Garbage Collection in Java

- In C/C++, programmer is responsible for both creation and destruction of objects. Usually programmer neglects destruction of useless objects. Due to this negligence, at certain point, for creation of new objects, sufficient memory may not be available and entire program will terminate causing **OutOfMemoryErrors**.

- But in Java, the programmer need not to care for all those objects which are no longer in use. Garbage collector destroys these objects.

- Garbage Collection is the process of **freeing** the memory blocks occupied by unreferenced objects **automatically** at runtime. In other words, it is a way to destroy the unused objects.
- Advantage of Garbage Collection
  - It makes java **memory efficient** because garbage collector removes the unreferenced objects from heap memory.
  - It is **automatically done** by the garbage collector(a part of JVM) so programmer don't need to make extra efforts.

How can an object be unreferenced?

- There are many ways:
  - By nulling the reference
  - By assigning a reference to another
  - By anonymous object etc.
  - Local object in a method not returned or received by calling method.

1) By nulling a reference:

```
Employee e=new Employee();
e=null;
```

2) By assigning a reference to another:

```
Employee e1=new Employee();
Employee e2=new Employee();
e1=e2;
//now the first object referred by e1 is available for garbage
collection
```

3) By annonymous object:

new Employee();

4) Local object in a method not returned or received by calling method:

```
class Alpha
{
  int a;
     void change()
         {
             Alpha al = new Alpha();
               al.a =10;
          }
}
```

```
class Test
{
    public static void main(String args[])
     {   Alpha a = new Alpha();
          a.change();
     }
}
```

```
class Test
{
    private Demo d;
    void start()
    {
        d = new Demo();
        this.takeDemo(d);
    }
    void takeDemo(Demo demo)
    {
        demo = null;
        demo = new Demo();
    }
}
```

```
public class  Alpha
{
    public static void main(String [] args)
    {
        Alpha x = new Alpha();
        Alpha x2 = fun(x);
        Alpha x4 = new Alpha();
        x2 = x4;
}
    static Alpha fun(Alpha mx)
    {
        mx = new Alpha();
        return mx;
    }}
```

```
class X2
{
    public X2 x;
    public static void main(String [] args)
    {
        X2 x2 = new X2();
        X2 x3 = new X2();
        x2.x = x3;
        x3.x = x2;
        x2 = new X2();
        x3 = x2;

    }
}
```

```
class Node {
    public  Object value;
    public Node next;
    public Node(Object o, Node n) { value = 0; next = n;}
}

//...some code
{
    Node a = new Node("a", null),
         b = new Node("b", a),
         c = new Node("c", b);
    a.next = c;
} //end of scope
```

finalize() Method in Java

- This method is part of class of Object.
- The **finalize()** is called by the garbage collector on an object when garbage collection determines that there are no more references to the object.
- Declaration of **finalize()** method:

  protected void finalize();

We can override this method in our class to free the unused resources.