CIS 201 Computer Science I
Fall 2009 Lab 11
Reading Files

November 30, 2009

- Working with files (for input and output).
- Working with `String`s.
- Different kinds of loops.

In this lab you will write three increasingly difficult programs: the first prompts the user for the name of a file and displays the text file on the screen; the second prompts the user for two file names, one for input and one for output and copies the contents of the input file into the output file; the final program copies a file in a similar way but before writing a `String` to the output file, the string is *enciphered* so that the contents of the copied file are secure from prying eyes.[1]

Checkpoint 1 ViewFile.java

Write a non-FANG program, `ViewFile.java`. Put the file in an appropriately named folder and so on.
The program should

- Prompt the user for an input file name and read in the file name from standard input. (Consider that you must prompt the user for multiple file names soon; how can you keep from repeating yourself?)
- Open the file for input.
- If there was a problem opening the file for input, provide a useful error message to the user. Be sure to include a description of the problem and the name of the offending file. Terminate the program after printing the error message.
- If the file opened properly, copy the lines of the file to standard output.
- You can test your program by viewing the `.java` file of the program itself.

Show your work on Checkpoint 1 to the lab monitor, answering any necessary questions for them. Have them sign before continuing.

Checkpoint 2 CopyFile.java

Starting with a copy of `ViewFile.java`, change the name to `CopyFile` and make it compile.

[1] For a sufficiently narrow definition of "prying eyes."

`System.out` is a `PrintStream`. You want to construct a new `PrintStream` associated with an output file. It is similar to associating a `Scanner` with an input file **except** that the `PrintStream` constructor takes the name of a file (not a `File` object) and you *must* `close` the file when you are done with it. The following loop gives an example of using a `PrintStream` for output.

```
String fname = /* get the file name somehow */;
PrintStream ps = null;
try {
   ps = new PrintStream(fname);
   ps.println("Hello, World!");
} catch (FileNotFoundException fnf) {
  System.err.println("There was a problem with " + fname);
  System.exit(1);
} finally {
  ps.close();
}
```

Your modified program should

- Prompt the user for an input file name and read in the file name from standard input. (Consider that you must prompt the user for multiple file names soon; how can you keep from repeating yourself?)
- Open the file for input.
- If there was a problem opening the file for input, provide a useful error message to the user. Be sure to include a description of the problem and the name of the offending file. Terminate the program after printing the error message.
- Prompt the user for an output file name and read the file name from standard input.
- Open the file for output.
- If there was a problem, give a reasonable error message and terminate the program.
- If both files opened properly, copy the lines of the input file to the output file.
- You can test your program by viewing the `.java` file of the program itself.

Show your work on Checkpoint 2 to the lab monitor, answering any necessary questions for them. Have them sign before continuing.

Checkpoint 3 EncipherFile.java

Starting with `CopyFile.java`, you should rename it to `EncipherFile.java`. There will be no visible change from the user's viewpoint: the program will prompt for two file names, one for input, one for output. It then reads the lines of the input file, enciphers each line character by character, and then prints out the enciphered line.

The guts of the program looks like this

```
while (there are more lines) {
  String original = nextLine;
  String converted = "";
  while (there are characters  in original) {
    char nextCh = nextChar;
    add nextCh to end of converted;
  }
```

```
  print out converted;
}
```

How do you loop across all the characters in a `String`?

It is a count-controlled loop based on the `length()` method of `String` and `charAt(i)` (look these methods up in the `String` documentation).

Modify `EncipherFile` so that its main loop converts each line before printing. The following loop does the real work:

```
// inside the hasNextLine loop...
String converted = "";
for (int characterNdx = 0;
     characterNdx < orginal.length();
     ++characterNdx) {
  char nextCh = orginal.charAt(characterNdx);
  // this is where you convert nextCh...
  converted = converted + Character.toString(nextCh);
}
// now print out converted
```

You should modify this code so that every semicolon (;) is replaced by a caret (^).

You can test the program by running it on its own code. You can then open the copy and see if the semicolons are all replaced with the right character (and that the rest of lines are unchanged).

Show your work on Checkpoint 3 to the lab monitor, answering any necessary questions for them. Have them sign before continuing.

We leave the program here; it is possible that next week you will extend the enciphering of the characters so that it uses the Caesar cipher to encipher the text read from the file. It is also possible that we need to work some more on sorting so that may be next week's lab.

Log off of the lab computer you are using before leaving they lab. Anyone entering the lab has unlimited access to your files if you remain logged on. **DO NOT** turn off lab computers! They are a shared resource and there might be someone else logged in to "your" machine.