## Lab 8

In lab 8, you will finish a program to count the number of lines, tokens, characters, and bytes in a file. You will need to handle exceptions as well.

The name of your Eclipse project should be abc123-lab8, where you replace abc123 with your abc123 id. The name of the file that contains the main method should remain Lab8.java. The remaining Java files should also keep their original names. To submit the project, export the project and upload the zip file to Blackboard.

### Task

Your task to modify the code in [lab8.zip](http://cs.utsa.edu/~cs3443/laboratories/lab8.zip) so that the user can select a file, the program counts the number of words, tokens, characters, and bytes in the file, and this information is printed. You can use [this file](http://cs.utsa.edu/~cs3443/laboratories/Language%20Learning%20and%20Teaching%20(more%20or%20less)%20in%2016%20or%20more%20languages.htm) as an example where the number of characters should differ from the number of bytes.

Lab8.java contains a main method and a getFileFromUser method. The getFileFromUser method gets a file name from the user using JFileChooser. THis method will throw a FileNotFoundException in some cases.

The main method of Lab8.java contains code with some problems. One problem is that no exceptions are handled. Another problem is that the code should continue to run until a file is successfully processed.

The FileCounts class is set up to store a File object. Each of the methods lineCount, tokenCount, charCount, and byteCount should:

* Open the file using the appropriate java.io class.
* Count the number of lines/tokens/chars/bytes and detect end of file using the appropriate methods of the class.
* Close the file and return the count.

For counting lines, use the Scanner class and its hasNextLine and nextLine methods.

For counting tokens (words), use the Scanner class and its hasNext and next methods.

For counting characters, use the FileReader class and its read method. Read the documentation for how you detect the end of file using theread method. To be more efficient in reading large files, you should also use the BufferedReader class.

For counting bytes, use the FileInputStream class and its read method. To be more efficient in reading large files, you should also use the BufferedInputStream class.

In all of the above, you will need to add code to handle exceptions. The methods in FileCounts should pass any exceptions to the mainmethod. The main method should keep trying to print out the information in a file selected by the user until no exception occurs.

### Comments

Add javadoc comments as needed for any additions or changes. Note that the initial version of FileCounts.java has no javadoc comments.

### UML Class Diagram

Create a UML class diagram using Violet.

### Rubric

* An incorrect submission will possibly get zero points. A project that does not compile will receive at most 50 points total.
* (40 pts.) If the user can select a file and the information about that file is correctly printed out.
* (40 pts.) If the program correctly handles exceptions. The response to an exception should be to print out the exception, and to start over again asking the user for a file. A couple of ways to generate exceptions is to select a directory rather than a file, or to type in the name of a file that doesn't exist.
* (10 pts.) If there are javadoc comments: at least one for each class and comments for each method, constructor and constant.
* (10 pts.) A UML class diagram is included that has all the classes (including the Lab8.java class) and all the constructors, methods, variables and constants of each class.