|  |  |
| --- | --- |
| Gerb-BMSTU_01 | **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 и системы управления**

КАФЕДРА **Компьютерные системы и сети (ИУ6)**

НАПРАВЛЕНИЕ ПОДГОТОВКИ **09.04.01 Информатика и вычислительная техника**

МАГИСТЕРСКАЯ ПРОГРАММА **09.04.01/12 Интеллектуальный анализ больших**

**данных в системах поддержки принятия решений.**

**Отчет**

**По лабораторной работе №9**

**Название:** Stream API

**Дисциплина:** Языки программирования для работы с большими данными

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Студент | ИУ6-23М |  |  | В.Д. Рожнов |
|  | (Группа) |  | (Подпись, дата) | (И.О. Фамилия) |
|  |  |  |  |  |
| Преподаватель |  |  |  | П.В. Степанов |
|  |  |  | (Подпись, дата) | (И.О. Фамилия) |

Москва, 2023

**Задание 1:** Использовать ТОЛЬКО методы Stream API. Циклов и условий быть не должно.

Задана коллекция строк. Вернуть последний элемент и третий элемент коллекции.

Решение задания представлено в листинге 1.

Листинг 1 – Программный код модуля Main.java для задания 1

import java.util.Arrays;

import java.util.List;

public class Main {

public static void main(String[] args) {

List<String> collection = Arrays.asList("A", "B", "C", "D", "E");

String lastElement = collection.stream()

.reduce((first, second) -> second)

.orElse(null);

String thirdElement = collection.stream()

.skip(2)

.findFirst()

.orElse(null);

System.out.println("Последний элемент: " + lastElement);

System.out.println("Третий элемент: " + thirdElement);

}

}

**Задание 2:** Использовать ТОЛЬКО методы Stream API. Циклов и условий быть не должно.

Задана коллекция строк. Вернуть первый элемент коллекции, а также существуют ли все совпадения с шаблоном. Шаблон можно выбрать произвольно.

Решение задания представлено в листинге 2.

Листинг 2 – Программный код модуля Main.java для задания 2

import java.util.Arrays;

import java.util.List;

public class Main {

public static void main(String[] args) {

List<String> collection = Arrays.asList("apple", "banana", "avocado", "blueberry");

String firstElement = collection.stream()

.findFirst()

.orElse(null);

boolean allMatch = collection.stream()

.allMatch(element -> element.matches("^[a-z]+$"));

System.out.println("Первый элемент: " + firstElement);

System.out.println("Все элементы совпадают с шаблоном: " + allMatch);

}

}

**Задание 3:** Использовать ТОЛЬКО методы Stream API. Циклов и условий быть не должно.

Задана коллекция строк. Отсортировать значения по алфавиту и убрать повторы.

Решение задания представлено в листинге 3.

Листинг 3 – Программный код модуля Main.java для задания 3

import java.util.Arrays;

import java.util.List;

public class Main {

public static void main(String[] args) {

List<String> collection = Arrays.asList("banana", "apple", "orange", "apple", "banana", "grape");

List<String> sortedUniqueValues = collection.stream()

.distinct()

.sorted()

.toList();

System.out.println("Отсортированные уникальные значения: " + sortedUniqueValues);

}

}

**Задание 4:** Использовать ТОЛЬКО методы Stream API. Циклов и условий быть не должно.

Задана коллекция:

(Класс People: имя и возраст)  
Collection<People> peoples = Arrays.*asList*(  
 **new** People(**"Ivan"**, 16),  
 **new** People(**"Petr"**, 23),  
 **new** People(**"Maria"**, 42)  
);

Отсортировать по имени в обратном алфавитном порядке.

Решение задания представлено в листинге 4.

Листинг 4 – Программный код модуля Main.java для задания 4

import java.util.Arrays;

import java.util.Collection;

import java.util.Comparator;

class People {

private String name;

private int age;

public People(String name, int age) {

this.name = name;

this.age = age;

}

public String getName() {

return name;

}

public int getAge() {

return age;

}

@Override

public String toString() {

return "People{" +

"name='" + name + '\'' +

", age=" + age +

'}';

}

}

public class Main {

public static void main(String[] args) {

Collection<People> peoples = Arrays.asList(

new People("Ivan", 16),

new People("Petr", 23),

new People("Maria", 42)

);

peoples.stream()

.sorted(Comparator.comparing(People::getName).reversed())

.forEach(System.out::println);

}

}

**Вывод:** в ходе выполнения данной лабораторной работы были изучены и применены на практике приемы использования Stream API на языке Java.