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 «Математической кибернетики и информационных технологий»

Информационные технологии и программирование

Лабораторная работа №8

Выполнил: студент группы

БВТ2306

Кесслер Алексей Сергеевич

Москва, 2024 г.

Цель работы: Изучение основ использования аннотаций.

Задача работы: Реализовать свою коллекцию, использовать коллекции при выполнении заданий.

Выполнение

Для начала создаем аннотацию @DataProcessor.

```java
import java.lang.annotation.Retention;
import java.lang.annotation.RetentionPolicy;

4 usages
@Retention(RetentionPolicy.RUNTIME)
public @interface DataProcessor {
}
```

Рисунок 1 - Аннотация DataProcessor

После реализуем три разных класса, представляющие различные обработчики данных:

```java
import java.util.HashMap;
import java.util.List;
import java.util.Map;

2 usages
public class NameSort {
    no usages
    @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();
    }
}
```

Рисунок 2 - Класс NameSort

```java
import java.util.HashMap;
import java.util.List;
import java.util.Map;

2 usages
public class ValueSort {
    no usages
    @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();
    }
}
```

Рисунок 3 - Класс ValueSort

```java
import java.util.HashMap;
import java.util.List;
import java.util.Map;

2 usages
public class MoreThanTen {
    no usages
    @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();
    }
}
```

Рисунок 4 - Класс MoreThanTen

Класс DataManager:

```
import java.io.*;
import java.util.*;
import java.util.concurrent.*;
import java.lang.reflect.Method;

2 usages
public class DataManager {
    3 usages
    private LinkedList<Object> processors;
    3 usages
    private ExecutorService service;
    3 usages
    private HashMap<String, Integer> data;
    4 usages
    private StringBuilder outputData;
    1 usage
    public DataManager() {
        processors = new LinkedList<>();
        service = Executors.newFixedThreadPool( nThreads: 10);
        data = new HashMap<>();
        outputData = new StringBuilder();
    }
```

Рисунок 5 - Поля и конструктор DataManager

Метод registerDataProcessor(Object processor) добавляет в DataManager новую задачу.

```
public void registerDataProcessor(Object processor) {
    processors.add(processor);
}
```

Рисунок 6 - Метод registerDataProcessor

Метод loadData(String source) получает путь к исходному файлу и считывает его содержимое.

```java
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( regex: " ");
            data.put(dataSet[0], Integer.parseInt(dataSet[1]));
            line = bufferedReader.readLine();
        }
    } catch (IOException e) {
        System.out.println(e.getMessage());
    }
}
```

Рисунок 7 - Метод loadData

Метод processData() выполняет все поставленные задачи, которые помечены аннотацией DataProcessor.

```java
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();
}
```

Рисунок 8 - Метод processData

Метод saveData(String outputFile) выводит результат работы менеджера в отдельный файл.

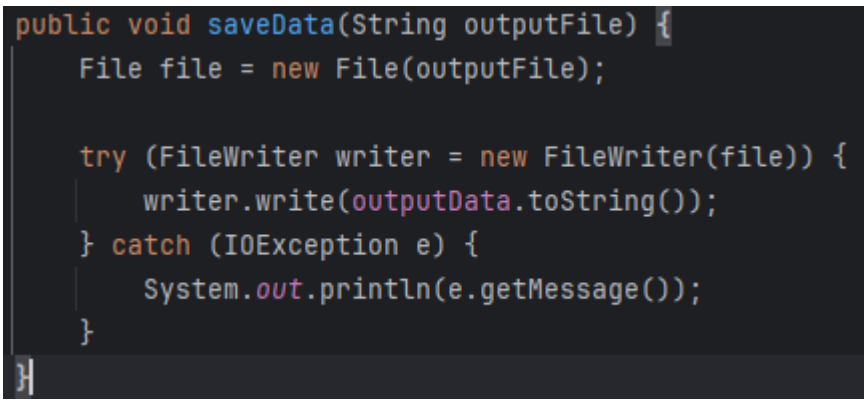

Рисунок 9 - Метод saveData