60647067s賈昊承

(a)Abstraction

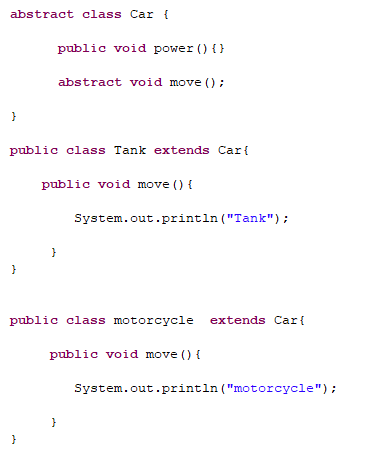
(1)Def : 當一個class被定義時，只宣告它的方法名稱，而沒有實作其內部功能與細節，稱為抽象方法(Abstract method)。若某個class中包含了抽象方法，則稱此class為抽象類別(Abstract class)。

(2)因為抽象類別是一個未定義完整的類別，所以不能直接使用。要使用它必須要先定義另一個class來繼承這個抽象類別，並從子類別(新定義的class)中把抽象方法定義完整，再產生子類別的實體(物件)即可。

(3)

Good use:

使用Abstract class來建立各種車子的類別可避免每種車子的類別內容被重複定義，如下圖power()只需要定義一次。



Bad use:

若每個class均無相同的method或屬性，但卻又使用Abstract class，則此Abstract class的意義就不大，因為每個class都需要各自定義自己的method或屬性。

(b)Encapsulation

(1)Def : 在class中能選擇性地讓class內部的成員(方法、屬性、事件等)不被其他class或程式所存取，稱為class的封裝性(Encapsulation)。能夠有效地隱藏class內部的一些資訊。

(2)將class內部的method或variable設為”private”即可防止其他class直接取用或修改此class內部的成員。

(3)

Good use:

根據Encapsulation的特性，適時地將重要的class或method設為”private”可增加安全性，想存取此資料的class或程式並不知道此method內部的運作方式。

Bad use:

每個class或method都設為”public”，較缺乏安全性。

(c)Cohesion

(1)Def : 一個class只做一件事，或只專注於一個目標(事或目標可大可小)，不插手做其他class做的事或目標，稱為內聚(cohesion)。

(2)將每個class與method所要執行的目標定義明確，避免多個class做相同的事或執行相同目標。

(3)

Good use:

每個class做的事都不相同，且只專注於一個目標可增加內聚力。

Bad use:

class間有許多目標相同或目標未定義明確。

(d)Coupling

(1)Def : 一個class與其他class的相關性稱為耦合(coupling)。一個好的物件導向設計或軟體設計必須符合高內聚、低耦合(high cohesion & low coupling)。

(2)應避免class間的相關性過高。若做一項改變會牽動許多class，表示耦合力高，程式不易修改、除錯與擴充。

(3)

Good use:

每個class有獨立執行的能力，且做一項修改不牽連許多其他的class。

Bad use:

每個class都息息相關，互相牽連，依賴性太高，造成耦合力太高。