* The [garbage collector](http://en.wikipedia.org/wiki/Garbage_collection_%28computer_science%29) is a program which runs on the [Java Virtual Machine](http://en.wikipedia.org/wiki/Java_Virtual_Machine) which gets rid of objects which are not being used by a Java application anymore.
* It is a form of *automatic memory management*.
* When a typical Java application is running, it is creating new objects, such as Strings and Files, but after a certain time, those objects are not used anymore. For example, take a look at the following code:

for (File f : files) {

String s = f.getName();

}

* In the above code, the String s is being created on each iteration of the for loop. This means that in every iteration, a little bit of memory is being allocated to make a String object.
* Going back to the code, we can see that once a single iteration is executed, in the next iteration, the String object that was created in the previous iteration is not being used anymore -- that object is now considered "garbage".
* Eventually, we'll start getting a lot of garbage, and memory will be used for objects which aren't being used anymore. If this keeps going on, **eventually the Java Virtual Machine will run out of space to make new objects.**

That's where the garbage collector steps in.

* The garbage collector will look for objects which aren't being used anymore, and gets rid of them, freeing up the memory so other new objects can use that piece of memory.
* In Java, memory management is taken care of by the garbage collector, but in other languages such **as C, one needs to perform memory management on their own using functions such as [malloc and free](http://en.wikipedia.org/wiki/Malloc).**[**Memory management**](http://en.wikipedia.org/wiki/Memory_management)**is one of those things which are easy to make mistakes, which can lead to what are called**[**memory leaks**](http://en.wikipedia.org/wiki/Memory_leak) -- places where memory is not reclaimed when they are not in use anymore.
* Automatic memory management schemes like garbage collection makes it so the programmer does not have to worry so much about memory management issues, so he or she can focus more on developing the applications they need to develop.

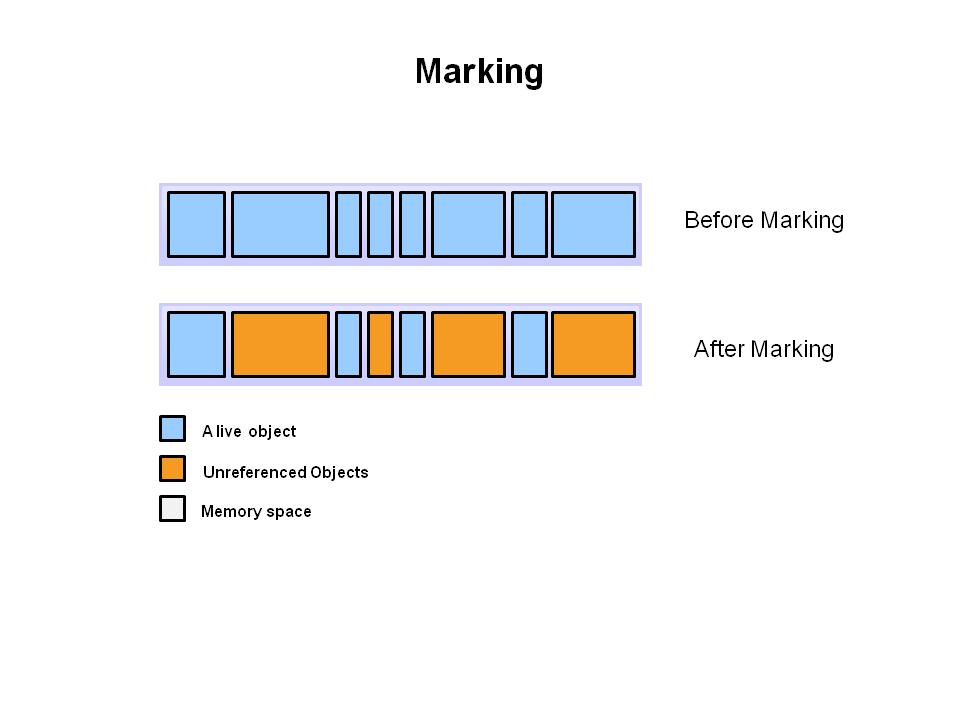
**What is Automatic Garbage Collection?**

**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 object, or unreferenced object, is no longer referenced by any part of your program. So the memory used by an unreferenced object can be reclaimed.

In a programming language like C, allocating and deallocating memory is a manual process. In Java, process of deallocating memory is handled automatically by the garbage collector. The basic process can be described as follows.

