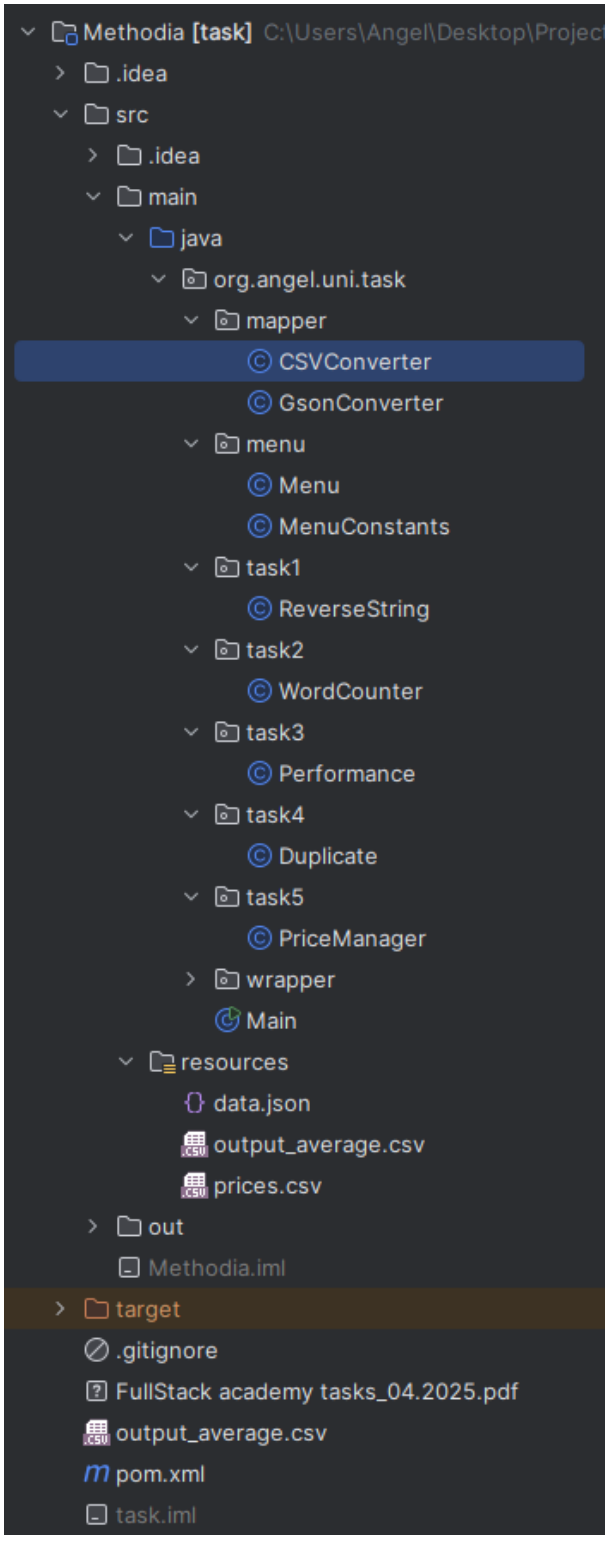
Съдържание

Документация по кода

Файлова структура



(фиг. 1)

Всяка задача е разделена в отделен package. Имам изглед в конзолата (Menu). В mapper package, имаме класове, които четат CSV и съответно JSON. Първият го ползваме в задачите, които изискват съхраняването и четенето от Excel. Вторият съдържа примерни низове (думи, специални знаци и емоджита).

Задача 1:

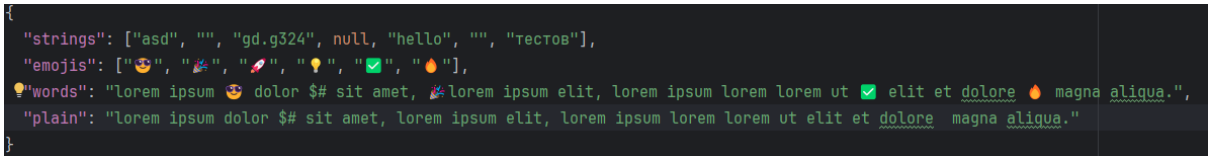
```
public class ReverseString {  2 usages  Angel L Stoynov +1 *

    private static final DataWrapper dataWrapper = GsonConverter.getContentFromJson();
    private static final String EXAMPLE = dataWrapper.getWords();  5 usages

    public static void reverseString() {  1 usage  Angel L Stoynov +1 *
        if (EXAMPLE == null || EXAMPLE.isEmpty()) {
            System.out.println("String cannot be null or empty. Provided: " + EXAMPLE);
            return;
        }

        String reversed = "";
        int[] codePoints = EXAMPLE.codePoints().toArray();
        for (int i = codePoints.length - 1; i >= 0; i--) {
            reversed += new String(Character.toChars(codePoints[i]));
        }
        System.out.println(EXAMPLE);
        System.out.println(reversed);
    }
}
```

(фиг. 2)

DataWrapper е помощен клас, който се използва да съхранява по горе описаните низове. EXAMPLE съдържа стойността на getWords().

