**Differences between Primitive Array and Reference Array.**

There is no difference between Primitive Array and Reference Array. When you initialize and construct the array, array with regardless of the data types is considered as Object. Object stores reference in the memory heap. Therefore, when you copy and make changes to the new array, the old array will also have the same change.

**The only difference between Primitive Array and Reference Array is that Primitive Array have default value and Reference Array will all be null, and how Java manage the Array:**

**Some Reference Array use Wrapper Class to wrap primitive data types. In theory, Reference Array that use Wrapper Class will be slower compared to Primitive Array. Explanation in ResizeCopyArray.pdf.**

```
int[] intArray = new int[2];
String[] stringArray = new String[2];
System.out.println(intArray[0]); // 0
System.out.println(stringArray[0]); // null

// Proof of Primitive Array and Reference Array is an object.
assert charArray instanceof Object; // true
Assert stringArray instanceof Object; // true
```

These lines of code is written on the permise of finding the behavior difference between Primitive Array and Reference Array. **The only known difference is when you tried to `sysout()` a char[] vs `sysout()` a int[]. But after more in-depth check, char[] attr is also an Object and stores "reference" instead of actual data, which is the behavior of a Primitive Data Types.**

```
public class ArrayPrimitiveReference {
  static char[] primitiveArray = new char[2];
  static String[] referenceArray = new String[2];
  public static void main(String[] args) {
    // 1. Where to store the actual data?
```

```
primitiveArray[0] = (char) 49;
primitiveArray[1] = (char) 50;
System.out.println(primitiveArray); // expected output: 12.


referenceArray[0] = "1";
referenceArray[1] = "2";
System.out.println(referenceArray); // expected output: reference.


System.out.print('\n' + "Copying & Changing The Data" + '\n');


// 2. Copying & Changing The Data
char[] copyPrimitiveArray = primitiveArray;
System.out.println(copyPrimitiveArray); // expected output: identical data.
copyPrimitiveArray[0] = (char) 51;
System.out.println(
  "Changing The Data: " + '\n' +
  "primitiveArray[0]: " + primitiveArray[0] + '\n' +
  "copyReferenceArray[0]: " + copyPrimitiveArray[0]
);


// copying primitive data type == copying the actual data.
char a = (char) 50;
char b = a;
b = (char) 51;
System.out.println("a: " + a + " b " + b);


String[] copyReferenceArray = referenceArray;
```

```
    System.out.println("copy" + copyReferenceArray); // expected output:
indentical reference.
    copyReferenceArray[0] = "3";
    System.out.println(
      "Changing The Data:" + '\n' +
      "referenceArray[0]: " + referenceArray[0] + '\n' +
      "copyReferenceArray[0]: " + copyReferenceArray[0]
    );


    System.out.print('\n' + "Initialization" + '\n');


    // 3. Initialization
    char[] charArray = new char[2];
    String[] stringArray = new String[2];


    System.out.println(charArray[0]); // expected output: does not throw error.
    System.out.println(stringArray[0]); // expected output: reference


    // 4. Everything is Object. Object stores reference.
    assert charArray instanceof Object;
    assert stringArray instanceof Object;


    /** Documentation:
    1. Where to store the actual data?
        Both primitive and referece array's attribute stores
        "reference" to the memory heap.
    2. Copying & Changing The Data conclusion:
        There is no difference between Primitive and Reference Array.
```

```
        There is difference between primitive data types and reference
        data types.
    3. Initialization
        Primitive Data Types always has a value. In theory empty
        Primitive Array should throw error but it does not.
        Reference Data Types can be null. There is still no evidence
        there is a difference between Primitive and Reference Array.
    4. Everything is Object. Object stores reference.
        Primitive Array && Rerference Array are an Object.
    */
  }
}
```