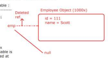
How many ways we can make an object eligible for Garbge Collector :

* Basically, There are 3 ways to make an object eligible for Garbage Collector.

1) Assigning a null literal on the reference variable :

```
Employee emp = new Employee(111, "Scott");
           emp = null;
```



emp reference variable was pointing to 1000x memory location but now emp reference variable is pointing to null hence Employee object created at 1000x memory location is eligible for GC.

2) Creating an object inside the method body :

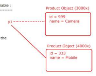
```
public void createObject()
{
  Employee e2 = new Employee();
}
```

e2 is a local reference variable which is holding the object address of Employee object, Once the method execution is over then e2 variable will be deleted from the stack frame so automatically Employee object is eligible for GC.

3) Assigning a new object on the existing reference variable :

```
Product p1 = new Product(999,"Camera");

        p1 = new Product(333,"Mobile");
```



p1 reference variable is pointing to 3000x memory location but when we assign a new object(4000x) on the p1 reference variable then automatically the product object created at 3000x memory location will be eligible for GC.

Memory In Java :

1) Method Area OR Class Area :

All the staic fields and static members are available in the Method Area.

2) HEAP Area :

All the non static fields and non static method are available in the HEAP Area.

3) STACK Area :

* All the local and parameter variables are available in the Stack Area.
* All these variables are stored in the Stack Frame, Once method execution is over then Stack Frame will be deleted and now with that Stack frame automatically local and parameter variables are also deleted.
* Whether, It is a static OR non static method but methods are always executed from STACK Area.

HEAP AND STACK Diagram :

Example :

```
public class Student
{
   private int roll = 101;

   public static void main(String [] args)
   {
       int val = 100;
       Student s1 = new Student();
       s1 = new Student();
   }
}
```

HEAP MEMORY

1000x : StudentObject, id : 101
2000x : StudentObject, id : 101

STACK MEMORY

main_stack

val : 100
s1 : ~~1000x~~ 2000x

* Whenever we execute any java program then JVM will get some memory from the Operating System, that entire memory is divided into two sections
  a) HEAP MEMORY
  b) STACK MEMORY

  javac Student.java
  java Student -> Execute
    a) HEAP MEMORY
    b) STACK MEMORY

HEAP and STACK Diagram for CustomerDemo.java :

HEAP MEMORY

1000x : CustomerObject, name :2000x, id : ~~2~~ 5
2000x : StringObject, Ravi
3000x : CustomerObject, name :4000x , id : ~~7~~ 9
4000x : StringObject, Rahul

STACK MEMORY

main_stack

val : 100
c : 1000x

m1_stack (crossed out)

cust : ~~1000x~~ 3000x

String object created in the String constant pool by using String literal is not eligible for Garbage Collector.

```
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 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());

    }
}
//Output :  9  5
```