

# Garbage Collection

---

*Source:*

<http://www.tech-archive.net/Archive/DotNet/microsoft.public.dotnet.framework.clr/2005-04/msg00052.html>

---

- *From:* [saurabhgarg2k@xxxxxxxxxx](mailto:saurabhgarg2k@xxxxxxxxxx)
  - *Date:* 6 Apr 2005 03:00:38 -0700
- 

Hi there,

I am attaching a snippet of code here which tries to explain that if u have a class which contains both Unamanged resources and allocates a lot of memeory and have a finalize method, then u should divides it into two classes , one contains the memory part and other manages the unmanaged resources and implement finalize in the second class so that the memory will be released quick enough for first Class which allocates large memory as classes implementing finalize method requires at least 2 GC cycles to release the objects.

Here is the first class with contains both large memory and Unamanged resources in one class

```

Class SimpleObject
Private Declare Function CreateSemaphore _
Lib "kernel32" Alias "CreateSemaphoreA" _
(ByVal lpSemaphoreAttributes As Integer, _
ByVal lInitialCount As Integer, _
ByVal lMaximumCount As Integer, _
ByVal lpName As String) As Integer
Private Declare Function ReleaseSemaphore _
Lib "kernel32" Alias "ReleaseSemaphore" _
(ByVal hSemaphore As Integer, _
ByVal lReleaseCount As Integer, ByVal _
lpPreviousCount As Integer) As Integer

Dim arr() As Integer
Dim handle As Integer

Sub New()
ReDim arr(10000) ' a lot of memory
' directly allocate an unmanaged resource
handle = CreateSemaphore(0, 1, 1000, _
"testsemaphore")
End Sub

Protected Overrides Sub Finalize()

```

## Garbage Collection

```
' release the unmanaged resource
If handle <> 0 Then _
ReleaseSemaphore(handle, 1, 0)
End Sub
End Class
```

Here is the Next Class which divides the functionality which is embedded in the outer class.

```
Class CompoundObject
Dim arr() As Integer
Dim wrapper As SemaphoreWrapper

Sub New()
ReDim arr(10000) ' a lot of memory
' allocate another (managed) resource
' (no need to release it in Finalize)
wrapper = New SemaphoreWrapper
End Sub

' a class that wraps the unmanaged resource
Private Class SemaphoreWrapper

Private Declare Function CreateSemaphore _
Lib "kernel32" Alias "CreateSemaphoreA" _
(ByVal lpSemaphoreAttributes As Integer, _
ByVal lInitialCount As Integer, _
ByVal lMaximumCount As Integer, _
ByVal lpName As String) As Integer

Private Declare Function ReleaseSemaphore _
Lib "kernel32" Alias "ReleaseSemaphore" _
(ByVal hSemaphore As Integer, _
ByVal lReleaseCount As Integer, _
ByVal lpPreviousCount As Integer) As Integer

Public Handle As Integer
Sub New()
' allocate the unmanaged resource
Handle = CreateSemaphore(0, 1, 1000, _
"testsemaphore")
End Sub

Protected Overrides Sub Finalize()
' release the unmanaged resource
If Handle <> 0 Then _
ReleaseSemaphore(Handle, 1, 0)
End Sub

End Class
```

Garbage Collection

## Garbage Collection

End Class

Here is the Client Code

```
Module Module1
Sub Main()
Dim i As Integer
Const TIMES As Integer = 10000

Dim t As Date = Now
For i = 1 To TIMES
Dim so As New SimpleObject
Next
Console.WriteLine("Simple object: {0}", _
Now.Subtract(t))

' force a garbage collection to clean up
GC.Collect()
GC.WaitForPendingFinalizers()

t = Now
For i = 1 To TIMES
Dim co As New CompoundObject
Next
Console.WriteLine("Compound object: {0}", _
Now.Subtract(t))
Console.ReadLine
End Sub
End Module
```

Now the author claims that the first for loop will take more time as it requires more GC cycles to deallocate the large chunk of memory it allocates and the second will release it quickly as the class allocating the Large memory does not implement finalize method so it will be deallocated in one GC cycle unlike the class with unmanaged resource

My question:

As GC will only fire when managed heap runs out of space, so in first for loop when after allocating a few objects, heap is full and the GC fires but no objects will be allocated as the class implements the Finalize method and it requires at least two GC cycles to deallocate, so if no object is deallocated, from where the runtime gets the space to allocate the next object, will GC be fired again for these kind of cases when it does not able to release enough memory for the next object.

Note:

The code works fine.No Exceptions so there is something missing in my understanding.

## Garbage Collection

Excuse me for such a long posting but i would highly appreciate if someone clears my doubt where i am wrong.

Thanks and Regards  
Saurabh Garg

- 
- Prev by Date: ***Re: Static Variables during Garbage Collection***
  - Next by Date: ***How GC Works***
  - Previous by thread: ***Re: Garbage Collection***
  - Next by thread: ***RE: Remote debug***
  - Index(es):
    - ◆ ***Date***
    - ◆ ***Thread***