類別與類別成員的封裝與修飾字

封裝	修飾字	類別

public
無
	final (禁止繼承)
	abstract (抽象的類別)
		class A001 {
		
		int x;
		
		void xyz(){
		
		}
		
		}

public protected 無 private	final (常數) static (類別等級)	
public protected 無 private	final (禁止覆蓋) static (類別等級) abstract (抽象的方法)	

成員的封裝與修飾字

1. 成員的封裝

	同 package		不同 package	
	new	繼承	new	繼承
public	V	V	V	V
protected	V	V	X	V
無	V	V	X	X
private	X	X	X	X

2. 成員的修飾字

<1>. 成員變數

final	常數
static	類別變數，當它所屬的類別一被載入，就配置它的實體空間，且所有同一個類別的物件，都會共用同一份 “static 成員變數”，而使用 static 成員變數時以 ‘類別.屬性” 或 “物件.屬性” 都是使用同一個 “已配置的實體空間”

<2>. 成員函數

final	不允許子類別覆蓋它(Override)
static	類別方法，當它所屬的類別一被載入，就配置它的實體空間，且所有同一個類別的物件，都會共用同一份 “static 方法” 實體，而使用 static 成員變數時以 “類別.方法” 或 “物件.方法” 都是使用同一個 “已配置的實體空間”
abstract	此成員函數是一個”abstract 方法” ，它只能有方法的 “宣告部份” ，而不能有實作部份，而且任何擁有 abstract 方法的類別，都必須宣告為”abstract 類別” 抽像類別不能產生物件

new 的封裝可視性

```java
package source;

import source.pack2.New3;

public class New1 {

    public void 使用New2() { //在同 package
        New2 a = new New2();
        a.x = 10;  // public ok
        a.y = 20;  // protected ok
        a.s1();  // 無 ok
        // a.s2();  //private error
    }

    public void 使用New3() { //在不同 package
        New3 a = new New3();
        a.x = 10;  // public ok
        // a.y=20;  // protected error
        // a.s1();  // 無 error
        // a.s2();  //private error
    }
}
```

```java
package source;

public class New2 {

    public int x;  //public
    protected int y;  //protected

    void s1() {  //無
    }
    private void s2() {  // private
    }
}
```

```java
package source.pack2;

public class New3 {

    public int x;  //public
    protected int y;  //protected

    void s1() {  //無
    }
    private void s2() {  // private
    }
}
```

繼承的封裝可視性

```java
package source;

public class Sub1 extends Super1 {

    public void 使用繼承而來的屬性與方法() {
        x = 10;  //public ok
        y = 20;  //protected ok
        s1();   //無 ok
        //  s2();   //private error
    }
}
```

```java
package source;

public class Super1 {

    public int x;
    protected int y;

    void s1() {
    }
    private void s2() {
    }
}
```

```java
package source;

import source.pack2.Super2;

public class Sub2 extends Super2 {

    public void 使用繼承而來的屬性與方法() {
        x = 10;  //public ok
        y = 20;  //protected ok
        //    s1();   //無 error
        //   s2();   //private error
    }
}
```

```java
package source.pack2;

public class Super2 {

    public int x;
    protected int y;

    void s1() {
    }

    private void s2() {
    }
}
```

父類別

```java
package source.pack1;

public class A001 {

    public int a;
    protected int b;
    int c;
    private int d;
}
```

```java
package source.pack1;

public class A001 {

    public int a;
    protected int b;
    int c;
    private int d;

    public void setd(int x){
        d=x;
    }
    public int getd(){
        return d;
    }
}
```

子類別

```java
package source;

import source.pack1.A001;

public class B001 extends A001 {
    /* 不用寫
     public int a;
     private int b;
     X int c;
     X private int d;
     */
}
```

```java
package source;

import source.pack1.A001;

public class B001 extends A001 {
    /* 不用寫
     public int a;
     private int b;
     X int c;
     X private int d;
     */
    //可透過公開的方法來設定
    public void setb(int x) {
        b = x;
    }

    public int getb() {
        return b;
    }
}
```

