Garbage Collection

- Garbage collection in Java is the process by which Java programs perform automatic memory management.
- Java programs compile to bytecode that can be run on a Java Virtual Machine, or JVM for short.
- When Java programs run on the JVM, objects are created on the heap, which is a portion of memory dedicated to the program.
- Eventually, some objects will no longer be needed.
- The garbage collector finds these unused objects and deletes them to free up memory.

Garbage Collection

What is Garbage Collection?

- In C/C++, a programmer is responsible for both the creation and destruction of objects.
- Usually, programmer neglects the destruction of useless objects.
- Due to this negligence, at a certain point, sufficient memory may not be available to create new objects, and the entire program will terminate abnormally, causing OutOfMemoryErrors.
- But in Java, the programmer need not care for all those objects which are no longer in use. Garbage collector destroys these objects.
- The main objective of Garbage Collector is to free heap memory by destroying unreachable objects.

Garbage Collection

How Does Garbage Collection in Java works?

- Java garbage collection is an automatic process.
- Automatic garbage collection is the process of looking at heap memory, identifying which objects are in use and which are not, and deleting the unused objects.
- An in-use object, or a referenced object, means that some part of your program still maintains a pointer to that object. An unused or unreferenced object is no longer referenced by any part of your program.
- So the memory used by an unreferenced object can be reclaimed. The programmer does not need to mark objects to be deleted explicitly.
- The garbage collection implementation lives in the JVM.

Garbage Collection

How Does Garbage Collection in Java works?

- Java garbage collection is an automatic process.
- Automatic garbage collection is the process of looking at heap memory, identifying which objects are in use and which are not, and deleting the unused objects.
- An in-use object, or a referenced object, means that some part of your program still maintains a pointer to that object. An unused or unreferenced object is no longer referenced by any part of your program.
- So the memory used by an unreferenced object can be reclaimed. The programmer does not need to mark objects to be deleted explicitly.
- The garbage collection implementation lives in the JVM.

Garbage Collection

Important Concepts Related to Garbage Collection in Java

1. Unreachable objects: An object is said to be unreachable if it doesn’t contain any reference to it. Also, note that objects which are part of the island of isolation are also unreachable.

```
Integer i = new Integer(4);
// the new Integer object is reachable  via the reference in 'i'
i = null;
// the Integer object is no longer reachable.
```

Garbage Collection

Important Concepts Related to Garbage Collection in Java

2. Eligibility for garbage collection:

An object is said to be eligible for GC(garbage collection) if it is unreachable. After i = null, integer object 4 in the heap area is suitable for garbage collection in the above example.

Garbage Collection

Ways to make an object eligible for Garbage Collector:

Even though the programmer is not responsible for destroying useless objects but it is highly recommended to make an object unreachable(thus eligible for GC) if it is no longer required.

There are generally four ways to make an object eligible for garbage collection.

- Nullifying the reference variable
- Re-assigning the reference variable
- An object created inside the method
- Island of Isolation

Garbage Collection

Ways for requesting JVM to run Garbage Collector

Once we make an object eligible for garbage collection, it may not destroy immediately by the garbage collector. Whenever JVM runs the Garbage Collector program, then only the object will be destroyed. But when JVM runs Garbage Collector, we can not expect.

We can also request JVM to run Garbage Collector. There are two ways to do it :

Using System.gc() method: System class contain static method gc() for requesting JVM to run Garbage Collector.

Using Runtime.getRuntime().gc() method: Runtime class allows the application to interface with the JVM in which the application is running. Hence by using its gc() method, we can request JVM to run Garbage Collector.

There is no guarantee that any of the above two methods will run Garbage Collector.

The call System.gc() is effectively equivalent to the call : Runtime.getRuntime().gc()

Finalization

Just before destroying an object, Garbage Collector calls finalize() method on the object to perform cleanup activities. Once finalize() method completes, Garbage Collector destroys that object.

finalize() method is present in Object class with the following prototype.

protected void finalize() throws Throwable

Garbage Collection

Based on our requirement, we can override finalize() method for performing our cleanup activities like closing connection from the database.

The finalize() method is called by Garbage Collector, not JVM. However, Garbage Collector is one of the modules of JVM.
Object class finalize() method has an empty implementation. Thus, it is recommended to override the finalize() method to dispose of system resources or perform other cleanups.
The finalize() method is never invoked more than once for any object.
If an uncaught exception is thrown by the finalize() method, the exception is ignored, and the finalization of that object terminates.
Advantages of Garbage Collection in Java
The advantages of Garbage Collection in Java are:

It makes java memory-efficient because the garbage collector removes the unreferenced objects from heap memory.
It is automatically done by the garbage collector(a part of JVM), so we don’t need extra effort.