第八章繼承

資訊科技系
林偉川

8-1 單一繼承

- 類別繼承使用extends關鍵字

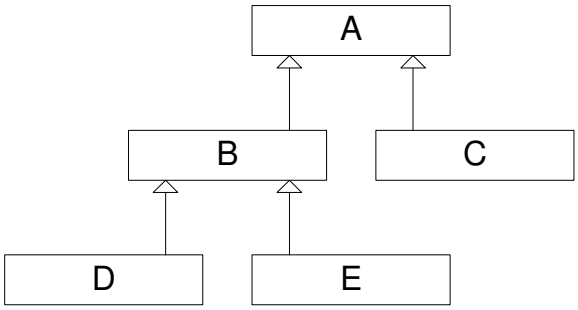
```
修飾字 class 子類別 extends 父類別1 {
        //區塊內的程式敘述
}
```

- 類別以final宣告時，不能做為父類別
- extends關鍵字之後只能接一個父類別

- 例子：

```
class A{}
class B extends A {}
class C extends A {}
class D extends B {}
final class E extends B {}
```

- 繼承關係：



3

預設繼承類別

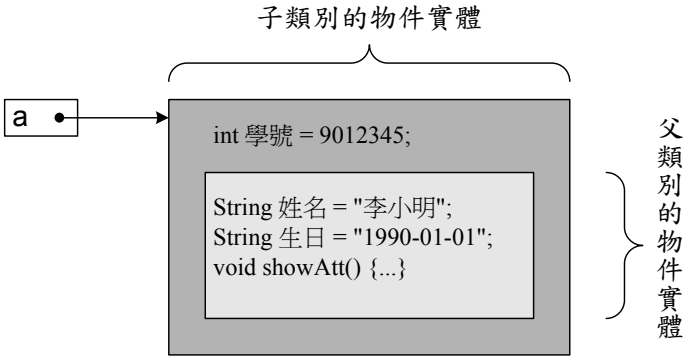
- 所有類別的基礎類別 Object
- 下列兩行的類別定義方式是相同的：

```
class A {}
class A extends Object {}
```

