Assignment 1

Thu Jan 28 (125 points)

Prerequisite

First, set up your Git account Java development environment by following the instructions on the software setup page.

Task

Suppose we want to compile a language with a comment syntax similar to C/C++. A comment begins with the two characters `(*` and ends with the characters `*)`. Those matching sequences and any characters between them should be ignored. Comments can contain newlines. They may not be nested. A program that ends with an unterminated comment is an error.

You should create a new Java project called `assn1` in a fresh sub-directory of your Git repository. Within the project, create a class `CommentLexer` with a method `stripComments` defined like this:

```
public static String stripComments(String source) {
  // TODO...
}
```

The objective of `stripComments` is to remove the comments (as described above) from the source string, and return a new string containing only the non-comment portions of the source string. For example, `stripComments("Hello(*cruel*)World")` should return `"HelloWorld"`.

You may want to think about the procedure in terms of a **finite state automaton.** What are the possible states and transitions? Are there any actions you need to take on certain transitions?

To submit your program, make sure the files in your `assn1` directory are added to Git VCS, then commit and push. Some helpful hints follow.

Tips

- I have set up a sample project at https://git.liucs.net/cs664s16/cs664pub/tree/master/assn1 (look in the `src` sub-directory). In addition to a template for the `CommentLexer` class, I provide a substantial test suite in `CommentLexerTest` and a class you can use as an exception to throw in the case of an unterminated

comment. To try it out, you can clone the entire `cs664pub` project into your system (just like you did for your own project in the software setup) – just don't do it *within* your own Git repository, save it somewhere else. The repository URL is `git@git.liucs.net:cs664s16/cs664pub.git`

- The test suite uses jUnit 4. It's very easy to add this library to an IntelliJ project. Go to **File** » **Project Structure** and select **Libraries** from the left panel. Then hit the green plus sign in the next column and select **From Maven.** In the search box, type `junit:junit:4.12` and hit the search icon. Once it finds it, you can press **OK.** Confirm that you want to incorporate the library into your `assn1` module.
- To run the test suite, select **Run** » **Run...** and it should automatically show `CommentLexerTest` as an option. Once it's configured, you can just use **Run** » **Run CommentLexerTest** or the green Play button.
- The best way to build up a string incrementally in Java is to use `StringBuilder`. So here is an example of an implementation that keeps only the odd-numbered characters in the `source` string. So "banana" would become "aaa". It won't pass the tests, but it does show the correct usage of `StringBuilder`.

```
public static String stripComments(String source) {
    StringBuilder buffer = new StringBuilder();
    for(int i = 0; i < source.length(); i++) {
        char current = source.charAt(i);
        if(i % 2 == 1) { // odd-numbered characters
            buffer.append(current);
        }
    }
    return buffer.toString(); // convert builder to regular string
}
```

If you try to use this implementation in the test suite, the failures will show you what it does:

```
org.junit.ComparisonFailure:
Expected :This (has balanced) parens
Actual   :hs(a aacd aes
```

- The example from the language theory notes on how to implement an FSA in Java might also be helpful.