**An array** - is a sequential data structure. They are used to store and process large amounts of data.

**Creating Arrays**

An array lets us refer to an entire collection of data with a single variable. Here are two ways to **define** and **initialize** an array of integers.

int[] a = new int[5];

This code says: create a new integer array, named a, that is of length 5. All 5 of the values are set to 0.

int[] b = {2, 4, 6, 8, 10};

This code says: create a new array, named a, that is of length 5. Set the values to 2, 4, 6, 8, 10.

You can also make an array of any other variable type, see examples:

double[] grades = new double[50]; //array of 50 doubles, all 0.0

String[] names = {"Joe", "Tim", "Sarah"} //array of 3 strings

Car[] lot = new Car[25]; //array of 25 car objects, all null to start

**Array Indexing**

You already know how to do this! Remember that Strings are just **arrays of characters**. Array indexing is just like String indexing, it starts at 0. Use bracket notation [] to index. So...

int[] b = {2, 4, 6, 8, 10};

int x = b[0]; //x is set to 2

int y = b[1]; //y is set to 4

int z = b[2]; //z is set to 6

etc

**Iterating Over an Array**

You iterate over an array the same way you iterate over a String. Start at 0, go to length-1. See the below example code that populates an array with values.

 int[] a = new int[10]; //new array of 10 0's

//this loop will populate the array with the numbers 1-10

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

 a[i] = i + 1;

}

**IMPORTANT NOTE:** "length" is a field in the array class NOT a method. So you use **a.length** to get the length of array a **NOT a.length()**. Parenthesis invoke a method. no parenthesis invoke a field.

How about printing the array?

System.out.println(a);

Prints --> **[I@106d69c** which is not very useful. Remember that arrays are objects and **objects are stored by reference**. What that String means is Integer array @ memory location 106d69c (hexadecimal number). So great, now you know where exactly in your computer's RAM that array is, so useful.

Let's print the actual values:

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

 System.out.print(a[i] + " ");

}

System.out.println();

You just iterate over the array. Retrieve each value, and print it.

**2D Arrays**

You can create a 2D Array that stores values at a specific row and column

int[][] table = new table[3][4]; //3 rows, 4 col

//after initializing with values could look like:

1 2 3 4

5 6 7 8

9 1 2 3

//indexing (row then column)

System.out.println(table[0][3]); //prints 4

System.out.println(table[1][1]); //prints 6

System.out.println(table[2][0]); //prints 9

To index into the 2D array you must provide both the row and column number. For example here are the indices for a tic tac toe board:

****

To traverse a 2D array you need to use a nested for loop.

//note

table.length; //gets the number of rows

table[0].length; //gets the number of columns

for (int row = 0; row < table.length; row++) {

 for (int column = 0; column < table[0].length; column++) {

 //table[row][column] is the current element

 //starts at (0,0) then (0,1) … through first row

 //then (1, 0) then (1, 1) … through second row

 //etc

 }

}

**Final Note**

Arrays are a **fixed length** data structure. That means that once you create an array, you **can not** **increase or decrease** it's size.

int[] a = new int[30];

That array is size 30. So what do I do if I get 20 more numbers that need to be stored in a?

1. Create a new array, say b, of the new size (50 in this case).

2. Copy all values from a into b (with a for loop).

3. Set a equal to b (a = b;).

4. The old a array (of size 30) will be cleaned up by Java's garbage collection.