切換導航 Farseerfc的小窩

• <u>菜</u>

働	て凪	曲
光	<u>ミ</u> 目	豆

<u>简体</u>

English

日本語

- <u>▲ About</u>
- <u> Links</u>
- ≜ About
- <u>Links</u>
- □ Import
- □ Life
- <u>□ Tech</u>

- <u>Import</u>
- □<u>Life</u>
- <u>Tech</u>
- Q

搜索

- 搜索
- 2

由記憶棒誤差故障引發的關於面向對象設計 的九點思考 <u></u> 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 mod 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 v 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.因爲上條,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。