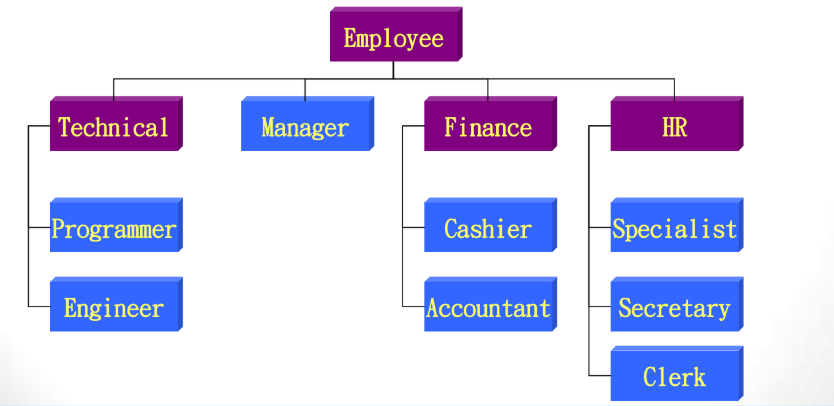
第十章 抽象與介面

* 抽象類別

→意指不用來建立實體也不能建立實體的類別。就像現實世界中動物vs狗的概念，動物就是一個抽象的觀念，而狗是具體的一個物種。

|  |  |
| --- | --- |
| **概念** | **實體** |
| 動物 | 狗 |
| 水果 | 香蕉 |
| 職業 | 走路工 |

→抽象類別的主要功能是只能被繼承來使用的類別



→如上圖各部門即是一個抽象的概念，目的是為了將同類型的人聚集在一起

→舉例：所有人資部的員工(會計、秘書、助理)都可以運用該部門的權利或資源，但人資部本身不是一個人的職稱

* 抽象類別

→關鍵字：abstract

→抽象類別：



→前面冠上abstract修飾詞的類別即為抽象類別

* 抽象方法：



→只宣告方法名稱，無實作

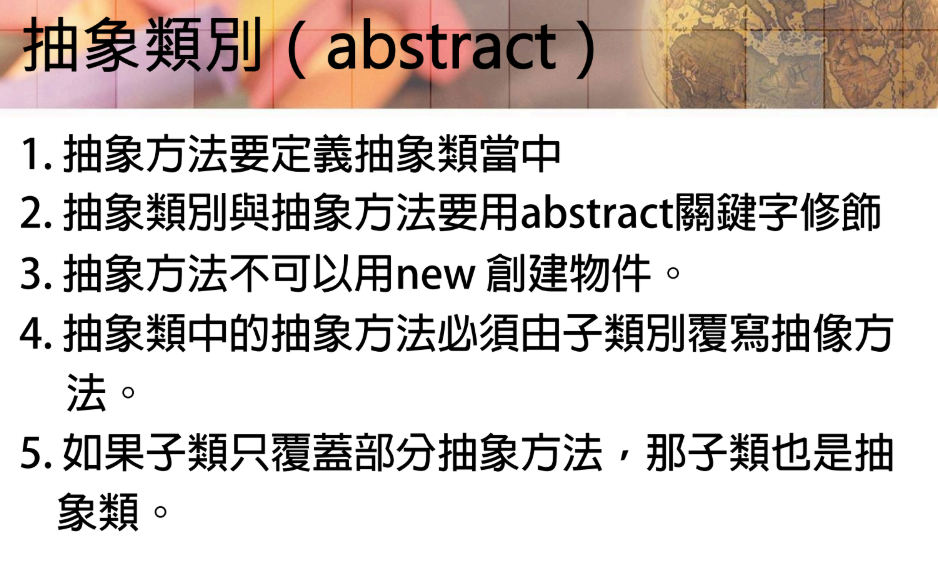
→強迫其它繼承它的類別一定要實作抽象方法

抽象類別的重要特性：(非常重要！！！)

