Reference Types vs. Value Types

In a previous video, I talked about the differences between a Reference vs. an Object, vs. an Instance, vs. a Class.

I want to revisit this a little and talk about why this matters when we're talking about arrays.

When you assign an object to a variable, the variable becomes a reference to that object.

This is true of arrays, but the array has yet another level of indirection if it's an array of objects.

This means every array element is also a reference.

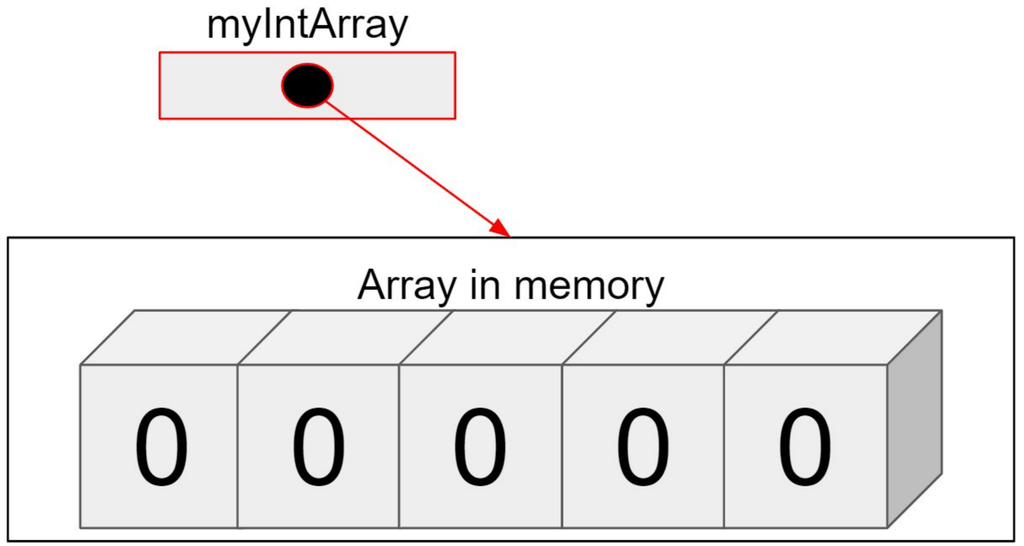
Reference Types vs. Value Types

```java
int[] myIntArray = new int[5];
int[] anotherArray = myIntArray;

System.out.println("myIntArray= " + Arrays.toString(myIntArray));
System.out.println("anotherArray= " + Arrays.toString(anotherArray));

anotherArray[0] = 1;

System.out.println("after change myIntArray= " + Arrays.toString(myIntArray));
System.out.println("after change anotherArray= " + Arrays.toString(anotherArray));
```



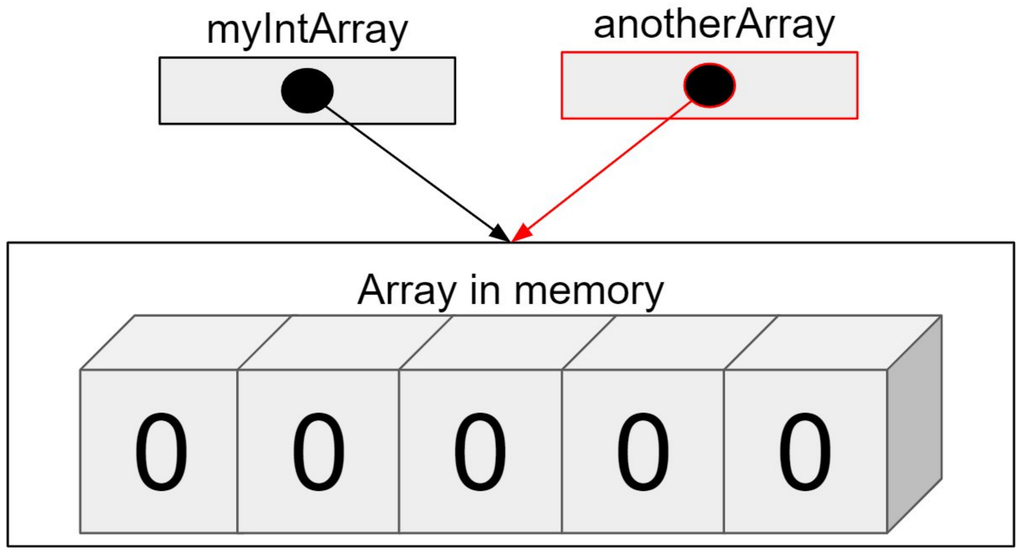
Reference Types vs. Value Types

```
int[] myIntArray = new int[5];
int[] anotherArray = myIntArray;

System.out.println("myIntArray= " + Arrays.toString(myIntArray));
System.out.println("anotherArray= " + Arrays.toString(anotherArray));

anotherArray[0] = 1;

System.out.println("after change myIntArray= " + Arrays.toString(myIntArray));
System.out.println("after change anotherArray= " + Arrays.toString(anotherArray));
```



