Individual Project:
Replace Utility -- Deliverable 1

Project 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* in a file and replaces it with string *to*.
- SYNOPSIS
  `replace OPT <from> <to> <filename>`

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

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

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

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

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

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

`-w`: if specified, the `replace` utility performs the search for string `from` only matching whole words or phrases that are whitespace delimited.

- EXAMPLES OF USAGE

  Example 1:
  `replace -i Howdy Hello file1.txt`
  would replace all occurrences of "Howdy" with "Hello" in the file specified. Because the "`-i`" option is specified, occurrences of "howdy", "HOwdy", "HOWDY", and so on would also be replaced.

  Example 2:
  `replace -w -f Bill William file1.txt`
  would replace the first whitespace delimited occurrence of "Bill" with "William" in the file specified. It would not replace non-whitespace delimited occurrences, such as "Billy".

  Example 3:
  `replace -w "Bill is" "William is" file1.txt`
  would replace all whitespace delimited occurrences of "Bill is" with "William is" in the file specified. It would not replace non-whitespace delimited strings, such as the occurrence of "Bill is" in the text "Bill is, in my opinion, ...".

  Example 4:
  `replace -w -i abc ABC file1.txt`
  would replace all whitespace delimited occurrences of "abc", where the letters in "abc" can be either uppercase or lowercase, with "ABC" in the file specified. This would include "Abc", or "aBc", but not "abc!" or "Abc's".

You must develop the `replace` utility using a test-driven development approach such as the one we saw in P4L4. Specifically, you will perform three activities within this project:

- For Deliverable 1, you will extend the set of test cases that you created for Assignment 6; that is, you will use your test specifications to prepare 15 additional JUnit tests (i.e., in addition to the 15 you developed for Assignment 6). Then, you will implement the actual `replace` utility and make sure that all of the test cases that you developed for Assignment 6 and Deliverable 1, plus the initial test cases provided by us, pass.
- Deliverable 2 will continue the test-driven development approach.
- Deliverable 3 will be revealed in due time.

Deliverables:

<u>DELIVERABLE 1</u> (this deliverable)

- **Provided (in Assignment 6):**
    - Skeleton of the main class for `replace`
    - Initial set of JUnit test cases (yours + ours)
    - Your A6 submission
- **expected:**
    - 15 **additional** JUnit test cases
    - Implementation of replace that makes **all** test cases pass

<u>DELIVERABLE 2</u>

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

<u>DELIVERABLE 3</u>

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

Deliverable 1: Instructions

Setup

1. In the root directory of the **personal GitHub repository that we assigned to you**, create a directory called "IndividualProject". Hereafter, we will refer to this directory as `<dir>`.
2. Copy, from your Assignment6 directory, the files in the `src` and `test` directories to a corresponding directory in `<dir>`:
    - `<dir>/replace/src/edu/gatech/seclass/replace/Main.java`
      This is the skeleton of the `Main` class of the `replace` utility, which you will have to complete for Deliverable 1 **after creating the additional test cases**. Make sure to keep the `usage` method unmodified, as it ensures that the error message is consistent across implementations.
    - `<dir>/replace/test/edu/gatech/seclass/replace/MainTest.java`
      This initial set of test cases for the `replace` utility must all pass, **unmodified**, on the implementation you create. The first four test cases correspond to the examples of usage of `replace` that we provided. The fifth shows an example of invoking the `usage` message for an invalid invocation.

  As you did for Assignment 6, you can leverage/adapt the utility methods in class `MainTest` to implement your own test cases.
- `<dir>/replace/test/edu/gatech/seclass/replace/MyMainTest.java`
  The test class with your original 15 test cases, to which you will add the additional 15 for Deliverable 1.

Test generation

3. Generate 15 or more JUnit test cases for the `replace` utility (in addition to the ones we provided and your original 15 from Assignment 6) and put them in class `MyMainTest`.
   - As you did for Assignment 6, you should add a concise comment to each test that states the test case's purpose and which test frame it implements,using the following format:

     ```
     // Purpose: <concise description of the purpose of the test>
     // Frame #: <test case number in the catpart.txt.tsl of A6>
     ```

   - As a reminder, you can 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.

Implementation

4. Implement the `replace` utility by completing the `Main` class, whose skeleton we provided in Assignment 6, and adding any other class you may need. The only requirements for your code are that it passes all the test cases that you created and the ones that we provided, and that it can be executed as follows:
   ```
   java -cp <classpath> edu.gatech.seclass.replace.Main <arguments>
   ```

Submission

5. Commit and push your code. You should commit, at a minimum, the content of directory `<dir>/replace/test and <dir>/replace/src`. As for previous assignments and projects, committing the whole IntelliJ IDEA project, in case you used this IDE, is fine.
6. Paste the commit ID for your submission on T-Square, as usual.