

由記憶棒誤差故障引發的關於面向對象設計的九點思考

從 farseerfc.wordpress.com 導入

故障描述: MMC Memory Stick Duo 記憶棒未經 Adapter 適配器, 直接插入 SD Reader, 致使 MMC 卡入 SD Reader 中。

棧展開: 某日下午, 無課。忙於數分作業, 想查詢用手機拍攝的板書照片。取出手機中的 MMC。 未經裝配 Adapter, 直接插入 SD Reader。(A runtime exception was thrown.) 嘗試翻轉筆記本機身, 倒出 MMC, 未果。(rethrow) 嘗試用手指甲取出, 未果。(rethrow) 考慮到有“推入反彈”機制, 嘗試將 MMC 推入更深, 反彈機制由於類型不匹配而失效, 未果。(rethrow) (The exception spread across the border of the model.) 電腦維修技師接手 (catch) 技師未能發現問題所在, 由我解說原委。(Because the exception lose the information, RTTI was asked to recall the information) 技師發現問題, 嘗試用鑷子鑷出 MMC, 未果。 技師開解機箱 (expose the data structure) 技師製作鉤子, 勾出 MMC(hooker link to the structure) 取出 MMC, 故障解除

故障總結 1. 接收到沒有完全瞭解、或沒有適當工具解決的 exception 時, 不要嘗試用不成熟的技術解決, 應儘快尋求能解決它的代碼。否則, 被反覆 rethrow 的 exception, 尤其是通過模塊邊界的 exception, 有可能由 subclass 退化爲 superclass, 並因此而喪失一些信息。儘量不要讓 exception 丟失信息, 必要時, 通過 RTTI 機制尋回信息。

2. 超負荷運轉，多線程執行，這種種複雜性都有可能導致錯誤，應避免。無論你有多麼信任你的代碼或能力。

3. 在設計 class 的 interface 時，相匹配的 interface 應該滿足 is-a 的關係。因此，任何能插入 SD Reader 的 object，即任何實現了 SD interface 的 object，都應該 is-a SD card。這次故障中，interface 接受了 MMC，但 MMC 不是 SD。即使這種情況下 throw an exception，都不能使事態緩和。能提供 compile-time error 時，儘量讓錯誤以 compile-time error 的形式展現，並在事先解決。類型匹配問題是應該能在事先解決的問題。

4. Design patterns 中的 Adapter pattern 應該只是迫不得已情況之下的解決方案。只有當你無權改變現狀時，才能使用 Adapter。如果能改變現狀，應該改變設計以符合 interface。

5. 因為上條，所有相似功能的對象應具有相同的 interface，不同的 interface 是本次故障的根源所在。

6. 特殊情況下，破壞封裝機制並 expose the data structure 是必要的，應該有方法支持這種做法。C 的指針和 C# 的 Reflection 技術都以不同的方式支持這種做法。其他的一些語言機制，比如 serializing(序列化) 或 streaming(流化)，也可以以某種方式間接支持這一做法。當然，機制還應避免這種做法被濫用。

7. 相反功能具有相同操作的設計，容易造成使用的混亂，應適當避免。比如 SD Reader 的推入反彈設計，即插入和彈出使用同一個向裏推的操作的設計。同樣的設計還包括，C++ 中的 setNewHandle 使用同一個函數，同時設置和返回 handle。以及有些書中提倡的，使用同名函數重載的方式，實現 setter/getter 的設計。

8. 特殊工具 (hooker) 對於解決特定問題，通常比手工解決有效。不要嫌麻煩而不願意構造特殊工具。

9. 棧語義，即 FILO 順序，總在不知不覺中影響我們。違反了 FILO 順序的操作極易造成混亂。本故障發生時正確的處理順序為：裝配 Adapter 插入 SD Reader 讀取數據 停用設備 拔出 SD Reader 拆解 Adapter 本次故障的原因就是違反了 FILO 順序，違反了棧語義。