Practice4

На странице представлена информация по практическому заданию №4.

Внимание!!!

Класс Demo перед запуском ваших тестов будет удален.
При подсчете покрытия кода тестами содержимое класса Demo учитываться не будет.
(*не вызывайте из тестов Demo.main*)

На стенде до запуска ваших тестов Jenkins запускает дополнительно тесты для каждой из частей. Если эти тесты не проходят, **в первую очередь проверьте правильность вывода на предмет отсутствия лишних пробелов и ограничителей строк**. Пример результата работы Demo.main приведен внизу страницы. Также см. лог сборки.

Для всех подзадач должны быть написаны JUnit тесты, покрытие кода тестами должно быть максимально приближено к 100%. На практическом занятии в первую очередь писать функционал всех подзадач и в самую последнюю очередь тесты.

Если приложение считывает информацию из файла, то необходимо указать кодировку, в которой эта информация записана. Если явно не указан язык, на котором записана текстовая информация, то брать текст на языке, который содержит кириллицу (русский, украинский) и латиницу (английский)

Заглушка проекта

Проект-заглушка для проекта: ~~**/examples/projects/Task4Stub**~~ Practice4Stub.zip
(можно менять как угодно или вообще не использовать).

Для решения задачи целесообразно ознакомиться с примерами:
/examples/M06/IOExample
/examples/M05/RegexpExample
/examples/M05/I18nExample
/examples/M06/JunitConsoleMenuDemo

Задание 1

Название класса: **ua.nure.your_last_name.Practice4.Part1**

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

Создать класс, который выводит содержимое текстового файла в консоль, заменяя в каждом слове длиннее трех символов все строчные символы (нижний регистр) прописными (верхний регистр). При решении задачи использовать регулярные выражения.

Файл брать размером не более 1 Кб (достаточно несколько строк).

Под словом понимать непрерывную последовательность букв национальных алфавитов (см. выше)

Задание 2

Название класса: **ua.nure.your_last_name.Practice4.Part2**

Сгенерированные случайные числа сохранять в файле **part2.txt**

Выходную информацию (отсортированные числа) записывать в файл **part2_sorted.txt**

Создать класс, который создает и заполняет файл случайными целыми числами от 0 до 50 (всего 10 чисел), затем читает файл и выводит его содержимое в другой файл, отсортировав числа по возрастанию. Содержимое обоих файлов (числа разделенные пробелом) вывести в консоль (всего будет две строки).

Для сортировки написать собственный метод, который осуществляет сортировку некоторым алгоритмом (например "пузырьком"). Выходной файл должен быть текстовым (читабельным). Вывести содержимое входного и выходного файла в консоль в следующем формате:

⚠ Обратите внимание на то, что в первой строке между **input** и **==>** находится **один** пробел.

Пример консольного вывода

```
input ==> 30 23 16 16 9 23 3 18 21 29
output ==> 3 9 16 16 18 21 23 23 29 30
```

Замечание. При написании тестов просто вызовите функционал заполнения файла случайными числами из тестового метода, но для тестирования сортировки используйте заранее подготовленный файл с числами.

part2.txt

```
30 23 16 16 9 23 3 18 21 29
```

part2_sorted.txt

```
3 9 16 16 18 21 23 23 29 30
```

Задание 3

Название класса: **ua.nure.your_last_name.Practice4.Part3**

Входную информацию загружать из файла **part3.txt**

Файл содержит символы, слова, целые числа и числа с плавающей точкой (**разделены пробелами**). Написать класс, который имеет следующую функциональность: в цикле пользователь вводит с консоли тип данных (один из: **char**, **String**, **int**, **double**), в ответ приложение печатает в консоль все значения соответствующих типов, которые существуют в файле. Задачу решить с использованием регулярных выражений.

Замечание: под строкой понимать последовательность символов два и более. Символы - латинские или кириллические буквы в верхнем или нижнем регистре (обязательно предусмотреть наличие кириллицы во входном файле).

Пример исходного файла:

part3.txt

a bcd 43.43 432 и л фвыа 89 .98

⚠ Признаком окончания ввода является слово **stop**.

Задание 4

Название класса: **ua.nure.your_last_name.Practice4.Part4**

Входную информацию загружать из файла **part4.txt**

Создать класс, который реализует интерфейс **java.lang.Iterable**. Класс должен разбирать текстовый файл и возвращать предложения из файла. Метод **iterator** данного класса должен возвращать объект итератор - экземпляр внутреннего класса.

⛔ Не допускается использовать существующие реализации итераторов из контейнерных классов! Используйте регулярные выражения.

Замечание. В методе **Iterator#remove** пропишите выброс исключения **UnsupportedOperationException** (т.е. данный метод наполнять логикой не нужно, но сам метод обязан присутствовать, т.к. на стенде используется Java 7, которая не поддерживает дефолтных реализаций методов интерфейсов).

