Implementing Finalize and Dispose to Clean Up Unmanaged Resources

Class instances often encapsulate control over resources that are not managed by the runtime, such as window handles (HWND), database connections, and so on. Therefore, you should provide both an explicit and an implicit way to free those resources. Provide implicit control by implementing the protected [Finalize](https://msdn.microsoft.com/en-us/library/vstudio/system.object.finalize(v=vs.100).aspx) on an object (destructor syntax in C# and C++). The garbage collector calls this method at some point after there are no longer any valid references to the object.

In some cases, you might want to provide programmers using an object with the ability to explicitly release these external resources before the garbage collector frees the object. If an external resource is scarce or expensive, better performance can be achieved if the programmer explicitly releases resources when they are no longer being used. To provide explicit control, implement the [Dispose](https://msdn.microsoft.com/en-us/library/vstudio/system.idisposable.dispose(v=vs.100).aspx) provided by the [IDisposable](https://msdn.microsoft.com/en-us/library/vstudio/system.idisposable(v=vs.100).aspx). The consumer of the object should call this method when it is finished using the object. **Dispose** can be called even if other references to the object are alive.

Note that even when you provide explicit control using **Dispose**, you should provide implicit cleanup using the **Finalize** method. **Finalize** provides a backup to prevent resources from permanently leaking if the programmer fails to call **Dispose**.

For more information about implementing **Finalize** and **Dispose** to clean up unmanaged resources, see [Garbage Collection](https://msdn.microsoft.com/en-us/library/vstudio/0xy59wtx(v=vs.100).aspx). The following example illustrates the basic design pattern for implementing **Dispose**. This example requires the [System](https://msdn.microsoft.com/en-us/library/vstudio/system(v=vs.100).aspx) namespace.

C#

**// Design pattern for a base class.**

public class Base: IDisposable

{

private bool disposed = false;

//Implement IDisposable.

public void Dispose()

{

Dispose(true);

GC.SuppressFinalize(this);

}

protected virtual void Dispose(bool disposing)

{

if (!disposed)

{

if (disposing)

{

// Free other state (managed objects).

}

// Free your own state (unmanaged objects).

// Set large fields to null.

disposed = true;

}

}

// Use C# destructor syntax for finalization code.

~Base()

{

// Simply call Dispose(false).

Dispose (false);

}

}

**// Design pattern for a derived class.**

public class Derived: Base

{

private bool disposed = false;

protected override void Dispose(bool disposing)

{

if (!disposed)

{

if (disposing)

{

// Release managed resources.

}

// Release unmanaged resources.

// Set large fields to null.

// Call Dispose on your base class.

disposed = true;

}

base.Dispose(disposing);

}

// The derived class does not have a Finalize method

// or a Dispose method without parameters because it inherits

// them from the base class.

}