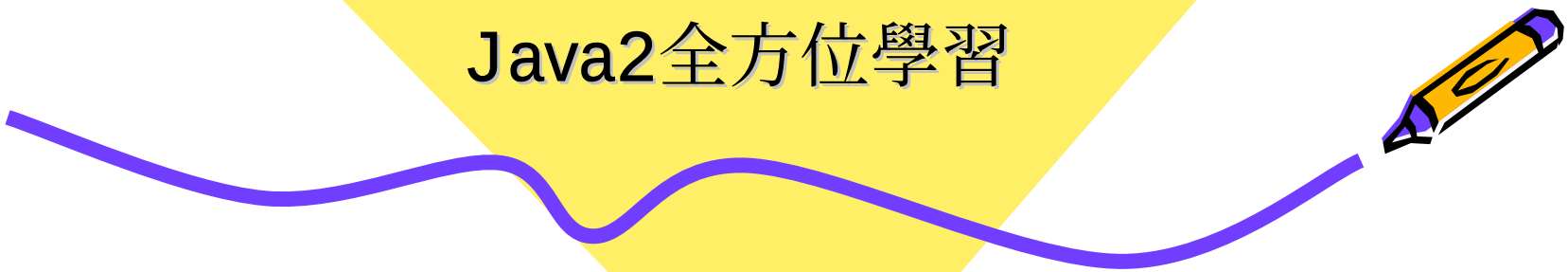
# Java物件導向程式設計

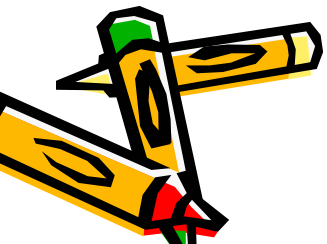
第七章

Java2全方位學習

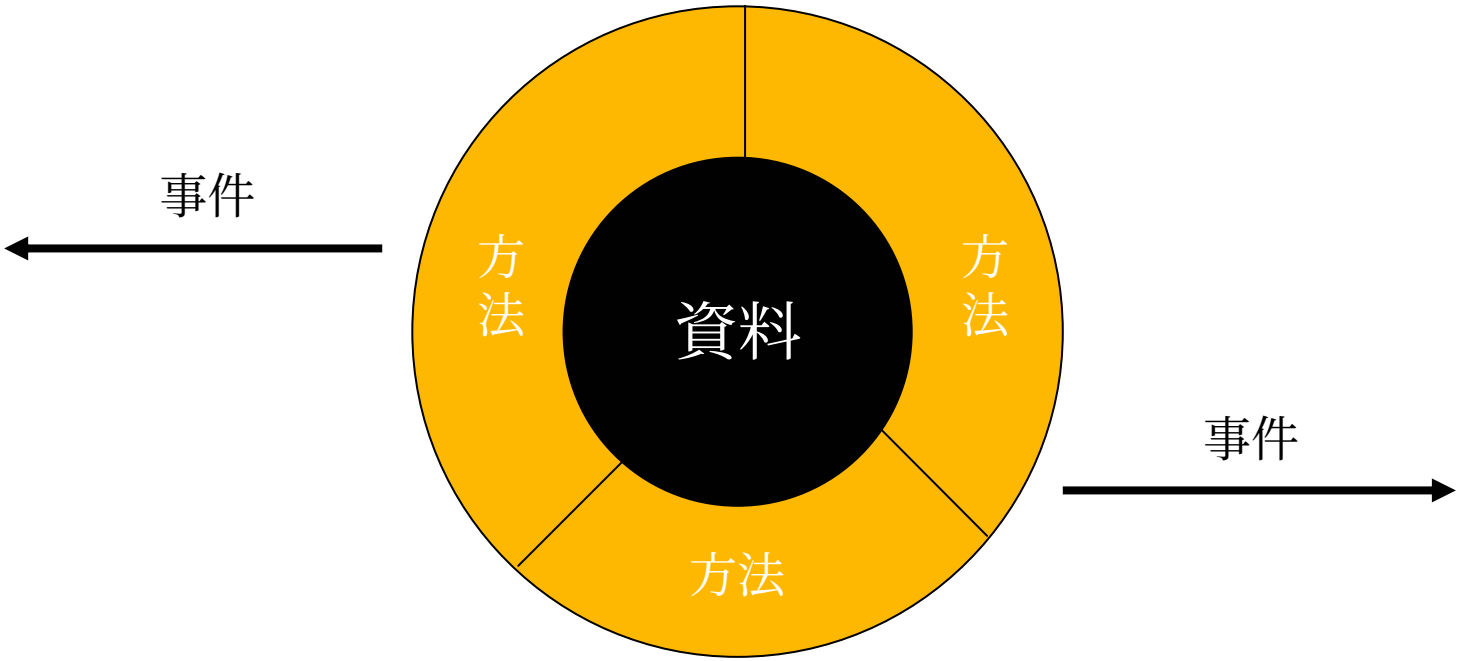


# 大綱

- 物件導向基本概念
- 繼承
- 封裝
- 多形
- 建構子使用
- Overload和Override



# 物件導向基本概念



# 類別與物件

類別

| 狗 |
| --- |
| +膚色 |
| +汪汪叫() |

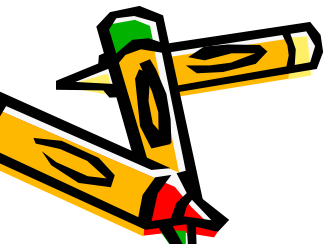
物件

物件

物件

# 類別與物件

- 類別是建構物件的藍圖(Blueprint)
- 物件是符合類別定義所產生的實體(instance)



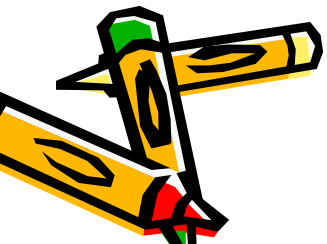
# Java類別與物件

- 類別

```
public class Dog
{
    private Color color;
    public void bark() {...}
    public void
      setColor(Color c)
    {
      color = c;
    }
}
```

