Pass by value and immutable Object :
---

* In java, few objects are **immutable object** (un-modifiable object) that means, If we modify the object then Original existing object will not be modified, the new changes will be done in another memory location.

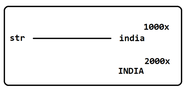
* String class and all the Wrapper classes are immutable object.

//Program with Diagram

```
public class ImmutableObject {

    public static void main(String[] args)
    {
        String str = "india";
        str.toUpperCase();
        System.out.println(str); //india
    }

}
```




```
    String str = "india";
    str = str.toUpperCase();
    System.out.println(str);
```

```
package com.ravi.pass_by_value;

public class PassByValueDemoS
{
    public static void main(String[] args)
    {
        String str = new String("Java"); //immutable
        accept(str);
        System.out.println(str);
    }

    public static void accept(String s1)
    {
        s1 = "technology";
    }
}
```

What is Garbage Collector in java ?
---

* It is a daemon thread which is running in the background to provide **memory management** in java.

* As we know all the objects are created in the HEAP memory, Garbage Collector (GC) will scan the heap memory to perform object clean up operation, It will automatically delete the object which are eligible for GC that the object which does not contain any references.

* Internally it uses an algorithm called **"Mark and sweep"** algorithm to delete the un-used objects from the HEAP memory.

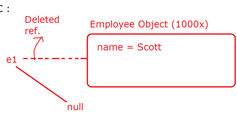
* At regular interval, It will visit the HEAP memory to delete the objects, We can explicitly call GC by using gc() static method of System class.

How many ways we can make an Object eligible for GC :

There are 3 ways to make an Object eligible for GC :

1) Assigning null literal to the reference variable :

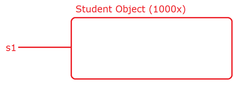
```
Employee e1 = new Employee("Scott");
        e1 = null;
```



Earlier, e1 reference variable was pointing to 1000x memory location but now e1 is pointing to null so automatically the object created at 1000x memory location will be eligible for GC.

2) Creating an Object inside a method :

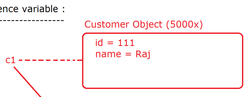
```
public void objectCreation()
{
    Student s1 = new Student();
}
```

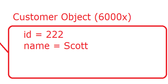


s1 is a local reference variable so once the method execution is over, s1 will be deleted from the Method Stack Frame so Student object created at 1000x memory location will be eligible for GC.

3) Assigning a new Object to the existing reference variable :

```
Customer c1 = new Customer(111,"Raj");
c1 = new Customer(222, "Scott");
```





Earlier, c1 reference variable was pointing to 5000x memory location but now it is pointing to 6000x memory location so automatically 5000x memory location will be eligible for GC

HEAP and STACK Diagram

After learning this concept, We will get :
1) Internal Working of Object
2) Output of any complex program
3) We can say easily, How many objects are eligible for GC

Demo Program on HEAP and STACK Memory :

```
public class Student
{
   private int roll;

   public static void main(String [] args)
   {
      int local = 500;
      Student s1 = new Student();
      s1 = new Student();
   }
}
```

HEAP MEMORY
1000x : StudentObject, roll : 0
2000x : StudentObject, roll : 0

STACK MEMORY
main_stack
local : 500
s1 : ~~1000x~~ 2000x

Points :
---

* In java whenever we execute a program (java Student) then JVM will get some memory from the Operating System, This memory is divided into two section

1) HEAP MEMORY SECTION
2) STACK MEMORY SECTION

* ANY PRIMTIVE VARIABLE WILL DIRECTLY HOLD THE **VALUE**
* ANY REFERENCE VARIABLE WILL DIRECTLY HOLD THE **ADDRESS**

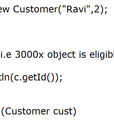
HEAP and STACK Digram for CustomerDemo.java

HEAP MEMORY
1000x : CustomerObject, id : ~~2~~, 5 name :2000x
2000x : String Object, Ravi
3000x : CustomerObject, id : ~~7~~, 9 name :4000x
4000x : String Obejct, Rahul

Output :

9
5

STACK MEMORY
main_stack
val : 100
c : 1000x



m1 method execution is over so it will be deleted from the Stack Frame

String object is never eligible for GC

```
class Customer
{
    private String name;
    private int id;

    public Customer(String name , int id)
    {
        super();
        this.name=name;
        this.id=id;
    }

    public void setId(int id)
    {
        this.id=id;
    }

    public int getId()
    {
        return this.id;
    }
}

public class CustomerDemo
{
    public static void main(String[] args)
    {
        int val = 100;

        Customer c = new Customer("Ravi",2);

      m1(c);

        //only 1 Object  i.e 3000x object is eligible for GC

        System.out.println(c.getId());
    }

    public static void m1(Customer cust)
    {
        cust.setId(5);

       cust = new Customer("Rahul",7);

        cust.setId(9);
        System.out.println(cust.getId());
    }
}
```