**Java null keyword**

**The null Keyword : Null Reference inside Instance Variable**

1. A reference variable refers to an object.
2. When a reference variable does not have a value (it is not referencing an object), such a reference variable is said to have a null value.



**Live Example : Null Value**

*class Rectangle {*

***double*** *length;*

***double*** *breadth;*

*}*

*class RectangleDemo {*

*public* ***static******void*** *main(String args[]) {*

*Rectangle myrect1;*

*System.out.println(myrect1.length);*

*}*

*}*

Output :

***javac RectangleDemo.java***

***Compile Time Error:***

*variable myrect1 might not have been initialized*

*System.out.println(myrect1.length); ^*

**Explanation : Null Value**

1. In the above example **myrect is not initialized**.
2. **Default value inside myrect1 is null**, means it does not contain any reference.
3. If you are declaring reference variable at “**Class Level**”, then you **don’t need to initialize instance variable** with **null**. (Above Error Message : error is at println stetement).

**Checking null Value**

*class Rectangle {*

***double*** *length;*

***double*** *breadth;*

*}*

*class RectangleDemo {*

*public* ***static******void*** *main(String args[]) {*

*Rectangle myrect1;*

***if****(myrect1 == null){*

*myrect1 = new Rectangle();*

*}*

*}*

*}*

We can check null value using “**==**” operator.

## Different Ways Of Null Value Statements:

### Way 1 : Class Level null Value

1. No need to initialize **instance variable with null**.
2. Instance Variable contains **default value as “null”**.
3. Meaning of “null” is that – **Instance Variable does not reference to any object**.

*Rectangle myrect1;*

is similar to –

*Rectangle myrect1 = null;*

### Way 2 : Method Level null Value

1. Suppose we have to create any object inside “Method” then we must initialize instance variable with Null value.
2. If we forgot to initialize instance variable with null, then it will throw compile time error.

**Valid Declaration :**

*Rectangle myrect1 = null;*

**Invalid Declaration :**

*Rectangle myrect1;*

**Java assigning object reference**

**Assigning Object Reference Variables: Class Concept in Java Programming**

1. We can assign value of reference variable to another reference variable.
2. Reference Variable is used to store the address of the variable.
3. Assigning Reference will not create distinct copies of Objects.
4. All reference variables are referring to same Object.

### Assigning Object Reference Variables does not –

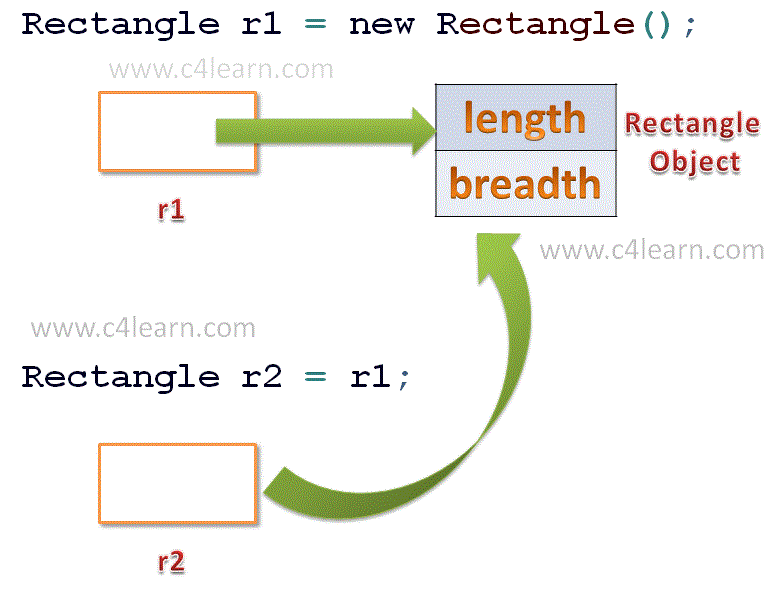
1. **Create Distinct Objects.**
2. **Allocate Memory**
3. **Create duplicate Copy**

**Consider This Example –**

Rectangle r1 = new Rectangle();

Rectangle r2 = r1;

* r1 is reference variable which contain the address of Actual Rectangle Object.
* r2 is another reference variable
* r2 is initialized with r1 means – “**r1 and r2**” both are referring same object, thus it does not create duplicate object, nor does it allocate extra memory.



**Live Example : Assigning Object Reference Variables**

*class Rectangle {*

***double*** *length;*

***double*** *breadth;*

*}*

*class RectangleDemo {*

*public* ***static******void*** *main(String args[]) {*

*Rectangle r1 = new Rectangle();*

*Rectangle r2 = r1;*

*r1.length = 10;*

*r2.length = 20;*

*System.out.println("Value of R1's Length : " + r1.length);*

*System.out.println("Value of R2's Length : " + r2.length);*

*}*

*}*

**Output :**

***java RectangleDemo***

*Value of R1's Length : 20.0*

*Value of R2's Length : 20.0*

**Typical Concept :**

Suppose we have assigned null value to r2 i.e

*Rectangle r1 = new Rectangle();*

*Rectangle r2 = r1;*

*...*

*r1 = null;*

**Note:**

Still r2 contain reference to an object. Thus We can create have multiple reference variables to hold single object.