在各开源项目中，为什么 DisposeObject 会比 Free 使用频率更高

要说 DisposeObject，必须先说 Interface 这种东西

Interface 在强类型编程语言中，是接口的意思，通过 Interface，可以实现各种调用的插入功能。同时，Interface 也是一个超级大坑，它非常不易于直接从源码跟踪分析，另外一方便，Interface 有一个 RefCount 引用计数器，置零时，接口到 Interface 的 class 将被释放。

Interface 有了以上两个坑，这是我非常不推荐使用的机制。

为了避免这种机制，在 CoreClasses 中，直接修改了 Interface 的引用计数，改为任何 class 在构建以后，都必须手动去释放。

DisposeObject 的由来就是这样的了

使用 DisposeObject 的另一个原因

因为 Delphi 是避源的编译工具链，同时 delphi 的 runtime 库自成一派

Fpc 基于 Lazarus 提供了高仿 delphi runtime 库的 LCL 框架

在 delphi runtime 中，LockObject 使用 Tobject 内置的一个变量作为并行线程的锁变量，在 Tmonitor.Enter(obj),exit(obj)时，可以高效交换。如果换做 fpc 的 runtime 体系，要实现 Tmoitor.Enter 机制，就很麻烦了，需要开 Mutex 锁。

在 CoreClasses 中为了兼容超级省事的 Tmoitor.Enter 机制，我开辟了原子锁的反查链表，但是随之而来会出现一个对象被使 free 释放时，原子锁无法同步释放。

为了在 fpc 兼容以上机制，DisposeObject 被调用时也会干一件释放原子锁的工作。

2019-7

By.qq600585