Test06 類別

```java
public static void 使用B001_1() {
    B001 p = new B001();
    p.a = 10;
    System.out.println("p.a=" + p.a);
    //不能使用 b
    // 因為 b 是 父類別特別保護給子類別用的
    // p.b = 20;
    //System.out.println("b=" + p.b);
}
```

```java
public static void 使用B001_2() {
    B001 p = new B001();
    p.a = 10;
    System.out.println("p.a=" + p.a);
    //p.b = 20;
    //System.out.println("b=" + p.b);
    p.setb(20); //透過公開的方法設定
    System.out.println("b=" + p.getb()); //透過公開的方法取得
}
```

設定 set ，與取得 get

```java
public class Animal {

    public int legs;

    public void eat() {
        System.out.println("Eating");
    }

    public void move() {
        System.out.println("Moving");
    }
}
```

```java
public static void 使用Animal() {
    Animal animal1 = new Animal();
    animal1.legs = 4;
    System.out.println("animal1 has " + animal1.legs + " legs.");
    animal1.eat();
    animal1.move();
}
```

```
animal1 has 4 legs.
Eating
Moving
```

```java
public class Animalx {

    private int legs;

    public void eat() {
        System.out.println("Eating");
    }

    public void move() {
        System.out.println("Moving");
    }
//把 legs的成員變數改成 private 再多加這兩段副程式

    public void setlegs(int x) {
        if (x <= 4) {
            legs = x;
        } else {
            System.out.println("腿設定值錯誤");
        }
    }

    public int getlegs() {
        return legs;
    }
}
```

```java
public static void 使用Animalx() {
    Animalx animal1 = new Animalx();
    animal1.legs=4;
    animal1.setlegs(4);
    System.out.println("animal1 has "+ animal1.legs+" legs.");
    System.out.println("animal1 has " + animal1.getlegs() + " legs.");
    animal1.eat();
    animal1.move();
}
```

類別間的存取權限

```java
package source.pack1;

public class P1 {

    private char blood;
    protected String DNA = "ABCDABCD";
    public String name;

    public char getBlood() {
        return blood;
    }
    public void setBlood(char b) {
        blood = b;
    }
}
```

```java
package source.pack2;

import source.pack1.P1;

public class ET extends P1 {
    /*  ET 從 person 繼承而來的成員
    private char blood;
    protected String DNA = "ABCDABCD";
    public String name;
    public char getBlood();
    public void setBlood(char b);
     */
    public void setDNA(String d) {
        DNA = d;
    }
    public String getDNA() {
        return DNA;
    }
}
```

```java
public static void 繼承存取權() {
    P1 fish = new P1();
    fish.blood = 'A'; //錯誤 , P1的 private 成員只能在 P1 裡存取
    fish.setBlood('A');
    System.out.println("fish的血型=" + fish.getBlood());
    fish.DNA = "AAAAABBBB"; //錯誤 , 不可在此存取 protected 的成員
    fish.name = "colorfish";
    System.out.println("fish的name=" + fish.name);

    ET Axel = new ET(); //只能透過子類別存取 DNA
    Axel.setDNA("oiujhgtffd");
    System.out.println("Axel 的 DNA=" + Axel.getDNA());
}
```

```
fish的血型=A
fish的name=colorfish
Axel 的 DNA=oiujhgtffd
```

方法覆載➔不小，同，同，同，子集

方法多載➔參數的 型態，個數，順序

副程式名稱(同)➔參數串 (同) ➔覆蓋➔封裝 (不小)，回傳型態 (同)

➔參數串 (不同)➔多載

覆載 (覆蓋) overriding	多載 (overloading)

protected void abc(int x) { 父類別的方法
 	
}
public void abc(int x) { //子類別的方法
 	
}
	protected void abc(int x) { 父類別的方法
	
	}
	void abc( ) { //子類別的方法
	
	}

```java
public class SuperX {

    protected void abc(int x) {//覆蓋
    }
}

class SubX extends SuperX {

    public void abc(int x) { //覆蓋
    }

    private int abc(int x) { //同類別內 不允許參數重覆
        return 1;
    }

    void abc() { //多載
    }

    static void abc(char x) { //多載
    }

    public int abc(int x, float y) { //多載
        return 1;
    }

    private void abc(double x) { //多載
    }

    protected void abc(float x, int y) { //多載
    }
}
```

