Hyperion Development

Task: Control Structures - Arrays

www.hyperiondev.com

Introduction

Welcome to the Control Structures - Arrays Task
Overview:

You should currently be very comfortable with the understanding and implementation of different conditional statements - if not, be sure to go through the content of your previous tasks! You'll now be exposed to loop statements to understand how they can be utilised in reducing lengthy code, preventing coding errors, as well as paving the way towards code reusability. The next structure you'll deal with is an array - a pivotal component of programming in Java!

-The Hyperion Team

What exactly is an array?
An *array* is similar to a table of objects or primitive types, keyed by index. You may have noticed the strange parameter of the default main() method (String[] args) since the beginning of the course. It is an array.

Array fundamentals
In Java, an array is an object. This object has a given type for the contained primitive types or objects (*int, char, String, ...*). An array can be declared in several ways:

Array declarations.

```
1 int[] array1 = null;
2 int array2[] = null;
```

Those syntaxes are identical but the first one is recommended. It can also be instantiated in several ways:

Array instantiations.

```
1 array1 = new int[10];
2 array1 = {1, 2, 3, 4, 5, 6, 7, 8, 9, 10};
```

At line 1, we instantiate an array of 10 items with non-initialized items. At line 2, we instantiate an array of 10 given items. It will each be given an index according to its order. We can know the size of the array using the length attribute:

The array size.

```
1 int nbItems = 10;
2 Object[] array3 = new Object[nbItems];
3 System.out.println(array3.length);
```

Arrays are allocated at runtime, so the specified size of an array creation expression may be a variable (rather than a constant expression as in C). However, the size of an instantiated array never changes. If you need to change the size, you have to create a new instance. Items can be accessed by their index. Beware! The first index is 0 (zero):

The array indexes.

```
1 char[] array4 = {'a', 'b', 'c', 'd', 'e'};
2 System.out.println(array4[2]);
3 array4[4] = 'z';
4 System.out.println(array4[4]);
```

If you attempt to access an index which is either negative or too high, you will get an *ArrayIndexOutOfBoundsException*.

That covers the fundamentals of array definitions and implementations. Navigate to the example.java file in your task folder to learn more in-depth content on arrays!

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 Factorial.java.
- Create an array called factorials with space to store 13 integer numbers.
- For each index in the factorials array, calculate its factorial and store it. (Use Java to calculate the factorials).
- Print out all calculated factorials
- Compile, save and run your file.

Compulsory Task 2

Follow these steps:

- Create a new file called Rebel.java.
- Create two arrays, one of them must store integer numbers and the other must store words. Both of them must be able to store 5 values.
- Fill the first array with numbers from 1 to 5 and the second one with numbers from 6 to 10 as strings.
- Swap the numbers in the first array with the ones in the second one (swap numbers with the same index).
- Multiply all the numbers in the first array by 4.
- Multiply all the numbers in the second array by 3.
- Print out the contents of both arrays.
- Swap all the numbers in the first array with the ones in the second one, but this time only swap the numbers at even positions.
- For every number from 1 to 40, print it out if it is in the array that stores numbers, the one that stores words or neither of them.
- For every number in the numbers array:
  - Print it out.
  - Print out the positions of the divisors it has in numbers array.
  - Print out the positions of the divisors it has in the words array.
- Do the above for the words array as well
- 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: 20/01/2016.
Authors: Riaz Moola and Jared Ping.
Main trainer: Umar Randeree.
Task Feedback link: Hyperion Development Feedback.