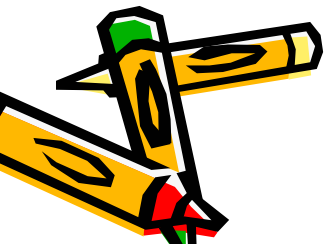
- 物件

```
Dog dog = new Dog();
```



# 成員

- 物件成員、類別成員
- 屬性(attribute)
  - 描術物件的特性
- 方法(method)
  - 操作物件

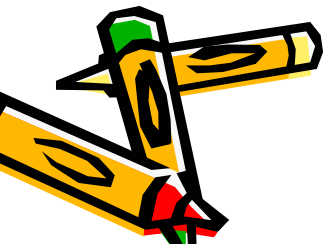


# 類別成員

- 伴隨類別存在
  - 類別被載入到記憶體時
- 宣告方法
  - static修飾子
- 使用方法
  - <類別名稱>.<成員名稱>
- 使用限制
  - 類別方法中無法存取物件成員

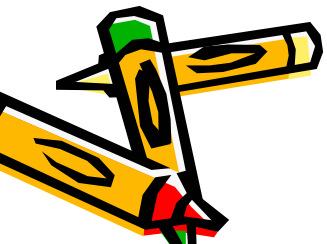
# 物件成員

- 伴隨物件存在
- 實體成員



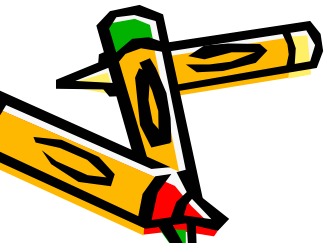
# 修飾子

- public
  - 每個類別都能使用
- private
  - 只有自己這個類別可以使用
  - 類別不能用



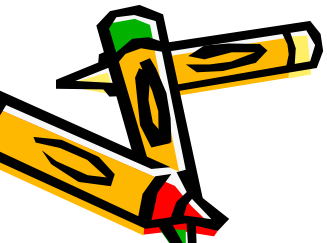
# 成員存取

- 存取方式
  - <類別/物件名稱>.<成員名稱>
  - g.color = Color.white;
  - g.setColor(Color.white);



