airline.printListOfFlightsForSelectedDayAndTime("FRIDAY",
            dateFormat.parse("13.10.2020 15:35:10"));
        System.out.println("\nFlights for selected type:");
        airline.printListOfFlightsForSelectedType("SMALL");
    }

    private static void getAirline() {
        try {
            File file = new File("fl.txt");
            FileReader fr = new FileReader(file);
            BufferedReader reader = new BufferedReader(fr);
            String line = reader.readLine();
            while (line != null) {
                String[] words = line.split("\t");

                airline.addFlights(new Flight(words[0], words[1], words[2], dateFormat.parse(words[3]),
words[4]));
                line = reader.readLine();
            }

        } catch (FileNotFoundException e) {
```

```

        e.printStackTrace();
    } catch (IOException e) {
        e.printStackTrace();
    } catch (ParseException e) {
        e.printStackTrace();
    }
}
}
}

```

## Airline.java

```

import java.util.Date;
import java.util.List;
public class Airline {
    private List<Flight> flights;
    public Airline(List<Flight> flights) {
        this.flights = flights;
    }
    public List<Flight> getFlights() {
        return flights;
    }

    public void addFlights(Flight flight) {
        flights.add(flight);
    }

    public void printListOfFlights() {
        flights.forEach(System.out::println);
    }
    public void printListOfFlightsForSelectedDestination(String destination) {
        flights.stream().filter(flight ->
            flight.getDestination().equals(destination)).forEach(System.out::println);
    }
    public void printListOfFlightsForSelectedDay(String dayOfWeek) {
        flights.stream().filter(flight ->
            flight.getDaysOfTheWeek().contains(dayOfWeek)).forEach(System.out::println);
    }
    public void printListOfFlightsForSelectedDayAndTime(String dayOfWeek, Date date)
    {
        flights.stream().filter(flight ->
            flight.getDaysOfTheWeek().contains(dayOfWeek))
            .filter(flight ->
                flight.getDepartureTime().before(date)).forEach(System.out::println);
    }
    public void printListOfFlightsForSelectedType(String typeOfAircraft) {
        flights.stream().filter(flight ->
            flight.getTypeOfAircraft().equals(typeOfAircraft)).forEach(System.out::println);
    }
}

```

## Flight.java

```

import java.util.Date;
public class Flight {
    private String destination;

```

```

private String flightNumber;
private String typeOfAircraft;
private Date departureTime;
private String daysOfTheWeek;
public Flight(String destination, String flightNumber, String typeOfAircraft,
              Date departureTime,
              String daysOfTheWeek) {
    this.destination = destination;
    this.flightNumber = flightNumber;
    this.typeOfAircraft = typeOfAircraft;
    this.departureTime = departureTime;
    this.daysOfTheWeek = daysOfTheWeek;
}

public String getDestination() {
    return destination;
}

public String getTypeOfAircraft() {
    return typeOfAircraft;
}

public Date getDepartureTime() {
    return departureTime;
}

public void setDepartureTime(Date departureTime) {
    this.departureTime = departureTime;
}

public String getDaysOfTheWeek() {
    return daysOfTheWeek;
}

public void setDaysOfTheWeek(String daysOfTheWeek) {
    this.daysOfTheWeek = daysOfTheWeek;
}

@Override
public String toString() {
    return "Flight{" +
        "destination=" + destination + "\" +
        ", flightNumber=" + flightNumber + "\" +
        ", typeOfAircraft=" + typeOfAircraft + "\" +
        ", departureTime=" + departureTime +
        ", daysOfTheWeek=" + daysOfTheWeek + "\" +
        '}'";
}
}

```

## Содержимое файла:



fl.txt – Блокнот

Файл Правка Формат Вид Справка

Brest	1	SMALL	12.09.2020	09:55:15	MONDAY, TUESDAY
Minsk	2	AVERAGE	13.10.2020	15:35:10	FRIDAY
Vilnius	3	AVERAGE	25.11.2020	17:12:20	SATURDAY, SUNDAY
Warsaw	4	MAJOR	13.10.2020	15:35:10	THURSDAY, FRIDAY
Moscow	5	MAJOR	15.12.2020	18:11:30	MONDAY, WENSDAY
Tallinn	6	AVERAGE	30.10.2020	07:40:50	TUESDAY
Kiev	7	SMALL	28.09.2020	05:20:10	MONDAY
New York	8	MAJOR	12.10.2020	01:15:00	SUNDAY

## Вывод программы:

