1. 為什麼要引入封裝性?  
   ●我們程式設計追求＂高內聚、低耦合＂。  
   　高內聚：類的內部數據操作細節自己完成，不允許外部干涉；  
   　低耦合：僅對外暴露少量的方法用於使用。  
   ●隱藏對象內部的複雜性，只對外公開簡單的接口。便于外界調用，從而提　  
   　高系統的可擴展性、可維護性。通俗的說，把該隱藏起來的隱藏起來，該  
   　暴露出來的暴露出來。這就是封裝性的設計思想。
2. 問題引入：  
   當我們創建一個類的對像以後，我們可以通過"對象.屬性"的方式，對對象的屬性進行賦值。這裡，賦值操作要受到

屬性的數據類型和存儲範圍的製約。除此之外，沒有其他制約條件。但  
是，在實際問題中，我們往往需要給屬性賦值

加入額外的限制條件。這個條件就不能在屬性聲明時體現，我們只能通過方法進行限制條件的添加。（比如：setLegs()）

同時，我們需要避免用戶再使用"對象.屬性"的方式對屬性進行賦值。則需要將屬性聲明為私有的(private).

\* -->此時，針對於屬性就體現了封裝性。

3. 封裝性思想具體的代碼體現：  
體現一：將類的屬性xxx私有化(private),同時，提供公共的(public)方法來獲取(getXxx)和設置(setXxx)此屬性的值  
**private** **double** radius;

**public** **void** setRadius(**double** radius) {

**this**.radius = radius;

}

**public** **double** getRadius() {

**return** radius;

}  
體現二：不對外暴露的私有的方法  
體現三：單例模式（將構造器私有化）  
體現四：如果不希望類在包外被調用，可以將類設置為缺省的。

1. Java規定的四種權限修飾符
   1. 權限從小到大順序為：private < 缺省 < protected < public
   2. 具體的修飾範圍：  
      
   3. 權限修飾符可用來修飾的結構說明：

4種權限都可以用來修飾類的內部結構：屬性、方法、構造器、內部類

修飾類的話，只能使用：缺省、public