Practice4

На странице представлена информация по практическому заданию №4.

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

Замечания.
(1) Если приложение считывает информацию из файла, то необходимо указать кодировку, в которой она (информация) записана.
(2) Если явно не указан язык, на котором записана текстовая информация, то брать текст на языке, который может содержать кириллицу (русский, украинский языки) и латиницу (английский язык).
(3) В корневом пакете создать класс Demo, который демонстрирует работу всего написанного функционала.
(4) Для всех подзадач должны быть написаны JUnit тесты, покрытие кода тестами - 100% (или как можно ближе к 100%).
(5) Имена входных и выходных файлов указаны относительно значения системного свойства user.dir (= корневой каталог проекта).
(6) Обязательно посмотреть в лог сборки проекта (Jenkins), вывод должен совпадать с тем выводом, который получается на вашей локальной машине.
(7) При выводе информации используйте платформонезависимый ограничитель строки, иначе при запуске в др. ОС вы можете не увидеть то, что ожидаете увидеть.
(8) Не используйте абсолютные адреса файлов, задавайте относительные пути от корня проекта (иначе проект скорее всего не будет собран).

(9) Timeoute сборки проекта в Jenkins 2 минуты. Если при сборке проекта будет вызвана функциональность ожидания консольного ввода, то максимум через 2 минуты проект будет снят с выполнения, а сама сборка помечена как Aborted.
(10) Если Jenkins не собирает проект по причине выброса IllegalAccessError с сообщением "tried to access class XXX" поставьте уровень доступа типа с именем XXX в public.

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).

Задание 2

Название класса: **ua.nure.your_last_name.Practice4.Part2**
Входную информацию загружать из файла **part2.txt**
Выходную информацию загружать в файл **part2_sorted.txt**

Создать класс, который:

1) Создает и заполняет файл (part2.txt) случайными целыми числами от 0 до 50 (всего 10 чисел)
2) Затем читает файл и выводит его содержимое в другой файл (part2_sorted.txt), отсортировав числа по возрастанию.
3) Читает содержимое обоих файлов как текст (числа разделенные пробелом) и выводит в консоль.

Для сортировки написать собственный метод, который осуществляет сортировку некоторым алгоритмом (например "пузырьком"). Выходной файл должен быть текстовым (читабельным).

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 функционал заполнения файла случайными числами просто вызовите из тестового метода, а для тестирования сортировки используйте заранее подготовленный файл с числами из примера выше.

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. Написать класс, который имеет следующую функциональность: в цикле пользователь вводит тип данных (один из: **char**, **String**, **int**, **double**), в ответ приложение печатает в консоль все значения соответствующих типов, которые существуют в файле. Если пользователь вводит слово stop, то происходит выход из цикла.

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 консольного вывода

```
----------------------------------------------------------
a bcd 43.43 432 и л фвыа 89 .98
----------------------------------------------------------
```

Пример вывода **char**:

```
----------------------------------------------------------
a и л
----------------------------------------------------------
```

Пример вывода **string**:

```
----------------------------------------------------------
bcd фвыа
----------------------------------------------------------
```

Пример вывода **int**:

```
----------------------------------------------------------
432 89
----------------------------------------------------------
```

Пример вывода **double**:

```
----------------------------------------------------------
43.43 .98
----------------------------------------------------------
```

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**
Входной пакет ресурсов, локаль **ru**: **resources_ru.properties**
Входной пакет ресурсов, локаль **en**: **resources_en.properties**
Пакеты ресурсов расположить непосредственно в каталоге **src**

Создать пакеты ресурсов (properties файлы) для двух локалей: ru и en. Пакеты содержат как минимум две записи, например:

resources_en.properties

```
table = table
apple = apple
```

resources_ru.properties

```
table = стол
apple = apple
```

Написать класс, который в цикле читает с консоли ключ (key) и имя локализации через пробел, в ответ печатает соответствующее значение в консоль. Признаком окончания ввода служит слово **stop**.

> Чтение из консоли и запись в консоль являются обязательными!

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 (после отрабатывания вашего кода необходимо предусмотреть восстановление стандартных потоков).

> **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 2 минуты оно будет снято с выполнения (на все задачи выставлен timeout).

Пример метода Demo.main

```
-------------------------------------------------------
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^String^int^double^stop".replace("^",
System.lineSeparator()).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^table en^apple ru^stop".replace("^",
System.lineSeparator()).getBytes(ENCODING));
         Part5.main(args);
         // restore the standard input
         System.setIn(STD_IN);
   }
}
```

Замечание по тестам.

