# Individual Project:
# Encode Utility -- Deliverable 1

## Project Goals

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

## Concise Specification of the `encode` Utility

---

- NAME:

  `encode` - encodes words in a file.

- SYNOPSIS

  `encode OPT <filename>`

  where `OPT` can be **zero or more** of
    - `-c <integer>`
    - `-r`
    - `-d <characters>`

- COMMAND-LINE ARGUMENTS
AND OPTIONS

  `<filename>`: the file on which the encode operation has to be performed.

  `-c <integer>`: if specified, the utility will apply a caesar cipher to all alphabetic characters (other characters remain unchanged and the original capitalization is preserved), incrementing letters by the integer passed in.

  `-r`: if specified, the utility reverse the order of characters in every word, with a word being any set of non-whitespace characters, including non-alphabetic characters, separated from others by any whitespace character.

`-d <characters>`: if specified, the `encode` utility will remove all characters from the file which match any of the characters passed into this option, before applying any other transformation.

If **none** of the OPT flags is specified, `encode` will default applying -c <length of file>, a caesar cipher incrementing by the number of characters in the input file.

- NOTES
  - While the last command-line parameter provided is always treated as the filename, OPT flags can be provided in any order (-d is always applied first).

- EXAMPLES OF USAGE

  **Example 1:**
  `encode file1.txt`
  (where the content of the file is "abcxyz")
  Increments all letters by the length of the input file, 6 (resulting in "ghidef").

  **Example 2:**
  `encode -r file1.txt`
  Reverses the characters in every whitespace delimited word.

  **Example 3:**
  `encode -d "aeiou" -c 3 file1.txt`
  Removes all 'a', 'e', 'i', 'o' and 'u' characters, then increments all remaining letters by 3.

  **Example 4:**
  `encode -c 2 file1.txt`
  Increments all letters by 2.

---

You must develop the `encode` utility using a test-driven development approach such as the one we saw in P4L4. Specifically, you will perform two 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 `encode` utility and make sure that all of the test cases that you developed for Assignment 6 and Deliverable 1, plus the test cases provided by us, pass.
- Deliverable 2 will be revealed in due time.

## Deliverables:

**DELIVERABLE 1 (this deliverable)**

- **Provided (in Assignment 6):**
  - Skeleton of the main class for `encode` (`Main.java` we provided in A6)
  - Initial set of JUnit test cases
    - Tests we provided for A6 (`MainTest.java`)
    - Tests you created for A6 (`MyMainTest.java`)
- **expected:**
  - 15 **additional** JUnit test cases
  - Implementation of encode that makes **all** test cases pass

**DELIVERABLE 2**

- **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>/encode/src/edu/gatech/seclass/encode/Main.java`
     This is the skeleton of the `Main` class of the `encode` 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>/encode/test/edu/gatech/seclass/encode/MainTest.java`
     This initial set of test cases for the `encode` utility must all pass, **unmodified**, on the implementation you create.

   - `<dir>/encode/test/edu/gatech/seclass/encode/MyMainTest.java`
     The test class with your original 15 test cases, to which you will add the additional 15 you create for Deliverable 1.

## Test generation

3. Generate 15 or more JUnit test cases for the `encode` 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, such as the utility methods, which we provided in the `MainTest` class (**without copying the provided test cases verbatim, of course**). You should also feel also free to implement your test cases differently.

## Implementation

4. Implement the `encode` utility by completing the `Main` class, whose skeleton we provided in Assignment 6, and adding any other class you may need. The 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.encode.Main <arguments>
   ```

## Submission

5. Commit and push your code. You should commit, at a minimum, the content of directory `<dir>/encode/test and <dir>/encode/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 Canvas, as usual.