Practice3

На странице представлена информация по практическому заданию №3.

Практическое занятие №3.

Если приложение считывает информацию из файла, то необходимо указать кодировку, в которой она (информация) записана. См. пример "Как получить входную информацию".

Задание 1

Определить класс с методами (они могут быть статическими или не статическими, в последнем случае использовать ООП подход), 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

```
Login;Name;Email
ivanov;Ivan Ivanov;ivanov@mail.ru
petrov;Petr Petrov;petrov@google.com
obama;Barack Obama;obama@google.com
bush;George Bush;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) {
    ...
}
```

или, если будет использован ООП подход, то вид будет такой

convertN

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

1.1. Метод convert1

Должен преобразовывать input data в строку следующего вида:

Output of convert1

```
ivanov ==> ivanov@mail.ru
petrov ==> 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

```
Ivanov Ivan (email: ivanov@mail.ru)
Petrov Petr (email: petrov@google.com)
Obama Barack (email: obama@google.com)
Bush George (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 ==> petrov, 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;2344
petrov;Petr Petrov;petrov@google.com;3423
obama;Barack Obama;obama@google.com;6554
bush;George Bush;bush@mail.ru;4534
```

Задание 2

Дан текст. Найти и напечатать все слова максимальной и все слова минимальной длины. Словом считать последовательность содержащую только буквы (все остальные символы в состав слова не входят).

Пример

Input data

```
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

Дан текст. Напечатать текст, поставив первый символ каждого слова в верхний регистр.

Пример

Input data

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

Output

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

Задание 4

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, 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

```
import java.security.*;

public class Part4 {

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

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

```
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"}
не годится.

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

Входные данные могут быть прочитаны из файла (сам файл должен находится в корне проекта)

Read file example

```
package ua.nure.name.Practice3;

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

public class Util {
    public static String getInput(String fileName) {
        StringBuilder sb = new StringBuilder();
        try {
            Scanner scanner = new Scanner(new File(fileName), "Cp1251");
            while (scanner.hasNextLine()) {
                sb.append(scanner.nextLine()).append(System.lineSeparator());
            }
            return sb.toString().trim();
        } catch (IOException ex) {
            ex.printStackTrace();
        }
        return sb.toString();
    }
    public static void main(String[] args) {
        String input = Util.getInput("part1.txt");
        System.out.println(input);
    }
}
```

или же могут быть определены как значение константы в программе:

INPUT field

```
private static final String INPUT = "Login;Name;Email\nivanov;Ivan Ivanov;ivanov@mail.ru\n..."
```

Рекомендуемая структура пакетов

Рекомендуемая структура пакетов

Рекомендуемая структура пакетов (name = your last name without project key):

```
ua.nure.name.Practice3
        Demo
ua.nure.name.Practice3.part1
        Part1
ua.nure.name.Practice3.part2
        Part2
ua.nure.name.Practice3.part3
        Part3
ua.nure.name.Practice3.part4
        Part4

ua.nure.name.Practice3.part5
        Part5
```