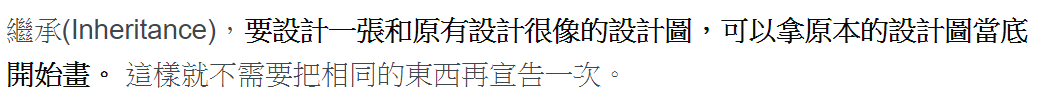
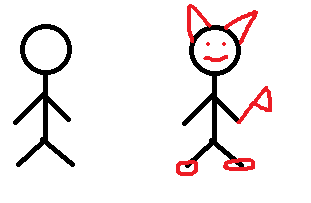
第八章 繼承(Inheritance)

Java語言的第二項重要特性，使用時機相當廣泛與多元，也是物件導向在撰寫時不可或缺的一項技能，實際應用上有將近60%的程式碼會使用到此項特性，所以對此一定要非常透徹了解！

1. 繼承的意義

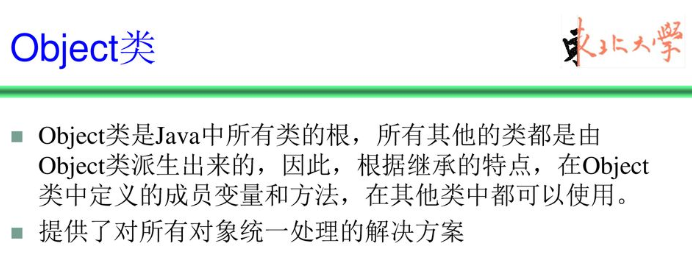
* 是類別與類別的關係，彼此間的資料結構與方法可以共享的概念
* 有著子類別(繼承者) vs父類別(被繼承者)的關係，子類別能承載父類別大部份的內容並自行再擴充使用
* 程式撰寫上常利用已建立好的類別或函式庫，使用繼承功能接續它的相關變數與方法，製作成符合自身需要的功能，可節省撰寫時間與提升設計上的靈活性
* 子類別就是直接承接父類別已經寫好的方法，降低自行撰寫可能產生的錯誤與時間浪費，是一種延伸與重複使用的概念
* 一旦類別如果冠上final，即不可被繼承 (復習一下final的特性)



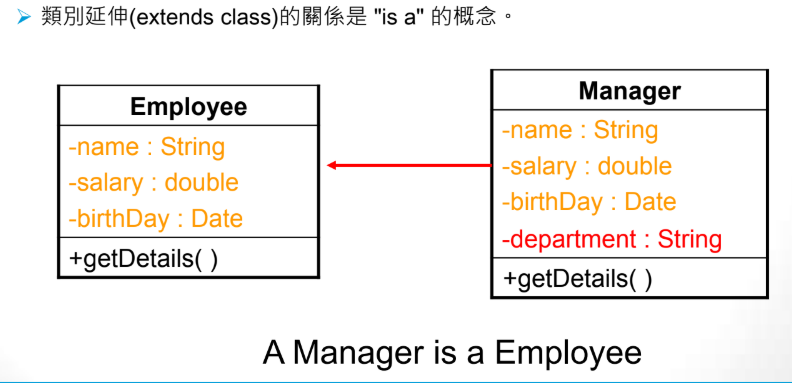


→拿原本的人物模板做延伸，就不用再重畫一次人物模型了！

* 所有類別的父類別 >> Object類！



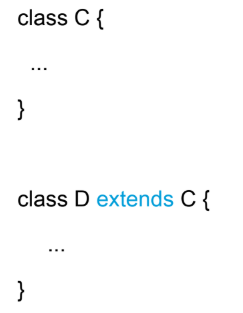
* 程式預設就已繼承了object類，所以才可以使用很多看起來預設可用，實際上是繼承後得到的方法



父類別

子類別

1. 類別繼承的寫法



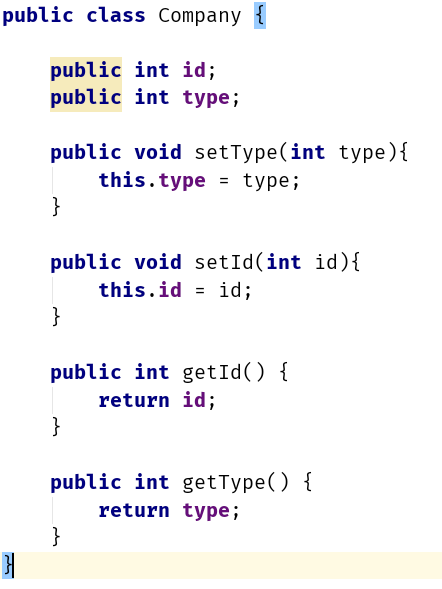
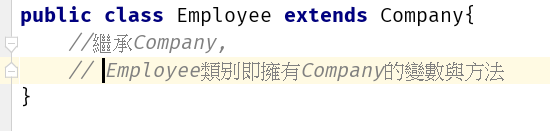
父類別

子類別

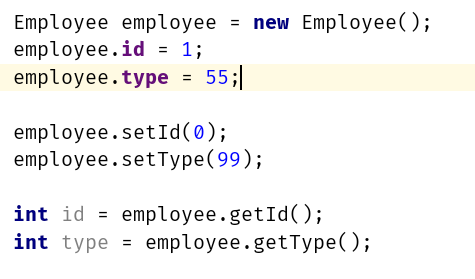
→可理解為類別D繼承了類別C

→至此類別D可以使用類別C所有可被繼承的項目(private以外的屬性&方法)

