Individual Project:
Replace Utility -- Deliverable 3

Goals

- Develop a Java application for replacing strings within a file.
- Get experience with an agile, test-driven process.

Details

For this project, you must develop, using the Java language, a simple **command-line** utility called *replace*. For this last deliverable, you will have to perform two main tasks:

- Task 1: Slightly extend and modify the functionality of `replace`
- Task 2: Evaluate your tests on an alternative implementation of `replace`

Task 1: Extending `replace`

After your customers started using replace, they realized that it would be useful to be able to replace multiple from-to pairs at a time. They also decided that, if "-b" is specified but a backup copy of a given file already exists, no replace operation should be performed on that file. The updated specification below includes these changes, highlighted in red.

Concise Specification of the `replace` Utility

- NAME:
  `replace` - looks for all occurrences of string *from* and replaces it with string *to*. It is possible to specify one or more files on which to perform the replace operation(s) in a single replace command. It is also possible to specify one or more pairs of *from* and *to* strings (i.e., one or more replace operations) in a single replace command. In this case, the replacements are performed in sequence, in the order in which they appear on the command line.
- SYNOPSIS
  `replace OPT <fr> <to> [<fr> <to>]* -- <filename> [<filename>]*`

  where `OPT` can be zero or more of
  - `-b`
  - `-f`
  - `-l`

- `-i`

- COMMAND-LINE ARGUMENTS AND OPTIONS

  `fr`: string to be replaced with string `to`.

  `to`: string that will replace string `from`.

  `filename`: the file(s) on which the replace operation has to be performed.

  `-b`: if specified, the `replace` utility creates a backup copy of each file on which a replace operation is performed before modifying it. If a backup for a given file is already present, no replace operation will be performed on that file.

  `-f`: if specified, the `replace` utility only replaces the first occurrence of string `from` in each file.

  `-l`: if specified, the `replace` utility only replaces the last occurrence of string `from` in each file.

  `-i`: if specified, the `replace` utility performs the search for string `from` in a case insensitive way.

- EXAMPLES OF USAGE

  Example 1:

  ```
  replace -i Howdy Hello -- file1.txt file2.txt file3.txt
  ```

  would replace all occurrences of "Howdy" with "Hello" in the files specified. Because the "`-i`" option is specified, occurrences of "howdy", "HOwdy", "HOWDY", and so on would also be replaced.

  Example 2:

  ```
  replace -b -f Bill William -- file1.txt file2.txt
  ```

  would replace the first occurrence of "Bill" with "William" in the two files specified. It would also create a backup copy of the two files before performing the replace.

  Example 3:

  ```
  replace -f -l abc ABC -- file1.txt
  ```

  would replace both the first and the last occurrences of "abc" with "ABC" in the file specified.

  Example 4:

  ```
  replace Howdy Hello Bye Regards -- file1.txt file2.txt
  ```

  would replace all occurrences of "Howdy" with "Hello" and all occurrences of "Bye" with "Regards" in the files specified.

To complete Task 1, you must modify your current version of `replace` so that it satisfies these new requirements. In the spirit of TDD, one of your coworkers has already modified the `MainTestAddOn` test suite to incorporate these requirements in the form of test cases and provided this new test suite to you (see below). **All you have to do is to make sure that your implementation of `replace` makes all of these provided test cases and your existing ones (possibly modified) pass.**

Task 2: Evaluate your tests on an alternative implementation of `replace` (and do some extra black-box testing)

In this task, you will be provided with a precompiled, alternative version of the `replace` utility that was developed by one of your colleagues in parallel to yours. The goal of this task is to use this version of `replace` to evaluate the tests that you developed for D1 and (possibly) updated for D2. To complete this task you must (1) run the tests you defined (class `MyMainTest`) against the provided version of `replace` and measure the coverage they achieve and (2) extend your set of tests to try to achieve 100% line (i.e., statement) coverage. In the process, you will also keep track of how many bugs your tests can find in this version of `replace` (there are several, none of which is revealed by the test cases we provided). You must find at least a bug to get full points, and finding more than two bugs will qualify you for up to 10 extra points for this deliverable.

Deliverable 3: Detailed instructions

1. Download archive individualProject-d3.tar.gz
2. Unpack the archive in the root directory of the **personal GitHub repo that we assigned to you**. After unpacking, you should see the following files:

   Files for Task 1:

   - `<root>/IndividualProject/.../replace/MainTestAddOn.java`
     This is an updated version of the tests that we provided for D2 and that encode the requirements for the `replace` utility.

   Files for Task 2:

   - Under `<root>/IndividualProject/replace/D3Task2` :
     - `compileAndRunTests.bat` and `compileAndRunTests.sh`
       These scripts for Windows and Unix/Mac, respectively, will compile and run against the provided version of `replace` the set of tests in `.../D3Task2/testsrc/edu/.../MyMainTest.java` and save the corresponding coverage information.

- `generateReport.bat` and `generateReport.sh`
  These scripts for Windows and Unix/Mac, respectively, will take the coverage information generated using the previous scripts and generate a coverage report in file `report.txt`.
- `lib/*`
  Various libraries used to run tests, compute coverage, generate reports, and so on. You can safely ignore these files.
- `template.xml`
  This file contains metadata used when computing coverage. You can safely ignore this file.
- `.../D3Task2/testsrc/edu/.../MyMainTest.java`
  This is a placeholder that you will have to replace with your own version of this test class.
- `testclasses/*`
  This is the directory where the compiled version of your tests (i.e., class `MyMainTest`) will be saved. You can safely ignore this file.

