AP Computer Science A – Unit 8

Recursive Search Project

Over the course of this project, you will be writing a program that will ask the user for a search word and then recursively search through files on your computer to see if the word is found in any of them.

Working With Files

Java provides a class called `File` (in `java.io`), which represents a single path on your computer – either a file or a directory (folder). You can create a `File` object like this:

```
File f = new File("path/to/my/file.txt");
```

Keep in mind this **does not** create a file on your computer. In fact, the file may not even exist! All this does is create a Java object in your program pointing to a specific path. Once you have a `File` object, you can test whether it is a directory (folder) or a file by using the `isDirectory()` method on it:

```
// If this returns true, f is a directory. Otherwise, it's a file!
boolean isDir = f.isDirectory();
```

You can also get Java to return all of the file objects in a *single* directory by using the `listFiles()` method. Of course, this only works if `f` is a directory:

```
File[] files = f.listFiles();
```

You can print out the path of any `File` object simply by putting it in a print statement:

```
System.out.println(f);
```

Part 1: Recursively List All Files

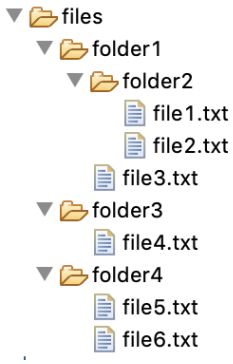
For this section of the assignment, you will be printing out the path of every file in a directory, even if the files are buried beneath many layers of folders. Because you will be traversing a folder structure of unknown depth, this must be done **recursively**.

Create a new Java project called `FileSearch`. In that project, create a new Java class called `SearchRunner` and give it a `main()` method and a `listDirectory()` method as shown on the next page. Complete the implementation of the

`listDirectory()` method, printing out the path of every file in every directory. You can assume that the `directory` parameter will be a valid directory and not a file.

```
public class SearchRunner {

	public static void listDirectory(File directory) {

		// Complete your method here

	}

	public static void main(String[] args) {

		File root = new File("files/");

		listDirectory(root);
	}

}
```

Example: If you are given a folder with the following contents:



Your program should output the following (order is not important):

```
files/folder1/folder2/file1.txt
files/folder1/folder2/file2.txt
files/folder1/file3.txt
files/folder3/file4.txt
files/folder4/file5.txt
files/folder4/file6.txt
```

As you complete your method, think about the following:

1. How would you iterate through the items in `directory`?
2. If an item you encounter is a file (not a directory), what do you need to do with it?
3. If an item you encounter is a directory, what do you need to do with it?
4. What is the **base case** of your recursive method? What is the **recursive case** of your method?

Part 2: Recursively Search Within a String

In this section, you will write a method `countOccurrences()` which takes two `String` parameters, `text` and `searchWord`, and returns an `int` of the number of times `searchWord` is found in `text`. For example:

```
String text = "He said that you said that I said you're old.";
int said = countOccurrences(text, "said");  // said == 3
int you = countOccurrences(text, "you");  // you == 2
int gorilla = countOccurrences(text, "gorilla");  // gorilla == 0
```

Copy the following method stub into your `SearchRunner` class:

```
public static int countOccurrences(String text, String searchWord) {
      // Complete your method here
}
```

There are many ways to count the number of times one `String` occurs inside another, but in this project, you must search the text recursively. This will require using some of the `String` methods from Unit 3, namely `indexOf()`, `length()`, and `substring()`. Taking the above example, this can be accomplished as follows:

1. Look for the index of the first occurrence of "said" in "He said that you said that I said you're old." The index of the first occurrence is 3 (0 = "H", 1 = "e", 2 = <*space*>, 3 = "s" — this is the beginning of the first occurrence).
   HINT: `indexOf()`
2. An occurrence was found, so we know that the number of total occurrences is *one plus the number of occurrences in the rest of the string*. The rest of the string starts at index 7. How did we determine this index?
   HINT: `length()`
3. Get the rest of the string after index 7. This is " that you said that I said you're old."
   HINT: `substring()`
4. Repeat steps 1-3, looking for the first occurrence of "said" in the rest of the string you determined in step 3. The first occurrence is at index 10. The rest of *that* string (starting at index 14) is " that I said you're old."
   HINT: recursive case
5. Repeat steps 1-3, looking for the first occurrence of "said" in the rest of the string you determined in step 4. The first occurrence is at index 8. The rest of *that* string (starting at index 12) is " you're old."
   HINT: recursive case

6. Repeat steps 1-3, looking for the first occurrence of “said” in the rest of the string you determined in step 5. The index is -1 so we know that “said” was not found. The number of occurrences in this string is zero.
   HINT: base case

Test your `countOccurrences()` method by adding the example from the beginning of this section to your `main()` method. Print out the variables `said`, `you`, and `gorilla`.

Part 3: Put It All Together

Here’s where the fun begins! Now you will modify your `listDirectory()` method, and instead of having it just print the path of every file, it’s going to search every file for the number of times a search word occurs inside it.

Let’s start by giving our method a better name - rename `listDirectory()` to `searchDirectory()`. The `searchDirectory()` method will also need to accept another parameter — the search word that we are looking for. The header for your method should now look like this:

```
public static void searchDirectory(File directory, String searchWord)
```

Keep in mind that because this method is recursive, you’ll have to modify the recursive case within your method to reflect the new name and additional parameter.

In the base case of your method, use your `countOccurrences()` method to find how many times the search word occurs in the file. In order to use `countOccurrences()`, you will need to get the contents of the file as a `String`. We have provided a helper method that does this for you. For a `File` object called `f`, get its contents like this:

```
String fileContents = SearchHelper.readFile(f);
```

Modify your print statement to include the number of times the search word was found next to the file’s path. **Do not print anything if the word was not found in the file.**

In your `main()` method, remove the `countOccurrences()` tests from part 2. Prompt the user to enter a search word and use `Scanner` to capture their input. Your `main()` method should now look like this:

```
public static void main(String[] args) {
      File root = new File("files/");
      // Prompt user for a search word and capture it with Scanner here
      searchDirectory(root, searchWord);
}
```

The final output of your program should look something like this:

```
Enter a search word: moon
files/random-words/discontents/…/turbidity/guitarists.txt: 1
files/random-words/procedure/…/inconveniencing/burps.txt: 1
files/books/the-scarlet-letter/introductory-the-custom-house.txt: 5
files/books/the-scarlet-letter/chapter-12-the-ministers-vigil.txt: 1
files/famous-speeches/kennedy-moon-speech.txt: 7
```