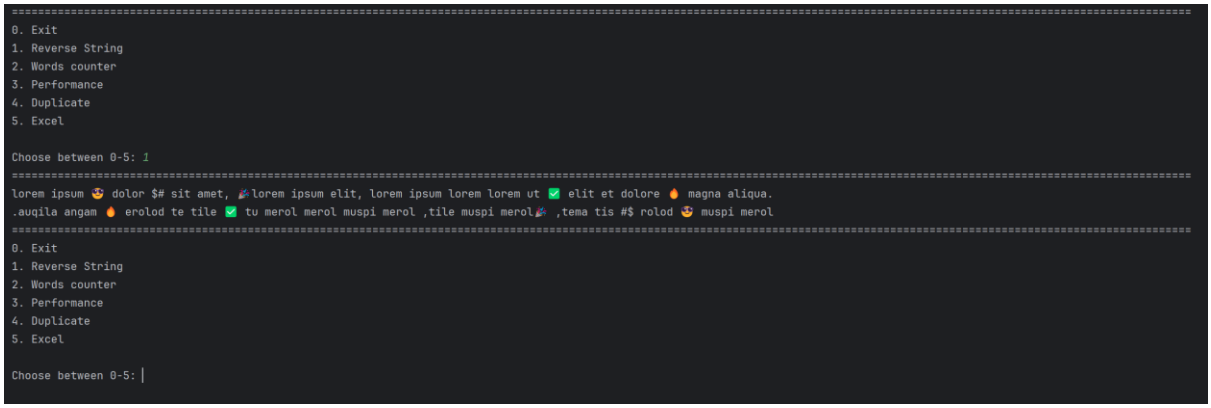
```
public class DataWrapper {  11 usages  Angel Stoynov +1 *
    private List<String> strings;  1 usage
    private List<String> emojis;  1 usage
    private String words;  1 usage
    private String plain;  1 usage

    public List<String> getStrings() { return strings; }

    public List<String> getEmojis() { return emojis; }

    public String getWords() { return words; }

    public String getPlain() {  no usages  new *
        return plain;
    }
}
```

(фиг. 3)



(фиг. 4)

В първа задача правим проверка дали не се съдържа стойност == null или ако низът е празен. Използваме codePoints, заради емоджитата. Понеже не може да се ползва StringBuilder/StringBuilder съм използва for loop, който итерира в обратен ред. Използва конкатенация, за да добави всяка следваща дума.



(фиг. 5)

От менюто избираме опция 1. Първо се визуализира низа, който искаме да обърнем и съответно резултата.

(фиг. 6)

При null или empty се визуализира това.

Задача 2:

```java
public class WordCounter {  2 usages  Angel Stoynov *

    private static final String REGEX_PATTERN = "[a-zA-Z.,!?]+";  1 usage
    private static final String REGEX_SPLIT = "[\\s+]";  1 usage
    private static final Map<String, Integer> wordCounterMap = new HashMap<>();;  4 usages

    public static void wordCounterWithMap() {  1 usage  Angel Stoynov *
        DataWrapper wrapper = GsonConverter.getContentFromJson();
        String sentence = wrapper.getPlain();
        System.out.println(sentence);

        if (sentence == null || sentence.isEmpty()) {
            System.out.println("Sentence cannot be null or empty. Provided: " + sentence);
            return;
        }

        List<String> words =  new ArrayList<>(Arrays.asList(sentence.split(REGEX_SPLIT)));
        Iterator<String> iterator = words.iterator();
        while (iterator.hasNext()) {
            String word = iterator.next();
            if (!word.matches(REGEX_PATTERN)) {
                iterator.remove();
            }
            if (wordCounterMap.containsKey(word)) {
                wordCounterMap.merge(word,  value: 1, Integer::sum);
            } else {
                wordCounterMap.put(word, 1);
            }
        }

        List<Map.Entry<String, Integer>> sortedEntries = new ArrayList<>(wordCounterMap.entrySet());
        sortedEntries.sort((entry1, entry2) -> {
            int countCompare = entry2.getValue().compareTo(entry1.getValue());
            if (countCompare != 0) {
                return countCompare;
            } else {
                return entry1.getKey().compareTo(entry2.getKey());
            }
        });

        for (Map.Entry<String, Integer> entry : sortedEntries) {
            System.out.println(entry.getKey() + " : " + entry.getValue());
        }
    }
}
```

