Hyperion Development

Task: Advanced Arrays

www.hyperiondev.com

Introduction

Welcome to the Advanced Arrays Task
Overview:

You should currently be very comfortable with the understanding and implementation of basic arrays - if not, be sure to go through the content of your previous task! You'll now be exposed to two dimensional and multidimensional arrays - the ultimate arsenal to tackle any possible array implementation in Java!

-The Hyperion Team

Into the second dimension!
You've gotten used to a standard array, however sometimes you'll find yourself needed something more complex. Adding another dimension can easily solve this problem - and thus the appeal of two dimensional arrays comes into effect. Actually, there are no two-dimensional arrays in Java. It's simply multiple subarrays within an array:

Two-dimensional arrays.

```
1 String[][] twoDimArray = {{"a", "b", "c", "d", "e"},
2                           {"f", "g", "h", "i", "j"},
3                           {"k", "l", "m", "n", "o"}};
4
5 int[][] twoDimIntArray = {{ 0,  1,  2,  3,  4},
6                           {10, 11, 12, 13, 14},
7                           {20, 21, 22, 23, 24}};
```

A multidimensional experience!

Multidimensional arrays are a useful structure in Java for storing data in a more structured way. They are often used for creating 2D or 3D systems. For example when creating coordinate systems. Suppose you wanted to store data in a 5x5 grid like below:

0;0	0;1	0;2	0;3	0;4
1;0	1;1	1;2	1;3	1;4
2;0	2;1	2;2	2;3	2;4
3;0	3;1	3;2	3;3	3;4
4;0	4;1	4;2	4;3	4;4

This could be accomplished using 5 separate arrays of length 5. Each position in each array would represent a position in the grid. Creating 5 arrays is tedious and referencing each of the positions would be difficult. A much more efficient way of doing this would be to use multidimensional arrays like below:

```
int [][] multi = new int[5][5];
```

Each set of square brackets references another 'dimension'. In this case, think of the first set of square brackets as the number of rows and the second set as the number of columns. You can add as many sets of square brackets as you wish but keep in mind that this increases memory usage exponentially and eventually your computer won't be able to handle it. For most cases you will never need to use more than three sets (this would be used for 3D coordinate systems).

You now have two indices to take care of, so when assigning values make sure you correctly reference both. For example, if we wanted to store the value '34' in the third row and the second column (i.e. position (2;1)), you would need to write the following line:

```
multi[2][1] = 34;
```

To do this for every position is even more tedious than doing it for a single dimension array. For past arrays we could just use an ordinary for loop to iterate through each position and assign a value to it. This, however, will not work for multidimensional arrays because we have more than one index. Trying to fill a multidimensional array with a single for loop would be very difficult. We can, however, make use of nested for loops. Having a loop for each index allows us to access every position. When creating multidimensional arrays greater than two dimensions a for loop should be nested for each dimension.

Below is an example of nested for loops being used to create a grid of random numbers between 1 and 10 in the 'multi' array:

```
for(int i = 0; i < multi.length; i++){
    for(int y = 0; y < multi[i].length; y++){
        multi[i][y] = (int)(Math.random() * 10) + 1;
    }
}
```

Note how each for loop references a different index. Also notice the lengths of each for loop: the first goes to multi.length which is the number of rows, the second goes till multi[i] which is the number of columns for each row (i).

It is important to understand how to access and edit each position in a multidimensional array. You will now put your understanding to the test.

Instructions

First read example.java, open it using jGRASP (Right click the file and select 'Open with jGRASP").

- In this folder there is a file called example.java
- Open this folder using the JGRASP program. You can either do this by right clicking on the example.java file, going to the 'Open with' menu, and selecting JGrasp.exe. Alternatively, you can run the JGRASP program on your computer, go to the top left corner of the program, select File->Open and navigate to example.java on your hard drive and double click on it to open it.
- Once example.java is open in JGRASP please read all of its content very carefully. This will help you understand the basic structure of a Java program.
- There is a compulsory task at the end of the example.java document. The instructions of what to do to complete the task can be found here. You must complete this exercise to continue onto the next task of this course.

Compulsory Task 1

Follow these steps:

- Create a new file called ThreeDim.java
- Within the main method, create a 10x10x10 three dimensional array.
- By making the use of nested for loops, store the value of the sum of each of its coordinate positions. For example:
  - position (3,4,8) = 3 + 4 + 8 = 15
- Print the array out to the user.
- Compile, save and run your file.

Compulsory Task 2

Follow these steps:

- Create a new file called MultiSort.java
- Within the main method, create a 7x7 two dimensional array.
- Assign random values in the range of 1 to 100 (inclusive) to each position of the array.
- Next, sort the values of the array in the following way:
  - Sort each row based on their column values in ascending order
  - Sort all rows based on their columns' totals in descending order
- Example:
  - (8,9) = Row total is higher (17>7). 8 swapped with 9 (8<9).
  - (3,4) = Row total is lower (descending). 3 was swapped with 4 (3<4).
- Print the array out to the user.
- Compile, save and run your file.

Things to look out for

1. Make sure that you have installed and setup all programs correctly. You have setup **Dropbox** correctly if you are reading this, but **jGRASP or Java** may not be installed correctly.
2. If you are not using Windows, please ask your tutor for alternative instructions.

Still need help?

Just write your queries in your comments.txt file and your tutor will respond.

Task Statistics

Last update to task: 28/01/2016.
Authors: Jared Ping and Tason Matthew.
Main trainer: Umar Randeree.
Task Feedback link: Hyperion Development Feedback.