Joanna Klukowska
joannakl@cs.nyu.edu

Programming Project 2: Why is my disk full?

Finding largest files in a directory structure.

Due date: Feb. 28, 11:55PM EST.

You may discuss any of the assignments with your classmates and tutors (or anyone else) but all work for all assignments must be entirely your own. Any sharing or copying of assignments will be considered cheating. If you get significant help from anyone, you should acknowledge it in your submission (and your grade will be proportional to the part that you completed on your own).

You are responsible for every line in your program: you need to know what it does and why.

If you are considering use of any Java API classes that are not discussed in CSCI101 curriculum and have not been covered yet in CSCI102, ask your instructor first. The objective of these projects is to practice the material that has been discussed in class and the solutions are not always the most efficient ones.

You are going to write a program that uses your new expertise in recursion to explore a directory on a user's computer. When your computer (or rather the hard drive that you are using) is running out of space, it might be hard to find the large files scattered all over your directory structure. Your program will provide a tool that given a name of a directory, explores all its sub-directories and files and does two things:

- computes the total size of all the files and sub-directories,
- prints a list of `n` largest files (their sizes and absolute paths.

The program that you write has to be command line based (no graphical interface) and it should not be interactive (do not ask user for anything).

Objectives

The goal of this programming project is for you to master (or at least get practice on) the following tasks:

- developing and writing recursive algorithms,
- working with existing code,
- using classes and methods that are part of the Java API,
- using command line arguments,
- implementing classes according to provided specification.

The Program Input and Output

Input The program should take two command line arguments.

First one is a directory name. If the directory is omitted from the command line, it is an error. The program should display an error message and terminate. The error message should indicate what went wrong. If the directory name is provided, but it is not a name of a valid directory, the program should display an error message and terminate. The error message should indicate what went wrong.

The second one is the maximum number of largest files found in the directory provided as the first argument. If the number is missing, the default value of 20 should be used.

Output Your program should determine and print to standard output the total space used by the directory in question (that includes the sizes or all the files and subdirectories contained in it). The program should also print the sizes and pathnames of the largest files.

Joanna Klukowska
joannakl@cs.nyu.edu

Example run When I run

```
java DirectorySize /home/asia 20
```

I get the following output

```
Total space used:  185.61 GB
Largest 20 files:
   2.84 GB     /home/asia/bin/dot_evolution_12820.tgz
   2.56 GB     /home/asia/bin/dot_evolution_21438.tgz
   1.83 GB     /home/asia/Data/GradSchool/Gabor/flu_data/set_01/FluB50K01.rec
   1.71 GB     /home/asia/Data/GradSchool/Gabor/flu_data/set_03_extra/FluB50K03.rec
   1.65 GB     /home/asia/Data/GradSchool/Gabor/flu_data/set_02/FluB50K02.rec
   1.64 GB     /home/asia/.thunderbird/3nbkrk21.default/Mail/pop.googlemail.com/Herman
   1.25 GB     /home/asia/xmipp/.git/objects/pack/pack-e519d7875d4ea8aaf90935c937e5ea48c4563d49.pack
   1.25 GB     /home/asia/xmipp_130108/.git/objects/pack/pack-e519d7875d4ea8aaf90935c937e5ea48c4563d49.pack
   1.25 GB     /home/asia/xmipp_130306/.git/objects/pack/pack-e519d7875d4ea8aaf90935c937e5ea48c4563d49.pack
   1.17 GB     /home/asia/Downloads/Mathematica_8.0.1_LINUX.sh
 994.68 MB     /home/asia/Data/Hunter/developper_tools/xcode313_2736_developerdvd.dmg
 994.68 MB     /home/asia/Data/HunterTeaching/CSci127/developper_tools/xcode313_2736_developerdvd.dmg
 913.91 MB     /home/asia/Data/GradSchool/Gabor/flu_data/FlueB9.tar.gz
 804.69 MB     /home/asia/Data/GradSchool/Gabor/Soft_X-ray/20110922 tomo3/tomograma/borrame.stk
 804.69 MB     /home/asia/Data/GradSchool/Gabor/people/Kino/From_Kino/20110922 tomo3/tomograma/borrame.stk
 774.22 MB     /home/asia/Data/GradSchool/Gabor/Matlab/FDR/proj_ddctf_cor_JK_nonoise_3200.mat
 769.85 MB     /home/asia/Data/GradSchool/Gabor/Matlab/FDR/cylinders_stacks/proj_ddctf_cor_cylinders_additive_noise_3200.mat
 728.11 MB     /home/asia/Data/Images/2011_06/P6110006.AVI
 704.00 MB     /home/asia/Data/GradSchool/Gabor/flu_data/set_01/recon_FluB50K01_noctf.ali
 676.39 MB     /home/asia/Data/GradSchool/Gabor/ExperimentalRuns/largeFileSupportInSnark/recfil
```

User Input This program is not interactive. It should not prompt or expect any input from the user.