1. 只要類別中有一個抽象方法，該類別必定為抽象類別。
2. 定義為抽象類別的類別不一定要有抽象方法
3. 抽象類別本身不能建立實體，但可以有屬性、建構子和方法
4. 繼承該抽象類別的類別，一定要實作抽象方法，不然要繼續宣告其為抽象類別

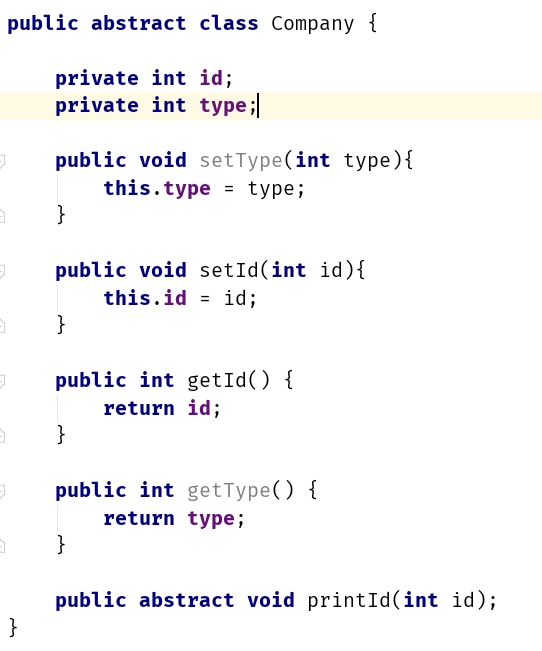
PS：實作的特徵：只要有大刮弧出現就是實作

1. abstract與final、private不能共存，因為抽象類別本身就是要給別人繼承後改寫的，所以不具改寫功能的修飾詞自然不能共存

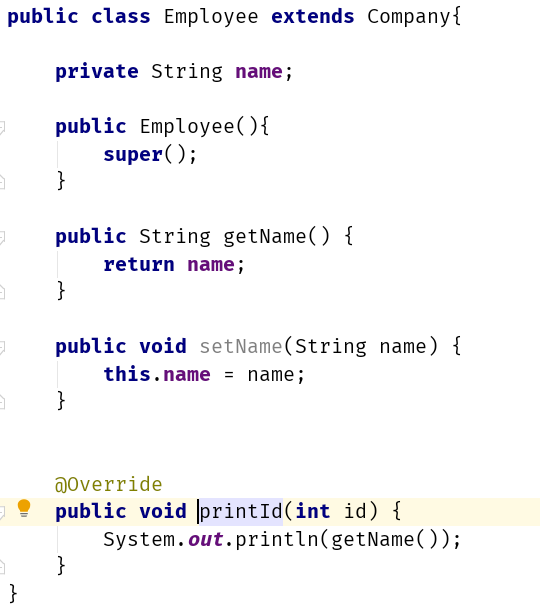


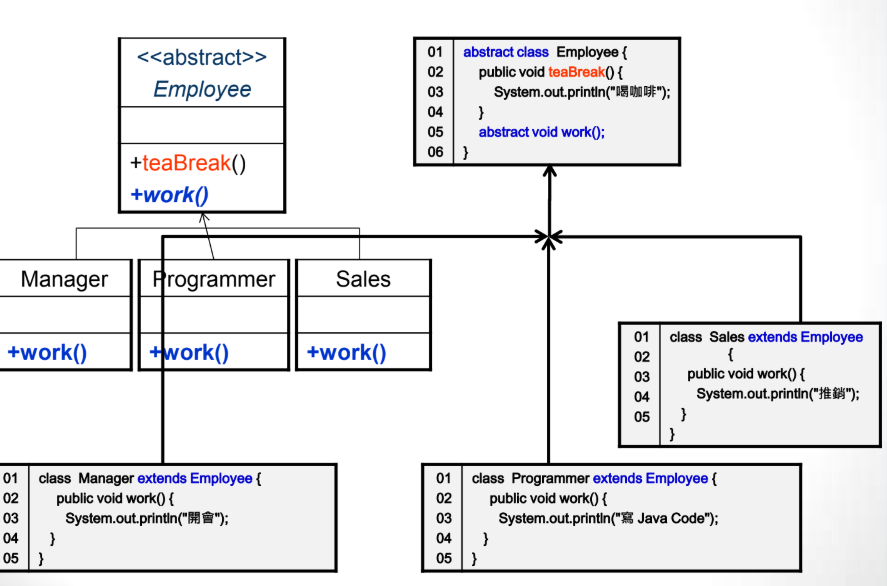
* 抽象類別的應用

→Company抽象類別



→繼承Company class的Employee類別

必須實作！！！





* 介面(interface)

→概念為使方法更彈性運用的一個類別

→一種規範或準則的概念