4

基本繼承

- 非private修飾的物件成員都可以被繼承
- 子類別的物件實體包含父類別的物件實體



5

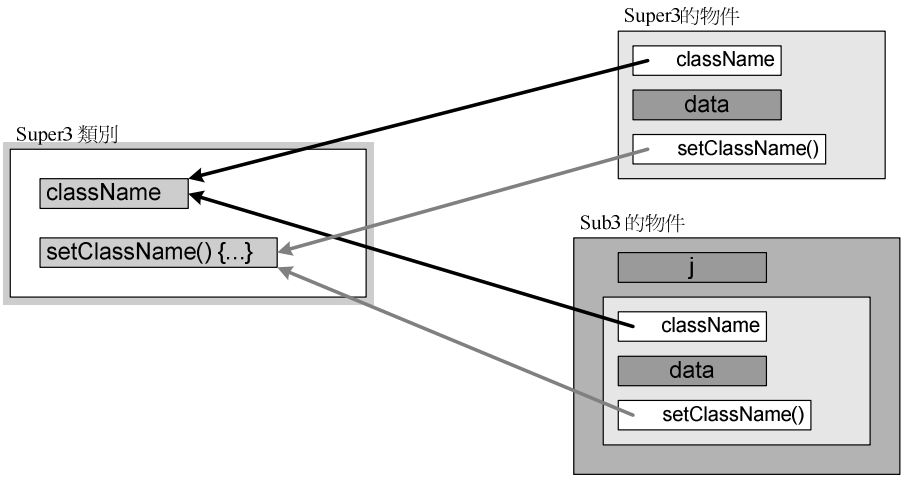
類別成員與繼承

- **類別屬性**經過繼承之後，還是**只有一份資料儲存區**。(宣告為static)
- 這份資料儲存區是父類別、子類別、父類別的物件及子類別的物件所共有。

6

類別成員與繼承

- 繼承之後，類別成員都還是只有一個

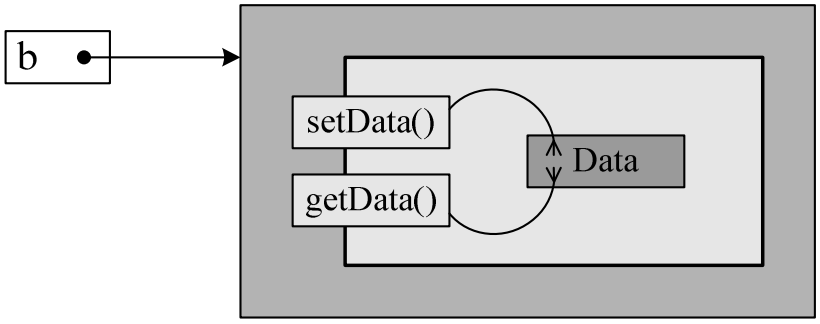


private成員不被繼承

- 以private宣告的父類別之物件成員不被子類別所繼承。(SCJP 32,94)
- 不被繼承的成員存在子類別物件中的父類別物件內，只是子類別物件無權使用。
- 子類別物件可以透過父類別的public方法操縱父類別的private屬性。

private成員不被繼承

- EX8_4：Sub4型別的b可以使用繼承而來的setData()和getData()。



9

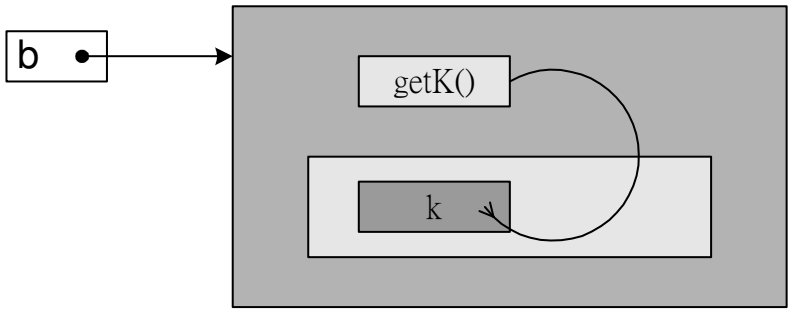
重新定義父類別的物件屬性

- 子類別新定義的成員和父類別的成員實際上是位於不同的領域(scope)。
- 子類別可重新定義與父類別物件屬性同名的屬性。
- 透過子類別物件參照無法存取已被重新定義的父類別屬性。
- 子類別可重新定義與父類別物件同名的方法且回傳型別需相同(SCJP 20)

10

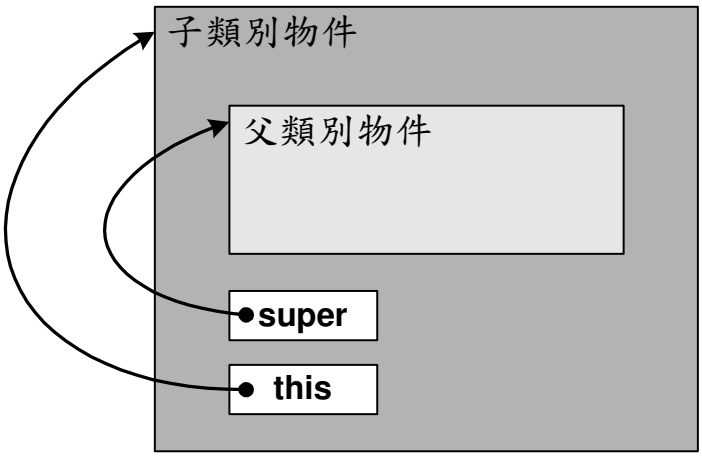
重新定義父類別的物件屬性

EX8_6：子類別的物件方法可以取用父類別的非private之物件屬性。



11

super－父類別物件之參照



12

- **建構子不被繼承**，無法透過子類別名稱去呼叫父類別的建構子。super()➔呼叫父類別的建構子
- 在父類別及子類別都沒有定義建構子的情形下，建立子類別物件時，父類別的預設建構子會先被呼叫，接著子類別的預設建構子才會被呼叫。(SCJP 36)
- 如果是自訂的建構子，父類別的無參數建構子會先被呼叫，接著子類別的無參數建構子才被呼叫。
- 建立延伸類別的物件時，延伸類別的每個基礎類別之建構子都會被呼叫。