Computational Task

Exploration of directory structure requires use of recursion (well, as we said in class, it could be done iteratively, but in this assignment you need to use a recursive solution). As you visit each directory, you need to process all the files that are in it and visit recursively all of its sub-directories. As you do that, you will need to update the total size of all visited files/directories and add all the files that you see to a list of files (so that later you can recover the `n` largest files.

Here is the pseudocode of the recursive algorithm you should use:

```
exploreDir ( potentialDirName )
    if potentialDirName is a directory
        add its size to totalSize
        get the list of all the files and sub-directories in potentialDirName
        for each of the files and sub-directories
            call exploreDir <-- this is the recursive call !!!
    otherwise potentialDirName is a file
        add file's size to totalSize
        add file (its name and size) to the list of files
```

WARNING: If implemented incorrectly, this algorithm results in infinite recursion when used on systems that allow shortcuts/links in the directory structure of dirName that produce circular paths. To avoid this, use the `getCanonicalPath()` method of the File class rather than `getAbsolutePath()` to obtain the name of the directory and make sure that you never visit the same directory twice.

Program Design

Your program must contain two classes:

- `FileOnDisk` class to represent a file in the program. (Note that this is just a very simplified version of the File class provided by Java API). The specification and requirements of this class are provide in its Javadoc at http://cs.nyu.edu/~joannakl/cs102_s15/assignments/doc/proj2/FileOnDisk.html
- `DirectorySize` class that is the runnable program containing the `main()` method. This class is partially written and you need to complete its implementation.

Joanna Klukowska
joannakl@cs.nyu.edu

Programming Rules

You must document all yokur code using Javadoc. Your class documentation needs to provide a description of what it is used for and the name of its author. Your methods need to have description, specification of parameters, return values, exceptions thrown and any assumptions that they are making.

Class's data fields and methods should not be declared `static` unless they are to be shared by all instances of the class or provide access to such data.

You may use `ArrayList` and `String` classes and any methods that are provided in them.

You may use any classes for reading the input from the file and writing the output to the file.

You may use any exception related classes (if you wish).

You may use the sort/search methods provided in the Arrays and/or Collections class (but this is going to be possible only if you design your `Tree` class appropriately).

Working on This Assignment

You should start right away! This program does not require you to write much code , but it will take some time to read through the code and specifications that are given to you. The testing of the implementation of the algorithm that computes the size recursively may also take significant amount of time.

You should break the development of the code into stages:

- Start by reading through the code provided with the assignment. Make sure you understand the parts that are implemented. Make sure you understand what you need to implement.
- Implement the `FileOnDisk` class according to the specifications.
- Implement the algorithm that recursively computes the size of a directory. Make sure that it also stores the list of all the files encountered during the recursive traversal.
- Complete the code in main() method.
- Test, test, test, test. Try to break your program.
- Submit your code.

You should backup your code after each time you spend some time working on it. Save it to a flash drive, email it to yourself, upload it to your Google drive, anything that gives you a second (or maybe third copy). Computers tend to break just a few days or even hours before the due date - make sure that you have working code if that happens.

Grading

If your program does not compile or crashes (almost) every time it is ran, you will get a zero on the assignment.

If the program does not adhere to the specification, the grade will be low and will depend on how easy it is to figure out what the program is doing.

15 points design, implementation and correctness of the missing code inside the `main()` method.

35 points design, implementation and correctness of the `getSize()` method.

25 points design, implementation and correctness of the `FileOnDisk` class

25 points proper documentation

How and What to Submit

Your should submit all your source code files (the ones with .java extensions only) in a single **zip** file to NYU Classes. You should have two different files: `DirectorySize.java` and `FileOnDisk.java` (make sure that these are the names of the files).

Once you submit the program make sure that you receive a confirmation email from NYU Classes. Read through it to double check that correct files were uploaded. Keep this email as your proof of submission until you receive your grade for the assignment.