Running Head: ARRAYS

Week 2 Discussion 1: Arrays and Their Dimensions

Jered McClure

Walden University

Week 2 Discussion 1: Arrays and Their Dimensions

The primary reason to use an array is to create a logical order between the elements of that array. When moving to a multi-dimensional array, this logical order extends itself into a relational space. This is similar to how relational databases store data in tables, where each cell is related to corresponding cells based on their similar row, column, or cube location (Farrell, 2012, p. 388).

On top of the relational bonuses of arrays, they offer the ability to dynamically create objects and variables (Farrell, 2012, p. 389). That is, if the programmer in question is unsure as to the number of items required for a specific task, they can create those objects and variables within an unconstrained array. This allows them to expand and contract the application as needed, without requiring any additional code. In essence, it actually removes a great amount of complexity from the code.

A one dimensional array is useful for when a programmer needs to initialize or create a number of elements that are only related to each other in a single direction, i.e. one dimension. For instance:

**public** **class** oneDimension {

**public** **static** **void** main(String[] args) {

String[] cake = {"Chocolate","Strawberry","Mud","Cheese",

"Double Decker","Wedding","Birthday","Lie","Caek"};

System.*out*.println("I've heard that cake is a "

+ cake[7].toLowerCase() + ".");

}

}//Output: I've heard that cake is a lie.

We know that we want one of the values stored in the array, and that all the values in the array are related to each other. However, the value for the item in the array that we want is specific only to that element, {7}.

Multi-dimensional arrays allow for the creation of multiple arrays within a single array. Essentially, a programmer could potentially create all of their arrays in a single variable and call the values within those arrays based on their index location. A simple example of this is as follows:

**public** **class** twoDimensions {

**public** **static** **void** main(String[] args) {

String[][] cakeAnd = {{"Chocolate","Strawberry","Mud","Cheese",

"Double Decker","Wedding","Birthday",

"Lie","Caek"},{"Ice-Cream","Soda",

"Portal Gun","Birds"}};

System.*out*.println("I've heard that cake is a "

+ cakeAnd[1][2].toLowerCase() + ".");

}

}//output: I've heard that cake is a portal gun.

The index of the element “Portal Gun” is located at the coordinates {1, 2} within the array cakeAnd[][]. A useful application of this function would be a program that needs to dynamically create different objects based on system parameters. Each object could exist within the multi-dimensional array. When needed, the application would pass the index location of that object and initialize it for use.

Reference

Farrell, J. (2012). *Java Programming, Sixth Edition.* Boston: Course Technology, Cengage Learning.