Methodia FullStack Academy 2025

Task 1

```
package Task1;

public class ReverseString {
    public static void main(String[] args) {
        String originalString = "This is a test string";
        char[] chars = originalString.toCharArray();

        int leftSide = 0;
        int rightSide = chars.length - 1;

        while (leftSide < rightSide) {
            // Размяна
            char temp = chars[leftSide];
            chars[leftSide] = chars[rightSide];
            chars[rightSide] = temp;

            leftSide++;
            rightSide--;
        }

        String reversedString = new String(chars);
        System.out.println("Original string: " + originalString);
        System.out.println("Reversed string: " + reversedString);
    }
}
```

Task 2

```
package Task2;

import java.util.*;

public class SortedDictionary {
    public static void main(String[] args) {
        String text = "This is a test. This TEST is only a Test! Is this a test? Yes, this is a TesT.";

        text = text.toLowerCase().replaceAll("[.,!?]", "");

        String[] words = text.split("\\s+");
        Map<String, Integer> wordCounts = new HashMap<>();
        for (String word : words) {
            wordCounts.put(word, wordCounts.getOrDefault(word, 0) + 1);
        }

        List<Map.Entry<String, Integer>> sortedList = new
ArrayList<>(wordCounts.entrySet());

        sortedList.sort((a, b) -> {
            int compare = b.getValue().compareTo(a.getValue());
            if (compare != 0)
            {
                return compare;
            }

            return a.getKey().compareTo(b.getKey());
        });

        Map<String, Integer> sortedMap = new HashMap<>();
        for (Map.Entry<String, Integer> entry : sortedList) {
            sortedMap.put(entry.getKey(), entry.getValue());
        }

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

Task 3

```
package Task3;

import java.util.*;

public class ArrayListTraversalMilliSeconds {
    public static void main(String[] args) {
        List<String> list = new ArrayList<>();
        for (int i = 0; i < 1000000; i++) {
            list.add("Element " + i);
        }
        long startTime = System.nanoTime();
        for (int i = 0; i < list.size(); i++) {
            String element = list.get(i);
        }
        long endTime = System.nanoTime();
        System.out.println("Time for for loop: " + (endTime - startTime) / 1000000 + " ms");

        startTime = System.nanoTime();
        int index = 0;
        while (index < list.size()) {
            String element = list.get(index);
            index++;
        }
        endTime = System.nanoTime();
        System.out.println("Time for while loop: " + (endTime - startTime) / 1000000 + "
ms");

        startTime = System.nanoTime();
        Iterator<String> iterator = list.iterator();
        while (iterator.hasNext()) {
            String element = iterator.next();
        }
        endTime = System.nanoTime();
        System.out.println("Time for Iterator: " + (endTime - startTime) / 1000000 + " ms");
    }
}
```

Task 4

```
package Task4;

import java.util.*;

public class DuplicateCharacters {
    public static void main(String[] args) {
        String input = "Methodia FullStack Academy";

        input = input.toLowerCase().replaceAll("[.,!? ]", "");

        Map<Character, Integer> charCount = new HashMap<>();

        for (char c : input.toCharArray()) {
            charCount.put(c, charCount.getOrDefault(c, 0) + 1);
        }
        System.out.println("Duplicate characters:");
        for (Map.Entry<Character, Integer> entry : charCount.entrySet()) {
            if (entry.getValue() > 1) {
                System.out.println(entry.getKey() + " → " + entry.getValue() + " times");
            }
        }
    }
}
```

Task 5

```
package Task5;

import org.apache.poi.ss.usermodel.*;
import org.apache.poi.xssf.usermodel.XSSFWorkbook;

import java.io.*;
import java.util.*;

public class ExcelProcessor {
    public static void main(String[] args) {
        String inputFile = "products.xlsx";
        String outputFile = "filtered_products.xlsx";

        double totalPrice = 0;
        int count = 0;

        List<Row> filteredRows = new ArrayList<>();

        try (FileInputStream fis = new FileInputStream(inputFile);
             Workbook workbook = new XSSFWorkbook(fis)) {

            Sheet sheet = workbook.getSheetAt(0);
            int priceColumnIndex = 2;

            Iterator<Row> rowIterator = sheet.iterator();
            Row header = rowIterator.next(); // заглавен ред

            while (rowIterator.hasNext()) {
                Row row = rowIterator.next();
                Cell priceCell = row.getCell(priceColumnIndex);

                if (priceCell != null && priceCell.getCellType() == CellType.NUMERIC) {
                    double price = priceCell.getNumericCellValue();
```

```
          if (price > 100) {
             filteredRows.add(row);
             totalPrice += price;
             count++;
          }
       }
    }

    double average = count > 0 ? totalPrice / count : 0;
    System.out.printf("Average price of filtered items: %.2f%n", average);

    Workbook newWorkbook = new XSSFWorkbook();
    Sheet newSheet = newWorkbook.createSheet("Filtered");

    Row newHeader = newSheet.createRow(0);
    for (int i = 0; i < header.getLastCellNum(); i++) {
       Cell cell = newHeader.createCell(i);
       cell.setCellValue(header.getCell(i).getStringCellValue());
    }

    int rowIndex = 1;
    for (Row originalRow : filteredRows) {
       Row newRow = newSheet.createRow(rowIndex++);
       for (int i = 0; i < originalRow.getLastCellNum(); i++) {
          Cell oldCell = originalRow.getCell(i);
          if (oldCell != null) {
             Cell newCell = newRow.createCell(i);
             switch (oldCell.getCellType()) {
                case STRING -> newCell.setCellValue(oldCell.getStringCellValue());
                case NUMERIC -> newCell.setCellValue(oldCell.getNumericCellValue());
             }
          }
       }
    }

    Row summaryRow = newSheet.createRow(rowIndex);
    summaryRow.createCell(0).setCellValue("Average price:");
    summaryRow.createCell(1).setCellValue(average);

    try (FileOutputStream fos = new FileOutputStream(outputFile)) {
       newWorkbook.write(fos);
    }

    newWorkbook.close();
    System.out.println("Filtered data written to: " + outputFile);

 }
```

```
catch (IOException e) {
          e.printStackTrace();
       }
    }
}
```