1. 實際應用

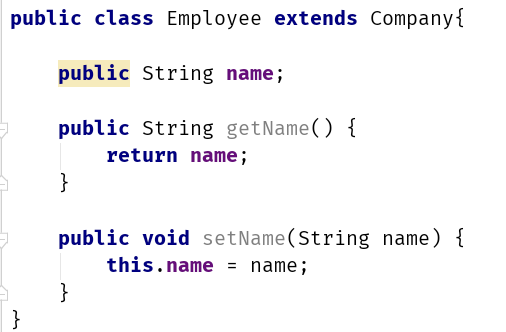


1. Main方法中

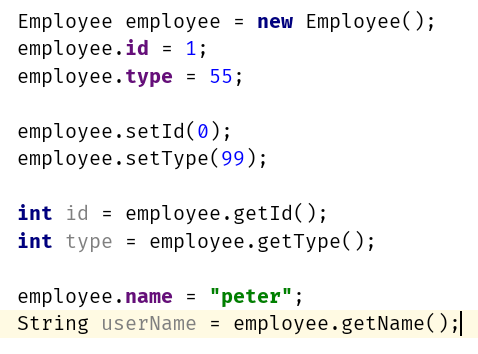


→Employee使用了Company的屬性&方法！

1. 除此之外，Employee類別也可以擁有自己的屬性與方法



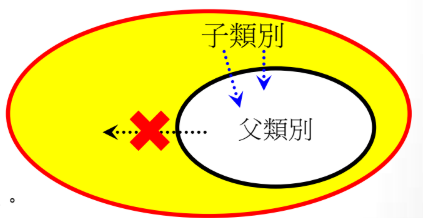
1. Main方法中的使用

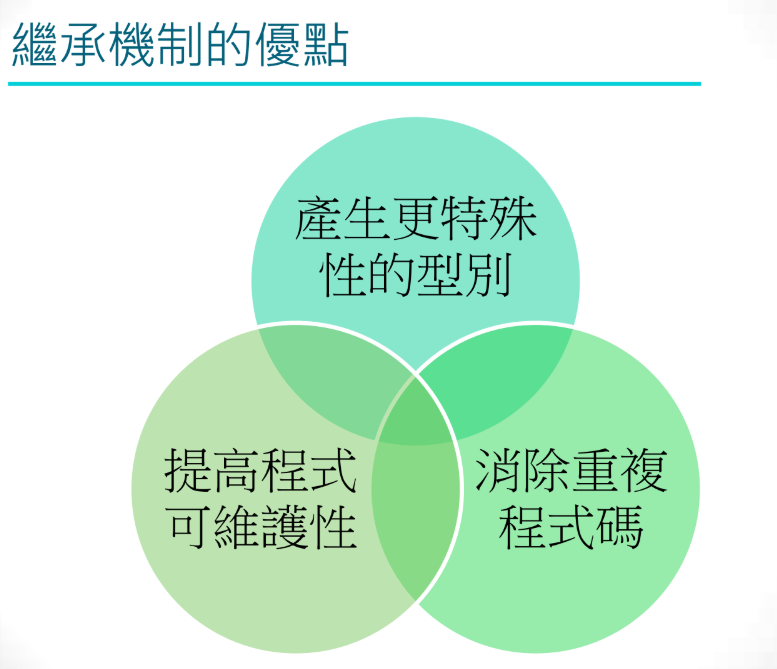
 Employee自己的東西

Employee繼承來的東西

1. 注意事項

* 子類別的屬性與方法只有它自己才認得，父類別的不能使用子類別的東西
* 父類別標記為private的不會被繼承，其它類別中是看不見的也不能用
* 父類別本身如果為final性質也不可被繼承
* 父類別的建構式不會被繼承，會被繼承的只有屬性跟方法，所以建構子的修飾詞對於繼承毫無影響



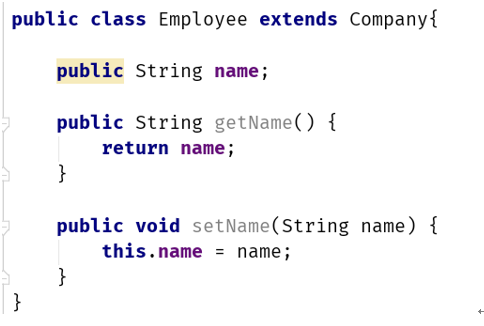


1. Is-a 和has-a的觀念

* has-a

→has-a有屬於的關係，例如a屬於b。

→簡單來說，物件與物件的內容有從屬關係



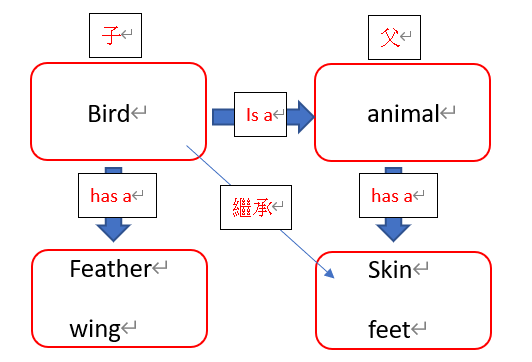
→Employee類別內有屬性name, 所以可以說，Employee has a name

