Inner 巢狀類別的宣告與應用

1. 在一個類別之內，可再宣告 “類別” ，內部的類別稱為 “巢狀類別” ，外部的類別稱為 “外圍類別”
2. 讓程式設計邏輯上相關的類別結合在一起
3. Inner Class 可以直接存取外部類別的成員，外部類別也可直接存取內部類別的成員（private 成員也可以 ）
4. 依 Inner Class 宣告的位置，我們可以分成

   <1>.成員式（ 宣告在全域區 ）

   (1).成員類別等級

   (2).成員物件等級

   <2>.區域式（ 宣告在方法內 ）

   (1).區域巢狀類別

   (2).區域匿名類別
5. 當我們在設計圖型化元件的 “事件” 時，經常就會使用 “巢狀類別”
6. 原則➔內用外 (直接用)，外用內 (new，或用類別去接)

4 種巢狀類別的架構

```
public interface InnerFather {

    public void m1();
}
```

```
public class InnerStruct1 {

    public int x;
    private static int y;
    public static int z;

    public void s1() {

    }

    public static void s2() {

    }

//類別等級巢狀類別
    public static class Inner1 implements InnerFather {

        private int a;
        public static int b;

        public Inner1() {
            a = 123;
        }

        public void m1() {
            // x = 123
            y = 789;
        }

        public static void n1() {

        }
    }
```

```
//物件等級巢狀類別
public class Inner2 implements InnerFather {

    private int a;
    //public static int b;

    public Inner2() {
        a = 123;
    }

    public void m1() {
        x = 123;
        y = 789;
    }
    // public static void n1() {

    // }
}
```

```
public void s3() {
    //區域巢狀類別
    class Inner3 implements InnerFather {

        private int a;

        public Inner3() {
            a = 123;
        }

        public void m1() {
            x = 123;
            y = 789;
        }
    }
    Inner3 p = new Inner3();
    p.m1();
}

public void s4() {
    //匿名類別
    InnerFather p = new InnerFather() {

        private int a;

        {
            a = 123;
        }

        public void m1() {
            x = 123;
            y = 789;
        }
    };
    p.m1();
}
```

```
public void test1() {
    //類別等級
    InnerStruct1.y = 123;
    InnerStruct1.z = 123;
    InnerStruct1.s2();
    InnerStruct1.Inner1.b = 123;
    InnerStruct1.Inner1.n1();
    //////
    InnerStruct1.Inner1 p = new InnerStruct1.Inner1();
    p.a = 123;
    p.m1();
    //物件等級
    this.x = 123;
    this.s1();
    /////////
    InnerStruct1.Inner2 q = new InnerStruct1.Inner2();
    q.a = 123;
    q.m1();
}

    public static void test2() {
        //類別等級
        InnerStruct1.y = 123;
        InnerStruct1.z = 123;
        InnerStruct1.s2();
        InnerStruct1.Inner1.b = 123;
        InnerStruct1.Inner1.n1();
        //////
        InnerStruct1.Inner1 p = new InnerStruct1.Inner1();
        p.a = 123;
        p.m1();
        //物件等級
        new InnerStruct1().x = 123;
        new InnerStruct1().s1();
        ///////
        InnerStruct1.Inner2 q = new InnerStruct1().new Inner2();
        q.a = 123;
        q.m1();
    }
}
```

```
//不相關的類別
class OtherA {

    public void test3() {
        //類別等級
        InnerStruct1.z = 123;
        InnerStruct1.s2();
        InnerStruct1.Inner1.b = 123;
        InnerStruct1.Inner1.n1();
        //////
        InnerStruct1.Inner1 p = new InnerStruct1.Inner1();
        //p.a = 123;
        p.m1();
        //物件等級
        new InnerStruct1().x = 123;
        new InnerStruct1().s1();
        /////////
        InnerStruct1.Inner2 q = new InnerStruct1().new Inner2();
        // q.a = 123;
        q.m1();
    }
}
```

簡潔版➔外圍類別使用到巢狀類別外圍類別名稱可以省略

```
public void test1() {
    //類別等級
    y = 123;
    z = 123;
    s2();
    Inner1.b = 123;
    Inner1.n1();
    //////
    Inner1 p = new Inner1();
    p.a = 123;
    p.m1();
    //物件等級
    x = 123;
    s1();
    /////////
    Inner2 q = new Inner2();
    q.a = 123;
    q.m1();
}
```

```
    public static void test2() {
        //類別等級
        y = 123;
        z = 123;
        s2();
        Inner1.b = 123;
        Inner1.n1();
        //////
        Inner1 p = new Inner1();
        p.a = 123;
        p.m1();
        //物件等級
//        new InnerStruct2().x = 123;
//        new InnerStruct2().s1();
//        ///////
//        Inner2 q = new InnerStruct2().new Inner2();
//        q.a = 123;
//        q.m1();
        ///////////
        InnerStruct2 t = new InnerStruct2();
        t.x = 123;
        t.s1();
        /////////
        Inner2 q = t.new Inner2();
        q.a = 123;
        q.m1();
    }
}
```

