Garbage Collection in Java

BY
M. BABY ANUSHA,
ASST.PROF IN CSE DEPT.,
RGUKT,NUZVID

Introduction :

- 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 abnormally 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

- Java Garbage collector tracks the live object and objects which are no more need are marked for garbage collection. It relieves developers to think of memory allocation/deallocation issues.
- JVM uses the heap, for dynamic allocation. In most of the cases, the operating systems allocate the heap in advance which is then to be managed by the JVM while the program is running.

Garbage Collection

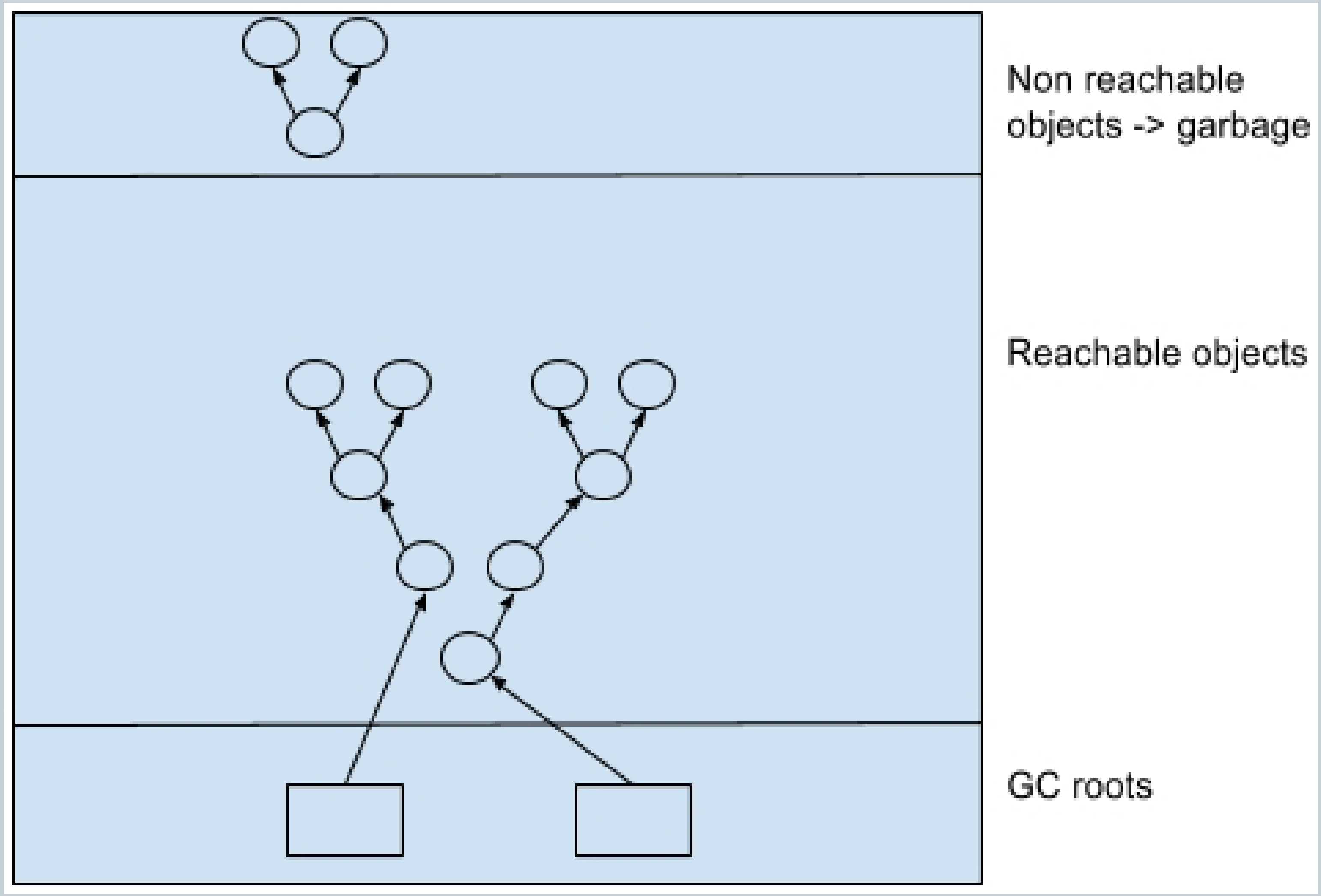
It helps in following ways –

- Faster object creation is possible when heap memory is used.. Object Allocation takes some memory .
- When an object is not required, garbage collector reuses the object's memory for further allocation.

