Module VII
物件導向概論

1. 何為物件(Object), 何為類別(Class)
2. 物件參考變數
3. 物件導向程式語言(Object-Oriented Programming, OOP)

物件與類別 (1/4)

- 五字箴言：**所見即物件**
- 只要是物件就一定有此兩項：

  - 屬性 (attribute) 或稱為特徵 (Characteristics)

  - 行為 (behavior) 或稱為操作 (Operation)
- Java透過類別(class)實現物件的概念，讓程式設計師能更具體化與直覺的方式進行資料處理
- 類別組成成員：

  - 資料成員 (Data Member → 變數 Variable)

  - 方法成員 (Method Member → 方法 Method)
- **類別是物件的資料型態(type)，一個物件是由某類別產生的一個實體(instance)**

物件與類別 (2/4)

- 一隻筆為物件的概念：

屬性 (Attributes) 有顏色, 有廠牌, 有價格

行為 (Behavior) 能寫, 能摔, 能丟…

物件與類別 (3/4)

- 設計類別並產生物件：

Pen
brand price size color
write() showInformation()

筆類別

筆物件

物件與類別 (4/4)

- 一個筆類別：

  產生兩個筆的物件實體(instance)

  Attributes: brand, price, color…

  Method: write, resupply…

brand: 萬寶龍
price: 14,800
color: 黑色

brand: S.K.B
price: 10
color: 藍色

物件參考變數 (1/6)

- 物件參考變數

```
1 public class PenTest {
2
3     public static void main(String[] args) {
4         Pen p = new Pen();
5         p.setBrand("Mont Blanc");
6         p.setPrice(14800);
```

- 宣告
  - · <類別名稱><變數名稱>
  - · 如：Pen myPen;
- 實體化物件
  - · 欲產生該物件真正的記憶體空間，必須以 **new** 關鍵字建立
  - · 如：new Pen();
- 初始化物件
  - · 用 **=** (指定運算子)指派該物件至物件參考變數
  - · 如myPen = new Pen();
- 注意：物件參考變數(Object Reference Variables)是一個儲存物件在記憶體中**位址**的變數

不同的記憶體空間

```
Pen myPen1 = new Pen();
Pen myPen2 = new Pen();
```

相同的記憶體空間

```
Pen myPen1 = new Pen();
Pen myPen2 = myPen1;
```

物件參考變數 (2/6)

- 物件參考變數

  - 操作資料：最普遍的作法是利用「.」運算子來操作物件的值

    - 如：myPen.brand = “SKB”;
    - 如：yourPen.price = 12000.0;

```
1  public class PenTest {
2
3      public static void main(String[] args) {
4          Pen myPen = new Pen();
5          myPen.brand = "SKB";
6          myPen.price = 10.0;
7
8          Pen yourPen = new Pen();
9          yourPen.brand = "MontBlanc";
10         yourPen.price = 12000.0;
11
12         myPen.showInfo();
13         yourPen.showInfo();
14     }
15 }
```

範例：Pen.java + PenTest.java

物件參考變數 (3/6)

- 在記憶體中儲存物件參考變數
  - **物件參考變數**儲存**記憶體位址** (memory address)
  - **基本資料型別變數**儲存**值** (value)

- 參考下面3頁的說明與範例：
  - Pass by value　　：傳遞基本型別時
  - Pass by reference　：傳遞物件或陣列時

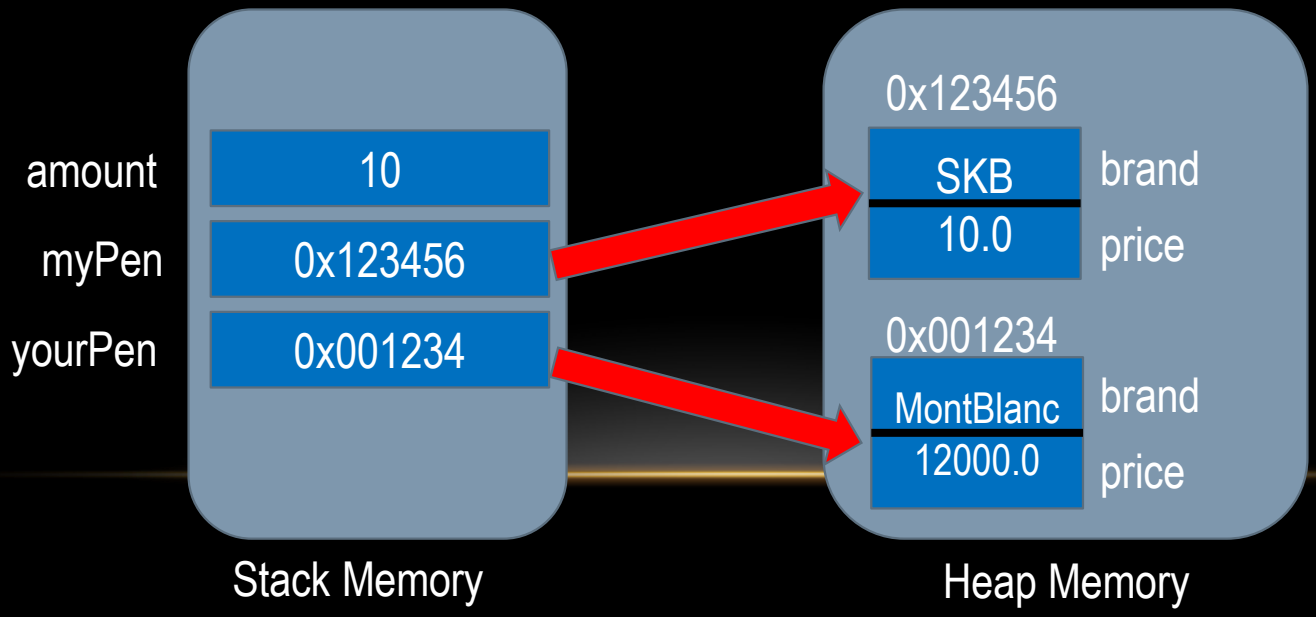
- 注意：

  Java的Pass by value, Pass by reference或是只有Pass by value，在定義上一直有所爭議，將於上課時說明

物件參考變數 (4/6)

