RegexTextReplacementInFiles

This package contains the skeleton structure for this exercise.

Use only built-in libraries come with Oracle's JDK 6/7/8, implement the process() method in the code (also shown below) to perform string replacement for each of the qualified files under a starting directory. Before the program terminates, display a summary report of the strings and occurrences count that were replaced (see sample output). Save the processed output to files in the same directories of the source files with new file names by appending the original names with ".processed".

Please assume the file size can be arbitrarily large. The file repository can also be arbitrarily large which may contain large number of sub directories and files. It might be crucial to process them as efficient as possible.

To demonstrate the level of your OOP understanding and programming skill, please structure your implementation to allow easy customization in the future for 1) Directory walking strategies. 2) Tasks to be performed against each file. 3) Stats to be collected and reports to be generated.

- The program takes 3 required parameters and one optional parameter:
  - Starting directory/file
  - String pattern to be replaced - in Java supported regular expression, see (1)
  - String to be replaced with
  - File naming pattern - UNIX wild-card filename syntax, see (2)
- Notes
  1. The string pattern is a JDK supported regular expression with exactly 1 match group. Find all texts, which match the pattern, and replace **ONLY** the match group 1 with the replacement. If the match group in the given pattern doesn't equal to one, the program can exit immediately by throwing and/or printing error.
  2. The file naming pattern will be the generally accepted file naming pattern in UNIX where '*' matches any number of characters, '?' matches one character.

RegexTextReplacementInFiles.java

```
public class RegexTextReplacementInFiles {
    public static void process(String startingDir, String regexPattern
        , String replacement, String fileAcceptPattern) {

    }

    public static void main(String[] args) {
        String startingDir = null
            , regexPattern = null
            , replacement = null
            , fileAcceptPattern = null
            ;

        if(args.length >= 3) {
            startingDir = args[0];
            regexPattern = args[1];
            replacement = args[2];
        }
        if(args.length >= 4) {
            fileAcceptPattern = args[3];
        }
        if(startingDir != null) {
            process(startingDir, regexPattern, replacement, fileAcceptPattern);
        } else {
            System.out.println("Expected at least 3 parameters but got " + args.length);
        }
```

```
    }

}
```

Sample Input / Output (sample in UNIX OS, please adjust for your OS when necessary)

```
./scripts/regexTextReplacementInFiles sample_dir '\w*(lan)\w+' '<-replaced->' *.txt
Processed 3 files
Replaced to '<-replaced->':
 * Planitia : 1 occurrence
 * lander : 1 occurrence
 * landing : 5 occurrences
 * lands : 1 occurrence
 * plans : 1 occurrence
 * plants : 1 occurrence

$ ls -R sample_dir

abcde.txt                       abcde.txt.processed
dummy.doc                       folder1

sample_dir/folder1:
folder1-sample1.txt             folder1-sample1.txt.processed
folder1-sample2.txt             folder1-sample2.txt.processed

$ cat sample_dir/folder1/folder1-sample2.txt.processed
…
Watney p<-- - replaced-- - >s to drive 3,200 kilometres (2,000 mi) to Schiaparelli crater when the
Ares 4 mission <-- - replaced-- - >ds there in four years. He begins modifying one of Ares 3's
rovers for the journey, adding solar cells and an additional battery. He makes a long test drive
to recover the unmanned Pathfinder <-- - replaced-- - >der and Sojourner rover and brings them back
…
```

Acceptance Criteria:

- Use only core JDK library.
- The program should not exit due to errors that were not handled.
- The processing report is printed
- All strings in the files that matches the string pattern are replaced with the string "to be replaced with"
- Only process the files matched the file-naming pattern. If the file-naming pattern was not given, apply the replacement to all files. In all cases, files can be in different subdirectories within the given starting directory.
- If the starting directory is a file, apply the replacement to the matching file, i.e. if file-naming pattern is given and the file name matches the pattern.

What we look for:

- Code quality
    - Following the OOP principals
    - Best programming practices
    - Readability with/without comments
- Correctness
- Completeness

What we are *not* looking for in this exercise (don't spend too much time):

- Fancy output of messages

- Print out Usage/Help for command line

Submission

- When finished, please zip up your code in the directory structure of the project and send it in as a zip file. Please make sure the zip file includes all your files
- It is ok not to finish all the functionalities, but please keep it compile-able and runnable for your submission