介面與多型與列舉

學習目標

- 使用介面定義行為
- 瞭解介面的多型操作
- 利用介面列舉常數
- 利用ENUM列舉常數

介面

介面變數

- 定義了一個介面，相當於定義了一種新的資料類型，就可以用這種新的資料類型定義變數。用介面定義的變數稱為介面變數。

介面變數的定義形式：

介面名 介面變數表列;

介面

1.介面的定義

介面的定義形式：

```
interface 介面名
{
//符號變數定義
//方法宣告
}
```

介面（Interface）

定義介面

- 介面(interface)與抽象類別(abstract class)的概念非常類似，而介面比抽象類別更抽象。
- 介面僅具有方法與變數的宣告，不包含任何方法的定義區塊。
- 介面必須藉由其他類別使用implements關鍵字來實作，所謂實作就是為介面中宣告的方法加上定義區塊。
- 可以將implements 「實作」當作「繼承」來看。
- 類別可以實作(implements)許多介面，而一個類別只能繼承一個類別而已，除此之外，介面也可以繼承(extends)一個以上的介面。

介面（Interface）

介面（interface）與抽象類別（abstract class）

- 介面裡不能實作任何 methods；abstract class 可以實作方法。
- 一個類別可以同時實作多個介面；但一個類別只能有一個 superclass（Java 不支援多重繼承）。
- 介面不屬於類別階層架構。不相干的類別也可以實作相同介面。
- 因為多重繼承會有問題，但實際上又有多重繼承的需要，所以 interface作為折衷的替代方案。

介面（Interface）

- 注意事項
  - 如果某介面繼承另一個介面，則 subclass 能從 superclass 繼承到的只有常數與函式宣告而已。
  - 介面開始使用後若有更改，已經使用此介面的程式會因無法實作新功能，而不能執行。
  - 如果真要更改需宣告新介面，就可以決定要實作一個介面或兩個介面都實作。

介面的繼承

1.介面的定義

介面還可以有多個父介面，介面的完整定義如下形式：

```
interface 介面名 extends 父介面表列
{
//符號常量定義
//方法宣告
}
```

介面的繼承

- 定義
  - 一段只有常數與函式宣告，但沒有函式實作的程式碼。
- 宣告
  - [public] interface 介面名稱 [extends 父介面名稱]
- 作用
  - 讓某個功能，不論由誰實作，都能夠有相同的函式名稱，傳入值，傳出值，與存取範圍。

介面的實現

介面的使用

介面必須通過類別才能使用。如果一個類別定義了介面中的所有方法，稱這個類別實現了介面。實現介面的形式：

```
class 類別名 implements 介面名稱[, 其他介面名稱, 其他介面名稱..., ...] {
//類體
}
```

可以用介面變數表示實現它的類別（可以看作子類別）的物件，介面變數可以稱作是物件的上轉型物件。

介面的實現

implements關鍵字

使用 implements 關鍵字可以同時繼承多個介面（介面跟介面之間採用逗號分隔）。

```
public interface A {
    public void eat();
    public void sleep();
 }

public interface B {
   public void show();
}

public class C  implements A,B { }
```

介面的實現

```
package ch7_1;
public interface Animal {
   public void eat();
   public void run();
}
```

```
package ch7_1;
public class Dog implements Animal {
    public void eat() {
        System.out.println("Dog eats");
    }

    public void run() {
        System.out.println("Dog runs");
    }

    public int noOfLegs() {
        return 0;
    }
}
```

```
package ch7_1;
public class ch7_2 {
  public static void main(String[] args) {
      Dog d = new Dog();
      d.eat();
      d.run();
  }
}
```

結果:

```
Dog eats
Dog runs
```