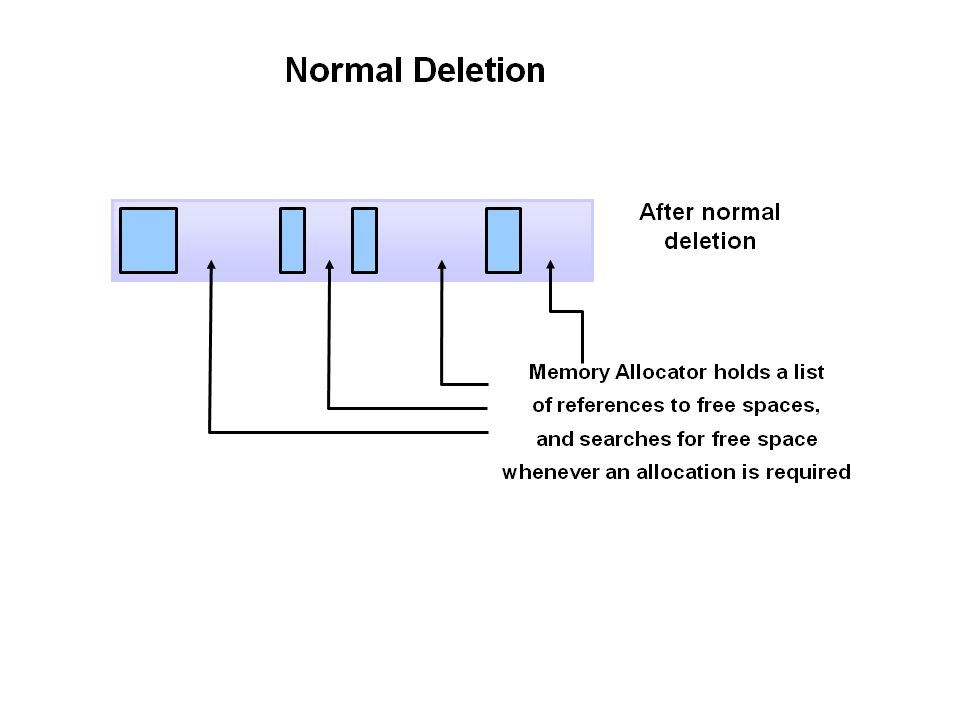
**Step 1: Marking**

The first step in the process is called marking. This is where the garbage collector identifies which pieces of memory are in use and which are not.



Referenced objects are shown in blue. Unreferenced objects are shown in gold. All objects are scanned in the marking phase to make this determination. This can be a very time consuming process if all objects in a system must be scanned.