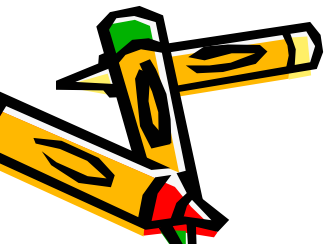
# 物件導向特性

- 繼承(inheritance)
- 多形(polymorphism)
- 封裝(encapsulation)

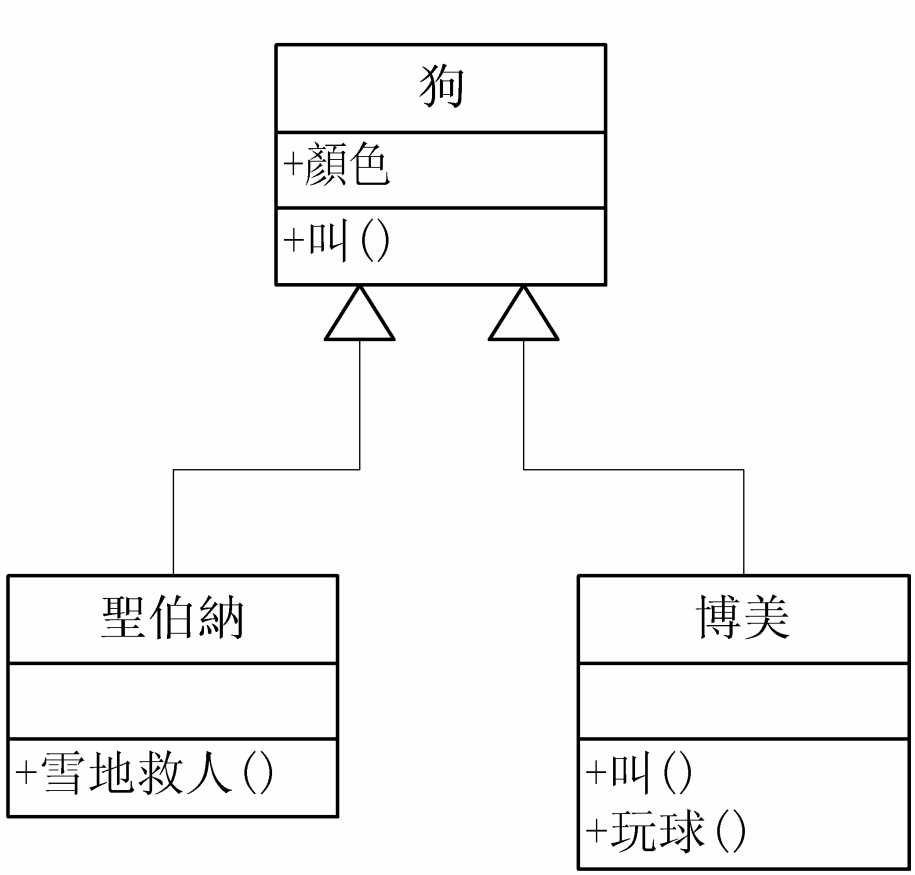


# 繼承

- 重覆使用(reuse)
- 延伸類別的資料與方法
- 擴展原有類別的功能
- 父、子類別

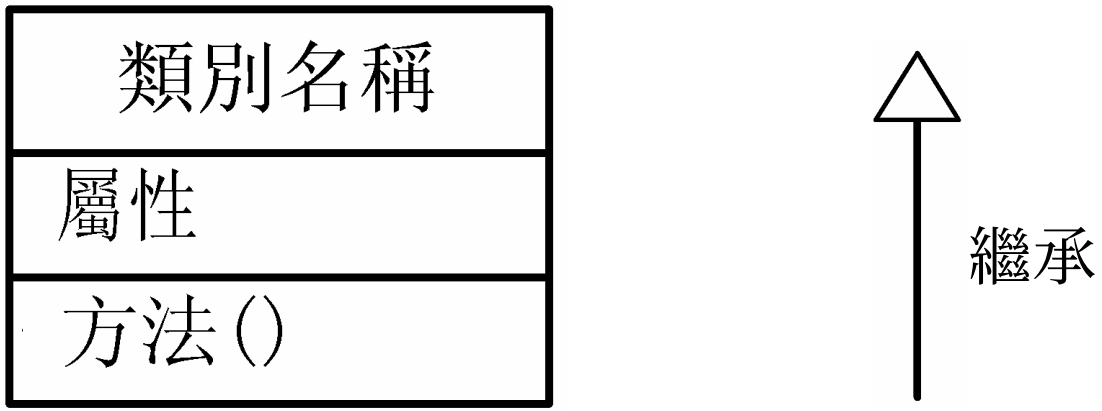


# 繼承



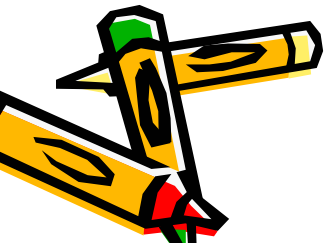
# UML

- Unified Modeling Language
  - http://www.omg.org/technology/uml/index.htm
- 物件導向程式的分析語設計



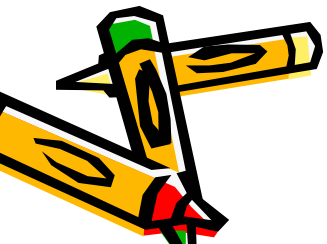
# Java繼承語法

- <子類別> extends <父類別>
- 單一繼承
- 例：
  - public class Dog extends Animal
- 所有的類別都繼承自java.lang.Object