不相關類別使用到巢狀類別➔外圍類別名稱不能省略

```
public void test3() {
    //類別等級
    InnerStruct2.z = 123;
    InnerStruct2.s2();
    InnerStruct2.Inner1.b = 123;
    InnerStruct2.Inner1.n1();
    //////
    InnerStruct2.Inner1 p = new InnerStruct2.Inner1();
    //p.a = 123;
    p.m1();
    //物件等級
    new InnerStruct2().x = 123;
    new InnerStruct2().s1();
    /////////
    InnerStruct2.Inner2 q = new InnerStruct2().new Inner2();
    // q.a = 123;
    q.m1();
}
```

巢狀類別的命名規則

```
class Father {

}

public class A001 {

    public static class B001 {

    }

    public class C001 {

    }

    public void s1() {

        class D001 {

        }
        class E001 {

        }
    }

public void s2() {

    class D001 {

    }
    class E001 {

    }
}

public void s3() {

    class F001 {

    }
    class G001 {

    }
}
```

```
    public void s4() {

        new Father() {
            void m1() {
            }
        };
        new Father() {
            void m2() {
            }
        };
    }

    public void s5() {
        new Father() {
            void m1() {
            }
        };
        new Father() {
            void m2() {
            }
        };
    }
}
```

- Father.class
- A001.rs
- A001.class
- A001$C001.class
- A001$B001.class
- A001$4.class
- A001$3.class
- A001$2E001.class
- A001$2D001.class
- A001$2.class
- A001$1G001.class
- A001$1F001.class
- A001$1E001.class
- A001$1D001.class
- A001$1.class

static Member Class

1. 只能使用外圍類別是 static 的成員
2. 類別等級的巢狀類別裡面可以宣告 物件等級 與 類別等級 的成員
3. 外圍類別使用巢狀類別的方式
4. 不相關的類別使用 巢狀類別 的方式

```java
public class Outer1 {

    public int x;
    public static int y;
    private static int z;

    public void s1() {
        Inner1.b = 123;
        Inner1.n2();
        Inner1 p = new Inner1();
        p.a = 123;
        p.c = 123;
        p.n1();
    }

    public static void s2() {
        Inner1.b = 123;
        Inner1.n2();
        Inner1 p = new Inner1();
        p.a = 123;
        p.c = 123;
        p.n1();
    }

    public static class Inner1 {

        public int a;
        public static int b;
        private int c;

        public void n1() {
            //x=123;
            y = 123;
            z = 123;
            // s1();
            s2();
        }

        public static void n2() {
            //x=123;
            y = 123;
            z = 123;
            // s1();
            s2();
        }
    }
}
```

```java
class Other1 {

    public void m1() {
        Outer1.Inner1.b = 123;
        Outer1.Inner1.n2();
        Outer1.Inner1 p = new Outer1.Inner1();
        p.a = 123;
        //p.c = 123;
        p.n1();
    }
}
```

instance Member Class

1. 可以使用外圍類別 物件等級 或 類別等級的成員
2. 物件等級的巢狀類別內不可宣告 static 的成員
3. 外圍類別使用巢狀類別的方式
4. 不相關的類別使用 巢狀類別 的方式

```
public class Outer2 {

    public int x;
    public static int y;
    private static int z;

    public void s1() {
        Inner2 p = new Inner2();
        p.a = 123;
        p.c = 123;
        p.n1();

    }

    public static void s2() {
        Inner2 p = new Outer2().new Inner2();
        p.a = 123;
        p.c = 123;
        p.n1();
    }

    public class Inner2 {

        public int a;
        // public static int b;
        private int c;

        public void n1() {
            x = 123;
            y = 123;
            z = 123;
            // s1();
            s2();
        }

       // public static void n2() {
            // x = 123;
            // y = 123;
            // z = 123;
            // s1();
            // s2();
       // }
    }
}

class Other2 {

    public void m1() {

        Outer2.Inner2 p = new Outer2().new Inner2();
        p.a = 123;
        //p.c = 123;
        p.n1();
    }
}
```

