R·I·T | Rochester Institute of Technology
Golisano College of Computing and Information Sciences
Department of Information Sciences & Technology

ISTE-121 Computational Problem Solving for the Information Domain II

Name: ________________________________

Lab 9c: Recursion & Sort

Overview

This lab is designed to help you think about creating recursive solutions to problems that would be more difficult with traditional iteration. To help you in this new adventure, design the flow of the code before typing code.

Part 1a – Sum the number of bytes, files and directories in a directory tree. (4 points)

<u>The exercise MUST be completed during the lab period.</u>

Overview

Given a starting directory name, find the <u>number of bytes</u> stored below that directory. To keep things “simple,” use only one file with the class named `DirectoryContents` that incorporates the main method. This lab will help in a future homework. You may have had a similar lab in the previous programming course, make sure you change it for these requirements. Also remember to design it first!

Requirements:

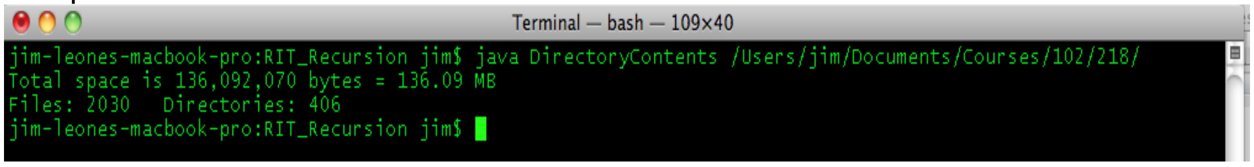
The main method instantiates an object of `DirectoryContents`. The program must be run from the command console so either an absolute pathname/directory_name or relative pathname/directory_name, such as “../../” must be given on the command line. The pathname and directory from the `args[]` array is the starting directory.

Using the `DirectoryContents` object, call a recursive method, `getSpace` that computes and returns the sum of the file sizes of all files under the given directory name. Method `getSpace` requires an initial directory name as a `File` object and returns a `long` result. The `getSpace` method obtains a list of all files within that directory. The `getSpace` method looks through the `File` array and, if the entry is a file, add the size of the file to a total counter.

However, if the item in the `File` array is a directory, add one to the directory counter. Call the getSpace method again with this directory name. Remember to sum the returned byte-count from calling getSpace again. At the end of the method, when the recursive method finishes processes a directory, the method returns the size of the files in that directory.

A starting shell or skeleton is supplied for this laboratory assignment. Modification of existing code in the shell is NOT allowed. Include the appropriate attributes and code within the existing skeleton to complete Exercise 1. A sample run is shown below from a personal laptop.

Sample run from Unix terminal:



```
jim-leones-macbook-pro:RIT_Recursion jim$ java DirectoryContents /Users/jim/Documents/Courses/102/218/
Total space is 136,092,070 bytes = 136.09 MB
Files: 2030   Directories: 406
jim-leones-macbook-pro:RIT_Recursion jim$
```

Sample run from jGRASP, showing fully qualified path, and relative path

```
java DirectoryContents /Users/jim/Documents/Courses/ISTE-121
Total space is 136,006,117 bytes = 136.01 MB
Files: 2033   Directories: 406

java DirectoryContents ../../../../
Total space is 900,409,969 bytes = 900.41 MB
Files: 3025   Directories: 1125
```

Signature: ____ Date: _______________
Have your instructor or TA sign here when Exercise 1 works correctly.

Part 1b – Exception Handler. (1 points)

The exercise MUST be completed during the lab period.

If the program is executed on a School of Informatics lab machine, it is quite likely that many directories will be accessible only with *admin* privileges. Attempting to traverse a directory without proper privileges will result in an exception. Add to the program from Exercise 1a, the appropriate `try/catch` block(s) to handle exceptions. Demo by selecting a path (absolute or relative) to a directory on a lab computer that includes a restricted subdirectory, such as C:/Public, or C:/

What exception(s) did you catch, besides the last Exception?

____

____

Signature: _________________________________ Date: _______________
Have your instructor or TA sign here when Part 1b works correctly.

Part 2 – Sorting (5 points)

Overview of problem:

From the zip download, open the program `SortAddresses`. This program sorts a list of addresses by zip code. Sorting only by the one value of zip code, this is considered a one level sort.

Problem to solve:

We need to sort not only by zip code but also alphabetically by state and city.
Look at the supplied code, do not delete the zip code Comparator, add a new state/city sort. Have the main print out the sorted state/city list at the indicated place in the code.

This is a two-level sort. Sorting on state code results in a -1, 0 or 1 as the compare() method provides as it did with the zip code compare() method. However, when a state is found to be the same as another state, the result is 0. In this instance you need to repeat the comparison for city. The discovery of how to do this is left up to you.

Questions to answer:

Explain the class signature of CustomerList.

____

____

What does "`new CustomerList();`"accomplish?

____

____

Signature: ______________________________ Date: _____________
Have your instructor or TA sign here when Part 2 works correctly.