final 加在方法➔子類別禁止改寫父類別的方法

```
public class Final1 {

    protected final void 習慣() {
        System.out.println("抽煙");
    }

    public void xyz() {
          System.out.println("xyz");
    }
}

class Final2 extends Final1 {

// 父類別方法 加了 final 修飾字 , 子類別不能覆蓋
    public void 習慣() {
        System.out.println("吃檳榔");
    }
    public void xyz() {
         System.out.println("abc");
    }

    public void m1() {
         System.out.println("m1");
    }
}
```

final 加在類別名稱➔禁止繼承

```
final class Final3 {

    protected void 習慣() {
        System.out.println("抽煙");
    }

    public void xyz() {
        System.out.println("xyz");
    }
}
//父類別加了 final 整個不能繼承
 class Final4 extends Final3{

}
```

abstract 修飾字

1. 一個特殊的機制，能讓 繼承 或 實作 的 子類別 一定要去 覆寫 或 實作 父類別 或 父介面的方法
2. 抽象類別 與 介面不能 產生物件 (被 new)

```java
public abstract class Abstract1 {

    public abstract void abc();

    public abstract void xyz();

    public void m1() {
    }
}

class Abstract2 extends Abstract1 {

    @Override
    public void abc() {
    }

    @Override
    public void xyz() {
    }

    @Override
    public void m1() {
    }
}

abstract class Abstract3 extends Abstract1 {

    @Override
    public void xyz() {
    }

    @Override
    public void m1() {
    }
}

class Abstract4 extends Abstract3 { //孫子

    @Override
    public void abc() {
    }
}
```

```java
interface Abstract5 {

    void abc();

    void xyz();
}

class Abstract6 implements Abstract5 {

    @Override
    public void abc() {
    }

    @Override
    public void xyz() {
    }
}

abstract class Abstract7 implements Abstract5 {

    @Override
    public void xyz() {
    }
}

class Abstract8 extends Abstract7 { //孫子

    @Override
    public void abc() {
    }
}
```

```java
public static void abstract修飾字() {
    //錯誤，abstract 類別 或介面 不能 new 產生實體
    Abstract1 p1 = new Abstract1();
    Abstract1 p2 = new Abstract2();

    Abstract5 q1 = new Abstract5();
    Abstract5 q2 = new Abstract6();
}
```

