

Module XIII

Object類別與包裝類別

1. Object類別
2. Wrapper類別
3. Auto boxing/unboxing

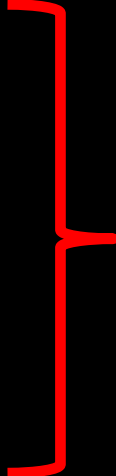
Object類別

- Java的所有類別，全部繼承自java.lang.Object類別
- 若一類別無繼承任何類別，則Java會自動用Object類別作為此類別的父類別
- Object類別常用的方法：
 - 為何Java要在(共同父類別)Object類別預備這些方法，其重要事項與觀念在上課時說明
 - boolean equals(Object obj)
 - String toString()
 - protected void finalize()
 - final void wait() notify() notifyAll() 【屬於執行緒的部份】

包裝類別 (Wrapper Class) (1/2)

- Java 每一個基本資料型態，都有一個相對應的 Wrapper 類別(包裝類別)

基本資料型態(Primitive Type)	Wrapper Class(包裝類別)
整數型態	
byte	java.lang.Byte
short	java.lang.Short
int	java.lang.Integer
long	java.lang.Long
浮點數型態	
float	java.lang.Float
double	java.lang.Double
其它型態	
boolean	java.lang.Boolean
char	java.lang.Character



java.lang.Number
的子類別

包裝類別(Wrapper Class) (2/2)

- boxing : 將基本型別，置入相對應的包裝類別中
 - **Integer** i = new **Integer**(1);
- unboxing : 從相對應的包裝類別取其值
 - 使用 **xxxValue()**方法
 - **int** x = i.**intValue()**;
- 字串轉成數字
 - 使用 static method **parseXxx**(String s)
 - **int** i = Integer.**parseInt**("1");
- 字串轉成包裝類別
 - 使用 static method **valueOf**(String s)
 - **Integer** i = Integer.**valueOf**("1");
- 比較兩個物件是否相等
 - 使用 boolean **equals**(Object obj)

記得其它如
float, double...等使
用均以此類推

自動裝箱/拆箱 (Autoboxing/Unboxing)

- Autoboxing/Unboxing

- 如果我們想將像是 `int` 的基本資料型別放到 `Collection` 中的話要怎麼辦呢？

- **Autoboxing(自動裝箱)**：

基本資料型別自動轉為包裝型態(Wrapper Types)，如`int`轉`Integer`

- **Unboxing(自動拆箱)**：包裝型態自動轉為基本資料型別，例如`Integer`轉`int`

```
33 public class Autoboxing1 {
34
35     public static void main(String[] args) {
36         Integer i1 = 1; //boxing
37         int i2 = i1;     //unboxing
38
39         int sum1 = i1 + i2;
40         Integer sum2 = i1 + i2;
41         System.out.println(sum1);
42         System.out.println(sum2);
43     }
44 }
```

範例：TestAutoboxing.java

章節整理

- Object類別為Java所有類別的父類別
- 所有類別均可使用Object類別內的方法
- 藉由Wrapper類別，我們可以讓基本型別做到更多複雜的計算與運用
- 8個Wrapper類別裡，4個整數型態與2兩個浮點數型態的共同父類別為java.lang.Number
- 把基本型別包成對應物件叫Boxing，而將物件還原成對應的基本型別叫Unboxing
- 注意資料型別的一致性