* is-a

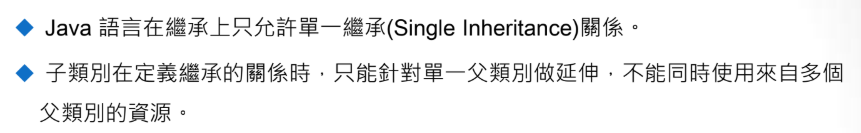
→有上下的關係，例如我是你的孩子、你是我的員工等等

→以繼承觀念來說的話, Bird is a Animal , Animal就是父類別的觀念，Bird是子類別

→上下關係要厘清，不能說Animal is a Bird, Animal是統稱，而bird是其中一種動物

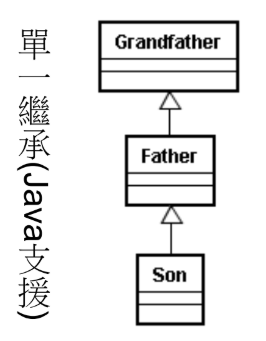


1. 單一繼承性

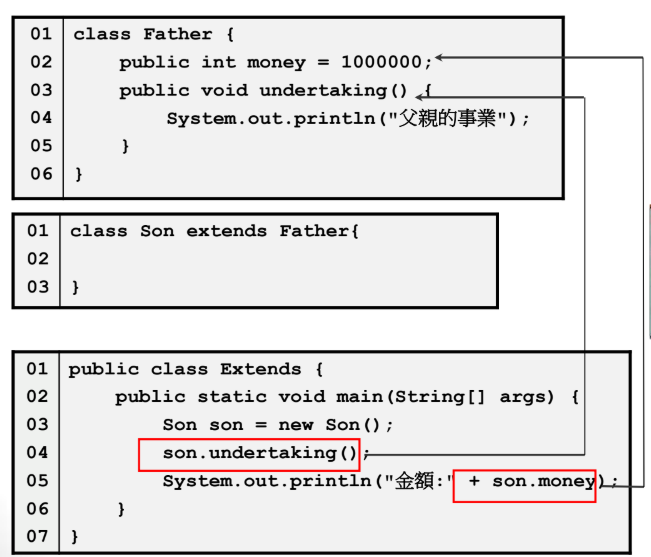




* class 繼承的項目只能有一件



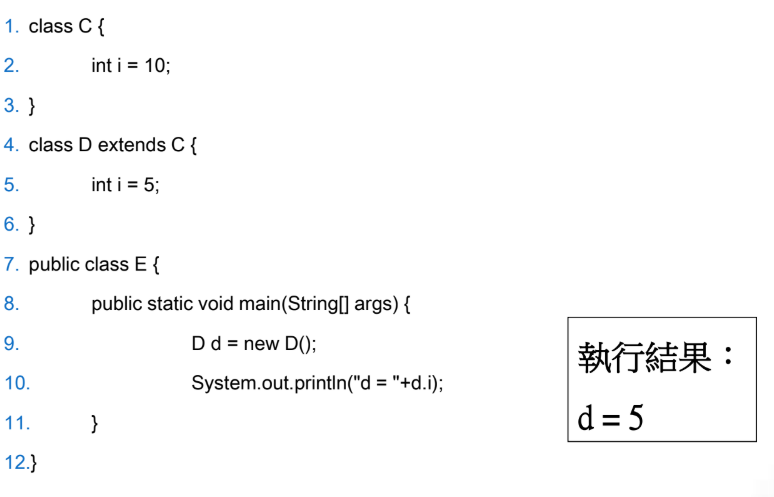
1. 繼承範例



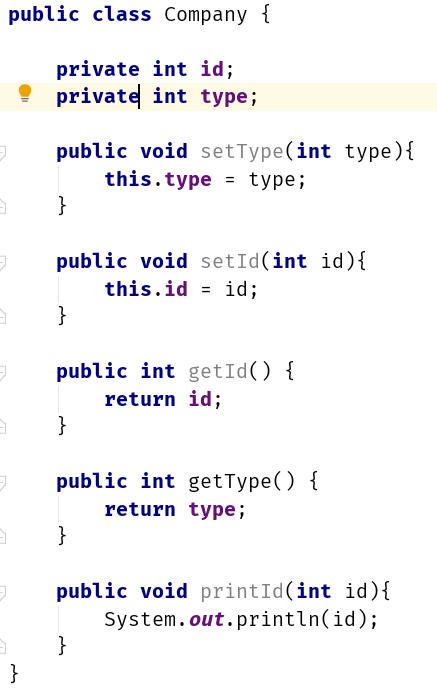
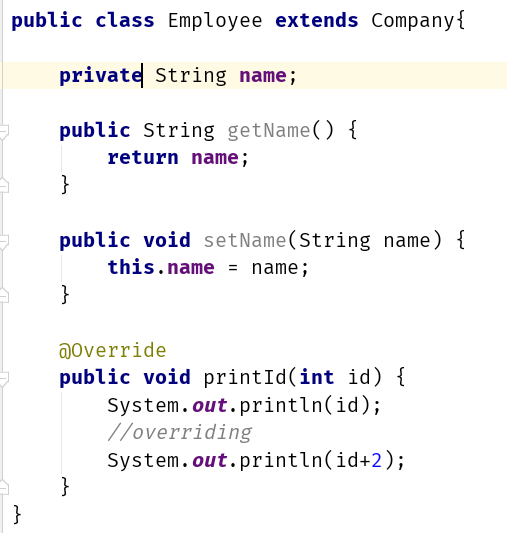
1. 覆寫技巧(overriding)