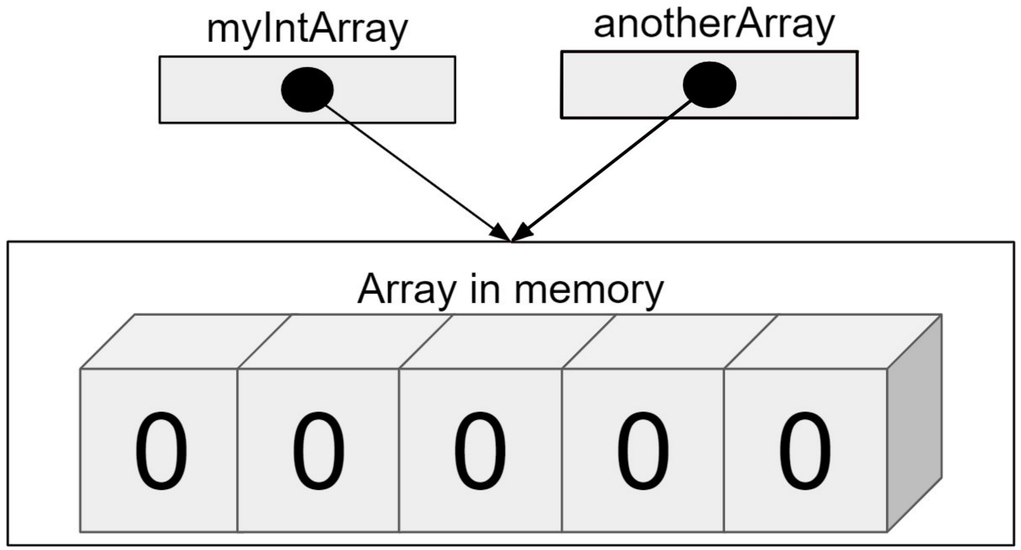
Reference Types vs. Value Types

```
int[] myIntArray = new int[5];
int[] anotherArray = myIntArray;

System.out.println("myIntArray= " + Arrays.toString(myIntArray));
System.out.println("anotherArray= " + Arrays.toString(anotherArray));

anotherArray[0] = 1;

System.out.println("after change myIntArray= " + Arrays.toString(myIntArray));
System.out.println("after change anotherArray= " + Arrays.toString(anotherArray));
```



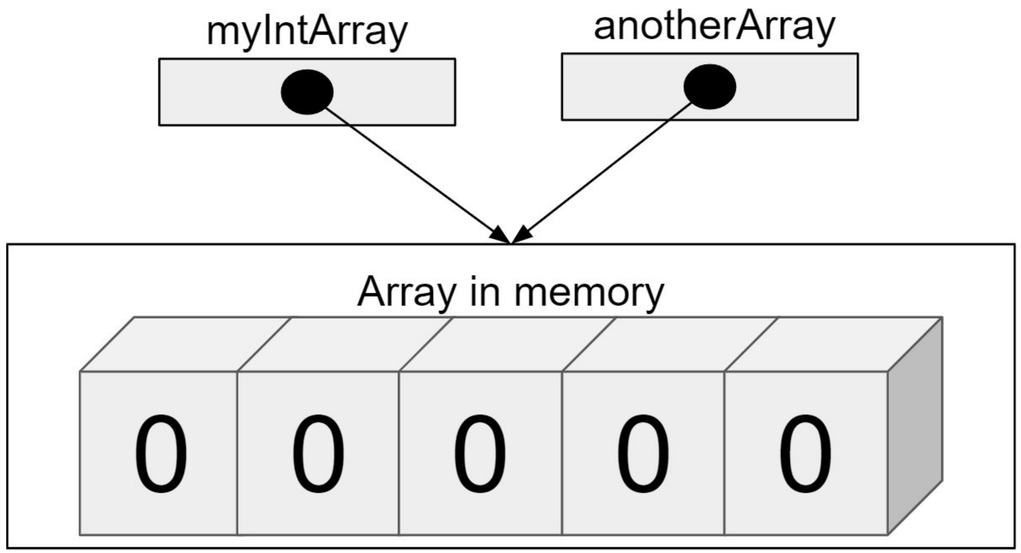
Reference Types vs. Value Types

```
int[] myIntArray = new int[5];
int[] anotherArray = myIntArray;

System.out.println("myIntArray= " + Arrays.toString(myIntArray));
System.out.println("anotherArray= " + Arrays.toString(anotherArray));

anotherArray[0] = 1;

System.out.println("after change myIntArray= " + Arrays.toString(myIntArray));
System.out.println("after change anotherArray= " + Arrays.toString(anotherArray));
```