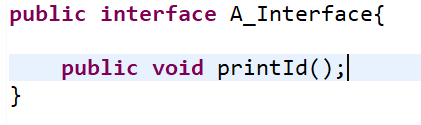
→可用來補足單一繼承無法辦到的其它事情

* 介面特性

1. 修飾詞一定為public
2. 所有的方法都是公開的抽象方法：public abstract(所以可省略abstract修飾詞
3. 沒有建構子，所以也不能實體化
4. 屬性皆為public static final常數，也是省略，設定常數時一定要有初始值
5. 一類別可以同時實作多個介面
6. 一類別實別該介面時，需要實作該介面的所有方法，否則自己也要變成抽象類別。

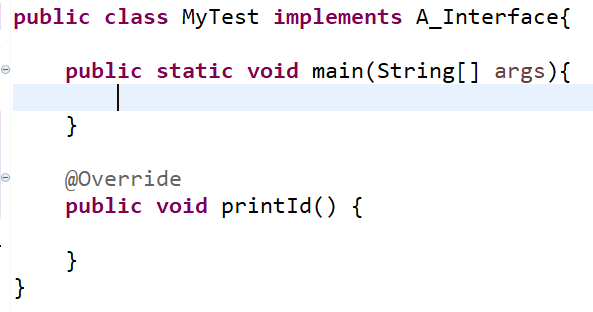


* 介面寫法

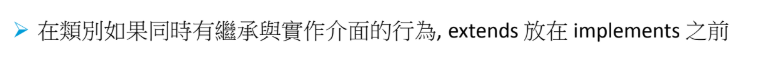


抽象方法沒有大刮弧

* 類別實作介面

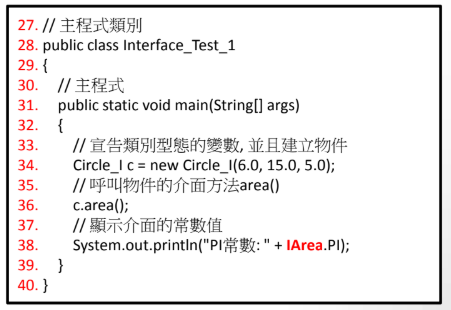
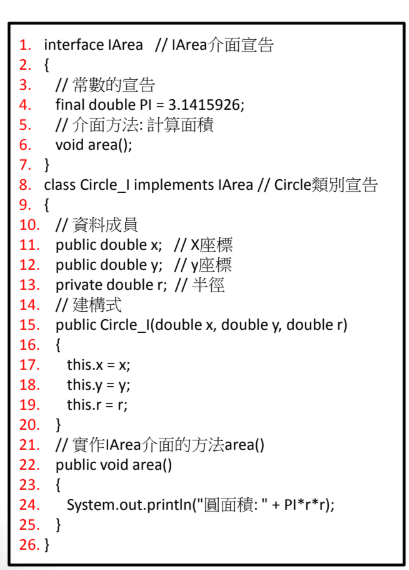