* 子類別繼承過來的屬性與方法加以改寫的技巧
* 繼承過來的變數可以重新定義數值，方法可以重寫，以達到符合該類別的行為

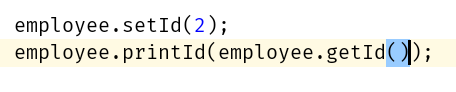
1. 變數覆寫



1. 方法覆寫



In main：



輸出結果：

結論：

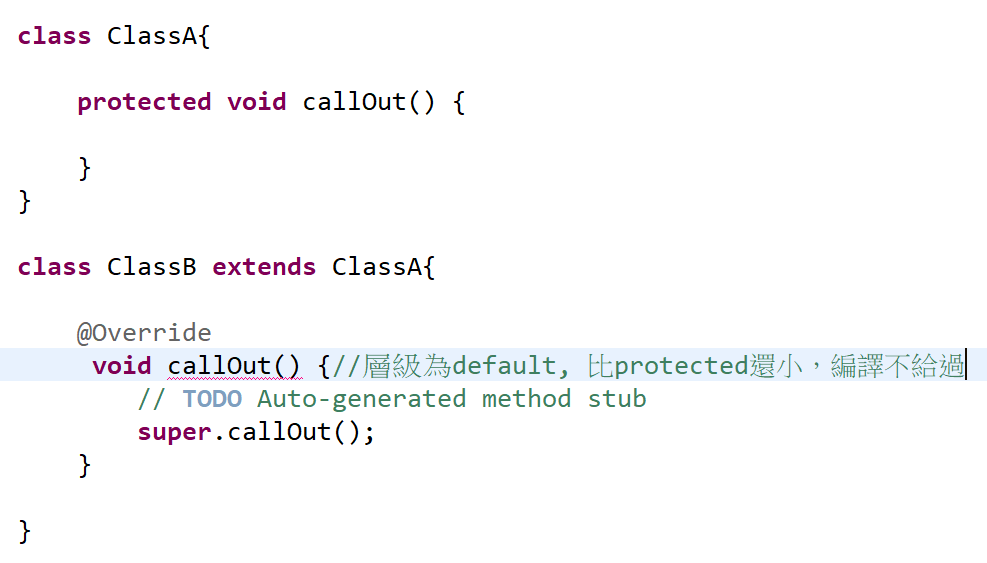
→子類別改寫父類別方法之後，使用子類別建立的物件呼叫的父方法，會執行子類別改寫的方法。這也是覆寫的意義所在

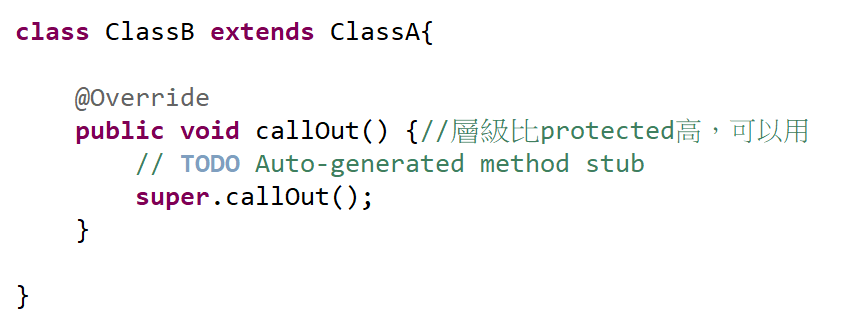
→承上，父類別原有的方法不變，只是隱藏起來，若使用父類別建立的物件呼叫的方法，仍會執行父類別原有的方法

(覆寫只是改寫，不會刪除原本父類別的方法)