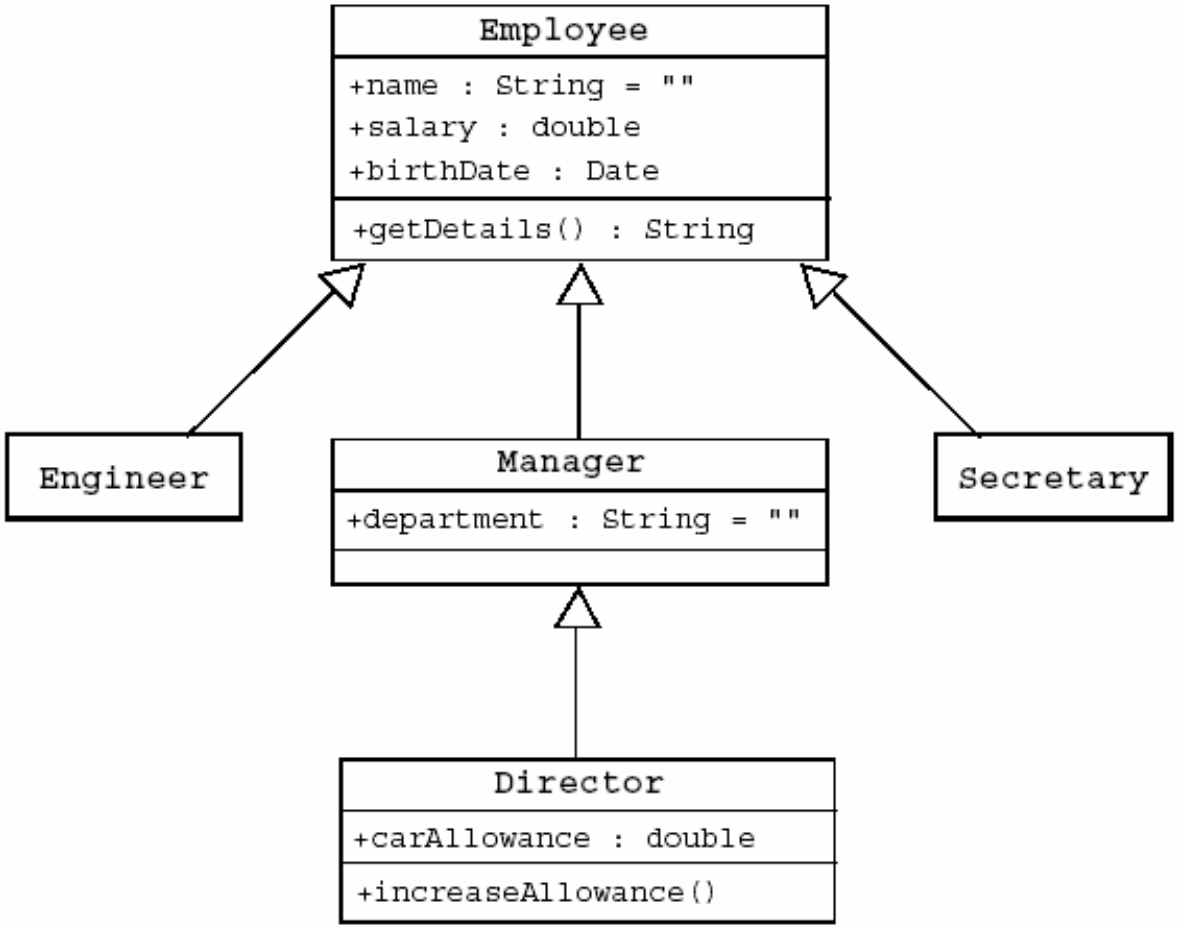


# 多形

- 一個類別有多種的表示方式
- 真正的實體物件只屬於某一個類別

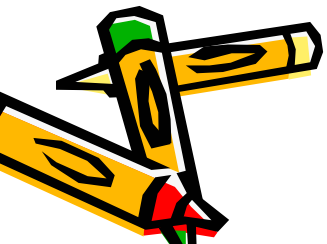


# 多形


Employee
+name : String = ""
+salary : double
+birthDate : Date
+getDetails() : String
Engineer
Manager
+department : String = ""
Secretary
Director
+carAllowance : double
+increaseAllowance()

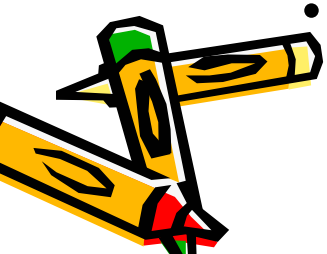
# 多形後成員的存取

- 遮閉問題
- 覆寫問題



# 多形轉形

- 物件產生
  - Employee e = new Manager();
  - Manager m = (Manager)e;
- 轉形錯誤
  - Compiler time錯誤
    - Engineer en = new Manager();