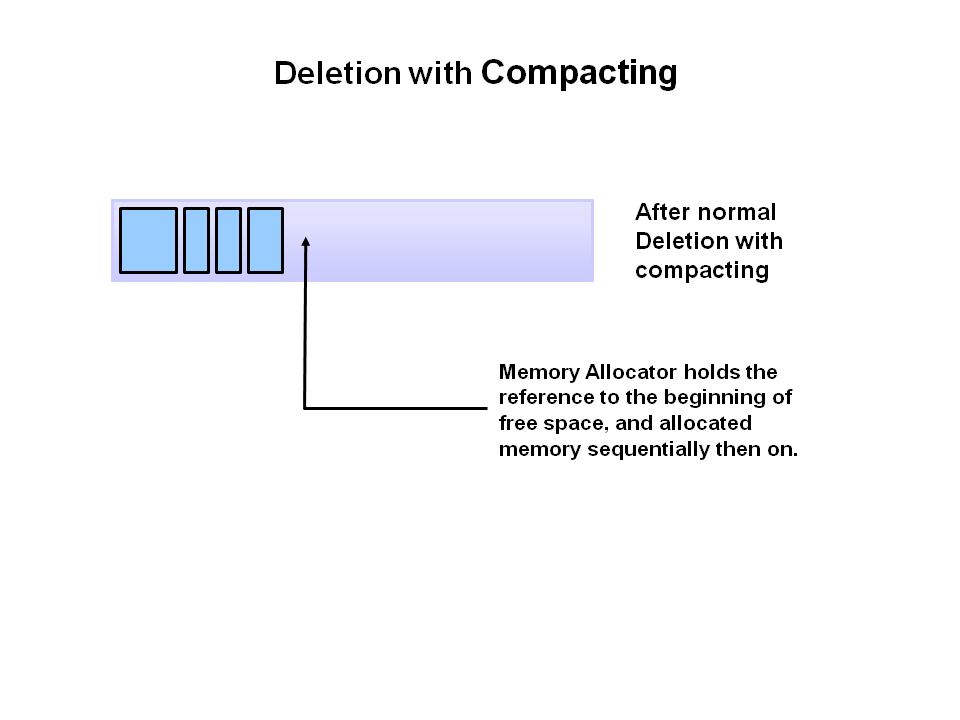
**Step 2: Normal Deletion**

Normal deletion removes unreferenced objects leaving referenced objects and pointers to free space.

The memory allocator holds references to blocks of free space where new object can be allocated.

**Step 2a: Deletion with Compacting**

To further improve performance, in addition to deleting unreferenced objects, you can also compact the remaining referenced objects. By moving referenced object together, this makes new memory allocation much easier and faster.



**Why Generational Garbage Collection?**

As stated earlier, having to mark and compact all the objects in a JVM is inefficient. As more and more objects are allocated, the list of objects grows and grows leading to longer and longer garbage collection time. However, empirical analysis of applications has shown that most objects are short lived.

1) Objects are created on the heap in Java irrespective of their scope e.g. local or member variable. while it's worth noting that class variables or static members are created in method area of [Java memory space](http://javarevisited.blogspot.com/2011/05/java-heap-space-memory-size-jvm.html) and both heap and method area is shared between different thread.  
  
2**) Garbage collection is a mechanism provided by Java Virtual Machine to reclaim heap space from objects which are eligible for Garbage collection.**  
  
3) Garbage collection relieves Java programmer from memory management which is an essential part of C++ programming and gives more time to focus on business logic.  
  
4) [**Garbage Collection in Java**](http://javarevisited.blogspot.com/2011/04/garbage-collection-in-java.html)**is carried by a daemon thread called Garbage Collector.**  
5) Before removing an object from memory, garbage collection thread invokes [finalize() method](http://javarevisited.blogspot.com/2012/03/finalize-method-in-java-tutorial.html) of that object and gives an opportunity to perform any sort of cleanup required.  
  
6) You as Java programmer can not force [garbage collection in Java](http://javarevisited.blogspot.com/2011/04/garbage-collection-in-java.html); it will only trigger if JVM thinks it needs a garbage collection based on Java heap size.  
  
7) **There are methods like** **System.gc() and Runtime.gc() which is used to send request of Garbage collection to JVM but it’s not guaranteed that garbage collection will happen.**  
8) If there is no memory space for creating a new object in Heap Java Virtual Machine throws OutOfMemoryError or [java.lang.OutOfMemoryError heap space](http://javarevisited.blogspot.com/2011/05/java-heap-space-memory-size-jvm.html)  
  
When an Object becomes Eligible for Garbage Collection

## **An object becomes eligible for Garbage collection or GC if it's not reachable from any live threads or by any static references.**

## **In other words, you can say that an object becomes eligible for garbage collection if its all references are null**. Cyclic dependencies are not counted as the reference so if object A has a reference to object B and object B has a reference to Object A and they don't have any other live reference then both Objects A and B will be eligible for Garbage collection. Generally, an object becomes eligible for [garbage collection in Java](http://javarevisited.blogspot.com/2011/04/garbage-collection-in-java.html) on following cases: 1) **All references to that object explicitly set to null e.g. object = null** 2) The object is created inside a block and reference goes out scope once control exit that block. 3) Parent object set to null if an object holds the reference to another object and when you set container object's reference null, child or contained object automatically becomes eligible for garbage collection. 4) If an object has only [lived weak references](http://javarevisited.blogspot.sg/2014/03/difference-between-weakreference-vs-softreference-phantom-strong-reference-java.html) via WeakHashMap it will be eligible for garbage collection.

An object becomes eligible for Garbage collection or GC if it's not reachable from any live threads or by any static references.

In other words, you can say that an object becomes eligible for garbage collection if its all references are null.

#### Cyclic dependencies are not counted as the reference so if object A has a reference to object B and object B has a reference to Object A and they don't have any other live reference then both Objects A and B will be eligible for Garbage collection. What is Garbage Collection in Java?

* Garbage Collection is an automatic memory management feature.
* The process of destroying unreferenced objects is called Garbage Collection.
* Once object is unreferenced it is considered as unused object, hence JVM automatically destroys that object.
* In java developers responsibility is only to creating objects and unreferencing those objects after usage.