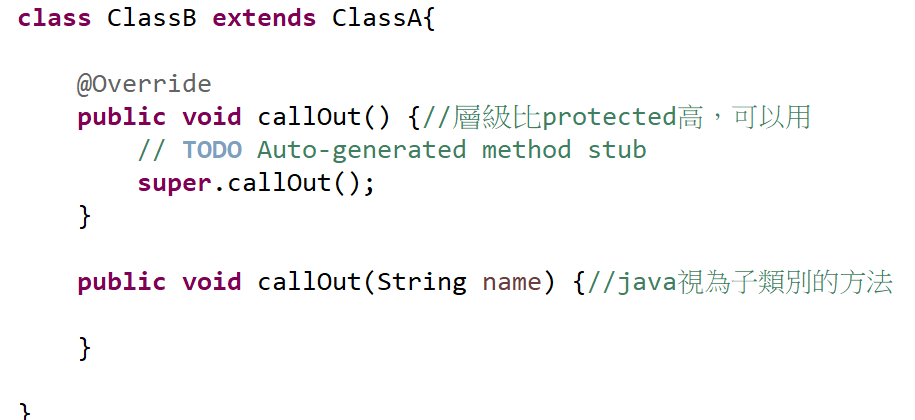
1. 繼承方法的限制

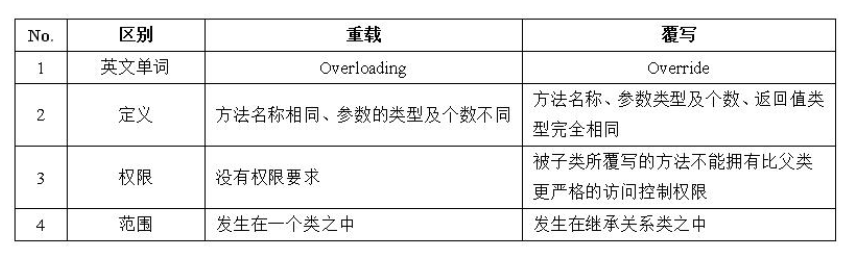
* 修飾詞範圍只能變大不能變小





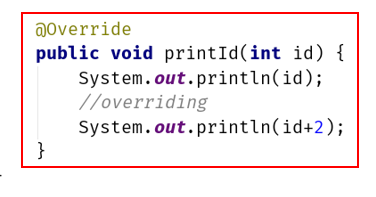
* 可多載，java會視為別的方法



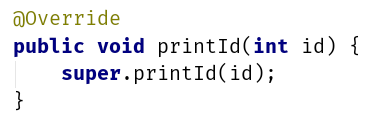


1. Super關鍵字

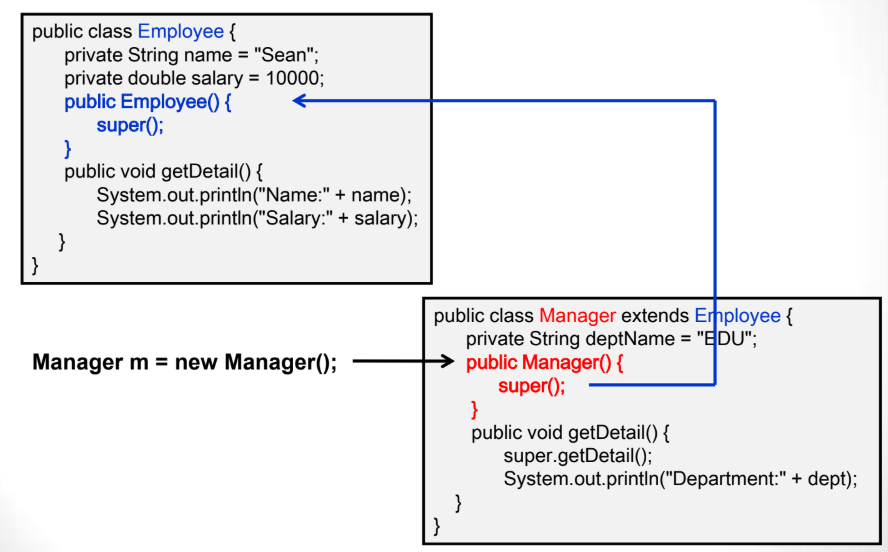
* 不是超人的意思！
* 父類別就是子類別的superClass
* super在程式中有”承上題”，承襲以上所述的意思



