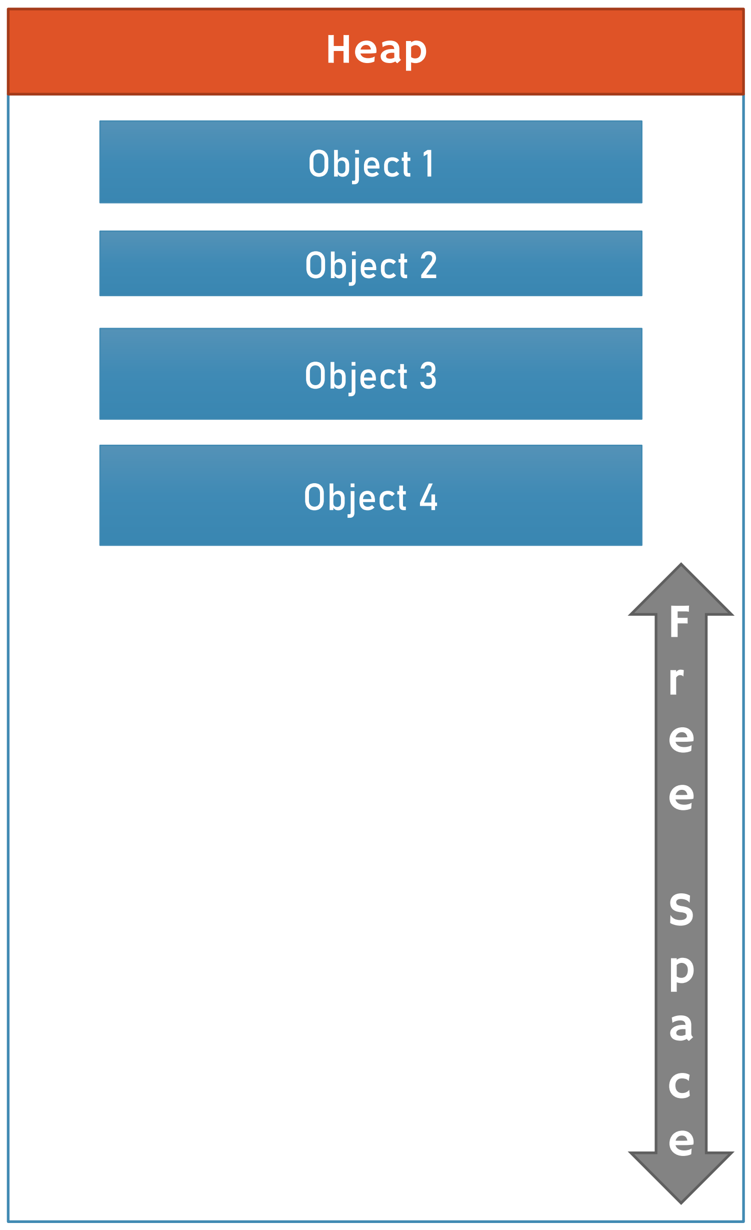
C#

Garbage Collection

What

- Garbage Collection is a process of deleting objects from memory, to free-up memory; so the same memory can be re-used.



- CLR automatically allocates memory for all objects created any where in the application, whenever it encounters "new ClassName( )" statement. This process is called as "Memory Management", which is done by "Memory Manager" component of CLR.
- All objects are stored in "Heap" (a.k.a. virtual memory).
- Heap is only-one for the entire application life time.
- The default heap size 64 MB (approx.), and extendable.
- When CLR can't find space for storing new objects, it performs a process called "Garbage Collection" automatically, which includes "identification of un-referenced objects and deleting them from heap; so that making room for new objects". This process is done by "Garbage Collector (GC)" component of CLR.

How GC decides if objects are alive?

How GC decides objects are alive?

- GC checks belongs information from the MSIL code:
  - It collects references of an object.
  - It identifies whether any object is referenced by static field.
- The objects that has at least one living reference variable in any stack or static field, are "alive objects"; others are "dead objects" or "un-used objects".

When GC gets triggered?

