Garbage Collection

OUTLINE

- What is Garbage Collection?
- Making objects eligible for GC
- Requesting JVM to run garbage collection
- finalize() method

Garbage Collection

- Garbage Collection is the process of automatically identifying and deleting unused objects from the memory (Heap) to free up space.
- Java objects are stored in **Heap Memory** .

➢ Who Performs Garbage Collection?

JVM (Java Virtual Machine)

- GC is handled by JVM, not by programmer
- Runs in the **background**

Note:

Programmer **cannot force** GC, only **request** it

Garbage Collection

- Purpose
  - Find and delete unreachable objects.
  - Free space as much as possible.
  - Improves **application performance**
  - Programmer does **not** need to delete objects manually.

When does an object Become Garbage?

- Even though Programmer is Not Responsible to Destroy Useless Objects but it is Highly Recommended to Make an Object Eligible for GC if it is No Longer required.
- An Object is said to be Eligible for GC if and Only if it doesn't contain any **Reference**.

➢ Ways for making objects eligible for collection

- Nulling a reference
- Reassigning a reference variable
- Isolating a reference

1) **Nullifying the Reference Variable:-** If an Object is No Longer required, then Assign null to all its Reference Variables, Then that Object Automatically Eligible for Garbage Collection

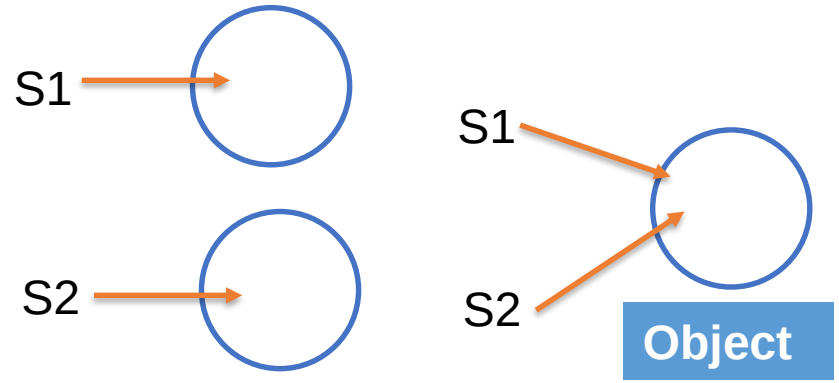
Example-

```
Student s1 = new Student();
s1 = null;
```


2) **Re Assigning the Reference Variable:-** If an Object is No Longer required then Re Assign its Reference Variable to any other object then Old Object is Automatically Eligible for GC.

Example-

```
Student s1 = new Student();
Student s2 = new Student();
s2=s1;
```



Requesting JVM to Run Garbage Collection

- Once we Made an Object Eligible for GC, it May Not Destroy Immediately by the Garbage Collector. Whenever JVM Runs Garbage Collector then Only Object will be Destroyed.
- But when exactly JVM runs GC, We can't Expect. It Depends on JVM and varied from **JVM to JVM.**
- Instead of waiting until JVM Runs GC, we can **Request JVM** to Run Garbage Collector.
- But there is No Guarantee whether JVM Accept Our Request OR Not. But Most of the Times JVM Accepts Our Request.

Ways to requesting JVM to Run Garbage Collector-

1) By Using System Class:-

System Class contains a Static Method gc() for this Purpose.

Example-

```
System.gc();
```

2) By Using Runtime Class:-

- A Java Application can Communicate with JVM by using Runtime Object. Runtime Class Present in java.lang **Package** and it is a **Singleton Class**.
- We can Create a Runtime Object by using **getRuntime().**

```
Runtime r = Runtime.getRuntime();
```

- Once we got Runtime Object we can Call this Method on that Object—

  gc():- Requesting JVM to Run Garbage Collector

NOTE-

- gc() method present in System Class is **Static** Method whereas gc() method Present in Runtime Class is **Instance** Method.

- With Respect to Performance,it is Recommended to Use Runtime class gc() method when compared with System class gc() method ,because Internally **System.gc()** method calls Runtime class gc() method.


```
class System {
        public static void gc() {
                Runtime.getRuntime().gc();
        }
}
```

Finalization

- Method called by JVM **before object is destroyed**
- Used for cleanup activities
- Once finalize() Completes their execution after that Automatically GC Destroys that Object.
- Called only once per object
- Not guaranteed to execute
- The finalize( ) method is a method of Object class [java.lang.Object.finalize()] with the following Prototype:-


```
protected void finalize() throws Throwable
```

- Based on Our Requirement we can Override finalize() method in Our Class to define our own Cleanup Activities.

Case 1

- Just before Destroying an Object Garbage Collector Always Calls finailze() on that Object, then the Corresponding Class finalize() will be executed.
- For Example, if String Object Eligible for GC, then String Class finalize() will be executed,

```
class DemoGc {
        public static void main(String[] args) {
                DemoGc s = new DemoGc();
                s = null;
                System.gc();
                System.out.println("End of main");
         }
         public void finalize(){
                System.out.println(" Finalize method called ");
          }
    }
```

Case 2

- Based on Our Requirement we can Call finalize() method Explicitly, then it will be executed Just Like a Normal Method Call and Object won't be Destroyed. But before destroying an Object Garbage Collector Always Calls finalize() method.
- Example-

```
public class Test {
        public static void main(String[] args) {
                Test t = new Test();
                t.finalize();
                t.finalize();
                t=null;
                System.gc();
                System.out.println("end of main()");
        }
        public void finalize() {
                System.out.println("finalize called");
        }
}
```

THANK YOU!!