Practical task #3

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

ATTENTION!

(1) the names of the classes and methods are specified;

(2) the names of the files with input data are specified;

(3) In case it’s not specifically indicated, input data can contain Cyrillic letters (letters of the Russian and Ukrainian alphabets) wherever;

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

Note.

If the application reads data from a file, it’s necessary to specify the same encoding using which the data is stored.

Use UTF-8/Cp1251 encoding, refer to the example ‘How to get the input data’ at the end of the text.

Each ‘PartX’ class should have ‘main’ method that demonstrates the functionality of the corresponding subtask.

In the root package create Demo class that demonstrates the actions of all the written functionality.

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Task 1

——————————————————————————

Class name: com.epam.rd.java.basic.practice3.Part1

The input data should be uploaded from the "part1.txt" file

——————————————————————————

The task should be resolved using regular expressions without using container classes.

Define a class with static methods that convert input data to output data.

As the input data, use a text of the following structure (the values of Login/Name/Email, in fact, can be random); Login and Name can contain both Cyrillic and Latin letters):

Input data (part1.txt)

——————————————————

Login;Name;Email

ivanov;Ivan Ivanov;ivanov@mail.com

петров;Петр Петров;petrov@google.com

obama;Barack Obama;obama@google.com

bush;Джордж Буш;bush@mail.com

——————————————————

The methods that you need to write have the following structure (N is a digit: 1, 2, 3, 4):

————————————————————————————

public static String convertN(String input) {

    ...

}

———————————————————————————

1.1. The method "convert1"

It should convert input data to a string of the following type:

Output of convert1

——————————————————

ivanov: ivanov@mail.com

петров: petrov@google.com

obama: obama@google.com

bush: bush@mail.com

——————————————————

1.2. The method "convert2"

It should convert input data to a string of the following type:

Output of convert2

——————————————————

Ivanov Ivan (email: ivanov@mail.com)

Петров Петр (email: petrov@google.com)

Obama Barack (email: obama@google.com)

Буш Джордж (email: bush@mail.com)

——————————————————

1.3. The method "convert3"

It should convert input data to a string of the following type (email domain ===> a list of the logins separated with a comma of those users whose emails are registered based on this domain):

Output of convert3

——————————————————

mail.com ==> ivanov, bush

google.com ==> петров, obama

——————————————————

1.4. The method "convert4"

It should convert input data to a string of the following type (Password column should be added, the password itself should contain exactly 4 digits that are generated randomly):

Output of convert4

——————————————————

Login;Name;Email;Password

ivanov;Ivan Ivanov;ivanov@mail.com;1707

петров;Петр Петров;petrov@google.com;9321

obama;Barack Obama;obama@google.com;4623

bush;Джордж Буш;bush@mail.com;7514

——————————————————

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Task 2

——————————————————————————

Class name: com.epam.rd.java.basic.practice3.Part2

The input data should be uploaded from the "part2.txt" file

——————————————————————————

The task should be resolved using regular expressions without using container classes.

Input: text (it can contain both Cyrillic and Latin letters).

Output: words of the minimum length and the maximum length in the format specified below (in the singular, minding the order of their occurrence in the text and considering the character case).

A "word" should be considered a sequence of characters containing letters only.

Create a static method "convert" that converts the input to the output.

Stub of the method:

——————————————————

public static String convert(String input) {

    ...

}

——————————————————

Example of the Input data (part2.txt)

——————————————————

When I was younger, so much younger than today

I never needed anybody's help in any way

But now these days are gone, I'm not so self-assured

Now I find I've changed my mind

I've opened up the doors

——————————————————

Example of the Output

——————————————————

Min: I, s, m

Max: younger, anybody, assured, changed

——————————————————

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Task 3

——————————————————————————

Class name: com.epam.rd.java.basic.practice3.Part3

The input data should be uploaded from the "part3.txt" file

——————————————————————————

The task should be resolved using regular expressions without using container classes.

Input: text (it can contain both Cyrillic and Latin letters).

Output: the original text, but the case of the first character of each word, that consists of three or more characters, should be inverted.

A "word" should be considered a sequence of characters containing letters only (all the other characters are not considered to be part of a word).

Create a static method "convert" that converts input to output.

Stub of the method:

——————————————————

public static String convert(String input) {

    ...

}

——————————————————

Example of Input data

——————————————————

When I was younger I never needed

——————————————————

Example of the Output

——————————————————

when I Was Younger

I Never Needed

——————————————————

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Task 4

——————————————————————————

Class name: com.epam.rd.java.basic.practice3.Part4

——————————————————————————

For data hashing (for example, passwords) it is used a method named MessageDigest#digest that returns hash as a byte array.