Рекомендация. Напишите регулярное выражение, которое "вырезает" предложения из текста, далее используйте объект Matcher при реализации методов интерфейса Iterator.

Задание 5

Название класса: **ua.nure.your_last_name.Practice4.Part5**

Имена *properties* файлов: **resources_en.properties**, **resources_ru.properties**

Создать пакеты ресурсов (*properties файлы*) для двух локализаций: **ru** и **en** (**поместить в каталог src**). Пакеты содержат как минимум две записи, например:

resources_en.properties

table = table
apple = apple

resources_ru.properties

table = стол
apple = яблоко

Написать класс, который в цикле читает с консоли ключ (key) и имя локализации через пробел, в ответ печатает соответствующее значение в консоль.

⛔ Чтение из консоли и запись в консоль являются обязательными!

⚠ Признаком окончания ввода является слово **stop**.

Содержимое класса Demo

В корневом пакете (**ua.nure.your_last_name.Practice4**) должен находится класс **Demo**, который демонстрирует работу всего функционала.

⛔ Для тех подзадач, которые требуют ввода с консоли, переназначить стандартный поток ввода таким образом, чтобы ввод осуществлялся из некоторой заданной строки. **Demo.main** должен отрабатывать без участия пользователя, никакого ожидания ввода с консоли при выполнении данного метода быть не должно. Пример переназначений см. в заглушке.

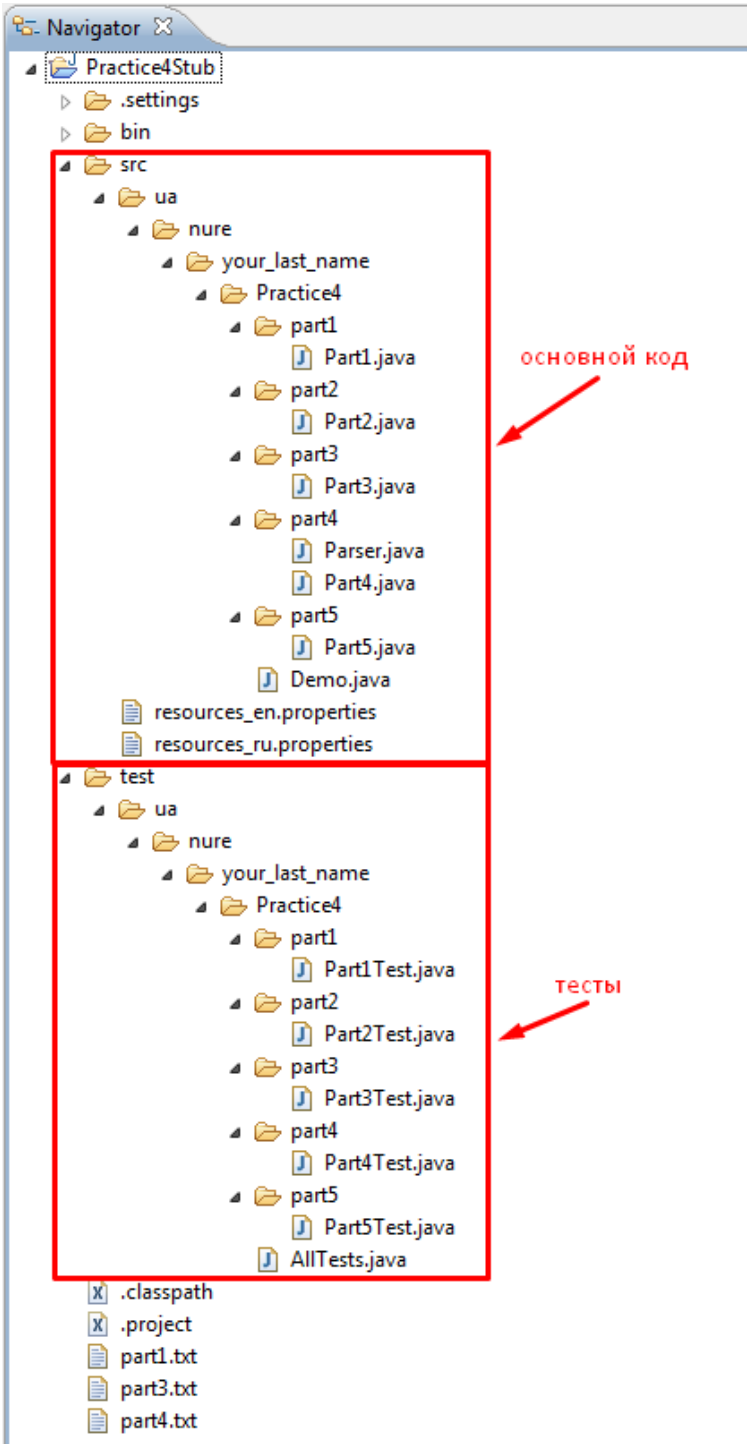
Если приложение на стенде зависнет в ожидании ввода с консоли, то не более чем через 3 минуты оно будет снято с выполнение (на все задачи выставлен timeout).