使用super()呼叫父類別建構子

- super()的功能是呼叫父類別的建構子。
- this()和super()都只能在建構子內使用，並且必須出現在建構子的第一個敘述。(SCJP111)
- this()和super()只能擇一而用，不能同時使用。

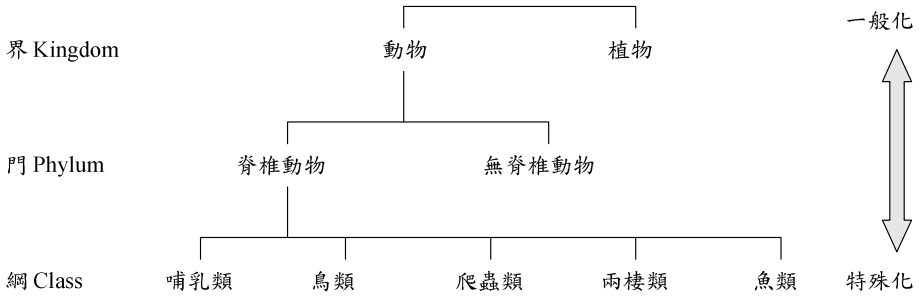
使用super()呼叫父類別建構子

- 子類別建構子未使用super(<參數列>)時，編譯器會自動加入預設的父類別建構子呼叫。

```
class 父類別名(){
  父類別名(){}
}
class 子類別名() extends父類別名 {//SCJP 67
  子類別名(){
    super();      //若無此敘述，則編譯器會自動加入
    //程式敘述
  }
  子類別名(形式參數列){
    super();      //若無此敘述，則編譯器會自動加入
    //程式敘述
  }
}
```

15

- 子類別物件也屬於父類別物件



小紅是一條魚。
小紅是一隻脊椎動物。
小紅是一隻動物。

16

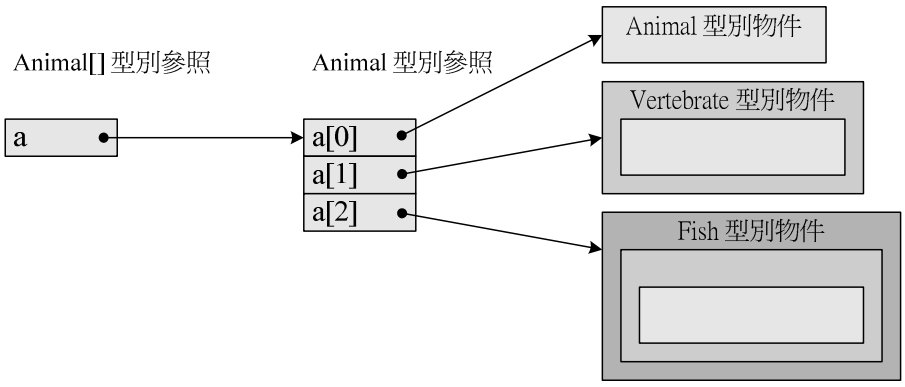
異質集合

- 相同類別的物件集合稱為**同質集合**(homogenous collections)。
- 不同類別的物件集合稱為**異質集合**(heterogeneous collections)。
- 基本型別的陣列為同質集合。
- 當父類別型別的**參照陣列**之元素分別指向不同的子類別物件時，此參照陣列稱為**異質集合**。
- **異質集合**建立在父類別型別參照變數可以指向子類別物件的原理上。