  - Runtime錯誤
    - Employee e = new Engineer();
    - Manager m = (Manager)e;



# 轉形規則

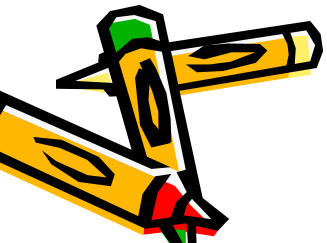
- 自動轉形
  - 子轉父
- 強迫轉形
  - 父轉子
- 錯誤轉形
  - 兄弟類別轉形
  - 直接由父類別物件轉成子類別

# 物件型別判斷

- instanceof運算子
  - boolean

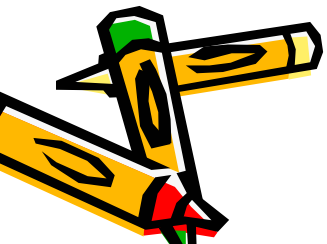
Employee e = new Manager();

(e instanceof Manager) => true



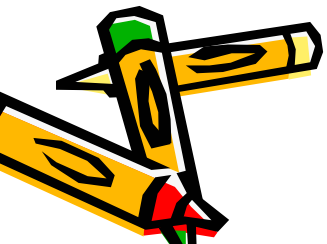
# 多形的好處

- 資料統一處理



# 資訊隱藏

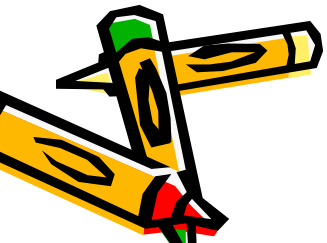
- Information hiding
- 封裝的基本條件
- 保護資料(屬性)
- 隱藏實作的方法