Local Class

1. 超出包圍它的方法，在別的方法內，就無法 new 它

```
public class Outer3 {

    public int x = 1;
    public int y = 2;

    public void s1(int r) {

        int m = 3;

        class Inner3 {

            public int a = 4;
            public int b = 5;

            public void n1() {
                int c;
                c = x + y + m + a + b + r;

            }
        }
        Inner3 inn = new Inner3();  //在方法內 new
        inn.n1();
        m = 123; //閉包內用到的 外圍區域變數值不可被更改 , 預設為 final
    }

    public void s2() {
        //Inner3 inn = new Inner3();  //不可在別的方法內 new , 超出範圍
        //inn.n1();
    }
}
```

local variables referenced from an inner class must be final or effectively final
---
(Alt-Enter shows hints)

匿名類別➔不具名的類別稱為 “匿名類別”

1. 一般 “有具名的類別” 是先定義類別這種型態，之後再利用它宣告，建構物件，整個類別還可以重複用來 宣告，建構 許多物件
2. “匿名類別” 是 宣告 與 定義 合而為一的類別，故此類別僅僅使用一次，不能來產生第二個物件
3. 所以 “ 匿名類別” 比起 “一般巢狀類別” ，是更純粹用在外圍類別中的巢狀類別，大多是用於成員函數之內
4. 匿名類別的父類別是指此匿名類別要 繼承 或 實作 的父類別或父介面➔匿名繼承
5. 匿名類別沒有自己的類別名稱，無法像一般巢狀類別那樣定義建構子，要替 “匿名類別” 產生的 “物件” 作初始化行為，可以在其內使用一個 “區塊敘述句” (instance 區塊)

```
public interface InnerDemo1P {

    void f();
}

class InnerDemo1C implements InnerDemo1P {

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

```
public class InnerAnonyClass1 {

    public void s1() {
        class InnerDemo1C implements InnerDemo1P {

            public void f() {
                System.out.println("B");
            }
        }
        new InnerDemo1C().f();
    }
}
```

```
public class InnerAnonyClass2 {

    public void s1() {
        new InnerDemo1P() {

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

```
public class InnerAnonyClass3 {

    public void s1() {
        s2(new InnerDemo1C());
    }

    public void s2(InnerDemo1C p) {
        p.f();
    }
}
```

```
public class InnerAnonyClass4 {

    public void s1() {
        s2(new InnerDemo1P() {
            @Override
            public void f() {
                System.out.println("D");
            }
        });
    }

    public void s2(InnerDemo1P p) {
        p.f();
    }
}
```

匿名類別的轉換過程

```java
public interface InnerDemo2P {

    void abc();
   // void xyz();
}
```

```java
class InnerDemo2C extends InnerDemo2P {

    public int a;
    public int b;

    public InnerDemo2C() {
        a = 10;
        b = 10;
    }

    public void abc() {
        System.out.println("a=" + a + "  b=" + b);

    }

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

```java
public static void 匿名巢狀類別轉變1() {
    new InnerDemo2C().abc();
}
```

```java
public static void 匿名巢狀類別轉變2() {

    class InnerDemo2C extends InnerDemo2P {

        public int a;
        public int b;

        public InnerDemo2C() {
            a = 20;
            b = 20;
        }

        public void abc() {
            System.out.println("a=" + a + "  b=" + b);
        }

        public void xyz() {
            System.out.println("xyz");
        }
    }
    new InnerDemo2C().abc();
}
```

```java
public static void 匿名巢狀類別轉變3() {

    new InnerDemo2P() {
        public int a;
        public int b;

        {
            a = 30;
            b = 30;
        }

        public void abc() {
            System.out.println("a=" + a + "  b=" + b);
        }

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

```
public static void 匿名巢狀類別轉變4() {

    // 請注意 p 在 compiler 階段會系統會認為是 InnerDemo2P 的型態
    InnerDemo2P p = new InnerDemo2P() {
        public int a;
        public int b;

        {
            a = 40;
            b = 40;
        }

        public void abc() {
            System.out.println("a=" + a + "  b=" + b);
        }

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

```
public static void 匿名巢狀類別轉變5() {

    呼叫匿名巢狀類別轉變5(new InnerDemo2C());
}

public static void 呼叫匿名巢狀類別轉變5(InnerDemo2C p) {
    p.abc();
    p.xyz();
}
```

```
public static void 匿名巢狀類別轉變6() {

    呼叫匿名巢狀類別轉變6(new InnerDemo2P() {

        public int a;
        public int b;

        {
            a = 50;
            b = 50;
        }

        public void abc() {
            System.out.println("a=" + a + "  b=" + b);
        }

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

public static void 呼叫匿名巢狀類別轉變6(InnerDemo2P p) {
    p.abc();
    // p.xyz();
}
```