17

異質集合

- 範例8_13：異質集合的物件陣列



18

instanceof運算子

- instanceof運算式會得到一個布林值，若物件名稱所指向的物件實體屬於欲判斷的類別則回傳true；反之，回傳false。

```
物件名稱  instanceof  類別名稱
```

- **物件名稱**所屬的**參照型別**必須和**類別名稱**有繼承關係，否則會發生錯誤。

- instanceof的優先權和關係運算子的 <、>、<=、>= 相同，它們的運算結果都是布林值。

19

8-4 物件的型別轉換

物件的強制型別轉換

- 物件型別轉換，其實是**針對指向物件的參照變數，而不是針對物件實體**。物件實體只要被建立，其佔用的記憶體位置及大小都不會被變更。

- 物件的強制型別轉換之語法

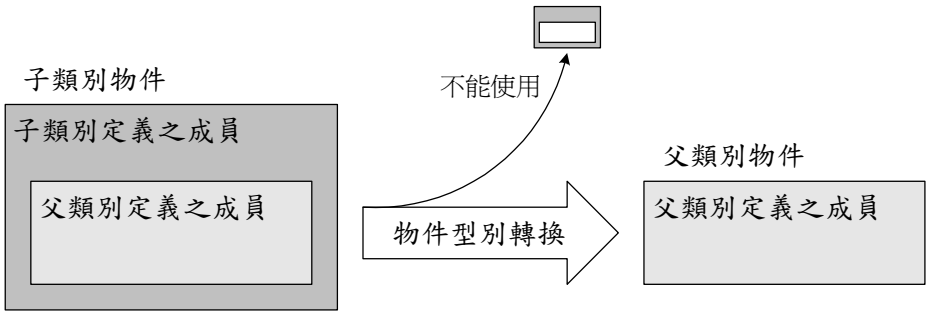
```
(目的類別)物件參照變數
```

  - **物件參照變數**所指向的物件實體，必須是**目的類別**的物件實體，或是其子類別的物件實體。

20

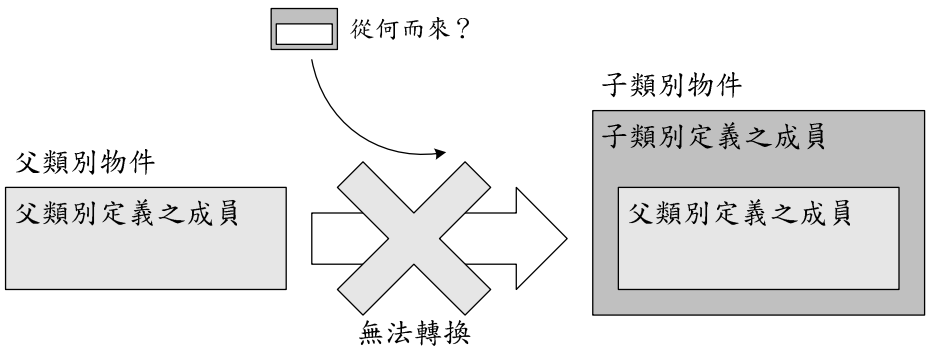
物件的強制型別轉換

- 子類別物件轉換成父類別物件



物件的強制型別轉換

- 父類別物件不具備**子類別定義之成員**，無法轉換成子類別物件。

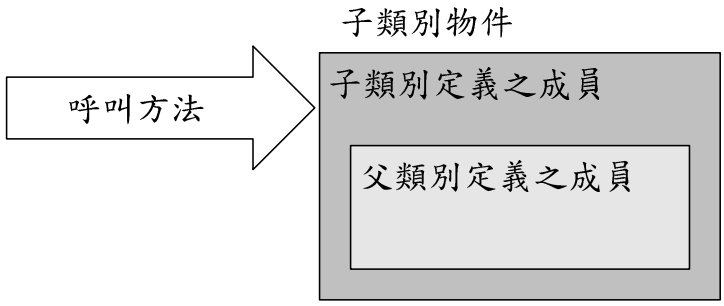


方法的覆蓋

- 子類別的重新定義物件方法稱為方法的覆蓋(Overriding)。(SCJP 33,52,70,109)
- 方法覆蓋時，下列各項都必須和父類別定義的方法相同：
  - 方法的回傳型別。
  - 方法名稱。
  - 形式參數列中的型別及順序。

