C# 7.0

Lesson 08 : Garbage Collection

Lesson Objectives

- Role of a Garbage Collector
- Garbage Collection – The Process
- Garbage Collection – Advantages
- Finalizers in Garbage Collection
- Overriding Object.Finalize
- Generational Garbage Collection
- Dispose Method
- Difference Between Finalize and Dispose
- More About Garbage Collection

Role of a Garbage Collector

- A Garbage Collector:
  - Simplifies memory management for the developers.
  - Manages the allocation and release of memory for an application.
    - Therefore no need to write code to perform memory management.
- Performs collection: releases the memory for objects that are no longer being used.
  - The GC's engine determines the best time to perform collection.
- Compacts the object in memory: freeing up blocks of address space allocated to unreachable objects.
- Sets the managed heap pointer after the last object.

Garbage Collection – The Process

- object birth: step 1 - allocation
  - acquire raw heap memory
  - using operator new
- object birth: step 2 - initialization
  - convert raw heap memory into an object
  - using a constructor
  - you can't use a constructor without new

Garbage Collection – The Process (Cont.)

- object death: step 1 – finalization
  - convert the object back to raw heap memory
  - using a special Finalize method
- object death: step 2 - deallocation
  - release the raw memory back to the heap
  - ready for another object to be created in

```
protected virtual void Finalize()   {
            ...
}
```

Garbage Collection – Advantages

- If programmers destroy objects:
  - they forget - creating memory leaks
  - they try to destroy an object more than once
  - they destroy reachable objects - creating dangling references
- In C#, only the runtime can destroy objects:
  - It never forgets - objects are destroyed
  - It never destroys an object more than once
  - It never destroys reachable objects

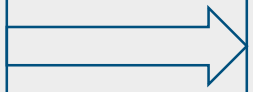
Finalizers in Garbage Collection

- The Garbage Collector finalizes objects:
  - when they're unreachable by calling Finalize()
  - You cannot override Object.Finalize yourself!
  - And, you cannot call Finalize yourself either!

Overriding Object.Finalize

- You can instead write a destructor which:
  - The compiler translates into Finalize.
  - Can be declared in a class but not a struct
  - Automatically calls its base class Finalize

```
public sealed class Resource
{
   ...
   ~Resource()
   {
   //...
   }
...
}
```



