Garbage Collection in Java

BY CHAITANYA SINGH | FILED UNDER: OOPS CONCEPT

When JVM starts up, it creates a heap area which is known as runtime data area. This is where all the objects (instances of class) are stored. Since this area is limited, it is required to manage this area efficiently by removing the objects that are no longer in use. The process of removing unused objects from heap memory is known as **Garbage collection** and this is a part of memory management in Java.

Languages like C/C++ **don't** support automatic garbage collection, however in java, the garbage collection is automatic.

Now we know that the garbage collection in java is automatic. Lets see when does java performs garbage collection.

When does java perform garbage collection?

1. When the object is no longer reachable:

```
BeginnersBook obj = new BeginnersBook();
obj = null;
```

Here the reference obj was pointing to the object of class BeginnersBook but since we have assigned a null value to it, this is no longer pointing to that object, which makes the BeginnersBook object unreachable and thus unusable. Such objects are automatically available for garbage collection in Java.

Another example is:

```
char[] sayhello = { 'h', 'e', 'l', 'l', 'o'};
String str = new String(sayhello);
str = null;
```

Here the reference str of String class was pointing to a string “hello” in the heap memory but since we have assigned the null value to str, the object “hello” present in the heap memory is unusable.

2. When one reference is copied to another reference:

```
BeginnersBook obj1 = new BeginnersBook();
BeginnersBook obj2 = new BeginnersBook();
obj2 = obj1;
```

Here we have assigned the reference obj1 to obj2, which means the instance (object) pointed by (referenced by) obj2 is not reachable and available for garbage collection.

How to request JVM for garbage collection

We now know that the unreachable and unusable objects are available for garbage collection but the garbage collection process doesn’t happen instantly. Which means once the objects are ready for garbage collection they must to have to wait for JVM to run the memory cleanup program that performs garbage collection. However you can request to JVM for garbage collection by calling **System.gc()** method (see the example below).

Garbage Collection Example in Java

In this example we are demonstrating the garbage collection by calling System.gc(). In this code we have overridden a finalize() method. This method is invoked just before a object is destroyed by java garbage collection process. This is the reason you would see in the output that this method has been invoked twice.

```
public class JavaExample{
   public static void main(String args[]){
        /* Here we are intentionally assigning a null
         * value to a reference so that the object becomes
         * non reachable
         */
        JavaExample obj=new JavaExample();
        obj=null;

        /* Here we are intentionally assigning reference a
         * to the another reference b to make the object referenced
         * by b unusable.
         */
        JavaExample a = new JavaExample();
        JavaExample b = new JavaExample();
        b = a;
        System.gc();
   }
   protected void finalize() throws Throwable
   {
        System.out.println("Garbage collection is performed by JVM");
   }
}
```

Output:

```
Garbage collection is performed by JVM
Garbage collection is performed by JVM
```