What is garbage collection

* Process in which programs try to free up memory space that no longer used by objects.
* High-level programming language have some sort of garbage collection built-in
  + Python
  + JavaScript
  + Java
  + C++
  + C#
  + Ruby
  + Swift
  + PHP
  + Go
  + Rust
* Low-level programming language may add garbage collection through libraries
  + X86 Assembly Language
  + ARM Assembly Language
  + MIPS Assembly Language
  + PowerPC Assembly language
  + SPARC Assembly Language
* In C programming, developers has to take care of memory allocation and deallocation using malloc() and dealloc() functions. But in C# do need to take care of GC .
* In C# , memory allocation of objects happens in managed heap, which is taken care by CLR -common language runtime.
* Memory allocation in heap is done by win32 dll in OS similarly to C.
* Memory mapping works based on LinkedList concepts. (one after the other)

Advantage of GC

* Its tool that saves time for programmers.
* It helps in memory leaks

Disadvantage of GC

* It has negative impact on performance
* While checking for object references and cleaning out memory it oftem requires program to pause.

Sufficient memory not available for creation of new objects and entire program terminates abnormally due to OutOfMemoryErrors.

Free() in C and delete() in C++ to perfrom Garbage collection.

* When java program runs on JVM , objects are created on heap, which is portion of memory dedicated to program.
* Heap memory has 2 types of objects
* - Live – The objects are being used and referenced from somewhere
* Dead – Objects are no longer used or refrenced from anywhere.

How to make eligible for Garbage collection ?

* By making reference null

|  |
| --- |
| * Student student = new Student(); * student = null; |

* By assigning a refrence to another

|  |
| --- |
| * Student studentOne = new Student(); * Student studentTwo = new Student(); * studentOne = studentTwo; // now the first object referred by studentOne is available for garbage collection |

* By using an anonymous object
  + Register(new Student();
* How garbage collection works?
* Implementation lives in JVM
* Each JVM can implement its own version of garbage collection
* Marking or identifying the unreachable objects and destroying them with compaction.
* GC works on concept of Garbage collection Roots(GC Roots ) to indentify live and dead objects .
* Examples of Grabage collection roots are
* - Classes loaded by system class loader (not custome class loaders)
* Live threads
* Local variables and parameters of currently executing methods
* Local variables and parameters of JNI methods
* Global JNI reference
* Objects used as monitor for synchronization
* Object held from garbage collection by JVM for its purpose

Phases of Garbage collection in Java

* In this step, the GC identifies all the live objects in memory by traversing the object graph.
* When GC visits an object, it marks it as accessible and thus alive. Every object the garbage collector visits is marked as alive. All the objects which are not reachable from GC Roots are garbage and considered as candidates for garbage collection.

**Sweep dead objects**

After marking phase, we have the memory space which is occupied by live (visited) and dead (unvisited) objects. The sweep phase releases the memory fragments which contain these dead objects.

### Compact remaining objects in memory

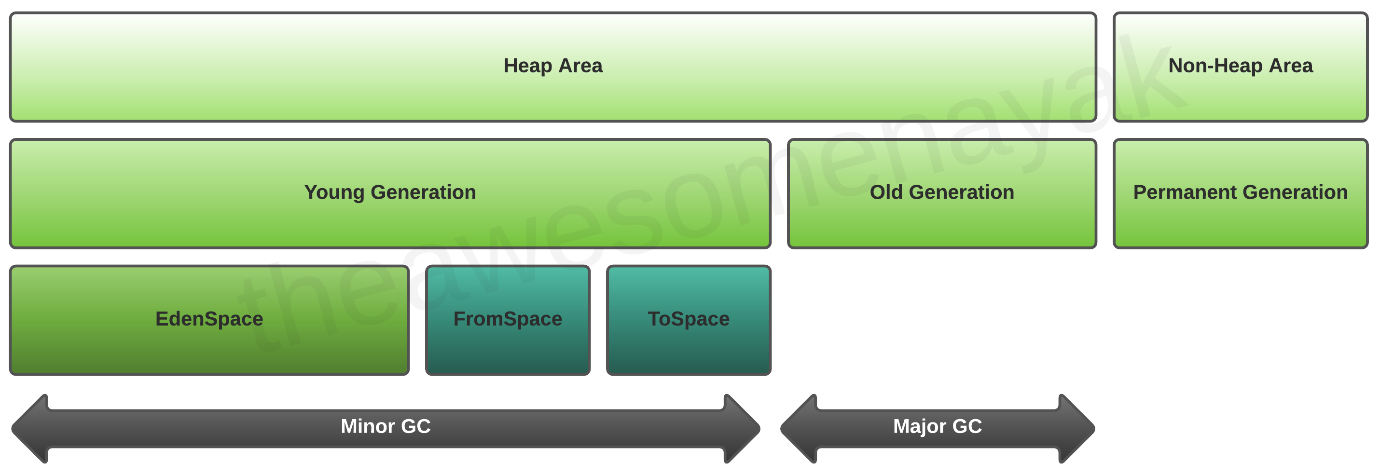
The dead objects that were removed during the sweep phase may not necessarily be next to each other. Thus, you can end up having fragmented memory space.

Memory can be compacted after the garbage collector deletes the dead objects, so that the remaining objects are in a contiguous block at the start of the heap.

The compaction process makes it easier to allocate memory to new objects sequentially.

Java cataegorizes the objects into generations and perfromsn the garbasge collection accordingly.

Heap memeory in JVM is divided into 3 secttions



The JVM heap is typically divided into three main areas: Young Generation, Old Generation, and Permanent Generation (or Metaspace, depending on the JVM version). The Young Generation is further divided into two Survivor Spaces and an Eden Space.

1. Eden Space: When objects are created in Java, they are initially allocated in the Eden Space, which is a part of the Young Generation. The Eden Space is usually where the majority of short-lived objects reside.
2. Survivor Spaces: The two Survivor Spaces (often called Survivor Space 0 and Survivor Space 1 or S0 and S1) are also part of the Young Generation. When a minor garbage collection (also known as minor GC or young GC) occurs, live objects from the Eden Space and one of the Survivor Spaces are moved to the other empty Survivor Space. The goal of this process is to identify and eliminate short-lived objects, leaving the long-lived ones to be moved to the Old Generation.
3. Old Generation: Objects that survive multiple rounds of minor GC are eventually promoted to the Old Generation. These are typically objects with longer lifetimes, and they are collected during a major garbage collection (full GC) cycle.

The division of the heap into these generations with separate Survivor Spaces allows the JVM to optimize garbage collection performance. Young Generation garbage collections are generally faster and more frequent, while Old Generation collections are more time-consuming and occur less often.

It's worth noting that JVM implementations can vary, and some may use different garbage collection algorithms or strategies. For example, the G1 (Garbage-First) Garbage Collector introduced in JDK 7 combines regions of the heap for young and old objects, removing the strict separation of Survivor Spaces. However, the concept of separating short-lived and long-lived objects remains fundamental to JVM garbage collection strategies.

Young Generation

Old Generation

Permanent Generation

MetaSpace

Types of garbage Collector

* Serial GC
* Parallel GC
* Parallel Old GC
* CMS (Concurrent Mark Sweep) GC
* G1 – Garbage First GC
* Epsilon Garbage collector
* Shenandoah
* ZGC

How to sleect right Garbage collector

Advantage of Garbage collector

Garbage collector Best practices

-Avoid Maual Triggers

Use tool for analysis

Defult setting as Good

Use JVM Flags for tuning

Select the right collector