方法的覆蓋

- 覆蓋的原理：先碰到子類別定義之成員



方法的覆蓋

- 方法覆蓋的主要用意是「**修改原有功能的實作**」。
- 使用 final 宣告的父類別之物件方法**不能被覆蓋**。
- **靜態方法**無法使用覆蓋(沒有覆蓋的特性)。(SCJP48)
- 覆蓋舊方法時，新定義的方法可以透過 super 參照呼叫舊方法。

方法的覆蓋

- 方法覆蓋要注意下列幾點：
  - 回傳型別必須相同。
  - 不能縮減方法的存取權限(public → private)
    - (SCJP92, 108, 142)
  - 丟出的例外型別必須相同
- 方法覆蓋時，沒有領域不同的區別，所有被覆蓋的方法一律打入冷宮。

8-5 方法覆蓋與多型

多型

- 不論參照的型別為何，呼叫方法時，呼叫最新的覆蓋方法。
- 多型指的是**使用相同的訊息呼叫，可以進行不同的功能操作**。
- **動態結合**(dynamic binding)：當某個物件方法，接收傳入的A類別之物件時，該方法無法得知傳入的是A類別的物件，或A之延伸類別的物件，直到執行時才會知道，也才能知道要呼叫的是哪個方法。

27

8-6 抽象類別

- 類別定義中，只要有一個或一個以上方法定義為**抽象方法**(abstract method)，則該類別為**抽象類別**(abstract class)。
- 抽象方法並沒有定義方法的主體(沒有實作)，因此無法利用該抽象類別建立物件。(SCJP 64)
- 抽象類別的用處是當作父類別，讓子類別繼承。
- 子類別必須將父類別中的抽象方法實作出來，才能建立物件。

28

8-6 抽象類別

抽象類別的定義語法：

```
修飾字 abstract class 類別名稱 {
        //屬性宣告
        修飾字 abstract 型別 方法名稱(參數列);
        //其它方法定義
}
```

某個類別的主體內，宣告了抽象方法，則該類別必須宣告為抽象類別。

抽象類別雖然不能用以建立物件，但是可以當作物件的**型別**，用來宣告參照變數。

抽象類別的子類別若不完全實作抽象方法，則依然是抽象類別。

29

8-6 抽象類別

使用抽象類別的用意，主要是「**制定固定的訊息接收管道，但不把焦點放在訊息的處理上**」。

抽象類別和一般類別一樣，都**可以定義建構子**，只是不能直接以抽象類別的建構子建立物件。

抽象方法的目的就是為了讓子類別實作，所以 abstract 不能同時和 final、static 或 private 一起使用。

30

8-7 內部類別與繼承

- 父類別包含內部類別時，子類別也可以繼承內部類別，如同繼承其它成員一樣。
- 內部類別可以有繼承關係，而且沒有特別限制。(SCJP 1, 59)

31

8-8 匿名類別

- **匿名類別**(Anonymous classes)是區域內部類別的一種，顧名思義匿名類別沒有名稱，所以也**不能定義建構子**。

```
new 類別名稱(參數) {
//類別主體
}
```

- **匿名類別**的目的只是要建立一個物件，物件建立之後就不會再使用該類別。
- **匿名類別**常當作覆蓋或實作抽象方法之用。

32

8-8 匿名類別

- 匿名類別的功能就是只建立一個物件，所以**類別主體的定義**和**建立物件**是一氣呵成。
- 繼承類別建立匿名類別：

```
new 父類別(參數列){
        // 類別主體
}
```

- 實作介面建立匿名類別：

```
new 介面(){
        // 類別主體
}
```

33