Java Programming

Array Concepts

What are arrays?

- Arrays are containers of more than one variable of a given type.
  - Think of an array as a list of objects.

Array Declaration & Initialization

```
String[] aArray = new String[5];

String[] bArray = {"a","b","c", "d", "e"};

String[] cArray = new String[]{"a","b","c","d","e"};
```

Printing an Array

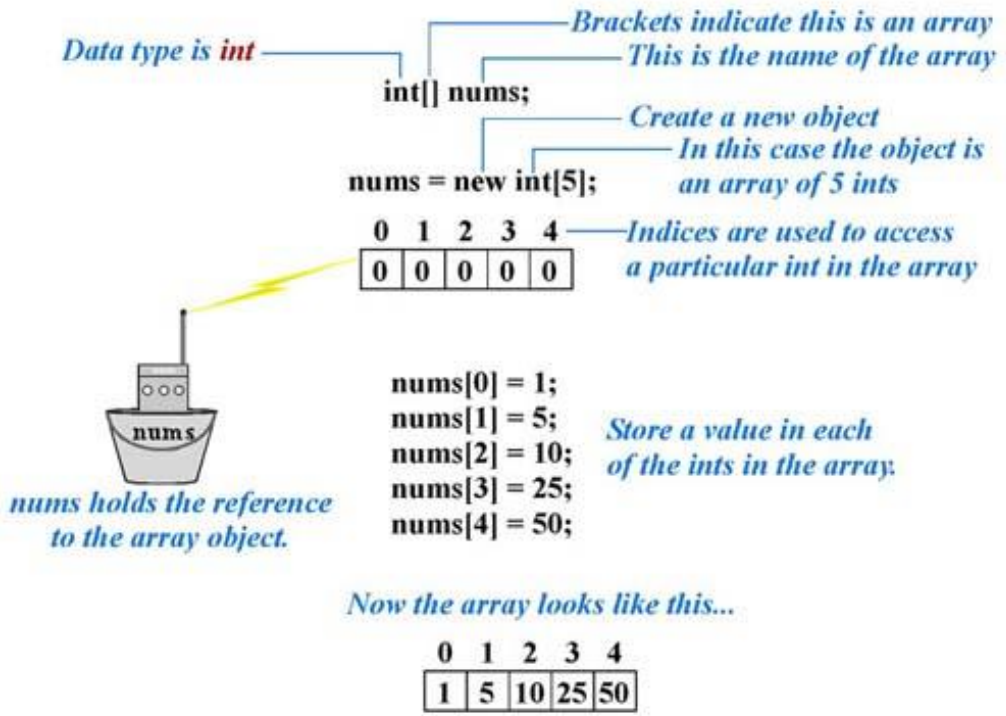
```
int[] intArray = { 1, 2, 3, 4, 5 };
String intArrayString = Arrays.toString(intArray);

// print directly will print reference value
System.out.println(intArray);
// [I@7150bd4d

System.out.println(intArrayString);
// [1, 2, 3, 4, 5]
```

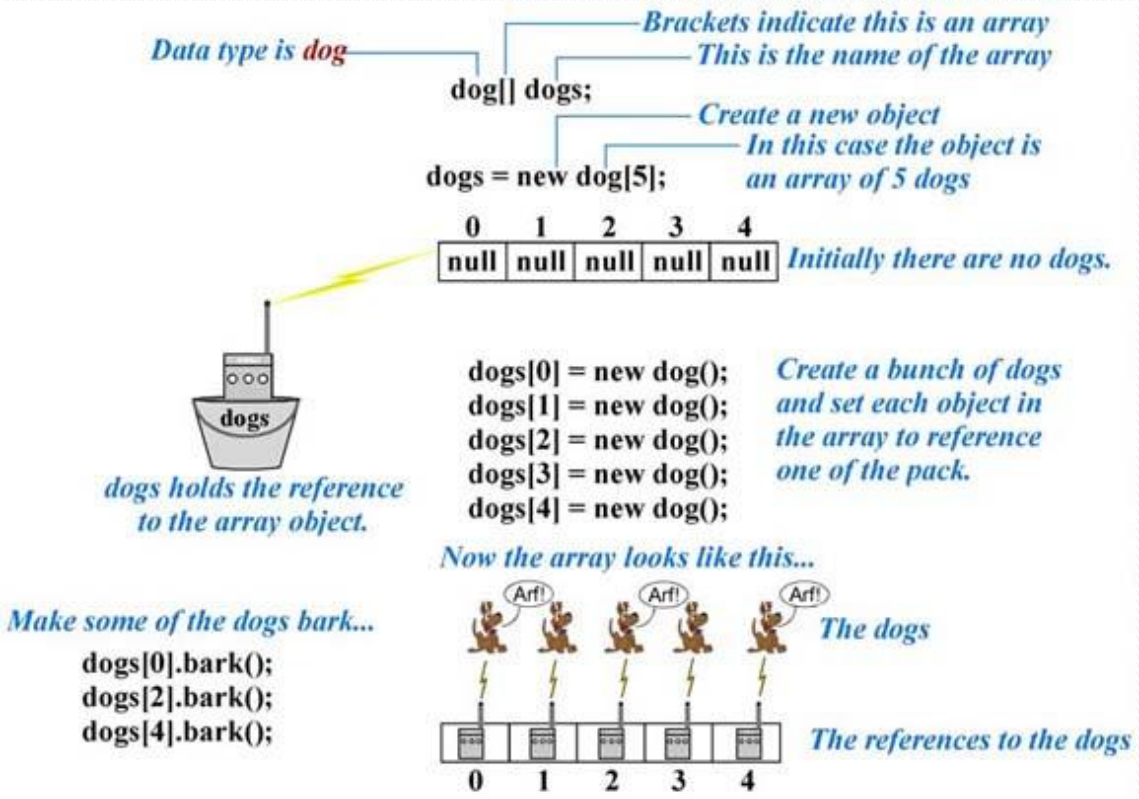
Array Example #1

- In the example below, an array of **int** variables is created and called **nums**.
- First you create a **reference** to an int array (**int[]**) using the square brackets after the data type to indicate that this is an array.
- Next you create the array with the **new** operator and set the **nums** reference variable to it.
- Now you can store values in or read values from the variables in this array by using the index into the array.



Array Example #2

- You can also create arrays of reference variables.
- In the example below an array of five reference variables to Dog objects is created.
- Note that we must create the array with the **new** operator, then we must create each of the Dog objects also with the **new** operator.



Array Copy

- For instance let's consider:

```
int [] numbers = { 2, 3, 4, 5};
int [] numbersCopy = numbers;
```

- The "numbersCopy" array now contains the same values, but more importantly the array object itself points to the same object reference as the "numbers" array.

- So if I were to do something like:

```
numbersCopy[2] = 0;
```

- What would be the output for the following statements?

```
System.out.println(numbers[2]);
System.out.println(numbersCopy[2]);
```

- Considering both arrays point to the same reference the output would look like the following:

```
0
0
```

Array Clone

- To make a distinct copy of the first array with its own reference we would want to ***clone*** the array. In doing so each array will now have its own object reference. Let's see how that will work.

```
int [] numbers = { 2, 3, 4, 5};
int [] numbersClone = (int[])numbers.clone();
```

- The "numbersClone" array now contains the same values, but in this case the array object itself points a different reference than the "numbers" array.

- So if I were to do something like:

```
numbersClone[2] = 0;
```

- What would be the output now for the following statements?

```
System.out.println(numbers[2]);
System.out.println(numbersClone[2]);
```

- You guessed it:

```
4
0
```