

# The IDisposable Interface

```
using System;
public class MyClass : IDisposable
{
    private string name;
    public MyClass(string name) { this.name = name; }
    override public string ToString() { return name; }

    ~MyClass()
    {
        Dispose();
        Console.WriteLine("~MyClass(): " + name);
    }

    public void Dispose()
    {
        Console.WriteLine("Dispose(): " + name);
        GC.SuppressFinalize(this);
    }
}

public class DisposableApp
{
    public static void Main(string[] args)
    {
        Console.WriteLine("start of Main, heap used: {0}",
            GC.GetTotalMemory(true));
        DoSomething();
        Console.WriteLine("end of Main, heap used: {0}",
            GC.GetTotalMemory(true));
    }

    public static void DoSomething()
    {
        MyClass[] ta = new MyClass[3];
    }
}
```

```
for (int i = 0; i < 3; i++) { ta[i] = new
MyClass(String.Format("object      #"      +i));
Console.WriteLine("Allocated {0} objects, heap used: {1}",
i+1, GC.GetTotalMemory(true)); } for (int i = 0; i < 3; i++) {
ta[i].Dispose(); ta[i] = null; GC.Collect();
GC.WaitForPendingFinalizers(); Console.WriteLine("Disposed {0}
objects, heap used: {1}",i+1, GC.GetTotalMemory(true)); } } }
[/csharp]
```