(50 pts) due: February 15, 2017 11:59pm

Important Notes

- **This assignment is to be done on your own.** If you end up using code you find on the Internet, you must disclose the origin of the code. **Concealing the origin of a piece of code is plagiarism.** If you need help, see the instructor.
- Please start the assignment as soon as possible and get your questions answered early.
- Read through this specification completely before you start.
- Some aspects of this specification are subject to change, in response to issues detected by students or the instructor.

Project Details

The goal of this project is to evaluate your Java programming skills to build the following text processing methods/functions.

1. Write a method `tokenize` that reads in a text file and returns a list of the tokens in that file. For the purposes of this project, a token is a sequence of alphanumeric characters, independent of capitalization. That is, *TCNJ* and *tcnj* are the same token.
2. Write a method `print` that takes a list of tokens as a parameter and prints out them onto the screen.
3. Write a method `computeWordFrequencies` that takes a list of tokens as a parameter, counts the total number of words and their frequencies in the list, and create a list of pairs, each of which contains a token and its frequency.
4. Write a method `print` that prints out the word frequency counts onto the screen. The result should be ordered by decreasing frequency.
5. Write a method `compute3GramFrequencies` that counts the total number of 3-grams and their frequencies in a token list. A 3-gram is three words that occur consecutively in a file. For example, in the sentence "April is the cruellest month", there are three 3-grams and they are: "April is the", "is the cruellest", and "the cruellest month".
6. Write a method that prints out the 3-gram frequency counts onto the screen.
7. Write a main method that
   - reads a text file, tokenizes it, counts the tokens and prints out the token frequencies.
   - reads a text file, tokenizes it, counts the 3-grams and prints out the 3-gram frequencies.

Part of the code skeleton is shown as follows:

```
public class Project1 {
  ......
  public static void main(String[] args) {
    ......
  }
}

public class Token {
  private String token;
  private int frequency;
  ......
  public String toString();
}

public class ThreeGram{
  private String 3gram;
  private int frequency;
  ......
  public String toString();
}
```

README files

Your README file for each project should contain your name, the project name, and all information you think is necessary for a typical user to run your program and for a software maintainer to understand and evaluate your code. Much of this information may be included on the original project description and you are responsible for filling in the parts not provided for you. More specifically:

- Information for the prospective user includes details on calling conventions, input and output data formats, limitations, bugs, and special features.
- Explain both the negative aspects of your program (limitations, known bugs) and the positive aspects (extensions, special features). Note that if you do not include the former, I will assume you did not realize the limitations were there.
- For the person who has to understand the insides of your code, you may need to describe your choice of modularization (abstractions), data structures, and algorithms. Be sure to explain anything you did that is likely to be different from what other students may have done, and justify any design decisions for which the rationale isn't immediately clear.

Your write-up should be coherently written, using full sentences and paragraphs. This write-up need not necessarily be long, but it is important. All write-ups must be in plain text (README.txt) or Adobe PDF (README.pdf) format. Other formats (e.g., Word or postscript) will not be accepted.

What to turn in:

JAR your *.java files and README into a file called Project1.jar. Upload the jar file to Canvas under category Project1.

Late Submission Policy:

-1% for every hour past the deadline for the first 24 hours.
Flat -25% until 7 days late.
No submission accepted past one week after the due date.

Grading scale

The project will be graded based on your source code and the README file:

- Code (correctness, completeness and efficiency) (80%): Your code should implement everything that was required in the assignment. It should produce the right output given normal, expected inputs, and some sort of reasonable response to unexpected inputs. Unless otherwise instructed, and as long as it does not severely compromise programming style, you are expected to use the most efficient data structures and algorithms.
- Documentation (20%)
  - Programming style (including internal documentation and program organization) (5%): Your code should have appropriate abstractions, data structures, algorithms, and variable names; declarations for all constants; and a judicious number of helpful comments. You should think of programming as explaining to the readers of your programs what you want the computer to do.
  - Completeness of README file (10%): Your write-up should include all the items discussed in the README files.
  - Readability of README file (5%): Your write-up should be well organized, clear, and concisely presented.