# 資訊隱藏

```
public class Dog
{
   public int weight;
}
```

```
Dog dog = new Dog();
dog.weight = 0; //不合理
```



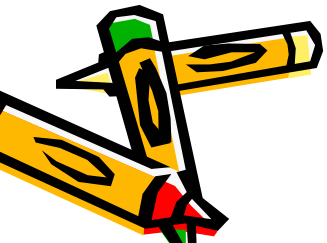
# 資訊隱藏

```
public class Dog
{
   private int weight;
   public void
   setWeight(int w)
   { <檢查>; weight = w; }
   public int getWeight()
   { return weight; }
}
```

```
Dog dog = new Dog();
dog.setWeight(10);
```

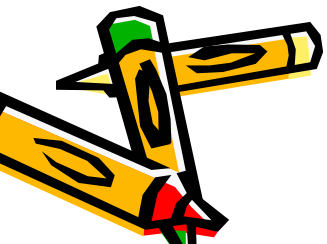
# 封裝

- 資訊隱藏
- 隱藏實作細節
- 統一成員使用方式
- 程式維護容易



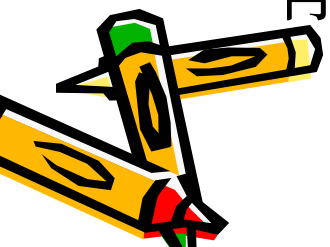
# 存取屬性方法命名規則

- 設定
  - setXXXX(<屬性資料>)
- 取得
  - getXXXX()
    - 傳回該屬性值
  - isXXXX()
    - boolean型態



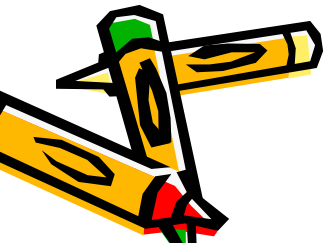
# 建構子

- 建構物件時呼叫
- 物件初始化的動作
- 不會被繼承
- 預設建構子
  - public <類別名稱>()
  - 沒有寫時，compiler會自動幫你加上
  - 有其它建構子時，則不會產生預設建構子



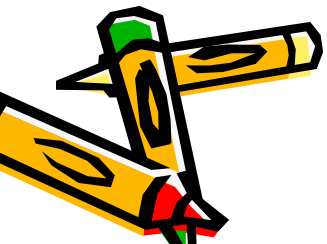
# 建構子間的呼叫

- super()
  - 呼叫父類別的建構子
- this()
  - 呼叫自已的建構子
- 物件初始化的設定



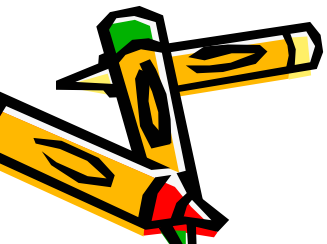
# super和this使用限制

- 一定要在建構子程式碼的第一行
- 兩個只能選一種
- 不能在其它程式區段中使用
- 如果建構子中沒有加上，compiler會自動加上super();



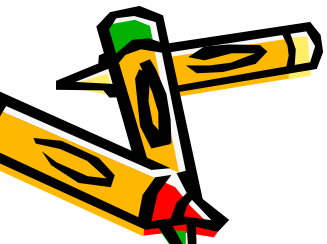
# Overload

- 方法的覆載
- 同一個方法名稱，有不同的使用方式
- 條件
  - 方法名稱一定要相同
  - 參數個數、型態一定要不同
  - 回傳值可不同



# 遮蔽

- Shadow
- 遮蔽父類別的屬性
- 存取方式
  - this
  - super



# Override

- 方法的覆寫
- 改寫父類別的方法
- 條件
  - 方法名稱一定要相同
  - 參數個數、型態一定要相同
  - 傳回值一定要相同
  - 修飾子只能愈來愈開放，不能愈來愈封閉
  - 類別方法不能被覆寫

# Q&A