An example of password hashing using MD5 hashing algorithm (other algorithms - SHA-256; SHA-512, etc.).

——————————————————

import java.security.\*;

import java.util.Arrays;

public class HashExample {

    public static void main(String[] args) throws NoSuchAlgorithmException {

        MessageDigest digest = MessageDigest.getInstance("MD5");

        digest.update("password to hash".getBytes());

        byte[] hash = digest.digest();

        System.out.println(Arrays.toString(hash));

        // output: [56, 55, 83, 50, 113, -114, -54, 115, -125, 86, 79, -109, 17, -65, 107, 84]

    }

}

——————————————————

Write a static method ‘hash’ that accepts two parameters:

(1) a string, the hash of which we need to obtain;

(2) the name of the hashing algorithm.

The output should be a string consisting of hexadecimal digits: each byte corresponds to two hexadecimal digits.

For example, in case an element of a byte array equals to -29, then in binary expansion it has "1110\_0011" format and corresponds to a couple of E3.

Stub of the class Part4

——————————————————

package com.epam.rd.java.basic.practice3.Part4;

import java.security.\*;

public class Part4 {

    public static String hash(String input, String algorithm) throws NoSuchAlgorithmException {

        // place yhour code here

        return null;

    }

    public static void main(String[] args) throws NoSuchAlgorithmException {

        System.out.println(hash("password", "SHA-256"));

        System.out.println(hash("passwort", "SHA-256"));

    }

}

——————————————————

If the code of the method Part4.main is the following:

——————————————————

System.out.println(hash("asdf", "MD5"));

System.out.println(hash("asdf", "SHA-256"));

——————————————————

then the result should be the following:

——————————————————

912EC803B2CE49E4A541068D495AB570

F0E4C2F76C58916EC258F246851BEA091D14D4247A2FC3E18694461B1816E13B

——————————————————

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Task 5

——————————————————————————

Class name: com.epam.rd.java.basic.practice3.Part5

——————————————————————————

Create a class with two static methods that convert data from the decimal system to the Roman number system and vice versa.

——————————————————

public static String decimal2Roman(int x) { ... }

public static int roman2Decimal(String s) { ... }

——————————————————

The operational range of the methods - from 1 to 100 inclusive.

Do not use container classes.

Demonstrate the methods functionality in the following way:

DECIMAL -decimal2Roman-> ROMAN -roman2Decimal-> DECIMAL

——————————————————

1 --> I --> 1

2 --> II --> 2

3 --> III --> 3

4 --> IV --> 4

5 --> V --> 5

...

94 --> XCIV --> 94

95 --> XCV --> 95

96 --> XCVI --> 96

97 --> XCVII --> 97

98 --> XCVIII --> 98

99 --> XCIX --> 99

100 --> C --> 100

——————————————————

Consider an algorithm  and create a program.

Brute-force is not allowed!

A solution using an array (a container) that contains 100 elements should not be considered:

——————————————————

String[] numbers = {"I", "II", "III", "IV", "V", ..., "XCV", "XCVI", "XCVII", ..., "C"}

——————————————————

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Task 6

——————————————————————————

Class name: com.epam.rd.java.basic.practice3.Part6

The input data should be uploaded from the "part6.txt" file

——————————————————————————

The task should be resolved using regular expressions without using container classes.

Input: text (it can contain both Cyrillic and Latin letters, but it does not contain \_).

Output: the original text, but all the recurring words should be preceded by underscore \_

A "word" should be considered a sequence of characters containing letters only (all the other characters are not part of a word).

Create a static "convert" method that converts the input to the output.

Stub of the method

——————————————————

public static String convert(String input) {

    ...

}

——————————————————

Example of the Input data

——————————————————

This is a test

And this is also a test

And these are also tests

test Это тест

Это также тест

И это также тесты

——————————————————

Example of the Output

——————————————————

This \_is \_a \_test

\_And this \_is \_also \_a \_test

\_And these are \_also tests

\_test

\_Это \_тест

\_Это \_также \_тест

И это \_также тесты

——————————————————

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

How to get the input data (the file should be placed in the root of the project, the file encoding - Ср1251)

——————————————————————————

package com.epam.rd.java.basic.practice3;

import java.io.IOException;

import java.nio.file.\*;

public class Util {

    private static final String ENCODING = "Cp1251"; //you can change it to UTF-8

    public static String readFile(String path) {

        String res = null;

        try {

            byte[] bytes = Files.readAllBytes(Paths.get(path));

            res = new String(bytes, ENCODING);

        } catch (IOException ex) {

            ex.printStackTrace();

        }

        return res;

    }

    public static void main(String[] args) {

        System.out.println(readFile("part1.txt"));

    }

}

——————————————————————————