```
All flights:
Flight{destination='Brest', flightNumber='1', typeOfAircraft='SMALL', departureTime=Sat Sep 12 09:55:15 MSK 2020, daysOfTheWeek='MONDAY, TUESDAY'}
Flight{destination='Minsk', flightNumber='2', typeOfAircraft='AVERAGE', departureTime=Tue Oct 13 15:35:10 MSK 2020, daysOfTheWeek='FRIDAY'}
Flight{destination='Vilnius', flightNumber='3', typeOfAircraft='AVERAGE', departureTime=Wed Nov 25 17:12:20 MSK 2020, daysOfTheWeek='SATURDAY, SUNDAY'}
Flight{destination='Warsaw ', flightNumber='4', typeOfAircraft='MAJOR', departureTime=Tue Oct 13 15:35:10 MSK 2020, daysOfTheWeek='THURSDAY, FRIDAY'}
Flight{destination='Moscow', flightNumber='5', typeOfAircraft='MAJOR', departureTime=Tue Dec 15 18:11:30 MSK 2020, daysOfTheWeek='MONDAY, WENSDAY'}
Flight{destination='Tallinn', flightNumber='6', typeOfAircraft='AVERAGE', departureTime=Fri Oct 30 07:40:50 MSK 2020, daysOfTheWeek='TUESDAY'}
Flight{destination='Kiev', flightNumber='7', typeOfAircraft='SMALL', departureTime=Mon Sep 28 05:20:10 MSK 2020, daysOfTheWeek='MONDAY'}
Flight{destination='New York', flightNumber='8', typeOfAircraft='MAJOR', departureTime=Mon Oct 12 01:15:00 MSK 2020, daysOfTheWeek='SUNDAY'}

Flights for selected destination:
Flight{destination='New York', flightNumber='8', typeOfAircraft='MAJOR', departureTime=Mon Oct 12 01:15:00 MSK 2020, daysOfTheWeek='SUNDAY'}

Flights for selected day:
Flight{destination='Brest', flightNumber='1', typeOfAircraft='SMALL', departureTime=Sat Sep 12 09:55:15 MSK 2020, daysOfTheWeek='MONDAY, TUESDAY'}
Flight{destination='Tallinn', flightNumber='6', typeOfAircraft='AVERAGE', departureTime=Fri Oct 30 07:40:50 MSK 2020, daysOfTheWeek='TUESDAY'}

Flights for selected day and time:
Flight{destination='Minsk', flightNumber='2', typeOfAircraft='AVERAGE', departureTime=Tue Oct 13 15:35:10 MSK 2020, daysOfTheWeek='FRIDAY'}
Flight{destination='Warsaw ', flightNumber='4', typeOfAircraft='MAJOR', departureTime=Tue Oct 13 15:35:10 MSK 2020, daysOfTheWeek='THURSDAY, FRIDAY'}

Flights for selected type:
Flight{destination='Brest', flightNumber='1', typeOfAircraft='SMALL', departureTime=Sat Sep 12 09:55:15 MSK 2020, daysOfTheWeek='MONDAY, TUESDAY'}
Flight{destination='Kiev', flightNumber='7', typeOfAircraft='SMALL', departureTime=Mon Sep 28 05:20:10 MSK 2020, daysOfTheWeek='MONDAY'}
```

Вывод: научился создавать и использовать классы на языке программирования Java.