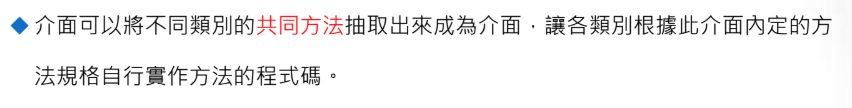


↓

→口訣：先繼承，再實作

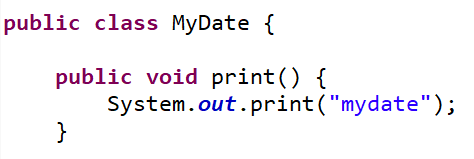


* 抽出共同方法成為介面

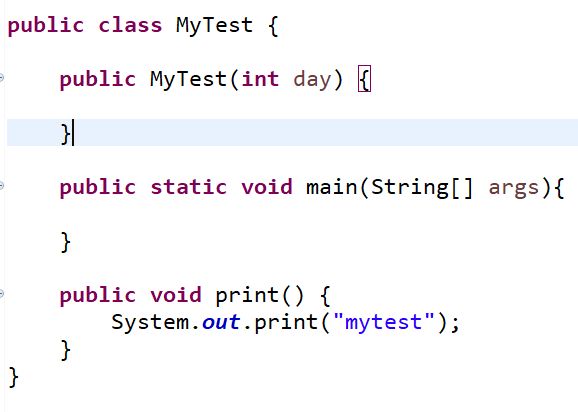




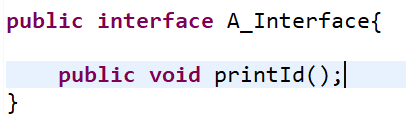
1、



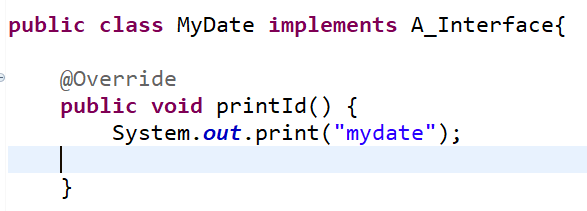
2、

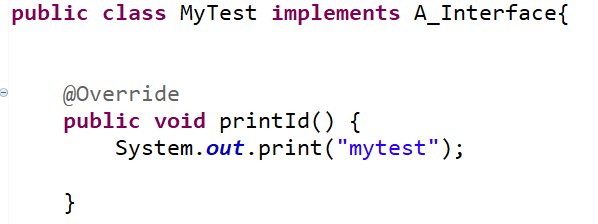


→抽出共同成為介面

