**繼承**

**繼承**

繼承是避免多個類別間重複定義了相同行為與實作，使用extends讓子類別繼承父類別，擁有父類別的行為與實作。

Ex:

Public class Magician **extends** Role{….

子類別只能繼承一個父類別。

**is-a**

子類別與父類別間有is-a的關係，也就是子類別繼承父類別，所以子類別是父類別的一種，把自己當編譯器將語法由右往左讀來檢查語法。

Ex:

Dog為父類別，Bullenbeisser為子類別繼承Dog

Dog dog1=new Bullenbeisser();從右邊讀過去Bullenbeisser是一種dog

如果是以下程式

EX:

Dog dog1= new Bullenbeisser();

Bullenbeisser bullenbeisser=dog1;

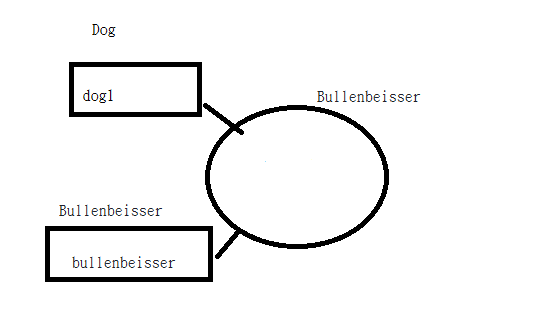
右邊讀過去dog不一定是Bullenbeisser，這時可讓dog**扮演(CAST)** Bullenbeisser，讓編譯器通過。

EX:

Dog dog1= new Bullenbeisser();

Bullenbeisser bullenbeisser= (Bullenbeisser) dog1;

因為dog1參考的物件是Bullenbeisser所以執行結果正確。



**多型**

多型是一個物件參考(父類別)可在不同環境下，扮演不同角色扮演(子類別)，指向不同物件實體。

**重新定義(override)**

子類別繼承父類別，定義與父類別相同方法，但實作方法不同，稱為重新定義，在子類別中重新定義。如果只是想在父類別的方法前後加工，取得父類別中的方法定義可使用super。

Ex:

Public String toString(){

Return “劍士”+super.toString();

}

**抽象方法與抽象類別**

當某方法區塊沒任何程式實作，可以用abstract表示為抽象方法，該方法不用撰寫{}區塊，直接;結束即可。

Java規定內含抽象方法的類別，一定要在class前標示abstract，表示這是一個定義不完整的抽象類別，無法用抽象類別建構實例。

子類別繼承抽象類別，對抽象方法有兩種做法，**一是繼續標示abstract(子類別也是抽象類別)，一是實作抽象方法。**

**建構式**

類別有繼承關係，在建構子類別實例時,會先進行父類別定義的初始流程，再定義子類別定義的初始流程。上面override的super()也可以用於呼叫父類別中某建構式

**Java.lang.Object**

在java中，子類別只能繼承一個父類別，如果定義類別時都沒有使用extend指定繼承類別，那一定是繼承java.lang.Object，因此任何類別追溯最上層父類別，一定是java.lang.Object。也就是java任何物件都是一種Object

Ex:

Object o1=”allen”;

Object o2=new Date();

**垃圾收集**

因為建構物件會佔據記憶體，當程式執行中已無法再使用某個物件，該物件只是徒耗記憶體的垃圾，Jvm有垃圾收集(GC)機制，當物件沒有指標參考是將會被回收。