Пример метода Demo.main (click here to expand...)

```
package ua.nure.your_last_name.Practice4;

import java.io.ByteArrayInputStream;
import java.io.IOException;
import java.io.InputStream;

import ua.nure.your_last_name.Practice4.part1.Part1;
import ua.nure.your_last_name.Practice4.part2.Part2;
import ua.nure.your_last_name.Practice4.part3.Part3;
import ua.nure.your_last_name.Practice4.part4.Part4;
import ua.nure.your_last_name.Practice4.part5.Part5;

public class Demo {
    private static final InputStream STD_IN = System.in;
    private static final String ENCODING = "Cp1251";
    public static void main(String[] args) throws IOException {
        System.out.println("========================== PART1");
        Part1.main(args);

        System.out.println("========================== PART2");
        Part2.main(args);

        System.out.println("========================== PART3");
        // set the mock input
        System.setIn(new ByteArrayInputStream("char\nString\nint\ndouble".getBytes(ENCODING)));
        Part3.main(args);
        // restore the standard input
        System.setIn(STD_IN);

        System.out.println("========================== PART4");
        Part4.main(args);

        System.out.println("========================== PART5");
        // set the mock input
        System.setIn(new ByteArrayInputStream("table ru\ntable en\napple
ru".getBytes(ENCODING)));
        Part5.main(args);
        // restore the standard input
    }
}
```

Тесты

Тесты должны располагаться в каталоге **test** в соответствующих пакетах. В корневом пакете создать тестовый набор **AllTests**, который объединяет все тестовые классы. Пример структуры пакетов (картинка кликабельна):



Порядок выполнения работы

1. Сделать checkout проекта заглушки из репозитория и отвязать его от узла: /examples/projects/Practice4Stub
2. Переименовать проект: **Task4Stub** ==> **Task4**
3. Переименовать корневой пакет проекта: your_last_name == > ваш логин, без последних трех букв, которые обозначают код проекта - jto.
4. Реализовать все подзадачи.
5. Реализовать класс Demo, который демонстрирует функциональность всех подзадач (его будет вызывать **Jenkins**). Для некоторых подзадач при этом необходимо моделировать консольный ввод, как это сделать см. класс Demo из заглушки.
6. Написать JUnit тесты, которые на 100% покрывают исходный код, тесты должны располагаться в отдельном каталоге test.
7. Создать тестовый набор (test suite) с именем AllTests (полное имя - ua.nure.your_last_name.Practice4.AllTests),
8. Привязать проект к нужному узлу в репозитории, сделать коммит проекта в репозиторий.
9. Добиться чтобы **Jenkins** успешно собрал проект.
10. Оптимизировать метрики в **Sonar**.

Замечания

1. Обязательно посмотреть в лог сборки проекта (Jenkins), вывод **должен совпадать** с тем выводом, который получается на вашей локальной машине.
2. При выводе информации используйте платформонезависимый ограничитель строки, иначе при запуске в др. ОС вы можете не увидеть то, что ожидаете увидеть;
3. ⚠️ Не используйте абсолютные адреса файлов, задавайте относительные пути от корня проекта (иначе проект, скорее всего, не будет собран).
4. **Timeoute** сборки проекта в Jenkins **3 минуты**. Если при сборке проекта будет вызвана функциональность ожидания консольного ввода, то максимум через 3 минуты проект будет снят с выполнения, а сама сборка помечена как Aborted.

Пример результата работы Demo.main

Click here to expand...

```
============================ PART1
ЕСЛИ ПРИЛОЖЕНИЕ СЧИТЫВАЕТ ИНФОРМАЦИЮ из ФАЙЛА,
то НЕОБХОДИМО УКАЗАТЬ КОДИРОВКУ, в
КОТОРОЙ она (ИНФОРМАЦИЯ) ЗАПИСАНА.
============================ PART2
input ==> 46 34 26 18 12 26 37 8 43 32 15 3 33 24 34 27 28 11 1 17
output ==> 1 3 8 11 12 15 17 18 24 26 26 27 28 32 33 34 34 37 43 46
============================ PART3
я f и л
bcd фвыа
432 89
43.43 .98
============================ PART4
Класс должен разбирать текстовый файл и возвращать предложения из файла.
Под предложением понимать последовательность, которая начинается с большой буквы и
заканчивается точкой.
Исходный файл брать небольшим по размеру.
Достаточно несколько строк и несколько предложений.
============================ PART5
стол
table
яблоко
```