**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 **09.04.01/07 Интеллектуальные системы анализа, обработки и интерпретации больших данных**

О Т Ч Е Т

|  |  |
| --- | --- |
| **по лабораторной работе № 9** |  |

**Дисциплина:** Stream API

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

|  |  |  |  |
| --- | --- | --- | --- |
| Студент | ИУ6-22М |  | Д.М. Карасёв\_\_ |
|  | (Группа) | (Подпись, дата) | (И.О. Фамилия) |
| Преподаватель |  | П.В. Степанов | |
|  |  | (Подпись, дата) | (И.О. Фамилия) |

Москва, 2023

**Цель:** ознакомиться с базовыми принципами языка Java для работы с большими данными

**Вариант 1:**

1. Коллекция из 10 задания. Вернуть средний балл.
2. Задана коллекция строк. Вернуть первые два элемента.

**Решение:**

package lr9\_v1\_task11;  
  
import java.util.Arrays;  
import java.util.Collection;  
import java.util.OptionalDouble;  
  
public class Main {  
 public static void main(String[] args) {  
 Collection<Student> students = Arrays.asList(  
 new Student("Ivan", 40),  
 new Student("Petr", 60),  
 new Student("Olga", 70)  
 );  
 OptionalDouble averageScore = students.stream().mapToInt(Student::getScore).average();  
  
 System.out.println(averageScore.getAsDouble());  
  
  
 }  
}

package lr9\_v1\_task11;  
  
public class Student {  
 private String name;  
 private int score;  
 public Student(String name, int score){  
 this.name = name;  
 this.score = score;  
 }  
  
 public int getScore() {  
 return score;  
 }  
  
 public String getName() {  
 return name;  
 }  
  
 public void setName(String name) {  
 this.name = name;  
 }  
  
 public void setScore(int score) {  
 this.score = score;  
 }  
}

На рисунке 1 представлен результат выполнения.



Рисунок 1 – Результат выполнения

package lr9\_v1\_task12;  
  
import java.util.Arrays;  
import java.util.Collection;  
  
public class Main {  
 public static void main(String[] args) {  
 Collection<String> strings = Arrays.asList(  
 "dfadsfsdf",  
 "zxvxcvxc",  
 "treteg",  
 "kygukuty54"  
 );  
  
 strings.stream().limit(2).forEach(x -> System.out.println(x));  
 }  
}

На рисунке 2 представлен результат выполнения.



Рисунок 2 – Результат выполнения

**Вариант 2:**

1. Задана коллекция:

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

Найти самого старшего человека мужского пола.

1. Коллекция из 11 задания. Найти самый минимальный возраст человека, у которого есть буква “e” в имени.

**Решение:**

package lr9\_v2\_task11;  
  
public enum Gender {  
 *MAN*,  
 *WOMAN*}

package lr9\_v2\_task11;  
  
import java.util.Arrays;  
import java.util.Collection;  
  
public class Main {  
 public static void main(String[] args) {  
 Collection<People> peoples = Arrays.*asList*(  
 new People("Ivan", 16, Gender.MAN),  
 new People("Petr", 23, Gender.MAN),  
 new People("Maria", 42, Gender.WOMAN)  
 );  
  
 int maxAgeMan = peoples.stream().filter(x -> x.getGender() == Gender.MAN).mapToInt(People::getAge).max().getAsInt();  
  
 peoples.stream().filter(x -> x.getAge() == maxAgeMan).forEach(x ->  
 System.out.println("Самый старший мужчина: " + x.getName() + ", возраст: " + x.getAge()));  
  
 }  
}

package lr9\_v2\_task11;  
  
public class People {  
 private String name;  
 private int age;  
 private Gender gender;  
  
 public People(String name, int age, Gender gender){  
 this.name = name;  
 this.age = age;  
 this.gender = gender;  
 }  
  
 public String getName() {  
 return name;  
 }  
  
 public Gender getGender() {  
 return gender;  
 }  
  
 public int getAge() {  
 return age;  
 }  
  
 public void setName(String name) {  
 this.name = name;  
 }  
  
 public void setAge(int age) {  
 this.age = age;  
 }  
  
 public void setGender(Gender gender) {  
 this.gender = gender;  
 }  
}

На рисунке 3 представлен результат выполнения.



Рисунок 3 – Результат выполнения

package lr9\_v2\_task12;  
  
public enum Gender {  
 *MAN*,  
 *WOMAN*}

package lr9\_v2\_task12;  
  
import java.util.Arrays;  
import java.util.Collection;  
  
public class Main {  
 public static void main(String[] args) {  
 Collection<People> peoples = Arrays.*asList*(  
 new People("Ivane", 16, Gender.*MAN*),  
 new People("Petr", 23, Gender.*MAN*),  
 new People("Maria", 42, Gender.*WOMAN*)  
 );  
  
 int minAge = peoples.stream().filter(x -> x.getName().contains("e")).mapToInt(People::getAge).min().getAsInt();  
  
 peoples.stream().filter(x -> (x.getName().contains("e") && x.getAge() == minAge)).forEach(x -> System.*out*.println(  
 "Имя человка с минимальны возрастом и буквой \'e\' в имени: " + x.getName() + ", возраст: " + x.getAge()  
 ));  
 }  
}

package lr9\_v2\_task12;  
  
public class People {  
  
 private String name;  
 private int age;  
 private Gender gender;  
  
 public People(String name, int age, Gender gender){  
 this.name = name;  
 this.age = age;  
 this.gender = gender;  
 }  
  
 public String getName() {  
 return name;  
 }  
  
 public Gender getGender() {  
 return gender;  
 }  
  
 public int getAge() {  
 return age;  
 }  
  
 public void setName(String name) {  
 this.name = name;  
 }  
  
 public void setAge(int age) {  
 this.age = age;  
 }  
  
 public void setGender(Gender gender) {  
 this.gender = gender;  
 }  
}

На рисунке 4 представлен результат выполнения.



Рисунок 4 – Результат выполнения

**Вывод:** были разработаны методы и классы согласно варианту