Array, Object Memory Reference Summary

Primitive Type

✤ Primitive type does not go into Java heap memory

✤ When used by functions, only the value is passed in

✤ Primitive type value will not be influenced by method

```java
public class MyProgram {
    public static void main(String[] args) {

        int a = 10;
        int b = function(a);
        System.out.print(a);
        System.out.print(b);

    }


    public int function(int a) {
        a = a + 10;

        return a;
    }
}
```

Object Type

* Object types are stored in Java heap memory

* When used by functions, the memory reference is passed in to the function

* Object type value will not be influenced by method, be careful

```
public class MyProgram {
        public static void main(String[] args) {

            int finalScore = 90;
            Student a = new Student(90);
            function(a);
            System.out.print(a.final);



        }


        public void function(Student a) {
                a.finalScore = 100;
        }
}
```

Java Heap Memory

```java
public class MyProgram {
    public static void main(String[] args) {

        int finalScore = 90;
        Student a = new Student(90);
        function(a);
        System.out.print(a.final);

    }


    public void function(Student a) {
        a.finalScore = 100;
    }
}
```

.

.

.

Address1: Student a

.

.

.

```java
public class MyProgram {
    public static void main(String[] args) {

        int[] myarray = {1,2,3,4,5};
        function(myarray);
        System.out.print(myarray[0]);

    }


    public void function(int[] input) {
        input[0] = 9;
    }
}
```

Object Copy

✤ Since Object types are stored in Java heap memory

✤ Directly use "=" will just point to the same memory address, this is shallow copy

✤ To deep copy, make sure all object values are copied and created a new object using new key word.

✤ For array, each value should be copied to the new array spot