OUTPUT:

myIntArray= 0 0 0 0 0

anotherArray= 0 0 0 0 0

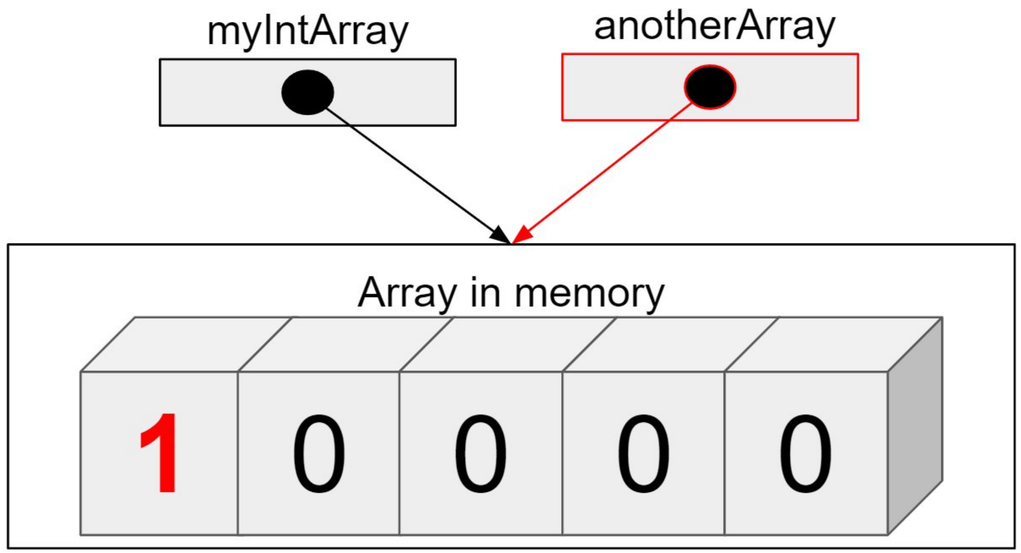
Reference Types vs. Value Types

```
int[] myIntArray = new int[5];
int[] anotherArray = myIntArray;

System.out.println("myIntArray= " + Arrays.toString(myIntArray));
System.out.println("anotherArray= " + Arrays.toString(anotherArray));

anotherArray[0] = 1;

System.out.println("after change myIntArray= " + Arrays.toString(myIntArray));
System.out.println("after change anotherArray= " + Arrays.toString(anotherArray));
```



OUTPUT:

myIntArray= 0 0 0 0 0

anotherArray= 0 0 0 0 0

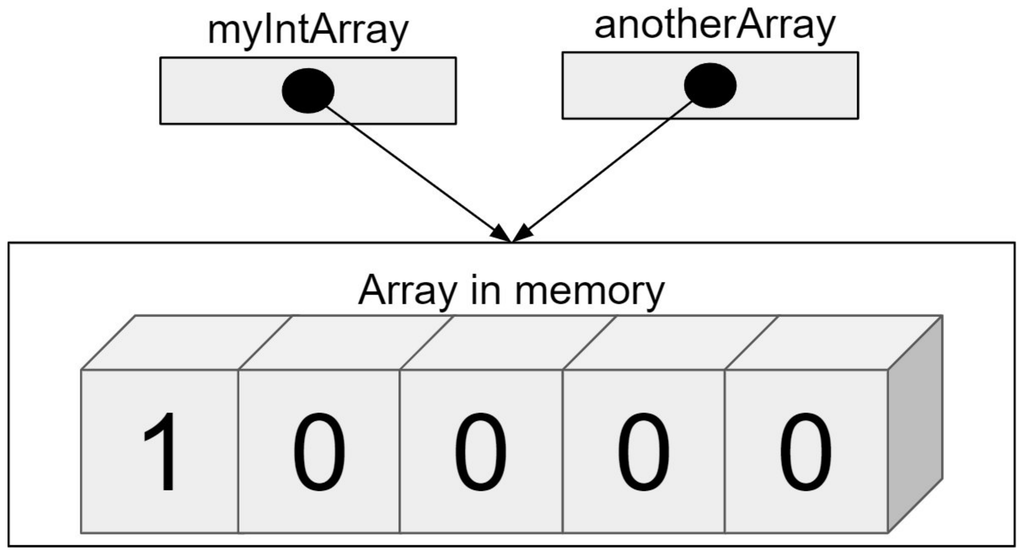
Reference Types vs. Value Types

```
int[] myIntArray = new int[5];
int[] anotherArray = myIntArray;

System.out.println("myIntArray= " + Arrays.toString(myIntArray));
System.out.println("anotherArray= " + Arrays.toString(anotherArray));

anotherArray[0] = 1;

System.out.println("after change myIntArray= " + Arrays.toString(myIntArray));
System.out.println("after change anotherArray= " + Arrays.toString(anotherArray));
```



OUTPUT:

myIntArray= 0 0 0 0 0
anotherArray= 0 0 0 0 0
after change myIntArray= 1 0 0 0 0
after change anotherArray= 1 0 0 0 0