由记忆棒误差故障引发的关于面向对象设计的九点思考

从 farseerfc.wordpress.com 导入

故障描述: MMC Memory Stick Duo记忆棒未经Adapter适配器，直接插入SD Reader，致使MMC卡入SD Reader中。

栈展开：某日下午，无课。忙于数分作业，想查询用手机拍摄的板书照片。取出手机中的MMC。未经装配Adapter，直接插入SD Reader。
(A runtime exception was thrown.) 尝试翻转笔记本机身，倒出MMC，未果。(rethrow) 尝试用手指甲取出，未果。(rethrow) 考虑到有“推入反弹”机制，尝试将MMC推入更深，反弹机制由于类型不匹配而失效，未果。(rethrow)
(The exception spread across the border of the model.) 电脑维修技师接手(catch) 技师未能发现问题所在，由我解说原委。
(Because the exception lose the information, RTTI was ask
技师发现问题，尝试用镊子镊出MMC，未果。技师开解机箱(expose the data structure) 技师制作钩子，勾出MMC(hooker link to the structure) 取出MMC，故障解除

故障总结 1.接收到没有完全了解、或没有适当工具解决的exception时，不要尝试用不成熟的技术解决，应尽快寻求能解决它的代码。否则，被反复rethrow的exception，尤其是通过模块边界的exception，有可能由subclass退化为superclass，并因此而丧失一些信息。尽量不要让exception丢失信息，必要时，通过RTTI机制寻回信息。

2.超负荷运转，多线程执行，这种种复杂性都有可能导致错误，应避免。无论你有多么信任你的代码或能力。

3.在设计class的interface时，相匹配的interface应该满足is-a的关系。因此，任何能插入SD Reader的object，即任何实现了SD interface的object，都应该is-a SD card。这次故障中，interface接受了MMC，但MMC不是SD。即使这种情况下throw an exception，都不能使事态缓和。能提供compile-time error时，尽量让错误以compile-time error的形式展现，并在事先解决。类型匹配问题是应该能在事先解决的问题。

4.Design patterns中的Adapter pattern应该只是迫不得已情况之下的解决方案。只有当你无权改变现状时，才能使用Adapter。如果能改变现状，应该改变设计以符合interface。

5.因为上条，所有相似功能的对象应具有相同的interface，不同的interface是本次故障的根源所在。

6.特殊情况下，破坏封装机制并expose the data structure是必要的，应该有方法支持这种做法。C的指针和C#的Reflection技术都以不同的方式支持这种做法。其他的一些语言机制，比如serializing(序列化)或streaming(流化)，也可以以某种方式间接支持这一做法。当然，机制还应避免这种做法被滥用。

7.相反功能具有相同操作的设计，容易造成使用的混乱，应适当避免。比如SD Reader的推入反弹设计，即插入和弹出使用同一个向里推的操作的设计。同样的设计

还包括，C++中的setNewHandle使用同一个函数，同时设置和返回handle。以及有些书中提倡的，使用同名函数重载的方式，实现setter/getter的设计。

8.特殊工具(hooker)对于解决特定问题，通常比手工解决有效。不要嫌麻烦而不愿意构造特殊工具。

9.栈语义，即FILO顺序，总在不知不觉中影响我们。违反了FILO顺序的操作极易造成混乱。本故障发生时正确的处理顺序为：装配Adapter　插入SD Reader　读取数据　停用设备　拔出SD Reader 拆解Adapter 本次故障的原因就是违反了FILO顺序，违反了栈语义。