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, ИСКУССТВЕННЫЙ ИНТЕЛЛЕКТ И СИСТЕМЫ УПРАВЛЕНИЯ**

КАФЕДРА **КОМПЬЮТЕРНЫЕ СИСТЕМЫ И СЕТИ (ИУ6)**

НАПРАВЛЕНИЕ ПОДГОТОВКИ **09.04.01 Информатика и вычислительная техника**
МАГИСТЕРСКАЯ ПРОГРАММА **09.04.01/07 Интеллектуальные системы анализа, обработки и интерпретации больших данных**

О Т Ч Е Т

по лабораторной работе №9

Название: Лабораторная работа №9

Дисциплина: Языки программирования для работы с большими данными

Студент	ИУ6-22М		М.И. Замула
	(Группа)	(Подпись, дата)	(И.О. Фамилия)
Преподаватель			П.В. Степанов
		(Подпись, дата)	(И.О. Фамилия)

Москва, 2023 г.

Вариант: 1

Использовать ТОЛЬКО методы Stream API. Циклов и условий быть не должно.

Задание 8

Задана коллекция строк. Получить список без дубликатов с сохранением порядка.

Выполнение

```
package lab9;

import java.util.Arrays;
import java.util.List;
import java.util.stream.Collectors;

public class Lab9_1_8 {
    public static void main(String[] args) {
        // Создание списка строк с дубликатами
        List<String> listWithDuplicates = Arrays.asList("apple", "banana",
"pear", "banana", "orange", "apple");

        // Получение списка строк без дубликатов с помощью метода distinct()
        List<String> listWithoutDuplicates = listWithDuplicates.stream()
                .distinct()
                .collect(Collectors.toList());

        // Вывод списка без дубликатов на консоль
        System.out.println(listWithoutDuplicates);
    }
}
```

Результаты

```
[apple, banana, pear, orange]
```

Задание 9

Задана коллекция строк. Вернуть количество вхождений строки.

Выполнение

```
package lab9;

import java.util.Arrays;
import java.util.List;

public class Lab9_1_9 {
    public static void main(String[] args) {
        // Создаем коллекцию строк
        List<String> strings = Arrays.asList("apple", "banana", "pear",
"banana", "orange", "apple");

        // Задаем строку, которую будем искать
        String searchString = "banana";
```

```
        // Фильтруем строки и находим количество вхождений искомой строки
        long count = strings.stream()
                .filter(s -> s.equals(searchString))
                .count();

        // Выводим количество вхождений на экран
        System.out.println(count);
    }
}
```

Результаты: 2

Вариант: 2

Использовать ТОЛЬКО методы Stream API. Циклов и условий быть не должно

Задание 8

Задана коллекция чисел. Получить сумму всех кратных 7

Выполнение

```
package lab9;

import java.util.Arrays;
import java.util.List;

public class Lab9_2_8 {
    public static void main(String[] args) {
        //Мы сначала создали коллекцию numbers с помощью метода
Arrays.asList.
        List<Integer> numbers = Arrays.asList(7, 14, 21, 28, 35, 42, 49, 50);
        // Затем мы использовали stream() для создания потока элементов этой
коллекции.
        int sum = numbers.stream()
                //Метод filter() используется для фильтрации элементов
потока, чтобы оставить только те, которые кратны 7.
                .filter(n -> n % 7 == 0)
                //Метод mapToInt() используется для преобразования каждого
элемента потока в примитивный тип int.
                .mapToInt(Integer::intValue)
                //Наконец, мы вызываем метод sum() для суммирования всех
элементов потока.
                // Результат сохраняется в переменной sum, которая затем
выводится на экран.
                .sum();
        System.out.println("Сумма всех кратных 7: " + sum);
    }
}
```

Результаты

Сумма всех кратных 7: 196

Задание 9

Задана коллекция чисел. С помощью метода reduce вернуть максимум и минимум.

Выполнение

```
package lab9;

import java.util.Arrays;
import java.util.List;

public class Lab9_2_9 {
    public static void main(String[] args) {
        List<Integer> numbers = Arrays.asList(4, 8, 1, 9, 3, 5, 2, 7, 6);

        // Находим максимум и минимум с помощью метода reduce()
        int max = numbers.stream().reduce(Integer.MIN_VALUE, (a, b) -> a > b
? a : b);
        int min = numbers.stream().reduce(Integer.MAX_VALUE, (a, b) -> a < b
? a : b);

        // Выводим результат
        System.out.println("Максимум: " + max);
        System.out.println("Минимум: " + min);
    }
}
```

Результаты

Максимум: 9

Минимум: 1