Garbage Collection

- As objects forms tree, they have one or more root objects. If root objects are reachable, the whole tree is reachable.
- There are some special objects as well which are garbage collection roots (GC roots) and are always reachable.

Garbage Collection



Garbage Collection

- Main objective of Garbage Collector is to free heap memory by destroying **unreachable objects**.

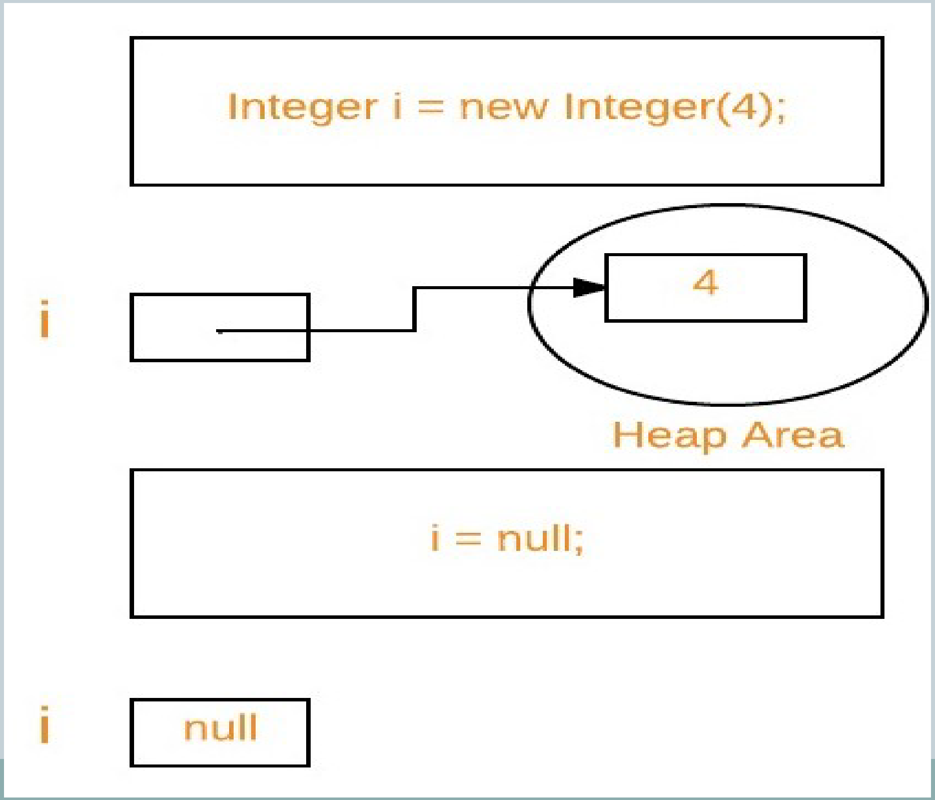
Important terms :

- **Unreachable objects** : An object is said to be unreachable iff it doesn’t contain any reference to it.
- **Eligibility for garbage collection** : An object is said to be eligible for GC(garbage collection) iff it is unreachable.

Garbage Collection

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.



Ways to make an object eligible for GC

- Even though the programmer is not responsible to destroy 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 different ways to make an object eligible for garbage collection.
  - Nullifying the reference variable
  - Re-assigning the reference variable
  - Object created inside method

Continued…

- Once we made 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.

Request JVM to run Garbage Collector

- 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

```
// Java program to demonstrate requesting
// JVM to run Garbage Collector
public class Test
{
   public static void main(String[] args) throws InterruptedException
   {
      Test t1 = new Test();
      Test t2 = new Test();

      // Nullifying the reference variable
      t1 = null;

      // requesting JVM for running Garbage Collector
      System.gc();

      // Nullifying the reference variable
      t2 = null;

      // requesting JVM for running Garbage Collector
      Runtime.getRuntime().gc();

   }
// finalize method is called on object once
    // before garbage collecting it
    protected void finalize() throws Throwable
    {
        ; System.out.println("Garbage collector called");
        System.out.println("Object garbage collected : " + this)
    }
}
```

Output:

```
Garbage collector called Object garbage collectedTest@46d08f12
Garbage collector called Object garbage collected : Test@481779b8
```

Note :

- There is no guarantee that any one of above two methods will definitely 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 following prototype.

  protected void finalize() throws Throwable

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

Thank you!
shutterstock.com · 567687052