МИНИСТЕРСТВО ЦИФРОВОГО РАЗВИТИЯ, СВЯЗИ И МАССОВЫХ

КОММУНИКАЦИЙ РОССИЙСКОЙ ФЕДЕРАЦИИ

Ордена Трудового Красного Знамени федеральное государственное бюджетное образовательное учреждение высшего образования   «Московский технический университет связи и информатики»



Кафедра"Системное программирование"

Лабораторная работа № 8

По дисциплине

“Информационные технологии и программирование”

                                      Выполнил: студент гр. БВТ2201

Новожилов Д.П.

Москва, 2023

Создаем аннотацию:

import java.lang.annotation.Retention;  
import java.lang.annotation.RetentionPolicy;  
  
@Retention(RetentionPolicy.RUNTIME)  
public @interface DataProcessor {  
}

Создаем класс Менеджер:

import java.io.\*;  
import java.util.\*;  
import java.util.concurrent.\*;  
import java.lang.reflect.Method;  
  
public class DataManager {  
 private LinkedList<Object> processors;  
 private ExecutorService service;  
 private HashMap<String, Integer> data;  
 private StringBuilder outputData;  
 public DataManager() {  
 processors = new LinkedList<>();  
 service = Executors.newFixedThreadPool(10);  
 data = new HashMap<>();  
 outputData = new StringBuilder();  
 }  
  
 public void registerDataProcessor(Object processor) {  
 processors.add(processor);  
 }  
  
 public void loadData(String source) {  
 File file = new File(source);  
 try (FileReader reader = new FileReader(file)) {  
 BufferedReader bufferedReader = new BufferedReader(reader);  
 String line = bufferedReader.readLine();  
  
 while (line != null) {  
 String[] dataSet = line.split(" ");  
 data.put(dataSet[0], Integer.parseInt(dataSet[1]));  
 line = bufferedReader.readLine();  
 }  
 } catch (IOException e) {  
 System.out.println(e.getMessage());  
 }  
 }  
  
 public void processData() {  
 ArrayList<Future<String>> futures = new ArrayList<>();  
 ArrayList<Integer> indexes = new ArrayList<>();  
 int i = 0;  
 for (Object processor : processors) {  
 for (Method method : processor.getClass().getMethods()) {  
 if (method.getAnnotation(DataProcessor.class) != null) {  
 indexes.add(0);  
 futures.add(service.submit(() -> {  
 try {  
 return (String) method.invoke(processor, data);  
 } catch (Exception e) {  
 System.out.println(e.getMessage());  
 return "";  
 }  
 }));  
 indexes.set(i, futures.size());  
 i++;  
 }  
 }  
 }  
 int j = 0;  
 for (Future<String> f : futures) {  
 try {  
 outputData.append(f.get());  
 j++;  
 for (int index: indexes) {  
 if (j == index) {  
 outputData.append("\n");  
 }  
 }  
 } catch (Exception e) {  
 System.out.println(e.getMessage());  
 }  
 }  
 service.shutdown();  
 }  
  
 public void saveData(String outputFile) {  
 File file = new File(outputFile);  
  
 try (FileWriter writer = new FileWriter(file)) {  
 writer.write(outputData.toString());  
 } catch (IOException e) {  
 System.out.println(e.getMessage());  
 }  
 }  
}

Создаем несколько видом различных обработчичков:

import java.util.HashMap;  
import java.util.List;  
import java.util.Map;  
  
public class MoreThanTen {  
 @DataProcessor  
 public String moreThanTen(HashMap<String, Integer> data) {  
 List<Map.Entry<String, Integer>> list = data.entrySet().stream().filter((x) -> x.getValue() > 10).toList();  
 StringBuilder output = new StringBuilder();  
 for (Map.Entry<String, Integer> entry : list) {  
 output.append(entry.getKey());  
 output.append(" ");  
 output.append(entry.getValue().toString());  
 output.append("\n");  
 }  
 return output.toString();  
 }  
}

import java.util.HashMap;  
import java.util.List;  
import java.util.Map;  
  
public class NameSort {  
 @DataProcessor  
 public String nameSort(HashMap<String, Integer> data) {  
 List<Map.Entry<String, Integer>> list = data.entrySet().stream().sorted((x, y) -> {  
 return x.getKey().compareTo(y.getKey());  
 }).toList();  
 StringBuilder output = new StringBuilder();  
 for (Map.Entry<String, Integer> entry : list) {  
 output.append(entry.getKey());  
 output.append(" ");  
 output.append(entry.getValue().toString());  
 output.append("\n");  
 }  
 return output.toString();  
 }  
}

import java.util.HashMap;  
import java.util.List;  
import java.util.Map;  
  
public class ValueSort {  
 @DataProcessor  
 public String valueSort(HashMap<String, Integer> data) {  
 List<Map.Entry<String, Integer>> list = data.entrySet().stream().sorted((x, y) -> {  
 return x.getValue().compareTo(y.getValue());  
 }).toList();  
 StringBuilder output = new StringBuilder();  
 for (Map.Entry<String, Integer> entry : list) {  
 output.append(entry.getKey());  
 output.append(" ");  
 output.append(entry.getValue().toString());  
 output.append("\n");  
 }  
 return output.toString();  
 }  
}

Запускаем всю программу в мейн файле:

public class Main {  
 public static void main(String[] args) {  
 DataManager manager = new DataManager();  
  
 NameSort processor1 = new NameSort();  
 ValueSort processor2 = new ValueSort();  
 MoreThanTen processor3 = new MoreThanTen();  
  
 manager.registerDataProcessor(processor1);  
 manager.registerDataProcessor(processor2);  
 manager.registerDataProcessor(processor3);  
  
 manager.loadData("fruitValues.txt");  
 manager.processData();  
 manager.saveData("newData.txt");  
 }  
}

Было:

Apple 12  
Orange 3  
Grape 5  
Banana 14  
Pear 7  
Melon 5

Стало:

Apple 12  
Banana 14  
Grape 5  
Melon 5  
Orange 3  
Pear 7  
  
Orange 3  
Grape 5  
Melon 5  
Pear 7  
Apple 12  
Banana 14  
  
Apple 12  
Banana 14

Вывод: Проделав данную работу мы научились обрабатывать данные параллельно с помощью многопоточности, а также довольно быстро за счет стрим апи

Ссылка на репозиторий: https://github.com/Nazhik123/inf\_labs/tree/master/laba8