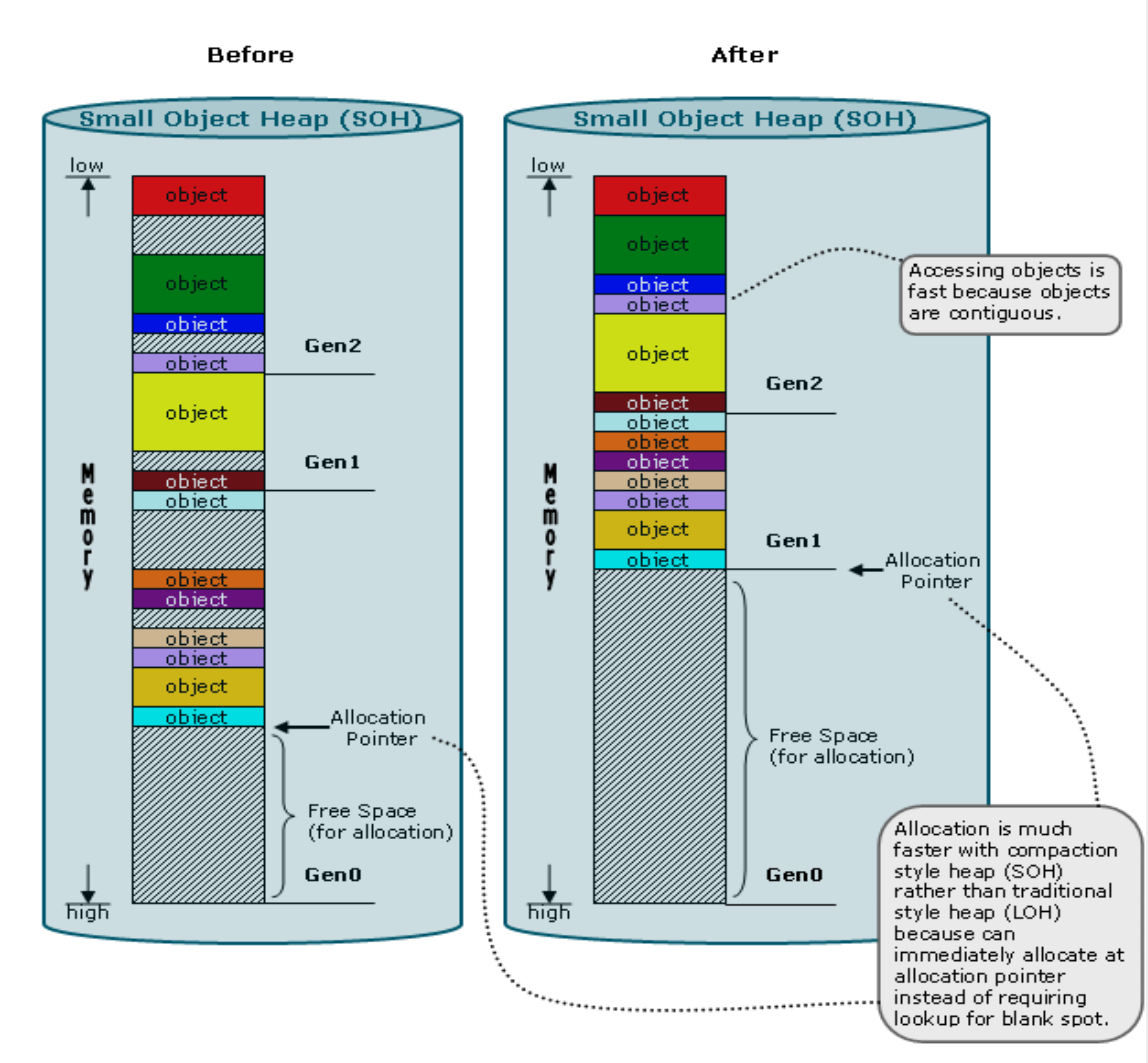
```
public sealed class Resource
{
   ...
   protected override void Finalize()
   {
      try {  //...  }

      finally {
         base.Finalize();
      }
   }
}
```

Generational Garbage Collection

- The heap is organized into generations so it can handle long-lived and short-lived objects.
- Garbage collection primarily occurs with the reclamation of short-lived objects that typically occupy only a small part of the heap.
- There are three generations of objects on the heap:
  - Generation 0
  - Generation 1
  - Generation 2
- Garbage collections occur on specific generations as conditions warrant.
- Collecting a generation means collecting objects in that generation and all its younger generations.
- A generation 2 garbage collection is also known as a full garbage collection, because it reclaims all objects in all generations (that is, all objects in the managed heap)

Generational Garbage Collection (Cont.)



Dispose Method

- The .NET Framework garbage collector does not allocate or release unmanaged memory
- Unmanaged resources are resources such as file and pipe handles, registry handles, wait handles, or pointers to blocks of unmanaged memory
- You implement a Dispose method to release unmanaged resources used by your application
- To implement a Dispose method, you need to implement IDisposable interface

Difference Between Finalize and Dispose

Finalize	Dispose
Belongs to the Object class.	Belongs to the IDisposable interface.
It is automatically called by the Garbage Collection mechanism when the object goes out of the scope(usually at the end of the program).	We have to manually write the code to implement it.
It is slower method and not suitable for instant disposing of the objects.	It is faster method for instant disposal of the objects.
It is non-deterministic function i.e., it is uncertain when Garbage Collector will call Finalize() method to reclaim memory.	It is deterministic function as Dispose() method is explicitly called by the User Code.

More About Garbage Collection

- Garbage collector does not guarantee...
  - when objects are finalized
  - this is known as non-deterministic finalization
- the order in which objects are finalized
  - so a destructor shouldn't call other objects since they might have already been finalized!
  - you still need to consider ownership in design, even in a language that supports GC

Summary

In this lesson, you have learnt

- Memory management in C#
- Role of a Garbage Collector
- Garbage collection in C# using Finalize and Dispose methods

Review Question

- Question 1 : How does memory management take place in C#?
- Question 2 : What is the use of SuppressFinalize method of GC class?
- Question 3 : What is a difference between Dispose() & Finalize() Methods?