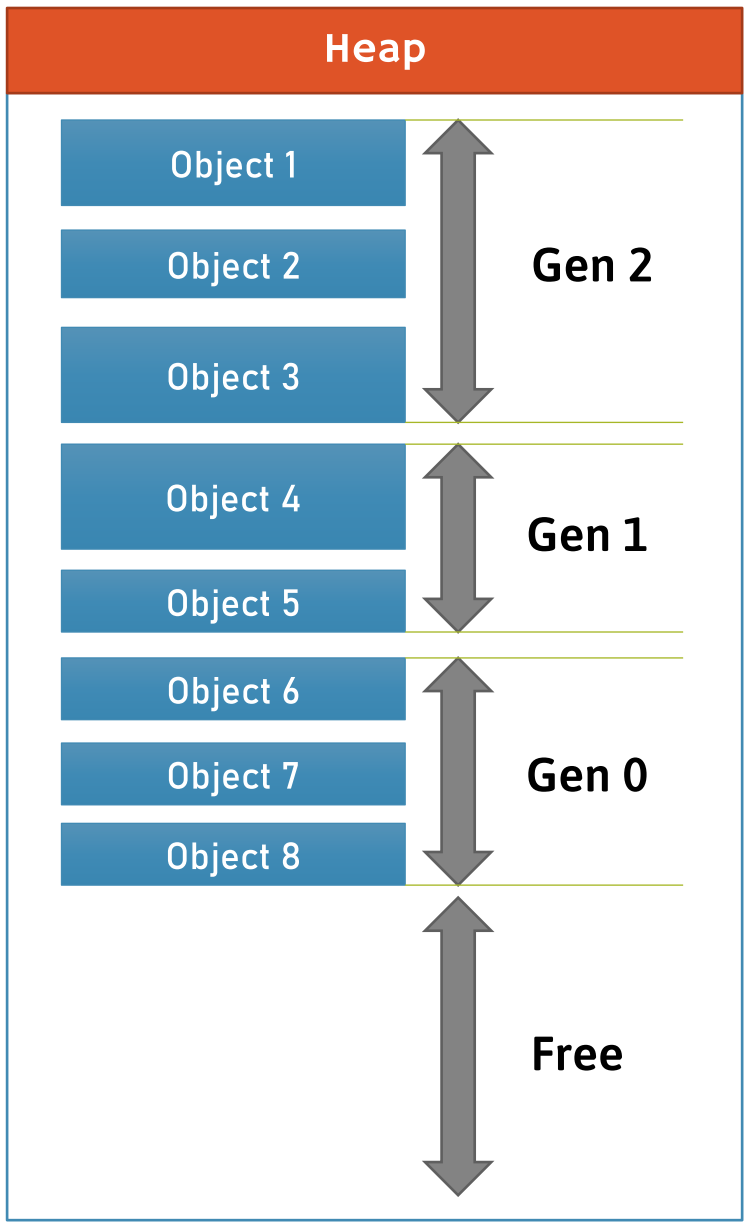
When GC gets triggered?

- There are NO specific timings for GC to get triggered.
- GC automatically gets trigged in the following conditions:
  - When the "heap" is full or free space is too low.
  - When we call GC.Collect() explicitly.

