Assignment objectives

Practice writing a Brute Force algorithm with a common design pattern ("generate and test", a type of exhaustive search).

Introduction

Here is the basic problem we are going to solve:

You are given a list of numbers. Somewhere distributed among this list, there are three separate numbers (a *triple*) that add up to a multiple of 37. The numbers could be anywhere on the list. There is only one such combination. Find it.

For example, suppose you are given the following numbers as input:

{149, 95, 59, 60, 247, 178}

Among these numbers 95+60+178 = 333, which is a multiple of 37. So the answer is {95, 60, 178}.

(A side question: If there are N numbers in the input data, how many different triples are there—multiples of 37 or otherwise?)

Description of program

Your program will:

1. Read a list of integers from a data file and store them in an array or ArrayList
    - No other Java Collection types – arrays only, please
2. Use a Brute Force algorithm to find the desired triple
3. Display the result (console output)

There is no need to write error-checking code. You can assume that the data is properly constructed, that it contains only integers, one number per line, and that the data file contains at least three numbers. It is allowable for the same number to appear more than one time in a data file. I have no idea if this actually happened in any of the sample data files you are given.

Brute Force means that your program should (if necessary) generate *every possible combination of three numbers in the file*, and test each one to see if it works. ("Generate and test"!) The test data files (see below) happen to be lovingly crafted so that *there is only one* valid triple.

If your program is given a data file that has more than one triple, it is okay (but not required) to report all of them.

Additional code requirements

You *must* follow the COMP 3760 Coding Requirements (separate handout available on Learning Hub).

In addition, your code *must* follow the structure below in order for me to grade your assignment.

Class name: `Lab2` (Note the exact spelling and capitalization.)

Required public method: `void doTheStuff(String) throws FileNotFoundException`
The only argument is the name of a data file. This function performs the tasks described in the above section "Description of program", including printing the results to the console.
This method must not alter the filename, but use it *exactly* as passed. If your program needs path information to find a file when running within your Java environment, this path information should be included when the filename is passed to the function.

Required public method: `void main(String[]) throws FileNotFoundException`
The argument is the default required program arguments. (Don't use them!) All your main method *must* do is call `doTheStuff()` with a filename, but you can do other stuff if you want. For example, if you want to run some additional code to solve the  problem. Or you could set it up to call `doTheStuff()` repeatedly with all the data files.

You are free to break up your code into any other private/public methods (e.g. helper functions) that you wish to use. A helper function to read the data file and return the array would be a great example. Or how about one that takes an array and a "value" and performs the brute force search for triples? These are just ideas ... be creative!

Test data files

There are 7 data files available for you to test your algorithm. Here are the expected answers for the given data files.

```
data0.txt  {11, 12, 14}
data1.txt  {95, 60, 178}
data2.txt  {21, 60, 30}
data3.txt  {95, 60, 67}
data4.txt  {813577, 902305, 304318}
data5.txt  {359920, 352374, 249854}
data6.txt  {582512, 773989, 427787}
```

Note: When I grade your program I may be using different data files!

Submission information

Due date: As shown on Learning Hub.

Submit the following to the drop box on Learning Hub:

- Just your Java source code (*.java file).
- File name is not important.
- Please *do not zip* or otherwise archive your code. Plain Java files only.
- Please *do not zip* or include your entire project directory.

Marking information

This lab is worth 20 points.

5/20 of these points are reserved for compliance with the COMP 3760 Coding Requirements.

Virtual donut

This part is not required; it is just for fun.

Find the answer for the additional data file "huge_file.txt".

This file is intentionally too big for you to use your brute force algorithm to find the answer. I estimated that my laptop would require about 400 hours to finish the computation!

To earn this it is enough to find the answer by any means. You do not need to write a program that can fully and quickly solve the problem on its own (although this is doable). You may be able to benefit in some way from some modification of your lab code, but you will also need to find some insight that goes beyond the straightforward BF algorithm. (Hint: You need some math.)

To get credit for solving this bonus problem, make sure it's somehow obvious to me that you did it, for example:

- Note/comment with your submission to Learning Hub
- Private message somewhere/somehow
- Have your main program run it after it runs `doTheStuff()`)
- Tell me if (and how) you used (or modified) your lab code to assist you
- *Don't spoil it for others!*

Last words

Coding requirements.

No, really

Did you put **your name** (and ID and set) on your program?