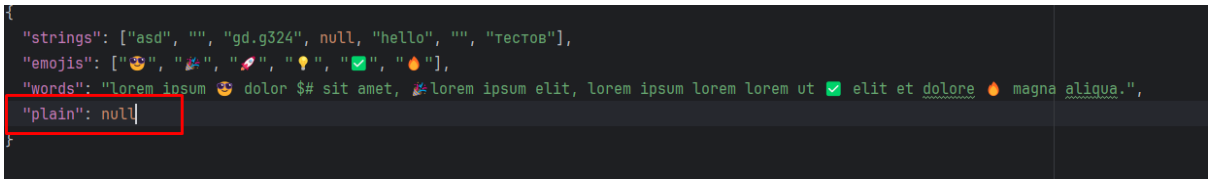
Font size: 15p

(фиг. 7)

REGEX_PATTERN се използва както е по условие. REGEX_SPLIT съм го отделил, за да не директно в .split(). Проверка дали е null или empty. Ползвал съм iterator, за да мога да премахвам, докато итерирам. Всяка дума, която не отговаря на regex се премахва. Ако hashMap съдържа тази дума само я инкрементираме. Иначе добавяне нови думи в map-а. По-късно сортираме първо по повтаряния и по азбука.

```
Choose between 0-5: 2
========================================================================================================================
lorem ipsum dolor $# sit amet, lorem ipsum elit, lorem ipsum lorem lorem ut elit et dolore magna aliqua.
lorem : 5
ipsum : 3
$# : 1
aliqua. : 1
amet, : 1
dolor : 1
dolore : 1
elit : 1
elit, : 1
et : 1
magna : 1
sit : 1
ut : 1
========================================================================================================================
0. Exit
1. Reverse String
2. Words counter
3. Performance
4. Duplicate
5. Excel

Choose between 0-5:
```

(фиг. 8)

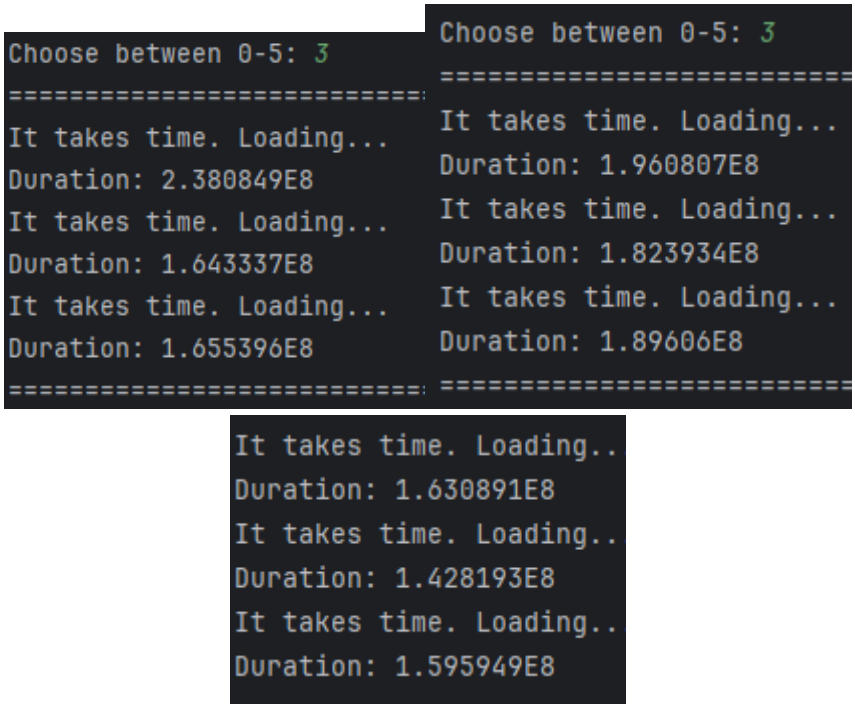


(фиг. 9)

```
Choose between 0-5: 2
======================================================
null
Sentence cannot be null or empty. Provided: null
======================================================
```

(фиг. 10)

Задача 3:



(фиг. 11-13)

Примерни резултати за for, while, iterator в тази последователност.

