**Data Structures**

If you want to store or use many objects without having to create each of them individually, data structures are very helpful.

**Arrays**

Arrays are one type of data structure that is commonly used. Arrays are a fixed length and they store only one type of data. Each object in the array is stored at a specific index. Arrays are generally instantiated with the call:

*type*[] *name* = new *type*[*size*];

Everything in italics is replaced by your code. Type can be any object or primitive type. You could have an array of ints, Strings, booleans, etc. or you could have an array of your own class. Name is the name you choose for the array, and size is the size of the array.

If you’re confused, look for the footnote at the end of this section.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| [0] | [1] | [2] | [3] | [4] |
|  |  |  |  |  |
|  |  |  |  |  |
| [0] | [1] | [2] | [3] | [4] |
| 1 | 2 | 3 | 4 | 5 |

int[] arr = new int[5]; 🡪

arr[0] = 1;

arr[1] = 2;

arr[2] = 3;

arr[3] = 4;

arr[4] = 5; 🡪

Array indexes start at zero. In this case, arr is the name of an array of ints. That means that every object in the array is an int. To access a certain value in the array, you need to know its index. For example, if you want to access the value that is last in the array, you would use the call arr[4].

int x = arr[4]; 🡪 //x = 5

You would get an error in this case if you tried to call arr[5], because the indexes only go up to four (remember, it starts at zero).

If you know the contents of the array before you create it, there is another shortcut to instantiating an array:

int[] array = {12,23,34,45,56,67,78,89,90,1};

This would create an array of ints with those values. If you wanted to access one of them, you would do the same thing:

int hi = array[2]; 🡪 //hi = 34

Note:

This declaration

int[] arr = new int[5];

is the same as saying:

int one;

int two;

int three;

int four;

int five;

It’s just a lot easier to have it condensed in one array. With arrays, the objects are each created in the array, but they don’t have individual names. Think of the indexes as the names for the objects. You can change and get their values in the same way.

Another note: Arrays are VERY important. Look at the videos or links if you don’t understand

**How to use Arrays**

When you use arrays, one of the most common things to do is to traverse the array, or go through the array stopping at each value. You can change them or just get the values. In this example, it will just change all of the values in the array to zero:

for(int i = 0; i<arr.length; i++)

{

arr[i] = 0;

}

Another way to traverse an array is to use a for-each loop. This automatically visits every cell in the array:

for(int i:array)

{

i++;

}

This code visits every int in the array and adds one to it. “i” acts as a temporary reference to the object, so each time the loop runs, i points to the next int until the end.

**Matrices**

A matrix is another name for a two-dimensional array. A matrix is instantiated in a similar way, and the values are accessed in a similar way.

int[][] matrix = new int[4][4];

//Assigning values – This would take up too much space. See code below...

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | [0] | [1] | [2] | [3] |
| [0] | 1 | 2 | 3 | 4 |
| [1] | 5 | 6 | 7 | 8 |
| [2] | 9 | 10 | 11 | 12 |
| [3] | 13 | 14 | 15 | 16 |

int test = matrix[3][0] 🡪 //test = 13;

You still need to traverse two-dimensional arrays as well. To do this, you can use a nested for-loop, which is basically a for-loop inside of a for-loop. Again, we can just set all of the values to zero for an example:

for(int x = 0; x < matrix.length; x++)

{

for(int y = 0; y < matrix[0].length; y++)

{

matrix[x][y] = 0;

}

}

matrix.length returns the number of rows and matrix[0].length returns the number of columns.

**Common Applications**

Arrays have many applications, mostly relating to storing data. It will be hard to find a good program that doesn’t use arrays. They are efficient and easy to use.

Matrices have many applications as well. One common application is in basic games with simple “animation”. Often, programmers will create a two dimensional array of primitive types, often ints, that control what is shown on the screen in a small section. For an example, make sure you check out the video on the website.

**Brief Description of Other Data Structures**

Arrays aren’t the only way to store data, but they are probably the most relevant for now. There are several different types of data structures:

* ArrayLists – Very similar to arrays, but they have a variable size, which means you can easily add and remove values after the array is initialized
* Lists – Easy to add and remove values, but they don’t store values by index and you have to go through and search every value to find one of them.
* Dictionaries/Maps – Store values by a key instead of an index. The only difference is that both the key and the value are objects

Although these are good to know about, you won’t need to use them for a while.

**Projects**

Here are some projects that will make sure that you know everything you need to from this lesson:

Don’t forget, you can always ask and answer questions on the website, and if that doesn’t help, bring your questions to our next meeting.