

Wanted: Java Programmer

Appliance Retailing Information Systems (ARIS) is an independent software vendor that develops software for major home appliance, furniture, and lawn/garden retailers. We are a highly successful, small but growing, family owned company and have been in business for about 7 years. Our software links retailers to their warehouses, manufacturers, financing companies and to buying cooperatives. Our software allows retailers to look up product specifications and graphics, print price quotes and price tags, order merchandise from manufacturers and more.

ARIS is looking to hire additional full time Java developers to assist us in our software development efforts. Specially, successful candidates will work to enhance our web crawling, data mining, and warehousing software. ARIS needs individuals who are already proficient in the Java programming language. Training will be provided on data mining techniques as well as on the other technologies we use in our products.

Although desired, a degree in a Computer Science related field is not necessary if candidate possesses Java programming skills. Qualified individuals must have strong verbal and written communication skills while willing to take initiative and be able to work remotely with minimal supervision. The ideal candidate is located in the immediate Phoenix/Scottsdale area and is willing to attend regular meetings in Scottsdale, AZ (but may, generally, work from home). Flexible options include candidate's choice of salary or contract based employment. Compensation includes competitive salary *and* generous performance based bonuses.

Interested applicants should complete one of the two sample programming tasks outlined below and email your code along with a resume to Jim Kane (jim.kane@gmail.com). Select candidates will be invited to an interview in our Scottsdale offices to discuss their code and resume. Should you have any questions, feel free to email Jim Kane. Be sure to include your telephone contact information in your email.

Code Sample for Interview

Introduction

This section outlines two small programming tasks. Before your interview, please complete one of the tasks and email the code to jim.kane@gmail.com. Should you have any questions, feel free to email. Be sure to include your telephone contact information in your email.

All code should be written in the Java programming language. You may use whatever version (e.g. 1.4, 1.5, 1.6 etc.) of Java you are most comfortable with. Please write your code entirely within one source file. If you need to use more than one class to solve the problem, use inner classes to keep your code entirely within one source file. Please include a suitable number of comments in your code but do not go crazy. We want to see code the way you would normally write it.

If you have specific questions, please email Jim Kane (jim.kane@gmail.com).

During the interview, we will discuss your code to gain a sense of your abilities as a programmer. If a trivial solution does exist to one of the problems (i.e. there is an API call that does it)

please do not use it as it will give us very little to go to gauge your skills. Feel free to use API calls in your solution, though. For example, in problem one, feel free to use the Collections Sort method.

Problem One: Folder Size Summary

Let's say that our disk is getting full and we need to free up some disk space. It would be very helpful to know which directories on our disk are consuming the bulk of the space.

Write a program that takes as input from the command line an absolute path and prints out a list of sub directories and files contained within that path; sorting by size from largest to smallest. The size of a subdirectory is the size of all of that sub directory's contents. The program should also write out the size of the files, in kilobytes, contained in a give sub-directory or file. Please write your code in a class called DU.

NOTE: When you design your answer, it is important to remember that a directory's size is the sum of the size of all the files it contains *plus* the sum of the size of all of that directory's sub directories. This process of summing the size of sub directories continues indefinitely: even if a file is 100 directories down, its size should still be added to the total.

NOTE: Do not list out the complete contents of subdirectories file by file. The idea is to get a high level view of things. The goal is to find out that our Program Files directory is, say, taking up 700MB of disk space while our temp directory is only taking up 10MB. This helps us in freeing up disk space: if we know, for example, that most of our disk space is being taken up by the Program Files directory, we can focus our efforts on deleting old stuff in that subdirectory and leave the smaller subdirectories alone.

For example, say that we have a directory on our computer, c:\test_du that contains the following directories and files:

```
C:\test_du\foo\a.dat (100kb)
C:\test_du\foo\b.dat (200kb)
C:\test_du\foo\another_dir\jim.dat (500kb)
C:\test_du\bar\ball.jpg (5kb)
C:\test_du\bar\sam\sam1.jpg (100kb)
C:\test_du\bar\sam\sam2.jpg (300kb)
C:\test_du\somefile.dat (700kb)
```

Running the command `java DU c:\test_du` should produce the following output:

```
DIR    C:\TEST_DU\FOO                800KB
FILE   C:\TEST_DU\SOMEFILE.DAT          700KB
DIR    C:\TEST_DU\BAR                   405KB
```

We can quickly verify that this is the correct output as follows: the total size of all of the files and subdirectories of c:\test_du\foo is 800KB (a.dat is 100KB + b.dat is 200KB + jim.dat is 500KB = 800 KB). The total size of all of the files and subdirectories of c:\test_du\bar is 405KB (ball.jpg is 5KB + sam1.jpg is 100KB + sam2.jpg is 300KB = 405KB). The size of SOMEFILE.DAT is 700KB. Therefore, FOO is the largest sub directory so it is listed first at 800 KB. SOMEFILE.DAT is listed second at 700KB and the BAR directory, whose contents total only 405KB is listed last.

Problem Two: Subset Sum

Say I give you a target value and a list of numbers to pick from and ask you to pick numbers from the list such that the numbers picked add up to the target value. This task is known as Subset Sum and is a classic computer programming problem. If you decide to do this problem, your task will be to write a program that solves subset sum.

Lets try a few examples. Say I give you a target value of 150 and a list of numbers to pick from consisting of 1, 2, 100, 22 and 28. You would pick the numbers 100, 22 and 28 because $100 + 22 + 28 = 150$. Say I give you a target value of 30 and the sample numbers to pick from, 1, 2, 100, 22 and 28. You would pick 28 and 2 because $28+2 = 30$.

NOTE: Once you use a number from the list you can not pick it again. This means, in the above example, you can't get 30 by picking the number 2 fifteen times.

NOTE: Sometimes, the answer is that there is no answer. For example, if I give you a target value of 100 and a list of numbers to pick from of 5, 8, and 10 its quite obvious that there is no way to get the total to add up to 100. Even if you pick every number from the list, the total is only 23.

NOTE: Sometimes there is more than one correct answer. For example, given the numbers 1, 2 and 3 and a target value of 3 correct answers include picking 1 and 2 (because $1+2 = 3$) and picking just the number 3. Your program does not need to find every correct answer, just one correct answer.

More formally: Given a target integer T and a set of Integers S , find a set L where L is a subset of S such that the sum of all the integers in $L = T$.

For example, when $T = 15$ and $S = \{1, 2, 3, 5, 7, 10\}$ $L = \{3,7,5\}$ because $3+7+5 = 15$.

For example, when $T = 9$ and $S = \{1, 2, 3, 5, 7, 10\}$ $L = \{2,7\}$ because $2+7 = 9$.

For example, when $T = 100$ and $S = \{1, 2, 3, 5, 7, 10\}$ L is undefined because no matter how we pick numbers from the set S we can't get a sum of 100.

Write a program that takes a list of numbers from the command line. The first number is the target value and the remainder of the numbers (however many their may be) is the set of numbers you have to pick from. The program should either output a subset whose sum equals the target or undefined if no such subset exists. Call this class SubsetSum.

For example, the following inputs should yield the following results:

```
Java SubsetSum 15 1 2 3 5 7 10
{3, 7, 5}
```

```
Java SubsetSum 9 1 2 3 5 7 10
{2, 7}
```

```
Java SubsetSum 100 1 2 3 5 7 10
Undefined
```

Note: many times there is more than one correct answer to a given input. As long as the subset your program outputs sums up to the total, your answer is correct.