Задача 4:

```java
public class Duplicate {  2 usages  Angel L Stoynov *

    private static final int SPACE_ASCII_CODE = 32;  1 usage
    private static final DataWrapper dataWrapper = GsonConverter.getContentFromJson();  1 usage
    private static final String EXAMPLE = dataWrapper.getWords();  3 usages

    public static void findDuplicateCharacters() {  1 usage  Angel L Stoynov *
        if (EXAMPLE == null || EXAMPLE.isEmpty()) {
            System.out.println("Input is null or empty.");
            return;
        }

        Map<Integer, Integer> frequencyMap = new HashMap<>();

        for (int codePoint : EXAMPLE.codePoints().toArray()) {
            frequencyMap.put(codePoint, frequencyMap.getOrDefault(codePoint,  defaultValue: 0) + 1);
        }

        System.out.println("Duplicate characters:");
        for (Map.Entry<Integer, Integer> entry : frequencyMap.entrySet()) {
            if (entry.getKey() != SPACE_ASCII_CODE && entry.getValue() > 1) {
                String character = new String(Character.toChars(entry.getKey()));
                System.out.println("'" + character + "' -> " + entry.getValue() + " times");
            }
        }
    }
}
```

(фиг. 14)

Тук съм добавил и ASCII кода на space – „ “, за да не брои колко шпации има.

```
Choose between 0-5: 4
================================================================================
Duplicate characters:
'a' -> 5 times
'd' -> 2 times
'e' -> 10 times
'i' -> 7 times
'l' -> 10 times
',' -> 2 times
'm' -> 10 times
'o' -> 9 times
'p' -> 3 times
'r' -> 7 times
's' -> 4 times
't' -> 6 times
'u' -> 5 times
================================================================================
```

(фиг. 15)

Ако символът се повтаря, повече от веднъж се принтира.

Задача 5:

```java
public class PriceManager {  2 usages  Angel L Stoynov *

    public static void saveAveragePriceToCSV() {  1 usage  Angel L Stoynov *
        List<String[]> rows = CSVConverter.getContentFromCSV();
        List<Integer> prices = new ArrayList<>();
        double sum = 0;
        int price;

        try {
            for (int i = 1; i < rows.size(); i++) {
                price = Integer.parseInt(rows.get(i)[0]);
                prices.add(price);
                sum += price;
            }
        } catch (NumberFormatException e) {
            System.out.println("String cannot be converted to integer. " + e.getMessage());
            return;
        }
        System.out.println(prices);
        CSVConverter.writeContentToCSV( average: sum / prices.size());
    }
}
```

(фиг. 16)

```java
public class CSVConverter {  3 usages  Angel L Stoynov

    private static final String PATH = "prices.csv";  2 usages
    private static final String OUTPUT_PATH = "output_average.csv";  1 usage

    public static List<String[]> getContentFromCSV() {  1 usage  Angel L Stoynov
        try (InputStream inputStream = GsonConverter.class.getClassLoader().getResourceAsStream(PATH)) {
            if (inputStream == null) {
                System.out.println("File not found: " + PATH);
                return new ArrayList<>();
            }

            try (CSVReader csvReader = new CSVReader(new InputStreamReader(inputStream))) {
                return csvReader.readAll();
            } catch (CsvException e) {
                System.out.println("Unexpected error happened in CsvReader: " + e.getMessage());
                return new ArrayList<>();
            }
        } catch (IOException e) {
            System.out.println("Unexpected error occurred: " + e.getMessage());
            return new ArrayList<>();
        }
    }

    public static void writeContentToCSV(double average) {  1 usage  Angel L Stoynov
        File file = new File(OUTPUT_PATH);

        try (FileWriter writer = new FileWriter(file);
             CSVWriter csvWriter = new CSVWriter(writer)) {

            String[] headers = {"average_price"};
            csvWriter.writeNext(headers);

            csvWriter.writeNext(new String[]{String.valueOf(average)});

        } catch (IOException e) {
            System.out.println("Unexpected error occurred: " + e.getMessage());
        } catch (NumberFormatException e) {
            System.out.println("Number format exception occurred when reading the average price: " + e.getMessage());
        }
    }
}
```

(фиг. 17)

Тези файлове се съхраняват в resources. При всяка грешка връщаме празен масив. Като записваме във файла сме означили header – average_price.

```
prices
100
200
6
300
5
400
12
33
```

```
"average_price"
"132.0"
```

(фиг. 18-19)