國立屏東大學 資訊工程學系 物件導向程式設計

12. Class Definitions and Object Instantiation

- 類別定義
- 物件實體化
- 建構函式/建構子

12.1 類別定義

類別的定義必須包含類別名稱、屬性與行為，使用Java語言的術語，屬性被稱為field□行為被稱為method□

在Java語言中，類別定義語法如下：

```
class className
{
   //field declarations
   [DataType variableName[=value]?[,variableName[=value]?]*;]*

   //method declarations
   [returnType methodName(parameters)
   {
      // method implementations
      statements
   }]*
}
```

我們以一個簡單的例子示範類別的定義：

```
class Person
{
   // field declarations
   String firstname;
   String lastname;

   // method declarations
   void showInfo()
   {
      System.out.println("Name: " + firstname + " " + lastname );
   }

}
```

上面這個程式必須命名為Person.java

12.2 物件的實體化Object Instantiation

下面則是使用類別來產生物件實體的例子：

```
class test
{
   //建構一個物件導向世界, 產生一個Person類別的物件
   public static void main(String[] args)
   {
      Person amy;          //amy是一個reference應該要指向Person類別的物件實體
      amy = new Person(); //產生實體, 並將實體所在的位置儲存在amy
      amy.showInfo();      //透過amy這個reference要求其執行showInfo() method
   }
}
```

執行結果：

```
Name: null null
```

因為沒有設定這個Person類別的物件的firstname與lastname所以再修改如下：

```
class test
{
   //建構一個物件導向世界, 產生一個Person類別的物件
   public static void main(String[] args)
   {
      Person amy = new Person(); //amy是一個reference指向Person類別的物件實體
      amy.firstname = "Amy";      //透過amy這個reference存取其firstname field
      amy.lastname = "Wang";      //透過amy這個reference存取其lastname field
      amy.showInfo();
   }
}
```

執行結果：

```
Name: Amy Wang
```

其實我們應該再增加一些field到Person類別, 例如phone number, address, email等, 讓它更完整, 不過為了保持範例的簡單性, 所以暫時不這樣做。

12.3 建構函式/建構子

有沒有注意到這行Person amy = new Person();其中的new Person()即為用來產生一個Person類別的實體。其實還不只是這樣，它還會幫你呼叫一個名為Person()的method

> 可是我沒有定義這個method啊
>
> > Java語言預設會幫你在類別定義內產生一個與類別同名的method稱為建構子(constructor)
> > 例如Person類別會自動包含以下的程式碼

```
Person() { }
```

> > 它是一個特殊的method沒有傳回型態的宣告，且其內容為空白
> > 在使用new Person()時，這個method會被自動呼叫
>
> 內容空白，那不就沒用處？
>
> > 看你想要做什麼？你可以自己定義啊！例如：

```
Person()
{
   firstname="undefined";
   lastname ="undefined";
}
```

> > 你還可以提供不同版本的constructor例如:

```
Person(String f, String l)
{
   firstname=f;
   lastname =l;
}
```

> > 這樣一來，你可以在實體化的同時順便把firstname與lastname設定好。例如以下的宣告:

```
Person amy = new Person("Amy", "Wang");
```

> > 不過要特別注意，如果定義了自己的constructor那麼預設的constructor就不會再產生，所以下面的程式碼是錯誤的：

```
class Person
```

```
{
   // field declarations
   String firstname;
   String lastname;

   // constructors
   Person(String f, String l)
   {
      firstname = f;
      lastname =l;
   }

   // method declarations
   void showInfo()
   {
      System.out.println("Name: " + firstname + " " + lastname );
   }

}

class test
{
   //建構一個物件導向世界, 產生一個Person類別的物件
   public static void main(String[] args)
   {
      Person amy = new Person(); //已定義Person(String, String)constructor預設的Person()就不存在
      amy.firstname = "Amy";
      amy.lastname = "Wang";
      amy.showInfo();
   }
}
```

> 瞭解了！所以如果有自定constructor就要小心實體化時呼叫的版本是否正確！
>
> > 您也可多提供一個預設的版本！例如：

```
Person()
{
   firstname="unknown";
   lastname="unkonwn";
}
```

> 我還有一個問題Java是不是和C++一樣有constructor與destructor那Java的destructor該怎麼寫？
>
> > Java採用記憶體自動管理機制garbage collection它會自己監看是否有不再使用的物件，它自己會回收！所以沒有destructor!

我們將類別定義語法加上constructor的定義：

```
class className
{
   //field declarations
   [DataType variableName[=value]?[,variableName[=value]?]*;]*

   //constructors
   [className(parameters)
   {
   }]*

   //method declarations
   [returnType methodName(parameters)
   {
      // method implementations
      statements
   }]*
}
```