Generations in Gargabge Collection

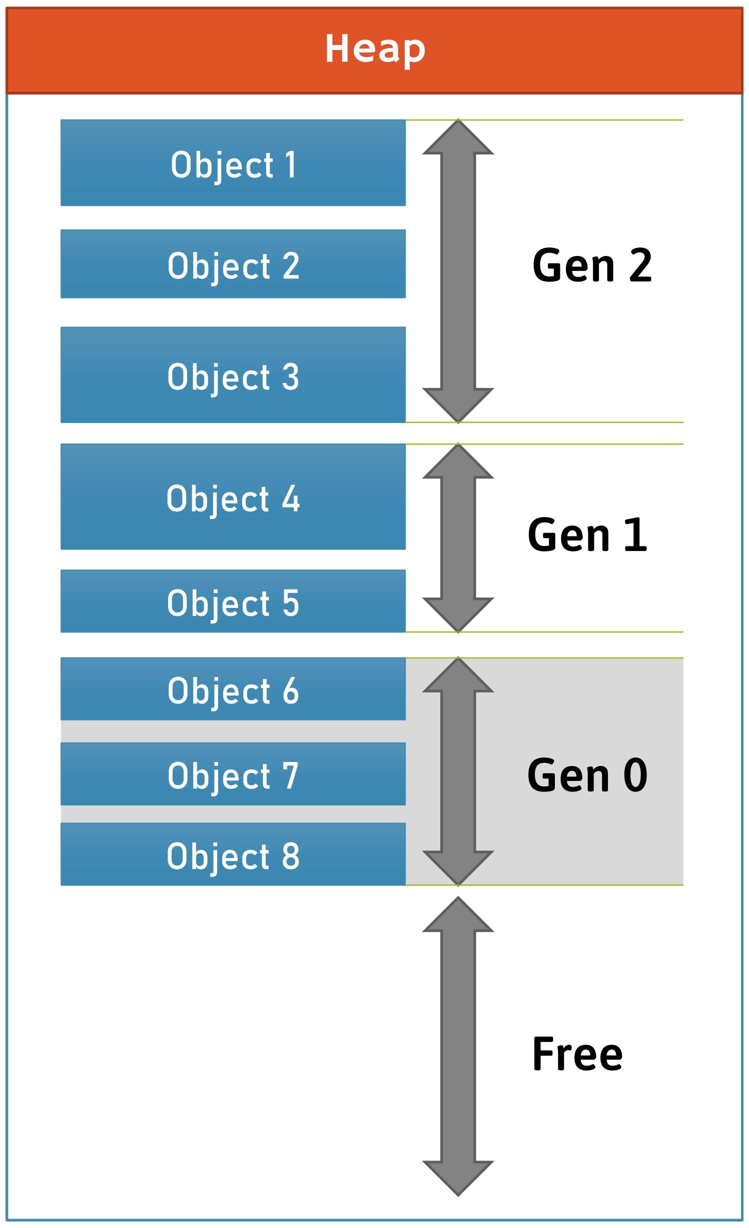
Generations

- Heap contains three segments (called generations):
  - Generation 2 [Long-Lived Generation]
  - Generation I [Survival Generation]
  - Generation 0 [Short-Lived Generation]



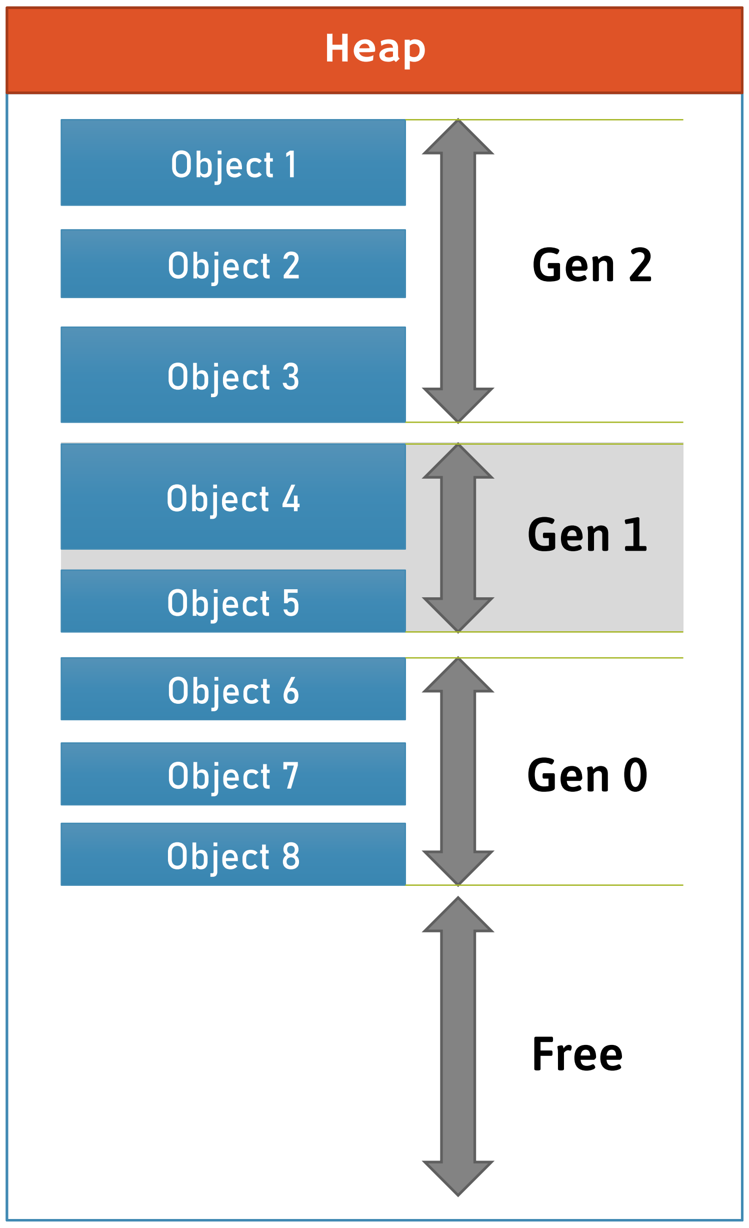
Generation 0

- The "Generation 0" is the youngest generation and contains newly created short-lived objects and collected at first priority. The objects survive longer, are promoted to "Generation I".
- <u>Ex:</u> The newly created objects.



