## Lake Washington Institute of Technology

## CSD 235/335 Algorithms and Data Structures

## Assignment 6 – Generalized Trees

### Introduction

The file system on most modern operating systems is organized as a generalized tree structure. The nodes of the tree are regular files and subdirectories. The subdirectories provide hierarchical structure, as they can contain other subdirectories as well as regular files.

For this assignment, write a Java program that recursively traverses the file system starting at a fully qualified subdirectory. The subdirectory must be provided as a command line parameter. When processing begins, the first line of output is the fully qualified subdirectory.

For each subdirectory encountered, get a list of the files in that subdirectory. For each entry in the list, if it is a regular file, output the file name and extension. Do not include the path name; just the file name and extension. If the file is a subdirectory, output just the subdirectory name; do not include the pathname that precedes the subdirectory name. In either case, after outputting the name, recursively descend into subdirectory and continue. You may recognized this as a pre-order traversal.

For each line of output, prefix the file or subdirectory name with indentation that provides a visual indication of the depth of the recursive descent.

### class FileTreeWalk

FileTreeWalk is a Java class that traverses the file system hierarchy. In addition to a constructor, the class provides two public methods for traversing the file system, and a public toString method for displaying the traversal results.

The public methods are as follows:

#### public FileTreeWalk (String pathname)

The constructor takes a String parameter that specifies a pathname. The pathname is a starting point for the traversal of the file system. The parameter is either a fully qualified path name, or a relative pathname. Fully qualified pathnames start with an optional drive letter, or a slash. On other operating systems, a fully qualified pathname starts with a slash. Relative pathnames start at the current working directory.

#### public void listAllFiles ()

This method outputs a string representation of the tree structure of the file system starting at the pathname specified in the constructor. Provided the argument to the constructor is the string, “C:\\Users\BigJoe\Documents”, the output might look like this:

C:\\Users\BigJoe\Documents

FileTreeWalk

Build

Classes

FileTreeWalk

FileTreeWalk.class

.netbeans\_automatic\_build

.netbeans\_update\_resources

nbproject

private

private.properties

build-impl.xml

henfiles.properties

project.properties

project.xml

src

filetreewalk

FileTreeWalk.java

test

build.xml

manifest.mf

assignment03.docx

Notice that the indentation reflects the position of the files within the files system hierarchy.

#### public String findFiles (String filename)

This method takes a string argument that specifies a file to search for. During the traversal if a file or subdirectory with the same name as the file name parameter is encountered, a string is added to the output that includes the full path name, file name, and extension.

#### public String toString ()

The traversal methods should build their output in a StringBuilder object. The toString() method should return a string representation of the string that was constructed during the traversal.

### Additional Note

Java 8 provides a class, java.nio.file.Files, that encapsulates all of the recursion needed to solve this assignment with no recursion in the calling code (code that you will write). Which is great. But this assignment is to help you better understand recursion. Do not use java.nio.file.Files for this assignment. If you do use it, you can pretty much count on receiving no credit for the assignment.

Prior to Java 8, the way you would approach this problem is to use java.io. File. Use this class and its methods to traverse the file system. An example of this method is provided on the canvas module.