可改寫為：

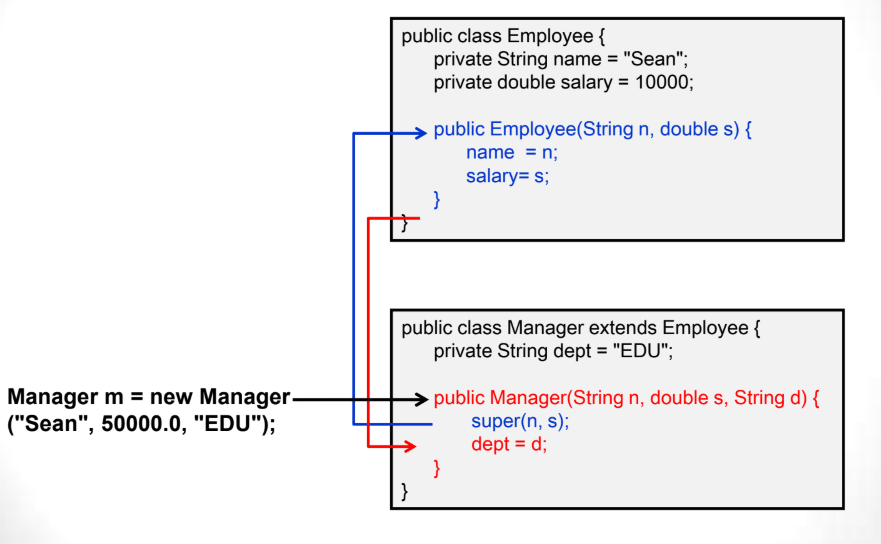


→Super代表的是父類別原先寫好的動作，下面再依需要的動作改寫

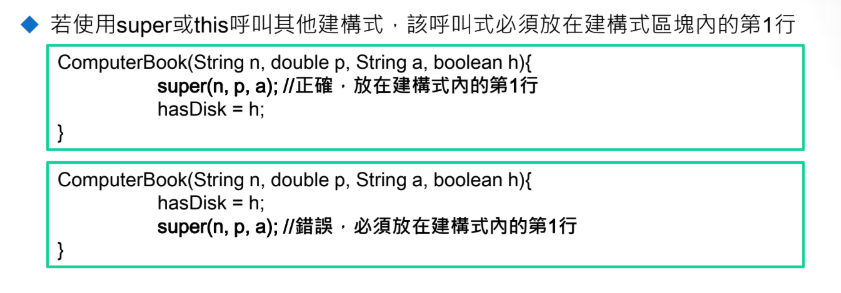


預設的super

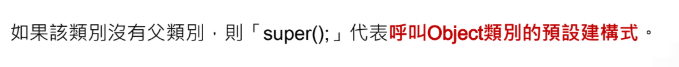
1. 建構式的super

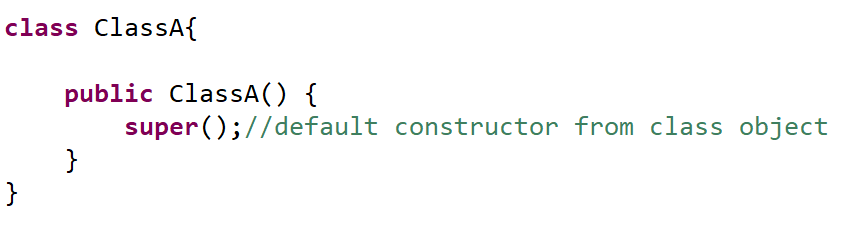


Super擺放的位置



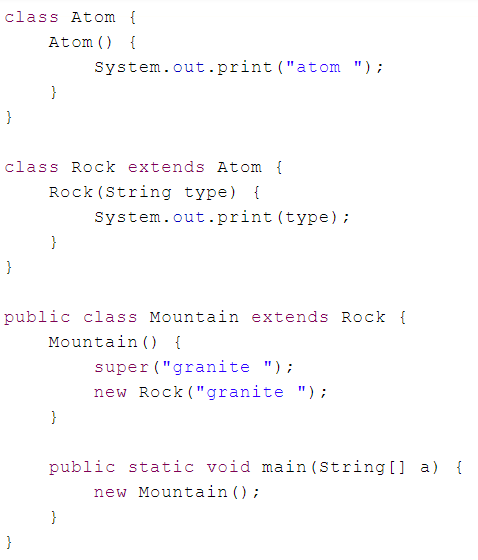
1. 補充說明

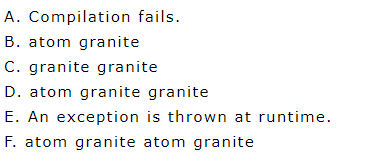




1. 父類別在子類別的預設建構式

**建構子的第一行如果沒有寫super或this的話，預設會給super()在第一行**。意思就是子類別的建構子在運作之前，一定會先運行父類別的建構子，以此類推，父類別中會運行父類的父類的建構子，如以下範例：

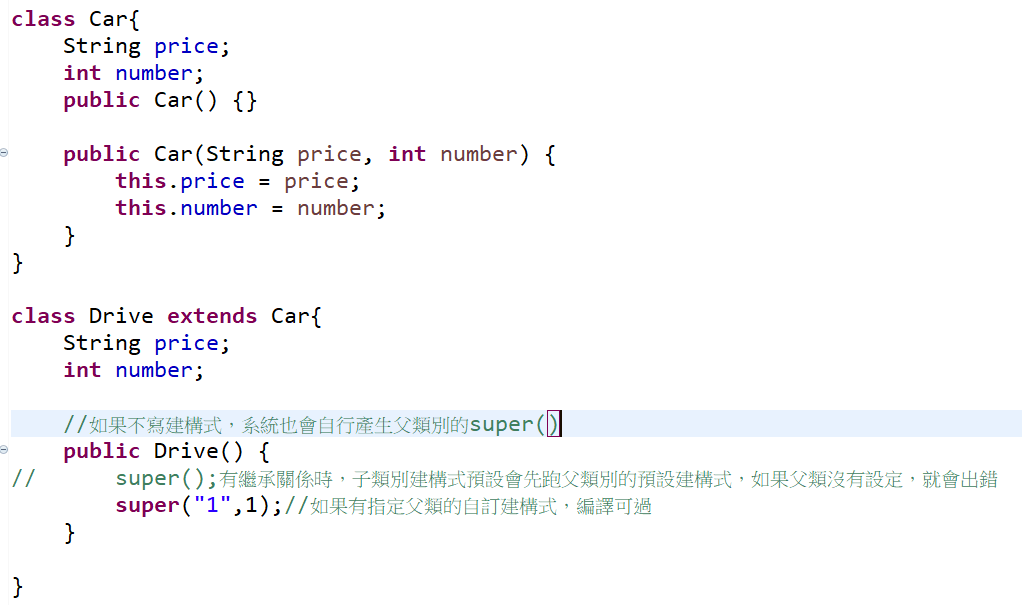
Ex：

Ans:  F



default

1. 繼承與建構子的關係



1. 建構式與super和this的交互應用

\*雲端檔案number8

|  |  |  |
| --- | --- | --- |
| 區別 | this | Super |
| 定義 | 表示該類別本身 | 表示父類的對象 |
| 使用 | This.屬性，this.方法、this() | 子類中操作，super.屬性，super.方法,super() |
| 調用時機 | 需放在第一行 | 需放在第一行 |
| 查找範圍 | 只找該類別 | 子類直接找父類 |
| 特殊情況 | 表當前對象 | 無 |

1. 程式的可讀性與維護性

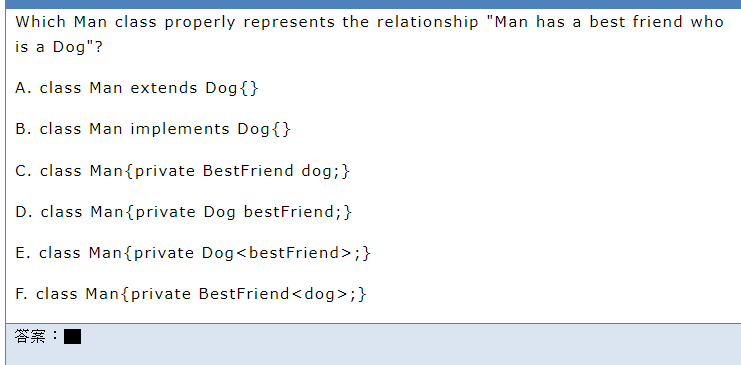
一個好的程式必須具備高度的凝聚性(cohesion)與較低的耦合性(coupling)，以下說明這兩種名詞的意義：

* Cohesion：凝聚力的意思，指的是程式中各個類別上具有良好的分工，一個好的物件導向程式，在分工上需要分配得恰當好，減少不必要重複的程式碼，但又不會漏掉重要的工作
* Coupling：指的是外部的接合、耦合的意思，指的是類別之間所要交流資訊的頻率，一般來說，需要交流的資訊次數愈多，愈容易產生資料丟失的風險，所以一個好的程式會盡量減少交流的次數

