Sunday, December 30, 2018 1:53 PM

Arrays

Meet Arrays

The syntax to declare an array is:

dataType[] arrayVariableName = new dataType[numberOfElementsToStore];

int[] studentGrades = new int[10];
When you first create an array it is filled with "zero values", where as the zero value for objects is null. Null literally means "no object".

update the value stored in an index:
arrayName[index] = value;

studentGrades[0] = 98;
studentGrades[1] = 86;
studentGrades[2] = 90;

create an Array and fill it with values in a single statement like so:

dataType[] name = {dataValue1, dataValue2, dataValue3};

int[] studentGrades = {98, 86, 90};

Arrays and Methods

public static int[] myMethod(int[] a) {}

IndexOutOfBoundsException
arrayName.length // returns the capacity of the array
arrayName[arrayName.length-1] // the last valid index
arrayVariableName.length // returns the capacity of the array

Array Methods

import java.util.Arrays;
Method example description

toString(array)
Arrays.toString(a)
returns a String representation of the array using square brackets and commas like so: [value, value, value]

equals(array1, array2)
Arrays.equal(a, b) OR a.equals(b)
returns true if the two arrays contain the same elements in the same order

fill(array, value)
Arrays.fill(a, 10)fills every index of the array with a copy of the given value

copyOf(array, newLength)
Arrays.copyOf(a, 10)
creates a new array object with the given length and fills it with values in the same order as the original array. If there are left over indexes those are filled with the data type's zero value

sort(array)
Arrays.sort(a)
arranges the values in the array in sorted order from smallest to largest

binarySearch(array, value)
Arrays.binarySearch(a, 100)
returns the index of the given value in a sorted array.
Will return a negative number if the value doesn't exist in the array.

If you forget to use that method and instead pass an array directly into a println you will get output that won't make much sense. You will see something that looks like:
[I@15db9742
This is actually the location in memory of the array. Chances are instead you want a view into what values the array stores, which is why you want to use the Arrays.toString method like so:

System.out.println(Arrays.toString(a));

Arrays and Loops

```
for (int index = 0; index < a.length; index++) {
   a[index] = value;
}
```

```
int[] a = new int[10];
for (int i = 0; i < a.length; i++) {
    a[i] = i;
}
```