CMSC 113: Computer Science I
Lab #7: Reading Files

Part I. Learning about files

To start off this lab, we'll be experimenting with two sample programs that read information from files.

Though we work with files every day, it can be easy not to really understand how they work. A *file* is a sequence of bytes with a filename and a location. (A byte is a sequence of 8 bits, where a bit [or *b*inary dig*it*] is a 0 or a 1.) These bytes are raw data – they have no meaning on their own but can only be given meaning when interpreted by some computer program. For example, a Microsoft Word document is a sequence of bytes that Word understands as text and formatting commands; without Word installed, the file would be gibberish.

Files also have a filename and a location. The file's location is a directory (or folder) in your computer; every directory has a unique *path* that describes how to find the directory. Paths in Windows begin with something like `C:\`; on Macs, they begin with something like `/Users/`. A file's filename is usually comprised of a base name and an extension. For example, this file's base name is *Lab7*; its extension is *.pdf*. File extensions are hints to the computer for how to process the file. When you try to open a file ending in *.pdf*, the computer knows to look for a PDF reader. Changing the extension *does not* change the contents of a file, which is why computers often warn about changing extensions. For example, changing *Lab7.pdf* to be named *Lab7.doc* wouldn't make it a Word document at all. Word would just get very confused opening the file and finding the wrong sequence of bytes there.

The files we will be concerned with are *text files*, where each byte encodes one character. (Depending on your computer settings, each character might be encoded with *two* bytes, in order to allow for non-Latin characters. Happily, Java makes the subtleties around character encoding irrelevant for the simple programs we will write.) Files ending with *.txt* are text files, as are many others, including *.java* files. Java provides good support for reading text files.

1. Create a new project in Eclipse for this lab.

2. From the syllabus page for our course, download the *ReadFile.java*, *Multiplier.java*, and *numbers.txt* files. Place them in the *src* folder for the project you just created.

3. Look at the *ReadFile.java* program. It demonstrates the code for reading a file and printing out the contents of the file for the user.

4. Run this program. It will ask for the name of a file. Enter `ReadFile.java`. The program will then read its own source code and print it out. (If you're fascinated by programs that print out their own code, look up *quine* on the internet. This program is not a proper quine because it requires file system access.)

5. Try running the *ReadFile* program but enter in a file that doesn't exist. What happens?

6. The *Multiplier.java* program reads in a file containing numbers and multiplies them all together. Try running it on *numbers.txt*. Try altering the contents of *numbers.txt* and running the multiplier program again. See how the multiplier program works by making good use of the features of `Scanner` to read numbers.

Part II. Writing programs

7. Write a program that reads from a file that contains a location (an *x*- and a *y*-coordinate) and a name. It should then make a label saying *Hello,* and that name.

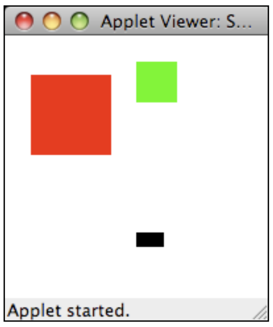
   Consider a file containing the following text:
   ```
   20 50
   Priscilla
   ```

   The program reading from this file would make a label saying *Hello, Priscilla* at the location (20, 50). You will need to create the file for your program to read. This is possible from Eclipse's *New*... dialog box (in the *File* menu).

8. Write a program that draws three rectangles of different colors on the applet. The locations and colors of the rectangles will be written in a file. Here is a sample file:

   ```
   20 30 60 60 255 0 0
   100 20 30 30 0 255 0
   100 150 20 10 0 0 0
   ```

   This would produce one rectangle with its upper-left corner at (20, 30), width and height of 60, and colored red. (The color numbers are 255, 0, and 0.) The next rectangle would be at location (100, 20), it would be 30 by 30, and colored green. The last rectangle would be at location (100, 150), it would be 20 pixels wide and 10 pixels tall, and it would be colored black.



9. Change your program in problem above to allow an arbitrary number of rectangles. You may assume that the file contains only complete descriptions of rectangles. (That is, the number of numbers in the file is a multiple of 7.)