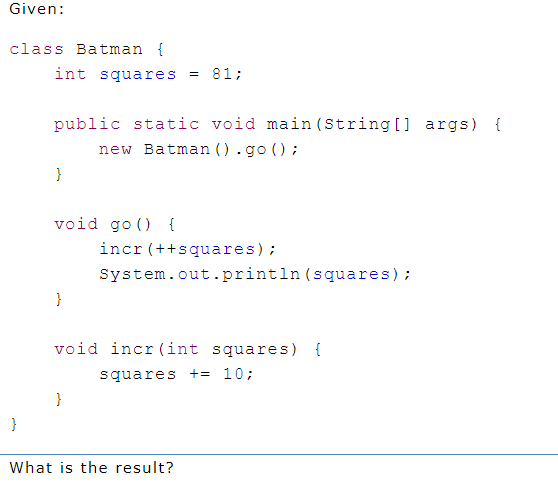


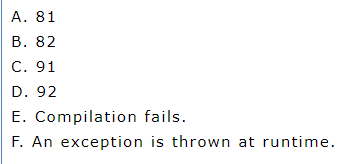
Ans: B C

\*練習：

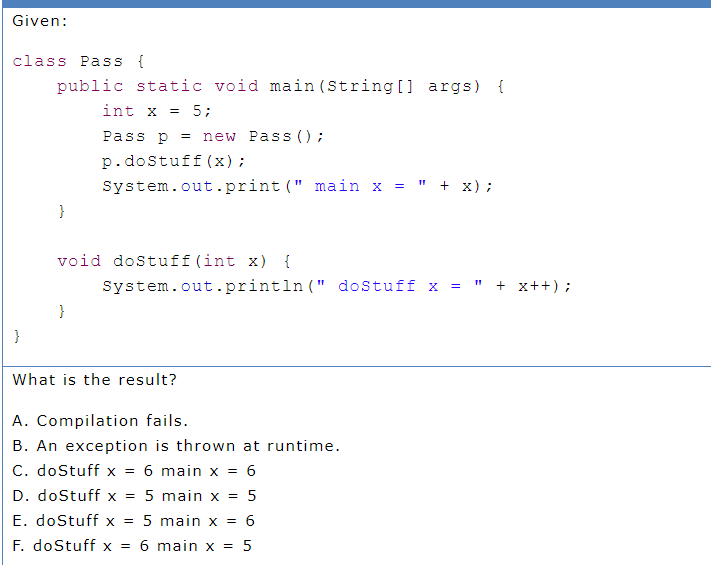


Ans: D



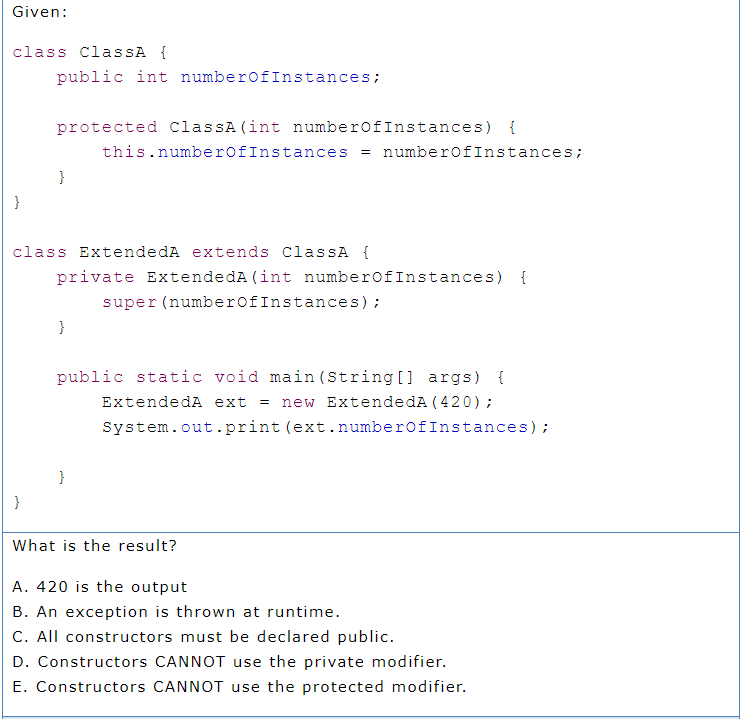


Ans: B



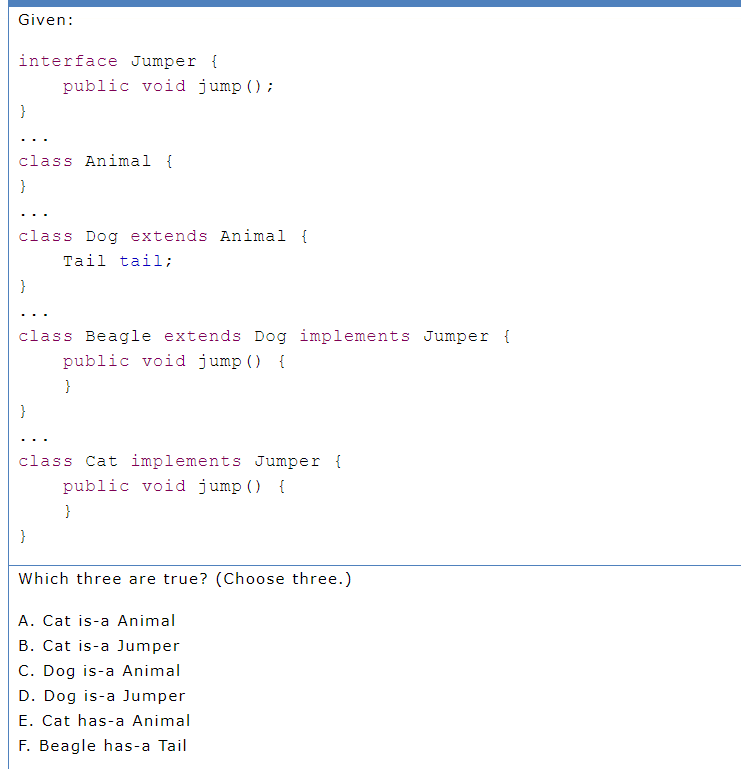
Ans: D

1. 建構式與修飾詞的關係

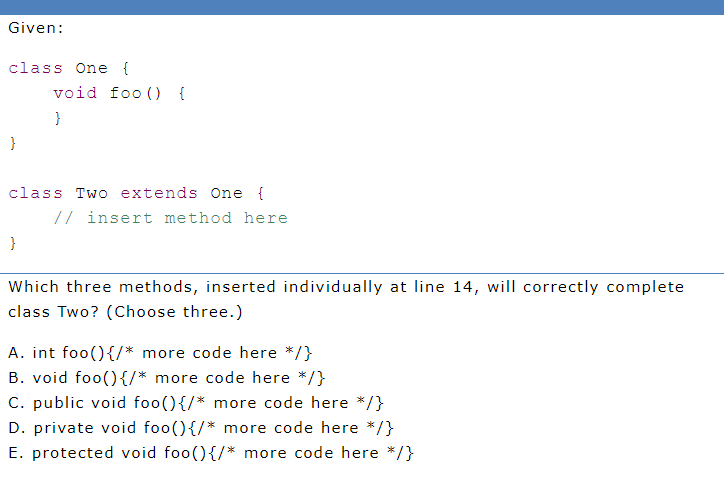


Ans: A

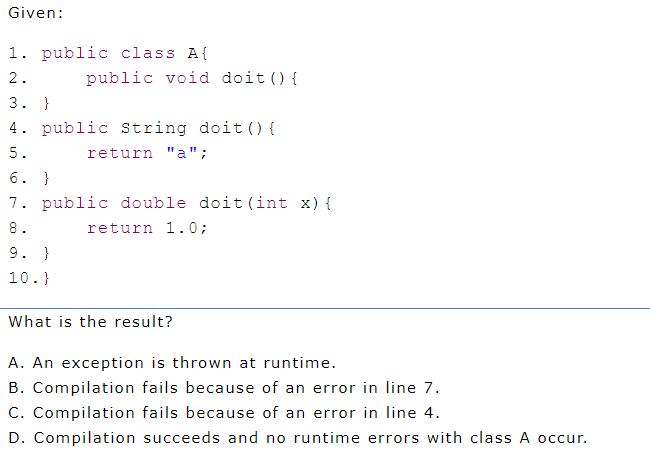
→建構式雖常寫成public, 但它與方法結構類似，可調整修飾詞，不會影響繼承時的讀取，因為建構子不會被繼承，所以就沒有像方法或屬性一樣有可見度的問題



