單元測試

TDD （Test-Driven Development），又稱測試驅動程式開發。

可能包含某些可拋棄的測試性驅動程式，我們可以手動與程式產生互動。

- **TDD的三大法則**

1. 在撰寫一個單元測試前，不可撰寫任何產品程式
2. 只撰寫剛好無法通過的單元測試，不能編譯也算無法通過
3. 只撰寫剛好能通過當前測試的產品程式

- **讓測試程式整潔**

測試程式與產品程式**一樣重要**，得用心設計與維護。~~它們不應該是二等公民。~~

測試程式若不保持整潔，當專案的規模越來越大，維護測試成本也將隨之提高。

最終，我們將會失去它們以及它們所帶來的好處。

我們會失去了「程式修改後是否能依預期般工作」的能力。

也將無法保證「對系統某部分修改是否搞爛其他程式」。

- **測試帶來更多的好處**

測試能為我們產品保持「修改的彈性」。

當測試涵蓋專案越廣，越能修改程式而無後顧之憂。~~即使你把人家搞壞了。~~

測試程式應遵循的最佳法則：

在一個概念裡最小化斷言的數量，一個測試函式只測試一個概念。

- **一個測試一次斷言（Assert）**

在允許的情況下，每個測試函式內應只有一個斷言。

也就是說一組測試函式僅會產生一個結論，使讀者可以更快的理解測試目的和內容。

所以我們應盡可能讓測試中的斷言數量減少。

- **一個測試一個概念**

同一個函式**只做一件事情**，我們不會希望測試函式過長或是細節過多使得閱讀不易。

假設測試的概念較廣且雜，應找到獨立不相依於其它函式的概念，然後合理的逐一測試。

- **整潔的測試**

可讀性、可讀性、可讀性。~~很重要所以說三次。~~

測試程式不需要多餘且惱人的細節與干擾，且遵循 **BUILD-OPERATE-CHECK** 模式。

BUILD-OPERATE-CHECK （建立、操作、檢查）

1. 建立測試資料
2. 操作測試資料
3. 檢查操作的測試資料結果是否如預期（通常是斷言 Assert）

- **F . I . R . S . T**

Fast：測試程式應可以被迅速的執行，如果執行的很慢通常你會~~發懶~~不想執行它們。

Independent：測試程式間不應該相互依賴，要能夠獨立測試。

Repeatable：應可在任何環境中重複執行，也可以在沒網路的情況下繼續測試。

~~（太嚴苛... 有時候這是幻想）~~

Self-Validating：測試程式應輸出布林值，應有自我驗證的能力而不用查看紀錄檔。

Timely：測試程式應被即時的撰寫，而不是寫完產品程式之後才回頭寫測試程式。