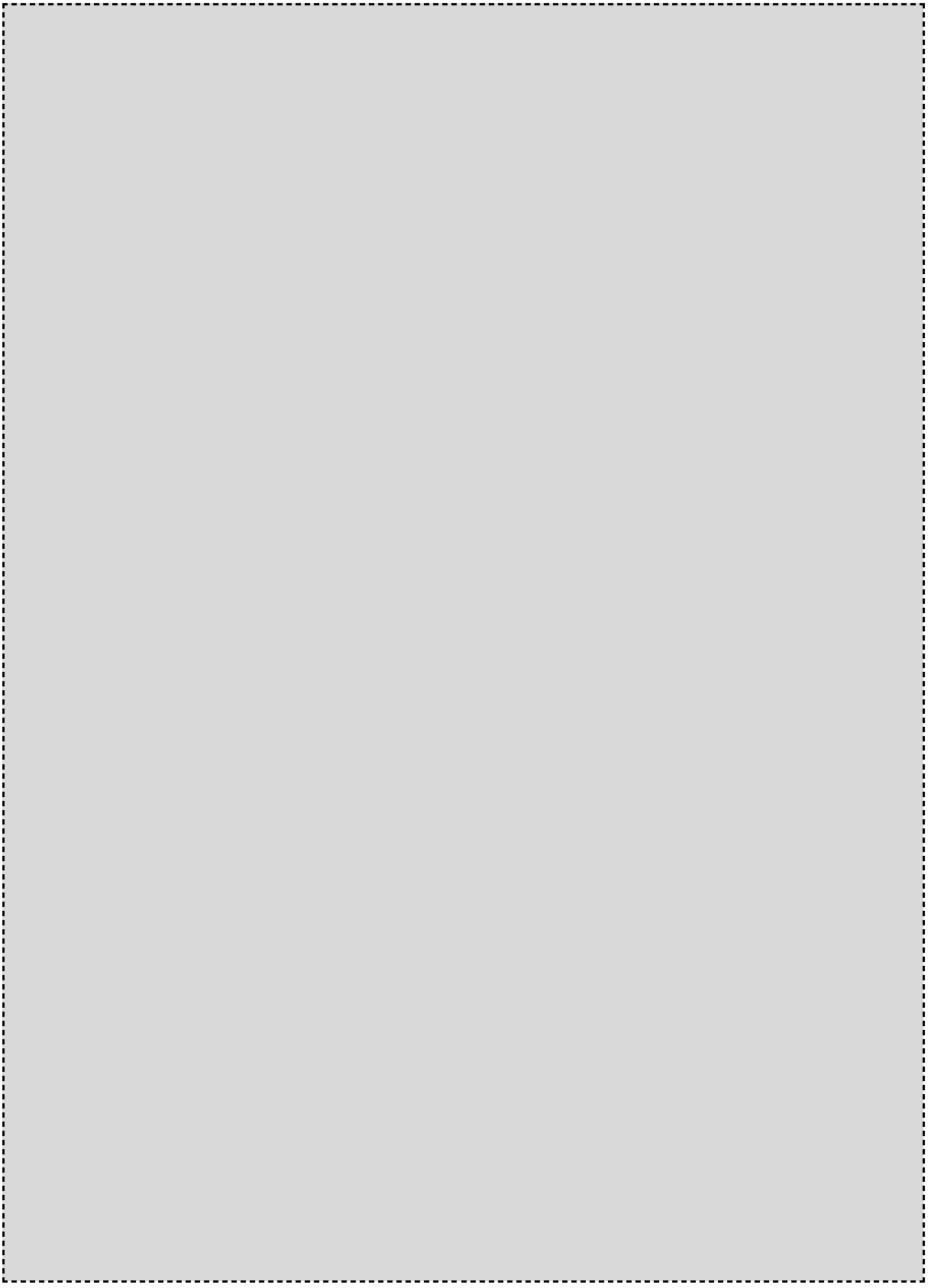
Тесты должны располагаться в каталоге **test** в соответствующих пакетах.

В корневом пакете создать тестовый набор **AllTests**, который объединяет все тестовые классы.

Пример файловой структуры проекта:

```
-------------------------------------------------------
Practice4
    src
```

```
        ua
            nure
                ivanov
                    Practice4
                        Demo.java
                        Part1.java
                        Part2.java
                        Part3.java
                        Part4.java
                        Part5.java
    test
        ua
            nure
                ivanov
                    Practice4
                        AllTests.java
                        Part1Test.java
                        Part1Test.java
                        Part1Test.java
                        Part1Test.java
                        Part1Test.java
-------------------------------------------------------
```



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):

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: yourlastname == > ваш логин, без последних трех букв, которые обозначают код проекта - jtk.
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 распологаться 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.