Individual Project:
Replace Utility -- Deliverable 1

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*, which is the same utility for which you developed test frames in the category-partition assignment. A concise specification for the `replace` utility was provided with that assignment and is repeated here for your convenience:

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.
- SYNOPSIS
  `replace OPT <from> <to> -- <filename> [<filename>]*`

  where `OPT` can be zero or more of
  - `-b`
  - `-f`
  - `-l`
  - `-i`
- COMMAND-LINE ARGUMENTS AND OPTIONS

  `from`: 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.

  `-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.

You must develop the `replace` utility using a test-driven development approach such as the one we saw in lesson. Specifically, you will perform three activities within this project:

- For Deliverable 1, you will generate 40 JUnit test cases for `replace`. At least 10 of the test cases that you generate should be instantiations of the test frames that you created for the category-partition assignment. (Each test frame is a test spec that can be instantiated as a concrete test case.) To generate the remaining 30 tests, you can (1) leverage additional test frames, (2) generate the test cases from scratch, or (3) do a combination of these two approaches (i.e., implement some of the test frames and create some new test cases).
- For Deliverable 2, you will develop the actual `replace` utility and make sure that all of the test cases that you developed for Deliverable 1, plus some test cases added by us, pass.
- The details of Deliverable 3 will be revealed in due time.

Deliverables:

DELIVERABLE 1 (this deliverable)

- **provided:**
    - Skeleton of main class
    - Initial set of JUnit test cases
- **expected:**
    - 40 or more **additional** JUnit test cases

DELIVERABLE 2

- **provided:**
    - Additional set of JUnit test cases (to be run in addition to yours)
- **expected:**
    - Implementation of replace that makes **all** test cases pass

DELIVERABLE 3

- **provided:** TBD
- **expected:** TBD

Deliverable 1: Instructions

1. Download archive individualProject-d1.zip
2. Unpack the archive in the root directory of the **personal GitHub repository that we assigned to you**. After unpacking, you should see the following structure:
    - `<root>/IndividualProject/replace/src/edu/qc/seclass/replace/Main.java`
      This is a skeleton of the `Main` class of the `replace` utility, which is provided so that the test cases for `replace` can be compiled and run. It contains an empty `main` method (that you will have to complete **for Deliverable 2**) and a method `usage`, which prints on standard error a usage message and should be called when the program is invoked incorrectly. In case you wonder, this method is provided so that the error message is consistent across implementations.
    - `<root>/IndividualProject/replace/test/edu/qc/seclass/replace/MainTest.java`
      This is a test class with an initial set of test cases for the `replace` utility (which must all pass, **unmodified**, on the implementation you will create **for Deliverable 2**).
        - The first three of these test cases correspond to the examples of usage of `replace` that we provided.
        - The remaining three are additional tests provided to further clarify the behavior of `replace`.
        - Test case `mainTest5`, in particular, shows how you can use `replace` to substitute strings that begin with a dash ("`-`"). To do so, we followed POSIX's utility argument syntax (see Guideline 10), which states the following:
          "*The first -- argument that is not an option-argument should be accepted as a delimiter indicating the end of options. Any following arguments should be treated as operands, even if they begin with the '-' character.*"
          Because `replace` also uses "`--`" to indicate when the list of filenames begins, there can be multiple occurrences of "`--`" with different meanings in a single command (as shown in `mainTest5`), which is an interesting case.
    - In addition to providing this initial set of tests, class `MainTest` also provides some utility methods that you can leverage/adapt and that may help you implement your own test cases:
        - `File createTmpFile()`
          Creates a `File` object for a new temporary file in a platform independent way.

        - `File createInputFile*()`
          Examples of how to create, leveraging method `createTmpFile`, input files with a given content as inputs for your test cases.
        - `String getFileContent(String filename)`
          Returns a `String` object with the content of the specified file, which is useful for checking the outcome of a test case.
3. Generate 40 or more JUnit test cases for the `replace` utility (in addition to the ones we provided) and put them in a test class called `MyMainTest`.
    - Each test case should contain a concise comment that states whether the test case (1) is created from scratch, and its main purpose, or (2) is the implementation of one of the test frames you created for the category-partition assignment, and which one. (As stated above, at least 10 test cases must be implementations of test frames.) Use the following format in the two cases:
        - Newly created test case. Purpose: <*concise description of the main purpose of the test (e.g., Testing that the -f and -l parameters work when used together)*>
        - Implementation of test frame #<*test case number in the* `catpart.txt.tsl` *you submitted as part of the category-partition assignment*>
    - Just like class `MainTest`, class `MyMainTest` should be in package `edu.qc.seclass.replace`, under directory `test`.
    - Feel free to reuse and adapt, when creating your test cases, some of the code we provided in the `MainTest` class (**without copying the provided test cases verbatim, of course**). But feel also free to implement your test cases differently. Basically, class `MainTest` is provided for your convenience and to help you get started.
    - Whether you leverage class `MainTest` or not, your test cases should assume (just like the test cases in `MainTest` do) that the `replace` utility will be executed from the command line, as follows:
      `java -cp <classpath> edu.qc.seclass.replace.Main <arguments>`
4. Important: **Do not implement the `replace` utility for this deliverable**. Doing so would defeat the purpose of the project.
5. Commit and push your code. You should commit, at a minimum, the content of directory `IndividualProject/replace/test`. As for previous assignments and projects, committing the whole IntelliJ IDEA or Eclipse project, in case you used one of those IDEs, is fine.
6. Paste the commit ID for your submission on Blackboard, as usual.