2.How JVM can destroy unreferenced object?

* **JVM internally uses a daemon thread called "garbage collector" to destroy all unreferenced objects.**
* **A daemon thread is a service thread. Garbage Collector thread is called daemon thread because it provides services to JVM to destroy unreferenced objects.**
* This thread is low priority thread. Since it is a low priority thread we can not guarantee this execution.

 3.So can you guarantee objects destruction?

* No, we can not guarantee objects destruction even though it is unreferenced, because we can not guarantee garbage collector execution.
* So, we can confirm whether object is eligible for garbage collection or not.

4.Can we force garbage collector?

* No, we can not force garbage collector to destroy objects , but we can request it.

5.How can we request JVM to start garbage collection process?

* We have a method called **gc()**in system class as static method and also in Runtime class as non static method to request JVM to start garbage collector execution.
* System.gc();
* Runtime.getRuntime().gc();

**6.What is the algorithm JVM internally uses for destroying objects?**

* **"mark and swap" is the algorithm JVM internally uses.**

7.Which part of the memory is involved in Garbage Collection?

* Heap.

8.What is responsibility of Garbage Collector?

* Garbage Collector frees the memory occupied by the unreachable objects during the java program by deleting these unreachable objects.
* It ensures that the available memory will be used efficiently, but does not guarantee that there will be sufficient memory for the program to run.

9. When does an object become eligible for garbage collection?

* An object becomes eligible for garbage collection when no live thread can access it.

10. What are the different ways to make an object eligible for garbage collection when it is no longer needed?

* Set all available object references to "null" once the purpose of creating object is served.

1. package com.instanceofjava;
3. class GarbageCollectionTest1{
5. public static void main(String [] args){
7. String str="garbage collection interview questions";
8. // String object referenced by variable str and is not eligible for GC yet.
10. str=null;
11. //String object referenced by variable str is eligible for GC
12. }
13. }

* Make the reference variable to refer to another object. Decouple the reference variable from the object and set it refer to another object, so the object which was referring to before reassigning is eligible for Garbage Collection

1. package com.instanceofjava;
3. class GarbageCollectionTest2{
5. public static void main(String [] args){
7. String str1="garbage collection interview questions";
8. String str2="Top 15 garbage collection interview questions";
9. // String object referenced by variable str1 and str2 and is not eligible for GC yet.
11. str1=str2;
12. //String object referenced by variable str1 is eligible for GC
14. }
15. }

**11.What is purpose of overriding finalize() method?**

* **The finalize() method should be overridden for an object to include the clean up code or to dispose of the system resources that should to be done before the object is garbage collected.**

**12.How many times does the garbage collector calls the finalize() method for an object?**

* **Only once.**

13.What happens if an uncaught exception is thrown from during the execution of finalize() method of  an object?

* The exception will be ignored and the garbage collection (finalization) of that object terminates

14.What are the different ways to call garbage collector?

* System.gc();
* Runtime.getRuntime().gc();

15. How to enable /disable call of finalize() method of exit of application?

* Runtime.getRuntime().runFinalizersOnExit(boolean value). passing the boolean value  true and false will enable or disable the finalize() call.
* **Q1) Which part of the memory is involved in Garbage Collection? Stack or Heap?**
* **Ans)** Heap
* **Q2)What is responsiblity of Garbage Collector?**
* **Ans)** Garbage collector frees the memory occupied by the unreachable objects during the java program by deleting these unreachable objects.  
  It ensures that the available memory will be used efficiently, but does not guarantee that there will be sufficient memory for the program to run.
* **Q3) Is garbage collector a dameon thread?**
* **Ans)** Yes GC is a dameon thread. A dameon thread runs behind the application. It is started by JVM. The thread stops when all non-dameon threads stop.
* **Q4)How is Garbage Collection managed?**
* **Ans)**The JVM controls the Garbage Collector; it decides when to run the Garbage Collector. JVM runs the Garbage Collector when it realizes that the memory is running low. The behavior of GC can be tuned by passing parameters to JVM. One can request the Garbage Collection to happen from within the java program but there is no guarantee that this request will be taken care of by jvm.
* **Q5) When does an object become eligible for garbage collection?**
* **Ans) An object becomes eligible for Garbage Collection when no live thread can access it.**