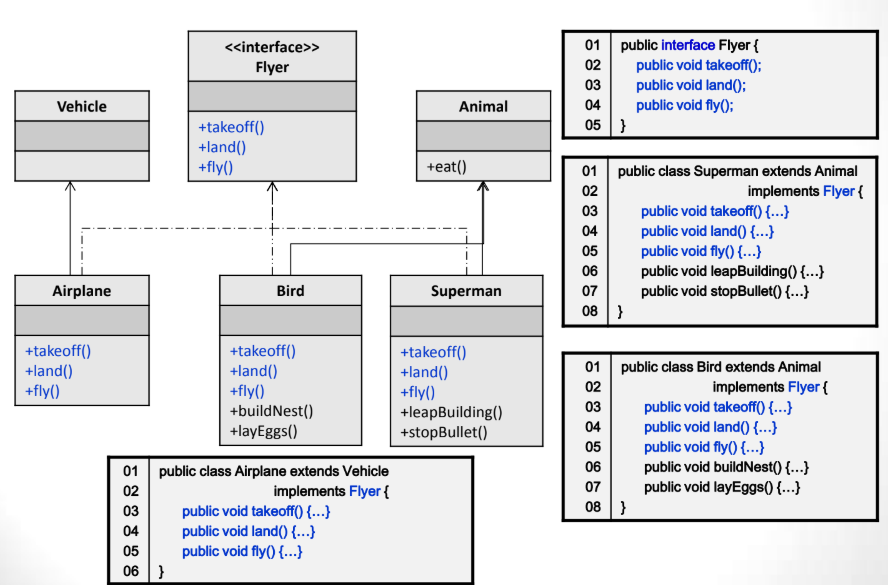


→改寫成介面實作



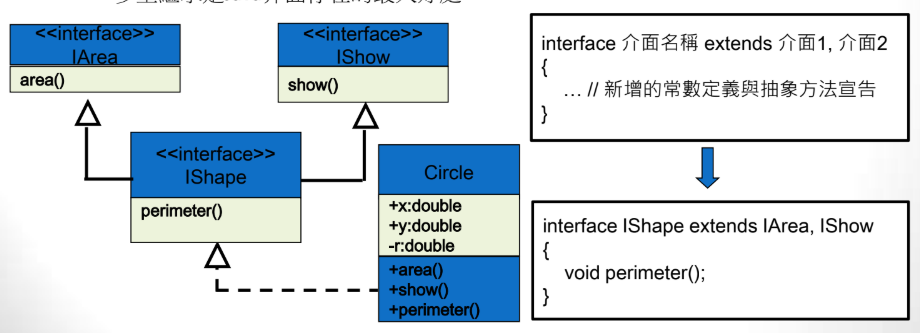


* 另一個範例

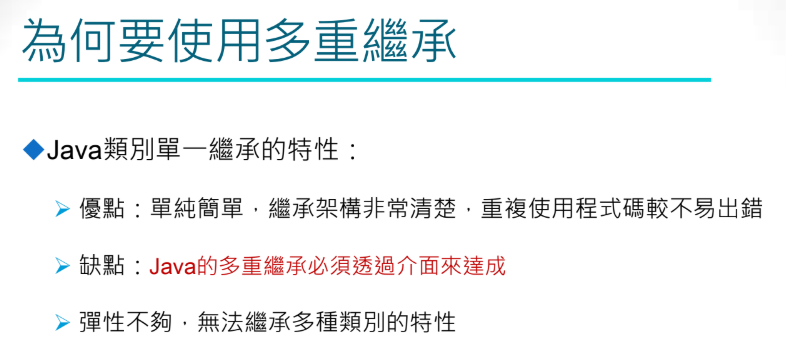


* 介面繼承

→介面可以多重繼承，類別只能單一繼承



→IShape繼承了IArea , IShow，所以Circle繼承Ishape時，同時需實作三個介面的抽象方法。



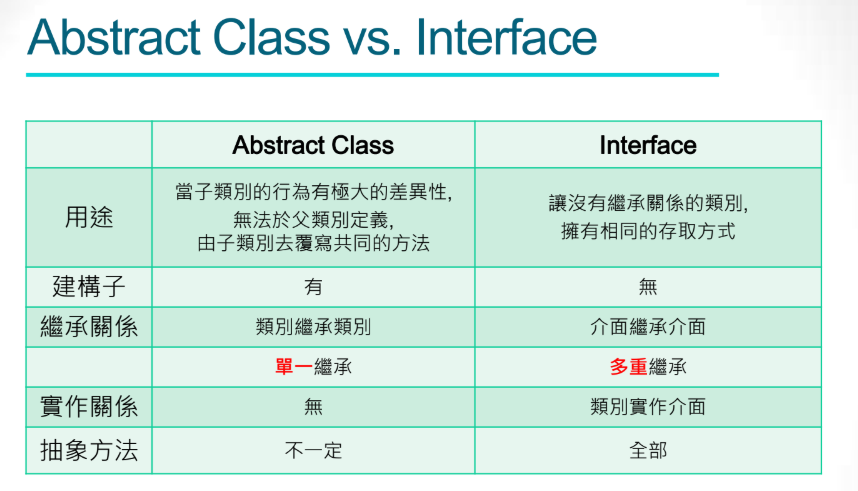
* 抽象與介面比較

相同處：

1. 皆不能實體化
2. 皆可宣告抽象方法

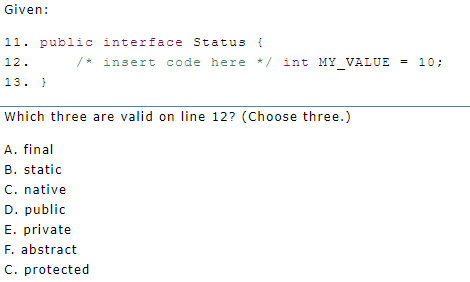