Instructions for Task 1

3. As it was the case for D2, all the tests in the updated `MainTestAddOn` set must pass, **unmodified**, on your implementation or `replace`. Therefore, you will have to adapt/refactor your code accordingly. In addition, if some of your own test cases (class `MyMainTest`) no longer pass after you modify `replace`, you will have to update those tests as well so that they comply with the new requirements. In short, you will be done with Task 1 when all the test cases in the new `MainTestAddOn` class will pass (unmodified) and all the test cases in the `MyMainTest` class will pass (possibly after modifying them) on your implementation of `replace`.
4. Commit and push your code. You should commit, at a minimum, the content of directories `IndividualProject/replace/test` and `IndividualProject/replace/src`. As for previous assignments and projects, committing the whole IntelliJ IDEA or Eclipse project, in case you used one of those IDEs, is fine. The comment for your commit should be "Task 1 completed", so that we can easily spot it.

Instructions for Task 2

5. Complete Task 1 before performing Task 2.
6. Before beginning Task 2 you should make sure that the provided files work as expected, by doing the following:

- Open a command shell
- Go to directory .../D3Task2
- Run ./compileAndRunTests.sh
  (if on Windows, run the corresponding bat file)
  You should see the following output (time may vary):

```
JUnit version 4.12
.......
Time: 0.084

OK (7 tests)
```

- Run ./generateReport.sh
  (if on Windows, run the corresponding bat file)
  This command should produce no output.
- Check the content of file report.txt, which should be as follows:

```
Coverage Report:
ALL: classes: 100% (2/2); branches:  72% (55/76); lines:  84%
(114/136);
PKG+: edu.gatech.seclass.replace classes: 100% (2/2); branches:
72% (55/76); lines:  84% (114/136);
CLS+: Main  branches:  69% (36/52); lines:  84% (59/70);

CLS+: Replace  branches:  79% (19/24); lines:  83% (55/66);
```

- If some of these steps do not work as expected, please post a public question on Piazza, as other student may have solved similar issues and may be able to help.

7. Copy your latest version of MyMainTest.java, the one you created for Task 1, to .../D3Task2/testsrc/edu/.../replace/MyMainTest.java, thus replacing the placeholder file currently there.
8. Run your test suite by executing compileAndRunTests (.sh or .bat, depending on your platform). If one of your test cases fails, it could be for one of several reasons:
   - Your test case hits a corner case that is not defined in the requirements and for which your version of replace makes different assumptions than the version we provided. You should (1) modify the test case (**in this copy of MyMainTest.java only**) so that it passes and (2) add the following comment right before the @test annotation for that test: "// Type a".

- Your test behaves incorrectly, which means that there is a bug in your version of `replace` that was not caught by our test suite (class `MainTestAddOn`). You should (1) modify the test case (**in this copy of `MyMainTest.java` only**) so that it passes and (2) add the following comment right before the `@test` annotation for that test: "// Type b". **Note**: you will not be penalized for this, so there is no need to fix the bug; we are actually interested in seeing what kind of bugs our test suite missed in your code.
- Your test triggered a bug in our version of `replace` and caused a failure: good job! In this case, you should leave the test case as is but add the following comment right before the `@test` annotation for that test: "// Type c: <short explanation of the failure and what you think it's the corresponding bug>". Do not worry too much about the explanation and just make your best guess (e.g., "// Type c: replace fails when passed more than 10 from-to pairs, probably due to the storing of the pairs in a fixed-size array), as you will not be penalized for getting the explanation wrong. **Note**: the fixed-size array example is not a hint...

9. Generate the coverage report for your test suite by running `generateReport` (`.sh` or `.bat`, depending on your platform).
10. Save this report as `report-initial.txt` under directory `.../D3Task2`.
11. If your **line coverage** of both class `Main` and class `Replace` is already 100%, and you found at least one bug in the version of `replace` we provided, you are done.
12. Otherwise, you should add test cases to your test suite (`.../D3Task2/testsrc/edu/.../replace/MyMainTest.java`) to try to (1) get to 100% **line coverage** for both class `Main` and class `Replace` and/or (2) make our version of `replace` fail at least once. If one of the tests you add does cause a failure in our version of `replace`, you should add the following comment right before the `@test` annotation for that test: "// Type d: <short explanation of the failure and what you think it's the corresponding bug>".
    Note: To extend your test suite, you can either use test cases from the test suite we provided (`MainTestAddOn`) or add new tests. The advantage of adding new tests is that our tests will not reveal any bug in our code.
13. Task 2 will be completed when either your tests will achieve 100% line coverage and cause at least a failure (both goals are possible) or you are stuck and cannot achieve one or both of these goals. (In this latter case, you will still get partial credit depending on how much you accomplished.)

14. Generate the coverage report for the final version of your test suite and save it as `report-final.txt` under directory `.../D3Task2`.
15. Commit and push the following files:
    - `.../D3Task2/testsrc/edu/gatech/seclass/replace/MyMainTest.java`
    - `.../D3Task2/report-initial.txt`
    - `.../D3Task2/report-final.txt`(unless you didn't have to modify your test suite, so you only have the initial report)
    - There is no need to push any other file, as none of the other existing files under `D3Task2` is supposed to be modified.
16. Paste this last commit ID on T-Square, as usual. Although you should make sure you paste the commit ID for Task 2 and not Task 1 as your submission, we request that you commit and push also your solution to Task 1, as an intermediate step and as we discussed above.