```java
public class Outer4 {

    public void abc(int p1, int p2) {
        int f1 = 30;
        int f2 = 40;
        class Inn1 {

            public int a;
            public int b;

            public Inn1() {
                a = p1;
                b = p2;
            }

            public void f() {
                System.out.println("晚上");
            }
        }
        Inn1 p = new Inn1();
        p.f();
        //==================================
        class Inn2 {

            public int c;
            public int d;

            public Inn2() {
                c = f1;
                d = f2;
            }

            public void g() {
                System.out.println("黃昏");
            }
        }
        Inn2 q = new Inn2();
        q.g();
    }
}
```

```java
public static void 匿名巢狀類別_轉換前1() {
    Outer4 p = new Outer4();
    p.abc(90, 100);
}
```

晚上
黃昏

```java
interface Inn1P {
    void f();
}

abstract class Inn2P {
    public abstract void g();
}

public class Outer5 {

    public void abc(int p1, int p2) {

        int f1 = 30;
        int f2 = 40;
        Inn1P p = new Inn1P() {
            public int a;
            public int b;

            {
                a = p1;
                b = p2;
            }

            public void f() {
                System.out.println("晚上");
            }
        };
        p.f();
        //==================================
        new Inn2P() {
            public int c;
            public int d;

            {
                c = f1;
                d = f2;
            }

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

```java
public static void 匿名巢狀類別_轉換後1() {
    Outer5 p = new Outer5();
    p.abc(90, 100);
}
```

晚上
黃昏

```java
public class Outer6 {

    public int x;
    public static int y;

    public class Inn3 {

        public int a;
        public int b;

        public Inn3() {
            a = 4;
            b = 5;
        }

        public void 習慣() {
            System.out.println("吃檳榔");
        }
    }

    public void s1(int p) {

        s2(new Inn3());
    }

    public void s2(Inn3 t) {
        t.習慣();
    }
}
```

```java
interface Inn3P {

    void 習慣();
}

public class Outer7 {

    public int x;
    public static int y;

    public void s1(int p) {

        s2(new Inn3P() {
            public int a;
            public int b;

            {
                a = 4;
                b = 5;
            }

            public void 習慣() {
                System.out.println("吃檳榔" + p + "顆");
            }
        });
    }

    public void s2(Inn3P t) {
        t.習慣();
    }
}
```

```java
public static void 匿名巢狀類別_轉換前2() {
    Outer6 p = new Outer6();
    p.s1(4);
}
```

```
吃檳榔
```

```java
public static void 匿名巢狀類別_轉換後2() {
    Outer7 p = new Outer7();
    p.s1(4);
}
```

```
吃檳榔4顆
```

```
public class Person10 {

    public Head10 theH;
    private int age;
    public String name;

    public Person10(int a, String n) {
        age = a;
        name = n;
    }
    public void setHead(String e, String f, String h, double s) {
        theH =new Head10(e, f, h, s);
    }
    public class Head10 {

        public String eye;
        public String face;
        public String hair;
        public double size;

        public Head10(String e, String f, String h, double s) {
            eye = e;
            face = f;
            hair = h;
            size = s;
            if (age > 80) {
                hair = hair + "，有白髮";
            }
        }
        public void showHeadProperties() {
            System.out.println(name + " 的頭：" + "\n眼睛： " + eye
                    + "\n臉： " + face + "\n頭髮： " + hair + "\n頭圍： " + size);
        }
    }
}
```

巢狀類別範例1

```
public static void 巢狀類別範例1() {
    Person10 cyh = new Person10(100, "cyh");
    cyh.setHead("咖啡黑", "不是方形", "黑色", 57.5);
    cyh.theH.showHeadProperties();
}
```

```
cyh 的頭：
眼睛： 咖啡黑
臉： 不是方形
頭髮： 黑色，有白髮
頭圍： 57.5
```

```
interface Head11P {

    void showHeadProperties();
}

public class Person11 {

    public Head11P theH;
    private int age;
    public String name;

    public Person11(int a, String n) {
        age = a;
        name = n;
    }

    public void setHead(String e, String f, String h, double s) {
        theH = new Head11P() {
            public String eye;
            public String face;
            public String hair;
            public double size;

            {
                eye = e;
                face = f;
                hair = h;
                size = s;
                if (age > 80) {
                    hair = hair + "，有白髮";
                }
            }

            public void showHeadProperties() {
                System.out.println(name + " 的頭：" + "\n眼睛： " + eye
                        + "\n臉： " + face + "\n頭髮： " + hair + "\n頭圍： " + size);
            }
        };
    }
}
```

```
public static void 巢狀類別範例2() {
    Person11 cyh = new Person11(100, "cyh");
    cyh.setHead("咖啡黑", "不是方形", "黑色", 57.5);
    cyh.theH.showHeadProperties();

}
```

```
cyh 的頭：
眼睛： 咖啡黑
臉： 不是方形
頭髮： 黑色，有白髮
頭圍： 57.5
```