相異處：

1. 抽象類別裡面不一定要有抽象方法(可以有一般方法)，介面一定全是抽象方法
2. 介面屬性一定要是public static final, 抽象可以有一般屬性
3. 介面可以多重繼承，抽象類別不行

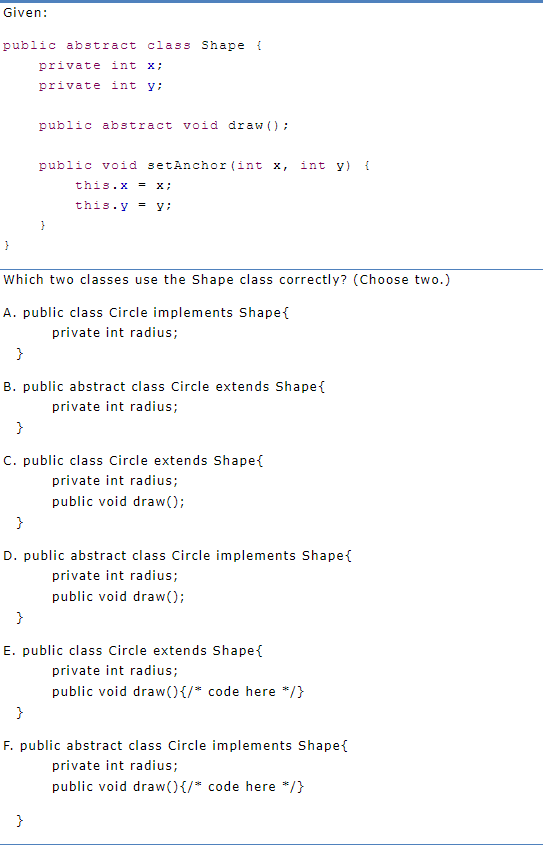


→總而言之，介面方法比較像通俗的行為，例如我跟你做事的方式就不同，但我們同樣要吃飯睡覺，吃飯、睡覺的這些身為人一定會有的動作，就是介面方法

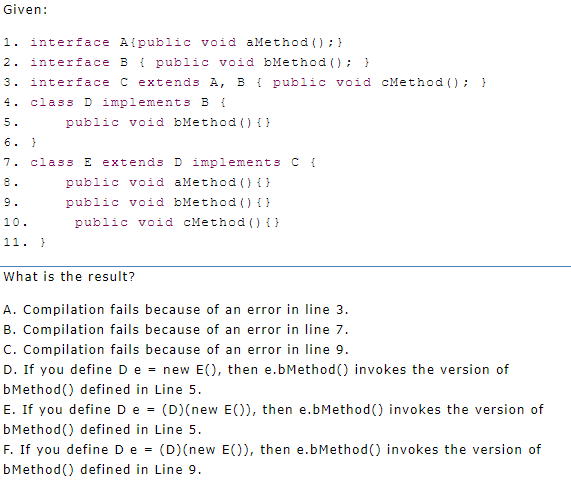
* 練習：某公司規定只要是員工只要開始上班，就必須出示各自的姓名、電話，試用程式表示之。



