List'Em Project

There are two commands you can execute on the command line in linux. The first is "grep". With it you can search through all files in a specified directory (or folder) whose names match a file selector specified as a regular expression and list the lines in each file that contain a substring that matches a second regular expression. With an optional command line argument, "-r", you can specify that you want the search to continue recursively through all directories in the specified directory, and then through their directories, and so forth until the end.

The second command line capability "lineCount" is implemented with the linux commands find and wc (word count). ". With it you can list all files in a specified directory whose names match a file selector specified as a regular expression and list the number of lines in each file. With an optional command line argument, "-r", you can specify that you want the search to continue recursively through all directories in a specified directory, and then through their directories, and so forth until there are no more directories to traverse.

As you can see, both operations are similar. You are to write two similar programs. One will be called "Grep", the other "LineCount". Both will be conceptually similar to the linux operations but their input syntax, output syntax and semantics will be altered.

Grep

The interface for Grep is defined here.

The *directory* is a directory on the machine.

The fileSelectionPattern is a regular expression specifying which files to search. The pattern must match a file's entire name for that file to be searched. The syntax of the regular expression must match that specified in the **Pattern** class in the java library.

The substringSelectionPattern is a regular expression specifying which lines in a selected file we are to match. A line is matched if it contains 1 or more substrings that match the substringSelectionPattern. The syntax of the regular expression must match that specified in the **Pattern** class in the java library.

Example:
Using the provided main class, RunGrep, the command line entry:

```
java listem.RunGrep -r c:\\Users\Scott\workspace\ExampleJavaCode ".*\.java" total
```

returned the output:

```
FILE: c:\Users\Scott\workspace\ExampleJavaCode\src\fileSearch\FileSearch.java
      private int _totalMatches;
```

```
            _totalMatches = 0;
            System.out.println("TOTAL MATCHES: " + _totalMatches);
                  ++_totalMatches;
MATCHES: 4

FILE: c:\Users\Scott\workspace\ExampleJavaCode\src\fileSearch\LineCount.java
      private int _totalLineCount;
            _totalLineCount = 0;
            System.out.println("TOTAL: " + _totalLineCount);
            ++_totalLineCount;
MATCHES: 4

TOTAL MATCHES: 8
```

LineCount

The interface for LineCount is defined here.

The *directory* is a directory on the machine.

The fileSelectionPattern is a regular expression specifying which files to search. The pattern must match a file's entire name for that file to be searched. The syntax of the regular expression must match that specified in the **Pattern** class in the java library.

Example:
Using the provided main class, RunLineCount, the command line entry:

```
java listem.RunLineCount -r c:\\Users\Scott\workspace\ExampleJavaCode ".*\.java"
```

returned the output:

```
355 c:\Users\Scott\workspace\ExampleJavaCode\src\evilHangMan\EvilHangman.java
39 c:\Users\Scott\workspace\ExampleJavaCode\src\evilHangMan\Result.java
30 c:\Users\Scott\workspace\ExampleJavaCode\src\evilHangMan\ValueResult.java
122 c:\Users\Scott\workspace\ExampleJavaCode\src\fileSearch\FileProcessor.java
91 c:\Users\Scott\workspace\ExampleJavaCode\src\fileSearch\FileSearch.java
71 c:\Users\Scott\workspace\ExampleJavaCode\src\fileSearch\LineCount.java
339 c:\Users\Scott\workspace\ExampleJavaCode\src\imageEditor\ImageEditor.java
132 c:\Users\Scott\workspace\ExampleJavaCode\src\spellingCorrector\SpellingCorrector.java
284 c:\Users\Scott\workspace\ExampleJavaCode\src\spellingCorrector\WordCounts.java
TOTAL: 1463
```

Deliverables

Build a class that correctly implements the Grep interface and one that correctly implements the LineCounter interface, according to this specification. Both classes should have a constructor that takes no arguments. Both grep() and countLines() should not save data from previous calls to the methods (i.e. repeated calls should not affect their behavior).

The two interfaces and two simple main classes can be downloaded here.

You are to minimize the amount of duplicate code in your programs. You do this be creating a third class that is a super class of both your Grep and LineCount classes. It will contain all fields and algorithms common to the other two programs. You are to do this using the **Template Method Pattern**. You can find other descriptions by going to Google and entering the keywords: “template”, “method”, and “pattern”.