- 將物件參考變數儲存在記憶體

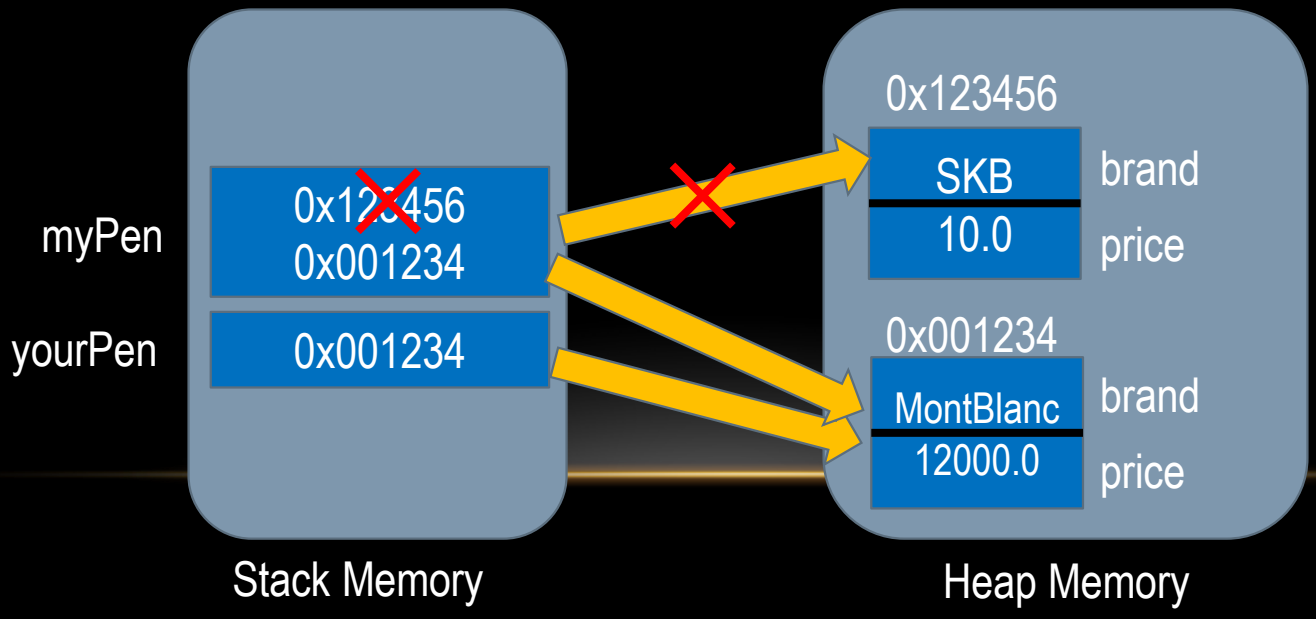
```
public static void main (String[] args) {
        int amount;
        amount = 10;
        Pen myPen = new Pen();
        Pen yourPen = new Pen();
}
```



物件參考變數 (5/6)

- 將物件參考變數指定給另一個物件

```
Pen myPen = new Pen();
Pen yourPen = new Pen();
myPen = yourPen;
```



物件參考變數 (6/6)

- 傳值 Pass by value
  - 如果傳入的參數為**基本資料型別**時，在方法內對參數的改變將不影響原來方法外變數的值
- 傳址 Pass by reference
  - 如果傳入的參數為**物件參考變數或陣列**時，無法更動reference對象，但可更動reference對象原變數的值，也可使用reference對象的方法與變數

```
1  public class PassArgTest {
2
3      static void passValue(double value) {
4          value = 20.0;
5      }
6
7      static void passReference(Pen reference) {
8          reference.price = 20.0;
9      }
10
11     public static void main(String[] args) {
12         double price = 10.0;
13         passValue(price);
14         System.out.println(price);          // 10
15
16         Pen myPen = new Pen();
17         myPen.price = 10.0;
18         passReference(myPen);
19         System.out.println(myPen.price);    // 20
20     }
21 }
```

範例：PassArgTest.java

物件導向程式語言 (OOP)

- 物件導向程式語言必定有以下三種特性：
  - **封裝** (Encapsulation)
    - 依類別成員存取權限分為private, default, protected與public
  - **繼承** (Inheritance)
    - 子類別可繼承父類別的成員，並可以修改或是新增自有成員
  - **多型** (Polymorphism)
    - 父類別指向子類別物件，並對應到子類別適用的方法
- OOP使用訊息傳遞(Message Passing)機制，透過物件接受訊息、處理訊息、傳送訊息來實現功能

整理與探討

- Java透過類別(class)來定義屬性與方法，並用該類別產生的物件實體(object instance)進行功能上的實現與架構設計
- 封裝、繼承與多型為物件導向程式語言的三大特性，我們必須藉由類別來達成以上三種概念，由此可知類別對Java的重要性有多大，我們可以說這是以類別為基礎(class-based)的程式設計
- 封裝(Encapsulation)讓Java實現隱藏資料與提昇資料存取安全性
- 繼承(Inheritance)讓父類別所定義的成員可以給子類別使用，子類別更能自行修改父類別定義或是進行擴充，以表現出自身的特質
- 多型(Polymorphism)讓程式設計師可用同樣方式去引用不同類別的物件，這對我們來說可以易於使用，並進一步到專案維護與功能擴充