原文地址：<https://www.cnblogs.com/nele/p/5673215.html>

**什么是GC**

GC是垃圾回收器，对于没有被其他对象所引用的对象，GC会在下一次垃圾回收中将其回收

**GC回收算法**

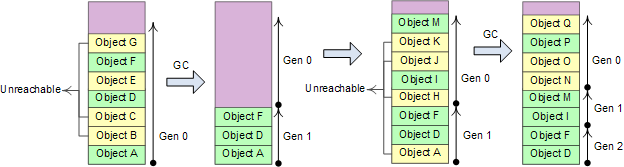
**Generational 分代算法**

目前的GC一般使用分代算法

Generational 分代算法如下：

1. 新实例的对象我们称为0代对象，在一个GC时间后，GC将会对0代对象进行遍历回收
2. 在0代对象中没有被回收的对象将会被放到1代中，在10个GC时间后，GC将会对1代对象进行遍历回收
3. 在1代对象中没有被回收的对象将会被放到2代中，在100个GC时间后，GC将会对2代对象进行遍历回收

GC使用三个指针将heap分为0，1，2代



**Finalization Queue和Freachable Queue**

当某个类具有析构函数时，其对象被生成时，Finalization Queue队列中会添加一个指向对象的指针

当对象要被回收时，如果Finalization Queue存在对象的指针，那么会跳过本次回收，并移除对象在Finalization Queue上的指针，然后再Freachable Queue会添加对象的指针，之后Freachable Queue队列的对象的析构函数会被调用，这时对象再下一个GC中就会被回收

GC方法：

ReRegisterForFinalize：将对象注册到Finalization Queue中（如果我们再析构函数中调用该方法将当前对象注册到Finalization Queue中，那么当前对象将永远不会被回收）

SuppressFinalize：禁止对象的Finalize方法（告诉GC回收时不必调用其析构方法了）

**值类型与引用类型**

.NET中的所有类型都是（直接或间接）从System.Object类型派生的。

CTS中的类型被分成两大类

引用类型：分配在内存堆上，如类，数组，String等

值类型：分配在堆栈上，如int，char，指针等

**托管资源与非托管资源**

**托管资源**

.NET中超过80%的资源都是托管资源，我们自定义的类都是托管资源，当某个托管资源不被任何其它资源引用时，GC会自动回收

**非托管资源**

GC无法回收系统资源，需要程序自己释放，如：

ApplicationContext, Context, FileStream, OdbcDataReader, OleDBDataReader, Regex, Socket, 文件句柄, GDI资源, 数据库连接等等资源

**非托管资源释放**

非托管资源释放的两种方式：

1. 手动释放：手动释放需要释放自身引用的非托管资源和所引用类引用的非托管资源，手动释放一般继承IDisposable（目的是我们可以使用using）
2. GC回收释放：在GC回收时释放，GC释放只需释放自身引用的非托管资源（因为所引用的类也会被GC回收，不需要帮其他类释放），需要实现析构函数

我们一般将手动释放和GC回收释放结合，当手动释放时会释放资源，当我们没有手动释放时，GC回收时也会释放

示例代码：

public class ResourceHolder : IDisposable

{

private bool isDisposabled = false;

~ResourceHolder()

{

// 释放自身资源

Dispose(false);

}

public void Dispose()

{

// 释放所有资源

Dispose(true);

}

/// <summary>

/// 释放资源

/// </summary>

/// <param name="disposing"></param>

protected virtual void Dispose(bool disposing)

{

if (isDisposabled == true)

return;

// 释放自身的资源

isDisposabled = true;

if (disposing == false)

return;

// 释放所引用的对象的资源（调用所有引用对象的 Dispose()）

}

}

**GC.Collect() 方法**

作用：强制进行垃圾回收

|  |  |
| --- | --- |
| 名称 | 说明 |
| Collect() | 强制对所有代进行即时垃圾回收。 |
| Collect(Int32) | 强制对零代到指定代进行即时垃圾回收。 |
| Collect(Int32, GCCollectionMode) | 强制在 GCCollectionMode 值所指定的时间对零代到指定代进行垃圾回收 |