static 修飾字

```java
public class Sale {

    public static final double MODE1 = 0.65;
    public static final double MODE2 = 0.78;
    public static final double MODE3 = 0.9;

    public static int count(int price, double mode) {

        return (int) (price * mode);
    }
}
```

```java
public static void static修飾字() {
    int price = 2100;

    int cost1 = Sale.count(price, Sale.MODE1);
    System.out.println("客人1 購買的價格= " + cost1);
    int cost2 = Sale.count(price, Sale.MODE2);
    System.out.println("客人2 購買的價格= " + cost2);
}
```

```
客人1 購買的價格= 1365
客人2 購買的價格= 1638
```

```java
public class Account1 {

    private String 帳號;    //  存放帳號
    private int 存款餘額;   //  存簿餘額

    public void 存款(int 存款) { // 存款

        存款餘額 += 存款;
        秀金額();
    }

    public void 提款(int 提款) { // 提款

        存款餘額 -= 提款;
        秀金額();
    }

    public void 設定資料(String id, int money) {// 設定資料

        帳號 = id;
        存款餘額 = money;
        秀金額();
    }
    public void 秀金額() { //  顯示餘額

        System.out.println(" 帳號 : " + 帳號);
        System.out.println(" 餘額 : " + 存款餘額);
    }
}
```

```java
public static void 使用Account1() {

    String id = "111-22-3333";
    int money = 10000;

    Account1 MyAccount = new Account1();
    MyAccount.設定資料(id, money);

    System.out.println(" == 存入 5000 元後");
    MyAccount.存款(5000);
    System.out.println(" == 領出 20000 元後");
    MyAccount.提款(20000);
}
```

```
帳號 : 111-22-3333
餘額 : 10000
== 存入 5000 元後
帳號 : 111-22-3333
餘額 : 15000
== 領出 2000 元後
帳號 : 111-22-3333
餘額 : -5000
```

```java
public class Account2 {

    private String 帳號;    //  存放帳號
    private int 存款餘額;    //  存簿餘額

    public void 存款(int 存款) { // 存款
        if (檢核存款(存款)) {
            存款餘額 += 存款;
            秀金額();
        } else {
            System.out.println("存款失敗--不能超過5萬元");
        }
    }

    public void 提款(int 提款) { // 提款

        if (檢核提款(提款)) {
            存款餘額 -= 提款;
            秀金額();
        } else {
            System.out.println("提款失敗--餘額不足");
        }
    }

    public void 轉帳(int 轉帳) { // 轉帳

        if (檢核提款(轉帳)) {
            存款餘額 -= 轉帳;
            秀金額();
        } else {
            System.out.println("轉帳失敗--餘額不足");
        }
    }

    private boolean 檢核存款(int 存款) {
        if (存款 > 50000) {
            return false;
        }
        return true;
    }

    private boolean 檢核提款(int 提款) {
        if (提款 > 存款餘額) {
            return false;
        }
        return true;
    }

    public void 設定資料(String id, int money) {
        帳號 = id;
        存款餘額 = money;
        秀金額();
    }

    public void 秀金額() {
        System.out.println(" 帳號 : " + 帳號);
        System.out.println(" 餘額 : " + 存款餘額);
    }
}
```

```java
public static void 使用Account2() {
    String id = "111-22-3333";
    int money = 10000;

    Account2 MyAccount = new Account2();
    MyAccount.設定資料(id, money);


    System.out.println(" == 存入 50000 元後");
    MyAccount.存款(50000);


    System.out.println(" == 領出 20000 元後");
    MyAccount.提款(20000);


    System.out.println(" == 轉帳 80000 元後");
    MyAccount.轉帳(80000);

}
```

```
 帳號 : 111-22-3333
 餘額 : 10000
 == 存入 50000 元後
 帳號 : 111-22-3333
 餘額 : 60000
 == 領出 20000 元後
 帳號 : 111-22-3333
 餘額 : 40000
 == 轉帳 80000 元後
轉帳失敗--餘額不足
```

作業

```
public class Account3 {


}
```

```
public class TestAccount {

    public static void main(String[] args) {

        String id = "111-22-3333";
        int money = 10000;

        Account3 MyAccount = new Account3();
        MyAccount.設定資料(id, money);

        if (!MyAccount.密碼驗證()) {
            System.out.println(" 非法使用,請離開 !!! ");
            return;
        }
        MyAccount.銀行作業();
    }
}
```

```
輸入密碼 : dsd
 密碼錯誤,請重新輸入1次
輸入密碼 : ssdf
 密碼錯誤,請重新輸入2次
輸入密碼 : sfgs
 密碼錯誤,請重新輸入3次
 非法使用,請離開 !!!
////////////////////////////////////////////////////////////

 帳號 : 111-22-3333
 餘額 : 10000
==========================================
輸入密碼 : 1234
 密碼錯誤,請重新輸入1次
輸入密碼 : sun2000
請問要 :  1.存錢 2.領錢 3.轉帳 4.其它, 離開 : 1
請輸入 存入  金額 : 2000
 存入 2000 元後
 帳號 : 111-22-3333
 餘額 : 12000
==========================================
請問要 :  1.存錢 2.領錢 3.轉帳 4.其它, 離開 : 1
請輸入 存入  金額 : 60000
 存入 60000 元後
存款失敗--不能超過5萬元
==========================================
請問要 :  1.存錢 2.領錢 3.轉帳 4.其它, 離開 : 2
請輸入 領出  金額 : 13000
 領出 13000 元後
提款失敗--餘額不足
==========================================
請問要 :  1.存錢 2.領錢 3.轉帳 4.其它, 離開 : 2
請輸入 領出  金額 : 5000
 領出 5000 元後
 帳號 : 111-22-3333
 餘額 : 7000
==========================================
請問要 :  1.存錢 2.領錢 3.轉帳 4.其它, 離開 : 3
請輸入 轉帳  金額 : 2000
 轉帳 2000 元後
 帳號 : 111-22-3333
 餘額 : 5000
```