ANS: A B D

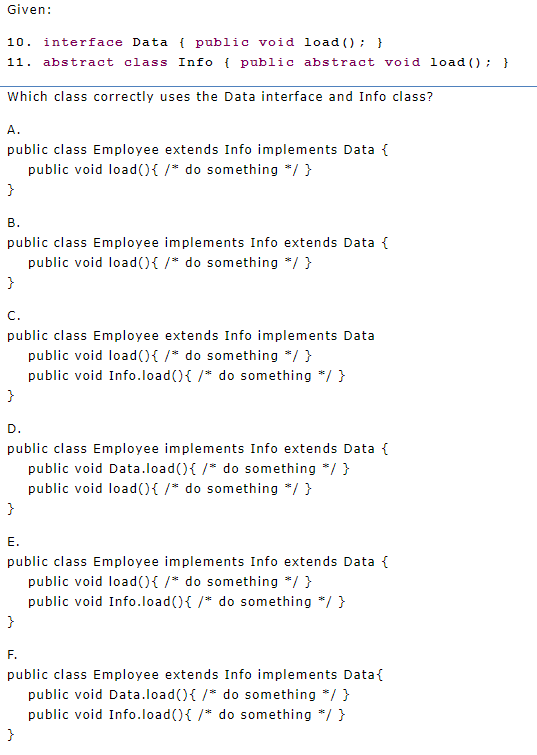


Ans: B E



//可以自己寫寫看，結果跟我說

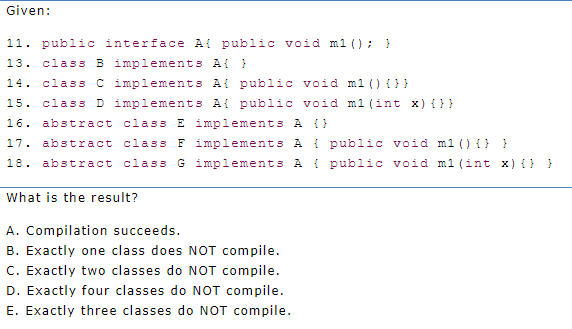
Ans: F

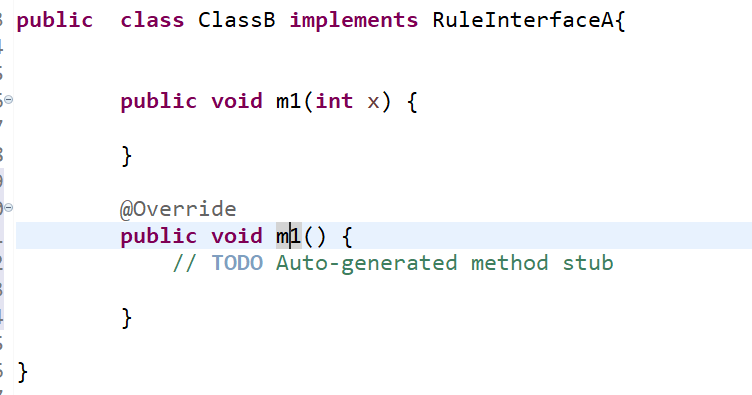


ANS: A

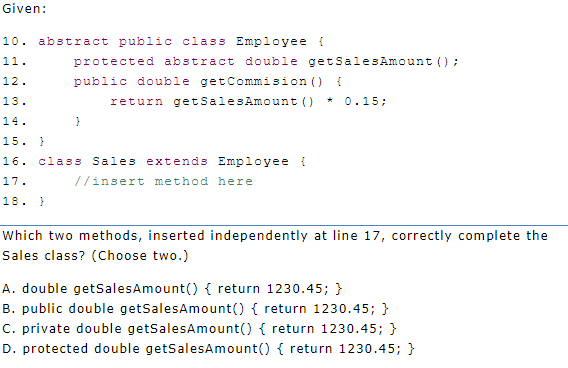


Ans: C D





ANS: C



Ans: B D