Primitive data type vs. Object data type in Java with Examples

Difficulty Level : Easy   Last Updated : 22 Mar, 2021

Primitive Data Type: In Java, the primitive data types are the predefined data types of Java. They specify the size and type of any standard values. Java has 8 primitive data types namely byte, short, int, long, float, double, char and boolean. When a primitive data type is stored, it is the stack that the values will be assigned. When a variable is copied then another copy of the variable is created and changes made to the copied variable will not reflect changes in the original variable. Here is a Java program to demonstrate all the primitive data types in Java.

Object Data Type: These are also referred to as Non-primitive or Reference Data Type. They are so-called because they refer to any particular objects. Unlike the primitive data types, the non-primitive ones are created by the users in Java. Examples include arrays, strings, classes, interfaces etc. When the reference variables will be stored, the variable will be stored in the stack and the original object will be stored in the heap. In Object data type although two copies will be created they both will point to the same variable in the heap, hence changes made to any variable will reflect the change in both the variables. Here is a Java program to demonstrate arrays(an object data type) in Java.

Difference between the primitive and object data types in Java:

Now let’s look at a program that demonstrates the difference between the primitive and object data types in Java.

```java
import java.lang.*;
import java.util.*;

class GeeksForGeeks {
    public static void main(String ar[])
    {
        System.out.println("PRIMITIVE DATA TYPES\n");
        int x = 10;
        int y = x;
        System.out.print("Initially: ");
        System.out.println("x = " + x + ", y = " + y);
        // Here the change in the value of y
        // will not affect the value of x
        y = 30;

        System.out.print("After changing y to 30: ");
        System.out.println("x = " + x + ", y = " + y);
        System.out.println(
            "**Only value of y is affected here "
            + "because of Primitive Data Type\n");

        System.out.println("REFERENCE DATA TYPES\n");
        int[] c = { 10, 20, 30, 40 };

        // Here complete reference of c is copied to d
        // and both point to same memory in Heap
        int[] d = c;

        System.out.println("Initially");
        System.out.println("Array c: "
                           + Arrays.toString(c));
        System.out.println("Array d: "
                           + Arrays.toString(d));

        // Modifying the value at
        // index 1 to 50 in array d
        System.out.println("\nModifying the value at "
                           + "index 1 to 50 in array d\n");
        d[1] = 50;

        System.out.println("After modification");
        System.out.println("Array c: "
                           + Arrays.toString(c));
        System.out.println("Array d: "
                           + Arrays.toString(d));
        System.out.println(
            "**Here value of c[1] is also affected "
            + "because of Reference Data Type\n");
    }
}
```

Output

```
PRIMITIVE DATA TYPES


Initially: x = 10, y = 10
After changing y to 30: x = 10, y = 30
**Only value of y is affected here because of Primitive Data Type


REFERENCE DATA TYPES
```

```
Initially
Array c: [10, 20, 30, 40]
Array d: [10, 20, 30, 40]

Modifying the value at index 1 to 50 in array d

After modification
Array c: [10, 50, 30, 40]
Array d: [10, 50, 30, 40]
**Here value of c[1] is also affected because of Reference Data Type
```

Let’s look at the difference between the primitive and object data type in a tabular manner.

Properties	Primitive data types	Objects
Origin	Pre-defined data types	User-defined data types
Stored structure	Stored in a stack	Reference variable is stored in stack and the original object is stored in heap
When copied	Two different variables is created along with different assignment(only values are same)	Two reference variable is created but both are pointing to the same object on the heap
When changes are made in the copied variable	Change does not reflect in the original ones.	Changes reflected in the original ones.
Default value	Primitive datatypes do not have null as default value	The default value for the reference variable is null
Example	byte, short, int, long, float, double, char, boolean	array, string class, interface etc.