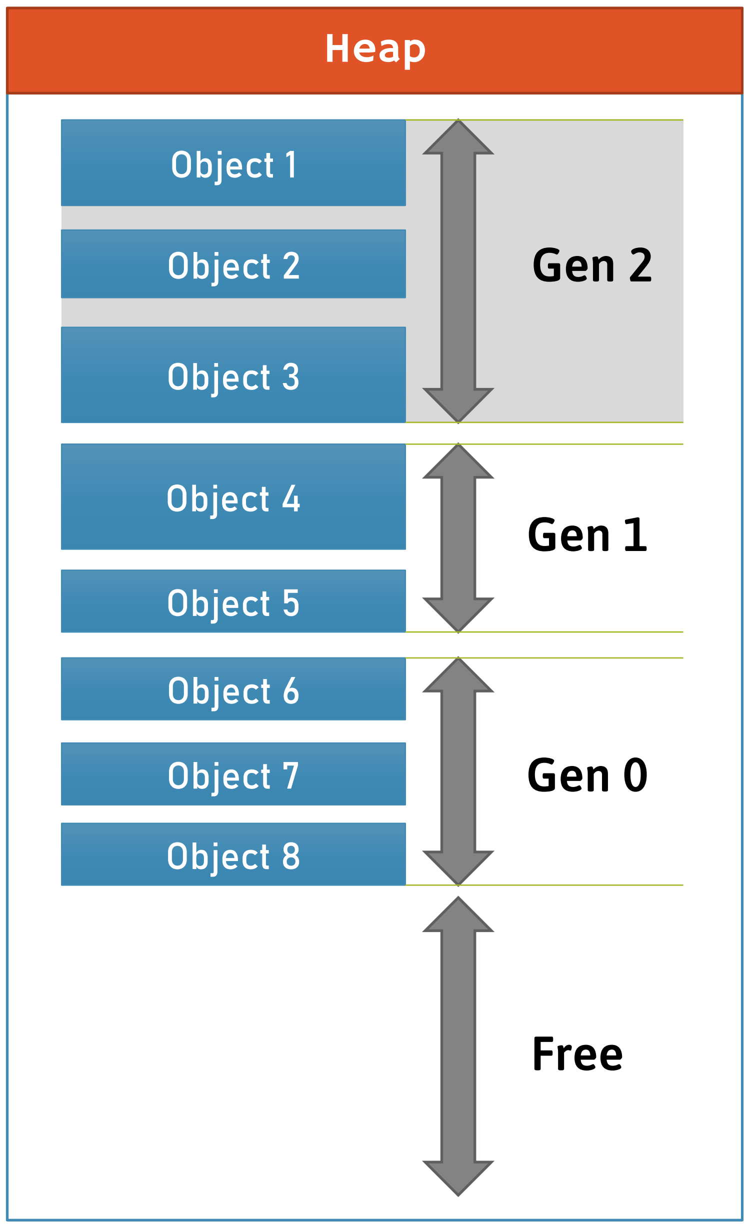
Generation 1

- The "Generation I" is buffer between "Generation 0" and "Generation 2".
- The "Generation I" mainly contains frequently-used and longer-lived objects.
- **Ex:** The objects created in the previously-executed methods, but still accessible.



Generation 2

- The "Generation 2" contains the longest-lived objects that were created long-back and still is being used, by different statements in the program.
- **Ex:** The objects that referenced with static fields.



Managed (vs) Unmanaged Resources

Managed Resources

- The objects that are created by CLR are called as "Managed Resources".
- These will participate in "Garbage Collection" process, which is a part of .NET Framework.

Unmanaged Resources

- The objects that are not created by CLR and not managed by CLR are called as "Un-managed resources".
- Ex: File streams, database connections

Clearing Unmanaged Resources

Destructor

- Clears unmanaged resources just before deleting the object; i.e. generally at the end of application execution.

Dispose

- Clears unmanaged resources after the specific task (work) is completed; so no need to wait till end of application execution.