Practice3

На странице представлена информация по практическому заданию №3.

Практическое занятие №3.

- Задание 1
  - 1.1. Метод convert1
  - 1.2. Метод convert2
  - 1.3. Метод convert3
  - 1.4. Метод convert4
- Задание 2
- Задание 3
- Задание 4
- Задание 5
- Как получить входную информацию

Для получения текстовой информации из файла см. пример "Как получить входную информацию".

Задание 1

Название класса: **ua.nure.your_last_name.Practice3.Part1**

Входную информацию загружать из файла **part1.txt**

Определить класс со статическими методами, которые преобразовывают входную информацию в выходную. В качестве входной информации (input data) использовать текст следующей структуры (значения Login/Name/Email в общем случае могут быть любыми):

Input data (part1.txt)

```
Login;Name;Email
ivanov;Ivan Ivanov;ivanov@mail.ru
петров;Петр Петров;petrov@google.com
obama;Barack Obama;obama@google.com
bush;Джордж Буш;bush@mail.ru
```

Заглушки методов, которые нужно написать

Методы, которые нужно написать, имеют следующий вид (N - цифра: 1, 2, 3, 4):

convertN

```
public static String convertN(String input) {
    ...
}
```

1.1. Метод convert1

Должен преобразовывать input data в строку следующего вида:

Output of convert1

```
ivanov ==> ivanov@mail.ru
петров ==> petrov@google.com
obama ==> obama@google.com
bush ==> bush@mail.ru
```

1.2. Метод convert2

Должен преобразовывать input data в строку следующего вида:

Output of convert2

```
Ivan Ivanov (email: ivanov@mail.ru)
Петр Петров (email: petrov@google.com)
Barack Obama (email: obama@google.com)
Джордж Буш (email: bush@mail.ru)
```

1.3. Метод convert3

Должен преобразовывать input data в строку следующего вида (почтовый домен ==> список логинов через запятую тех пользователей, чьи почтовые ящики зарегестрированны в данном домене):

Output of convert3

```
mail.ru ==> ivanov, bush
google.com ==> петров, obama
```

1.4. Метод convert4

Должен преобразовывать input data в строку следующего вида (должна быть добавлена колонка Password, сам пароль должен состоять ровно из 4 цифр, которые генерируются случайным образом):

Output of convert4

```
Login;Name;Email;Password
ivanov;Ivan Ivanov;ivanov@mail.ru;1707
петров;Петр Петров;petrov@google.com;9321
obama;Barack Obama;obama@google.com;4623
bush;Джордж Буш;bush@mail.ru;7514
```

Задание 2

Название класса: **ua.nure.your_last_name.Practice3.Part2**

Вход: текст (*может состоять из латиницы и кириллицы*).

Выход: слова максимальной и минимальной длины в формате, который дан ниже (в единственном числе и в порядке их появления в тексте). Словом считать последовательность содержащую только буквы (все остальные символы в состав слова не входят).

Входную информацию загружать из файла **part2.txt**

Создать статический метод convert, который преобразовывает вход в выход.

Заглушка метода

convertN

```
public static String convert(String input) {
    ...
}
```

Пример

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
```

Output

```
Min: I, s, m
Max: younger, anybody, assured, changed
```

Задание 3

Название класса: **ua.nure.your_last_name.Practice3.Part3**

Вход: текст (*может состоять из латиницы и кириллицы*).

Выход: исходный текст, но первый символ каждого слова, **состоящего из трех и более символов**, должен быть в верхнем регистре. Словом считать последовательность содержащую только буквы (все остальные символы в состав слова не входят). Входную информацию загружать из файла **part3.txt**

Создать статический метод **convert**, который преобразовывает вход в выход.

Заглушка метода

convertN

```
public static String convert(String input) {
    ...
}
```

Пример

Input data

```
When I was younger
I never needed
```

Output

```
When I Was Younger
I Never Needed
```

Задание 4

Название класса: **ua.nure.your_last_name.Practice3.Part4**

Для хеширования информации (например, паролей) используют метод MessageDigest#digest, который возвращает хеш в виде массива байт.

Пример хеширования пароля с помощью алгоритма хеширования MD5 (другие алгоритмы - SHA-256; SHA-512 и пр.)

Хеш пароля

```
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]
    }
}
```

Написать статический метод **hash**, который на вход принимает два параметра: (1) строку, хеш которой нужно получить; (2) названия алгоритма хеширования. Выход должен представлять из себя строку из шестнадцатеричных цифр: каждому байту соответствует две шестнадцатеричные цифры. Например, если некоторый элемент массива байт равен -29, то в двоичном разложении он имеет вид **1110**_**0011** и ему соответствует пара **E3**.

Stub

```java
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"));
    }

}
```

Задание 5

Название класса: **ua.nure.your_last_name.Practice3.Part5**

Создать класс с двумя статическими методами перевода из десятичной системы счисления в римскую и обратно.

```java
public static String decimal2Roman(int x) { ... }
public static int roman2Decimal(String s) { ... }
```

Рабочий диапазон методов - от 1 до 100 включительно.

Работу методов продемонстрировать так:

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
```

Брут-форс (полный перебор) не допускается! Решение которое использует массив из ста элементов
String[] numbers = {"I", "II", "III", "IV", "V", ..., "XCV", "XCVI", "XCVII", ..., "C"}
исключить из рассмотрения.

Продумать алгоритм и запрограммировать.

Возможно, окажет некоторую помощь информация по этим ссылкам: римские цифры, перевод из десятичной системы счисления в римскую.

В корневом пакете создать класс Demo, который демонстрирует работу всего написанного функционала.

Как получить входную информацию

Как получить входную информацию

Файл должен находится в корне проекта, кодировка файла - Cp1251.

Read file example

```
package ua.nure.your_last_name.Practice3;

import java.io.IOException;
import java.nio.file.*;

public class Util {

    private static final String ENCODING = "Cp1251";

    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"));
    }

}
```