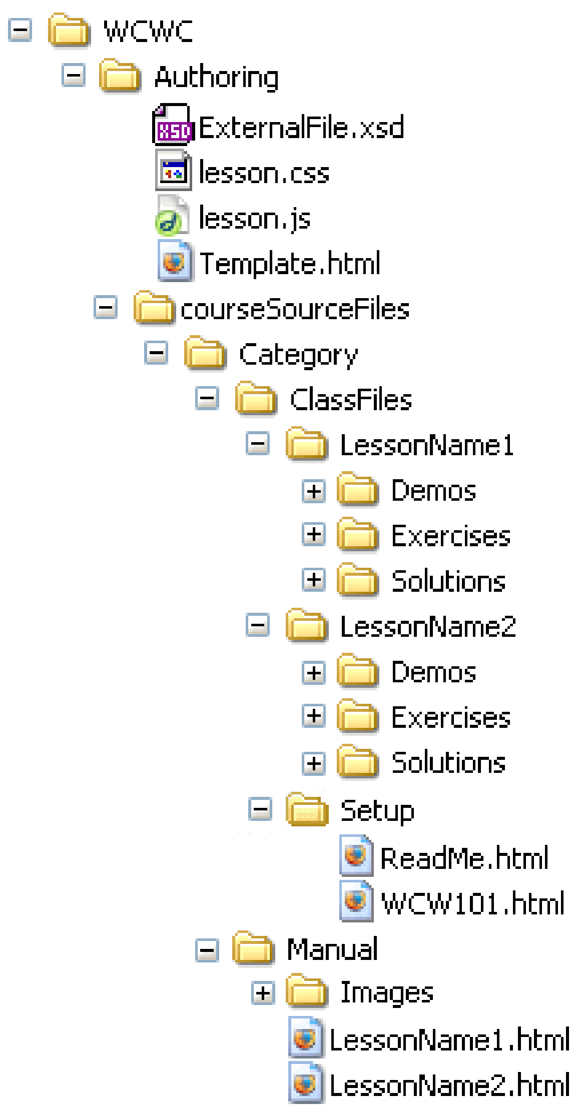
File Searching Homework
CSS 162 : Programming Methodology
Instructor Lesley Kalmin

Assignment by Rob Nash

Recursively Searching Files & Directories

Summary

Build a class and a driver for use in searching your computer's secondary storage (hard disk or flash memory) for a specific file from a set of files indicated by a starting path. Lets start by looking at a directory listing. Note that every element is either a file or a directory.



```
WCWC
    Authoring
        ExternalFile.xsd
        lesson.css
        lesson.js
        Template.html
      courseSourceFiles
          Category
              ClassFiles
                  LessonName1
                      Demos
                      Exercises
                      Solutions
                  LessonName2
                      Demos
                      Exercises
                      Solutions
                  Setup
                      ReadMe.html
                      WCW101.html
              Manual
                  Images
                  LessonName1.html
                  LessonName2.html
```

Introduction & Driver

In this assignment, your job is to write a class that searches through a file hierarchy (a tree) for a specified file. Your **FindFile** class will search a directory (and all subdirectories) for a target file name. For example, in the file hierarchy pictured above, the file "lesson.css" will be found once in a directory near the root or top-level drive name (e.g. "C:\") . Your **FindFile** class will start at the path indicated and will search each directory and subdirectory looking for a file match. Consider the following code that could help you build your **Driver**.java:

```
String targetFile = "lesson.css";
String pathToSearch ="C:\\WCWC";
FindFile finder = new FindFile(MAX_NUMBER_OF_FILES_TO_FIND);
Finder.directorySearch(targetFile, pathToSearch);
```

File Searching

In general, searching can take multiple forms depending on the structure and order of the set to search. If we can make promises about the data (this data is sorted, or deltas vary by no more than 10, etc.), then we can leverage those constraints to perform a more efficient search. Files in a file system are exposed to clients of the operating system and can be organized by filename, file creation date, size, and a number of other properties. We'll just be interested in the file names here, and we'll want perform a brute force (i.e., sequential) search of these files looking for a specific file. The way in which we'll get file information from the operating system will involve no ordering; as a result, a linear search is the best we can do. We'd like to search for a target file given a specified path and return the location of the file, if found. You should sketch out this logic linearly before attempting to tackle it recursively.

FindFile Class Interface:

- FindFile(int maxFiles)
  - This constructor accepts the maximum number of files to find.
- void directorySearch(String target, String dirName)
  - The parameters are the target file name to look for and the directory to start in.
- int getCount()
  - This accessor returns the number of matching files found
- String[] getFiles()
  - This getter returns the array of file locations, up to maxFiles in size.

Requirements

- Your program should be recursive.
- You should build and submit at least two files: **FindFile**.java,**Driver**.java,
- Throw an exception (IllegalArgumentException) if the path passed in as the starting directory is not a valid directory.
- Throw an exception if you've found the MAX_NUMBER_OF_FILES_TO_FIND, and catch and handle this in your main driver.
    - You're program shouldn't crash, but rather exit gracefully in the unusual situation that we've discovered the maximum number of files we were interested in, reporting each of the paths where the target files were found.
- The only structures you can use in this assignment are basic arrays and your Stack, Queue, or ArrayList from the previous homeworks.
    - Do not use built-in data structures like ArrayList.
    - To accomplish this, add the following constructor to your ArrayList, Stack, or Queue

        -
          ```
          public ArrayList(Object[] input) {
             data = input;
             numElements = input.length;
          }

          public Object get(int index) {
            return data[index];
          }
          ```

Notes and Hints

- Consider looking into the File class in Java for helpful methods like isDirectory() and toString().
    - http://docs.oracle.com/javase/7/docs/api/java/io/File.html
    - Useful File details have been taken from the link above and copied here for reference
        - import java.io.File;
            - Import the File class to use it.
        - File f = new File(dirName);
            - Next, create a File object.
        - String[] fileList = f.list()
            - This list files in current directory as Strings
        - File aFile = new File(dirName + "\\" + fileList[i]);
            - Notice the concatenation in the new File object
        - if (aFile.isDirectory()) {
            - check whether it is a directory
- Test your FindFile.java class.
    - Try to find a file that exists once
    - Try to find a file that doesn't exist
    - Try to find a file that exists twice

    - Try to find a file that exists MAX_NUMBER_OF_FILES_TO_FIND
    - …
- In your recursive call, make sure you don’t loose the directory path by concatenating it with the file name.