Training materials

- The Java Tutorials. Lessons: Numbers and Strings, Exceptions, Regular Expressions.
- **Course on learn.epam.com Java.Strings.**
- И. Блинов. Глава 7 (Классы String, StringBuilder, Регулярные выражения, Интернационализация текста, чисел и дат).
- http://skipy.ru/technics/strings.html
- И. Блинов. Глава 8 (основы обработки исключений, стр. 201-209).

Code Exercise

A csv file (text data separated by semicolon) contains some sequence of lines with any number of elements in each line.

Let the initial element of any line have an index 0. Its value determines the index of an element in the line you should work with.

Calculate:

1. the sum of elements, determined by zero elements. Present this result in the string of the output format shown in examples below.

2. the number of lines with “errors”.

The example of the file in1.csv:

```
3;qw;4;5.2;2.7
15;;;k;5
1;-3.14;fgh;5
0;;e1;2;3
-2.3;a;b;c
b;d;e
```

Results:

```
result(5.2 - 3.14 + 0.0) = 2.06
error-lines = 3
```

The example of the file in2.csv:

```
3;qw;4;-3.1;2.7
1;-1;fgh;5
```

Results:

```
result(-3.1 - 1.0) = -4.1
error-lines = 0
```

The example of the file in3.csv:

```
1;7.5e-1;2.7
```

Results:

```
result(0.75) = 0.75
error-lines = 0
```

The example of the file in4.csv:

```
1;0
```

Results:

```
result(0.0) = 0.0
error-lines = 0
```

The example of the file in5.csv:

```
1;da
```

Results:

```
result() = 0.0
error-lines = 1
```

Define the TestRunner class in the **default** package with unit tests for the static method
static int getResult(String csvName, StringBuilder csvName) throws FileNotFoundException
csvName is the name of input csv file,
strResult is the result string of the above format,
returned value is the number of lines with errors.

Замечания и ограничения к задаче 1

– Не использовать коллекции.

– Информационных классов не создавать. Реализовать в одном классе, т.е. в процедурном стиле (как скрипт).

Один из вариантов структуры класса:

```
public class TestRunner {

    private static int getResult(String csvName, StringBuilder strResult) throws FileNotFoundException {
        try (Scanner scanner = new Scanner(new FileReader(csvName))) {
            …
            strResult.insert(0, RESULT_HEAD).append(RESULT_TAIL).append(numResult);
            return errorLines;
        }
    }

    @Test
    public void testMainScenario() throws FileNotFoundException {
        StringBuilder result = new StringBuilder();
        int errorLines = getResult("src/in1.csv", result);
        Assert.assertEquals(3, errorLines);
        String expectedIn1 = "...";
        Assert.assertEquals(expectedIn1, result.toString());
    }

    @Test
    public void testXxx() throws FileNotFoundException {
        ...
    }

    …

    @Test(expected = FileNotFoundException.class)
    public void testWrongCsvName() throws FileNotFoundException {
        ...
    }
}
```

– Строки csv–файла разбивать на элементы методом split( ).

– Не использовать регулярные выражения (см. И. Блинов, стр. 177).

– Если код в блоках catch одинаковый, то блоки объединяют через операцию ИЛИ.

```
} catch (Exception0 | Exception1 e) {
        // some code
}
```

– Согласно условию нужно сформировать ответ в виде экземпляра StringBuilder. Обратите внимание на наличие пробелов в выходной строке. В частности, если первое число отрицательное, то после знака минус нет пробела. Далее каждый знак (плюс, минус, равно) окружен пробелами.

– Попробуйте найти такой алгоритм, когда особый случай, связанный с первым отрицательным числом в выходной строке, обрабатывается после цикла. Т.е. внутри цикла не должно быть подобной обработки:

```
int numLine = 0;
while (sc.hasNext()) {
    line = sc.nextLine();
    …
    if(numLine == 0) {
        …
    } else {
        …
    }
    numLine++;
}
```

– Вызов метода replaceAll() – это скрытый цикл. Поэтому запрещаю его использование.

– Обязательно проверить тест, в котором отсутствует входной файл.

– Самый распространенный антипаттерн – magic numbers, т.е. константы (особенно повторяющиеся), смысл которых очевиден исключительно автору кода. На выводимые строковые комментарии к результатам я пока еще закрываю глаза. Но числа или строки, которые используются в алгоритме, обязательно нужно инициализировать в константных локальных переменных либо вообще выносить в отдельный класс статических констант. Рекомендую отличную статью http://skipy.ru/philosophy/constantsUsage.html).

Примеры (локальные константы).

```
final String PLUS = " + ";
final String MINUS = " - ";
final int PLUS_LENGTH = PLUS.length();
```

– Чтобы предыдущее замечание имело **очевидный** практический смысл, добавляю следующее ограничение. Код обработки начала выходной строки не должен измениться, если знак будет окружен любой комбинацией символов. Например, двумя пробелами, или слева пробелом, а справа табуляцией и т.п.

Следствие: позиции редактируемого текста необходимо регулировать константами.