For this part of the assignment, you will design, implement and test a Java class CompilerIO that will handle input and output for the compiler.

The idea behind this class is that the compiler will use a single CompilerIO object to manage input and output, which will shield the rest of the compiler from the details of file handling. The input and output files are ordinary text files.

CompilerIO constructors

When the compiler starts, it creates an instance of the CompilerIO class. The CompilerIO constructor opens the input and output files and stores references to them for later use.

CompilerIO is required to have two constructors: CompilerIO(String in) and CompilerIO(String in, String out). If an exception occurs while opening the files, the constructors should throw the exception and let the caller handle it.

The parameters for the two-argument constructor are the names of the input and output files. The constructor opens both files. The input file name is used to create a BufferedReader object, and the output file name is used to create a PrintWriter object. References to these objects are stored in instance variables.

The parameter for the single-argument constructor is the name of the input file. The constructor creates the output file name by taking the input file name (say test.txt) and replacing the extension with ".asm" (test.asm). Don't make assumptions about the extension in the input file name. It might be ".txt", but it might also be ".stuff", or ".d" or something else. (Implementation hint: class String includes methods that locate the first or last occurrence of a character or substring in a String.) Once having created the output file name, the constructor opens both files to create a BufferedReader object and a PrintWriter object and stores references in the appropriate instance variables.

String readSrcLine() and **void printAsmLine(String s)**

The scanner will use the CompilerIO object's readSrcLine() method to read the source program text. The code generation parts of the compiler will use printAsmLine(s) to write the generated assembly code to the output file. Both readSrcLine() and printAsmLine(s) should throw an exception if an error occurs while reading or writing.

In addition to the generated assembly code, we will want to include in the output file the original source code as assembly-language comments to make the output easier to read. So, besides just reading the input file, readSrcLine() should be able to automatically print each input line to the output file as it is read, if requested by the user.

Instance variables `echoing` and `echoPrefix` should control whether the input lines are printed to the output file, and how. If `echoing` is true, then each input line should be written to the output file at the time it is read, with the `echoPrefix` string concatenated

to the front of it. The idea behind `echoPrefix` is that we can set it to something appropriate so the echoed source program lines copied to the assembly language output file are treated as comments in the generated program (more details in a future assignment). You should provide methods to get and set both of these properties (`void setEchoing(boolean b), boolean getEchoing(), void setEchoPrefix(String s), String getEchoPrefix()`).

public static void main(String[] arg)

Class CompilerIO should include its own test program (method main). The test program should accept zero, one or two file names as command line arguments and use the appropriate constructors to create a CompilerIO object. If no file name is specified, your main method can supply a default name (eg, CompilerIO.java). Once the files are open, the test program should verify that the input and output methods of CompilerIO work properly.