Introduction to IO: files, directories, paths

Java packages and methods for I/O

Initially, the main package of classes and interfaces for IO was package `java.io`. But with release Java SE 7 many years ago, a newer package `java.nio` was released. And, instead of using class `java.io.File` for working with files and directories, the Java people advocated using the newer one:

	Old package	**Package since version 7**
packages	java.io	java.nio
Dealing with files:	java.io.File	java.nio.file.Path, java.nio.file.Files

This webpage explains the reasons for preferring the newer package:

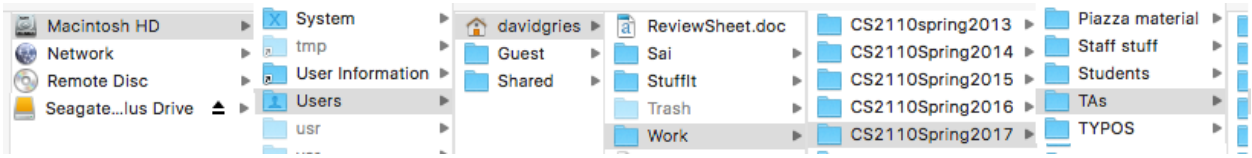
http://docs.oracle.com/javase/tutorial/essential/io/legacy.html

In our discussions of I/O, we will use the new package, but you may come across legacy code that uses the old package. When you see uses of `java.io.File`, read the Java API documentation to learn about the methods being used. The webpage listed above tells you about dealing with legacy code and how you could modify the legacy code to use the new package instead.

Files, directories, paths

The hard drive on your computer is structured as a tree —or perhaps a forest of trees. The leaves are either files or empty directories (also called folders). The internal nodes are directories. The image below displays part of the tree on one laptop in horizontal fashion. The root is named with the empty string (although it looks like it should be `Macintosh HD`). It has several children, mostly directories. Child `Users` is selected. It contains three directories, one of which is selected, and so on.

This image actually shows a forest of four trees. Besides the tree for the internal hard drive, there are trees with roots `Network`, `Remote Disc` (a portable CD unit), and `Seagate ...` (an attached external hard drive).



A path to a file or directory is simply the list of names from the root of the tree to that file or directory. In the above image, the path to directory Work is: `""`, `Users`, `davidgries`, `Work` (the first name is the empty string!)

Different operating systems use different delimiters to separate names on a path. Unix-based systems, including the Macintosh systems, use "/"; Microsoft windows systems use "\". Here are examples:

Unix operating system: `/Users/davidgries/Work`
Microsoft Windows system: `C:\home\sally\statusReport`

Absolute versus relative paths

The paths shown above are *absolute* paths; they show the path from a root of a file system to a node.

A *relative* path shows the path from some node other than the root. For example, here is a relative path in the image shown above:

`Work/CS2110Spring2017/Students`

Obviously, this relative path can't be used without having more information.

One can use relative paths in Eclipse projects. Consider the image to the right of Eclipse project `2017sp_a4`. The project contains directory `data`, which contains text file `names.txt`. Within classes in directory `src`, we can write code to read file `names.txt` using the relative path `data/names.txt`. By default, Eclipse assumes that directory `data` is in directory `2017sp_a4`, which it is.