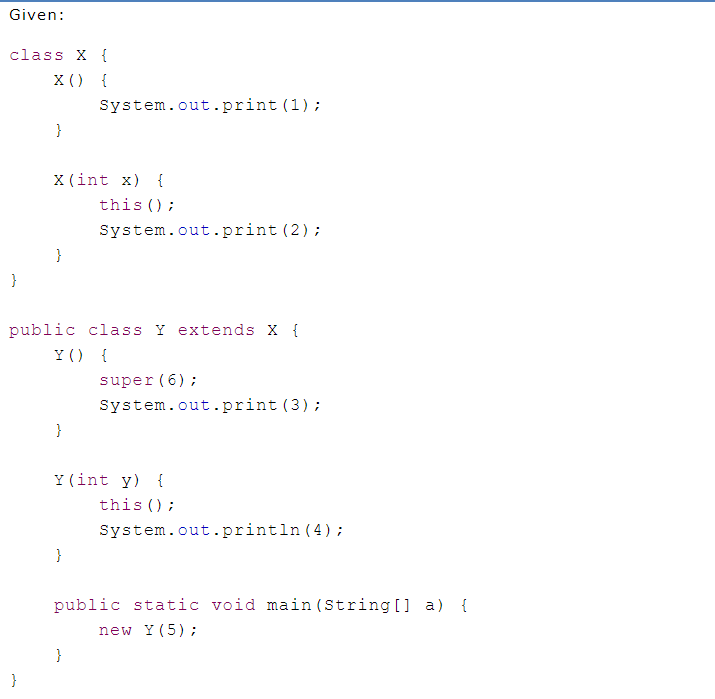
Ans: B C F

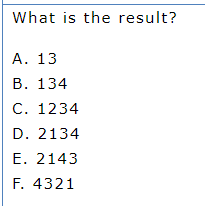


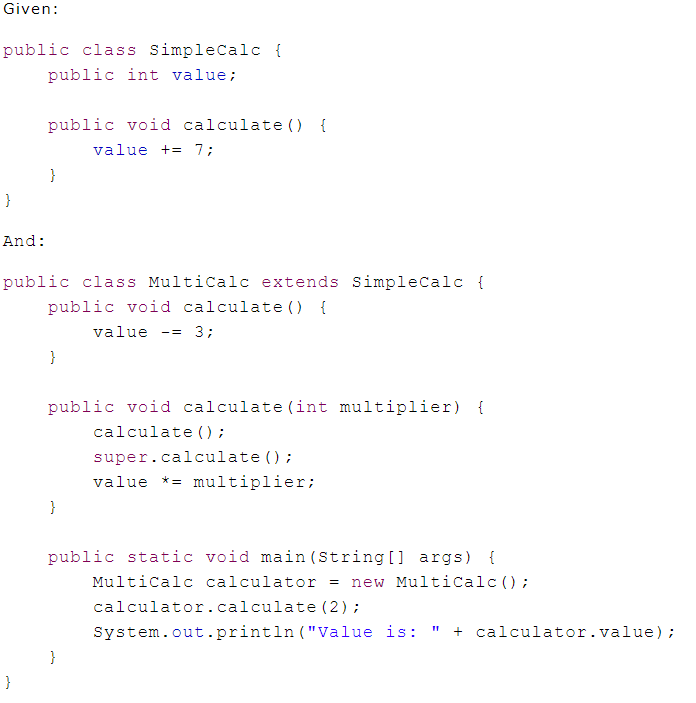
Ans:BCE

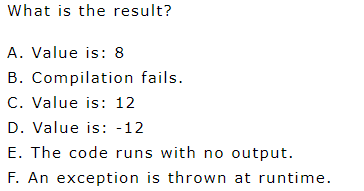


Ans: C



 ans: C





Ans: A

\*練習：某公寓內有三個不同的房客，要住在在這公寓內有一項規定，那就是只要在浴室洗完澡就必須把水漬擦乾，每位房客入住時都會有基本的房號、鑰匙跟是否可打擾的牌子各自保管並使用，廚房可以煮飯，可以依照自己喜好煮不一樣的食物。

而某天晚上三人洗完澡後各自煮了不同的菜餚。

上述劇本請試著用程式語言寫出(提示：class設計，繼承，覆寫，static技巧)