inner classes

เริ่มจาก Java 2 ภาษา Java ยอมให้มีการกำหนดคลาสขึ้นภายในคลาส เรียกว่า inner classes ซึ่งมี 4 ประเภท ดังนี้

1. Static Inner Classes คือ static class ที่ซ้อนอยู่ในคลาสอื่น

```
// StaticInnerTest.java
package com.mycomp.inner;
class A {
static class B {
void f() { System.out.println("Hello"); }
}
}
public class StaticInnerTest {
public static void main(String args[]) {
A.B b = new A.B();
b.f();
}
}
```

กรณีนี้ คลาส B เป็น static inner class ของคลาส A และเราเรียกคลาส A ว่าเป็น outer class
เราใช้ static inner class เป็น name space อีกระดับหนึ่งต่อจาก package ลงไป
ทำให้ต้องอ้างถึงคลาส B ที่อยู่ในคลาส A อย่าง static ด้วย A.B
เมื่อถูก compile แล้วคลาส B จะมีไฟล์เป็น A$B.class
Static inner classes จะซ้อนลึกลงไปอีกระดับก็ได้
Static classes ต้องเป็น inner class อยู่ใน outer class ใดคลาสหนึ่ง ไม่สามารถเป็น top level class

กฎเกณฑ์เกี่ยวกับ visibility ของ static inner class แสดงในตัวอย่างต่อไปนี้

```
//  StaticInnerVisibility.java
package com.mycomp.inner;
class S1 {
public int x = 1;
static private int y = 2;
static class S2 {
void print() {
// System.out.println(x); // x is not static
```

```
System.out.println(y + S3.z);
}
}
static class S3 {
static private int z = 3;
}
}
```

ภายใน static inner class จะอ้างถึง static members ทั้งหมด (แม้แต่ private) ของ outer class หรือของ static inner classes อื่นที่อยู่ใน outer class เดียวกัน

---

2. Member Classes คือ class ที่เป็น member อยู่ในคลาสอื่น

```
// MemberClassTest.java
package com.mycomp.inner;
class X {
class Y {
void f() { System.out.println("Hello"); }
}
}
class MemberClassTest {
public static void main(String args[]) {
X x = new X();
X.Y y = x.new Y();
y.f();
}
}
```

กรณีนี้เราจะอ้างถึงคลาส Y โดยใช้ X.Y ไม่ได้ แต่จะต้องสร้าง instance ของคลาส X ซึ่งเป็น outer class ก่อน แล้วนำ instance นั้นมา new ในกรณีนี้คือ x.new Y() จึงจะได้ instance ของคลาส Y ที่ติดกับ instance x นั้น เมื่อถูก compile แล้ว คลาส Y จะมีไฟล์เป็น X$Y.class

กฎเกณฑ์เกี่ยวกับ visibility ของ member class แสดงในตัวอย่างต่อไปนี้

```
// MemberClassVisibility.java
package com.mycomp.inner;
class M1 {
```

```
private int x = 1;
M1.M2 m = new M1.M2();
void print() {  System.out.println(m.y); } // *

class M2 {
private int y = 2;
void print() { System.out.println(x); } // **
}
class M3 {
M1.M2 m = new M1.M2();
void print () { System.out.println(m.y); } // ***
}
}
```

* ใน outer class จะสามารถอ้างถึง members (แม้แต่ private) ของ member class

** ใน member class จะสามารถอ้างถึง members ของ outer class

*** ใน member class จะสามารถอ้างถึง members ของ member class ใน outer class เดี่ยวกัน

---

3. Local Classes คือคลาสที่ถูกกำหนดขึ้นภายใน scope ของ block หมายถึงประโยคที่อยู่ระหว่าง { และ }

```
// LocalClassTest.java
package com.mycomp.inner;
public class LocalClassTest {
public static void main(String args[]) {
// A is not visible before the class definition
class A {
void f() { System.out.println("Hello"); }
}
new A().f();
}
}
```

Scope rules ระบุว่า ชื่อที่ถูกกำหนดขึ้นภายใน block จะถูกอ้างถึงได้หลังจากประโยคที่กำหนดชื่อนั้นขึ้น ไปจนถึง } ที่สิ้นสุด scope ของ block นั้น แสดงว่าในกรณีนี้ คลาส A ที่เป็น local class นี้ จะถูกอ้างได้หลังจาก class definition ไปจนถึง }ของ main() เท่านั้น

เมื่อถูก compile แล้วคลาส A จะมีไฟล์เป็น LocalClassTest$A1.class การมีเลข 1 ต่อท้ายแสดงว่าในคลาส LocalClassTest นี้อาจมีคลาสชื่อ A ได้หลายคลาส แต่ต้องอยู่ต่าง scope ของเราสร้างคลาสให้เป็น local class เพื่อจำกัดให้คลาสนั้นถูกใช้งานได้เพียงแค่ใน scope ที่กำหนดคลาสนั้นขึ้นเท่านั้น Blocks ที่เราสามารถกำหนด local class ได้อาจเป็น block ใดๆ ที่ไม่ใช่ scope ของคลาส เช่น

Free Block:

```
{
class B { }
new B();
}
```

For Loop Block:

```
for(int i = 0; i < 10; i++) {
class C{ }
new C();
}
```

If Statement Block:

```
if(x > 0) {
class D{ }
new D();
}
```

เป็นต้น

กฎเกณฑ์เกี่ยวกับ visibility ของ local class แสดงในตัวอย่างต่อไปนี้

```
// LocalClassVisibility.java
package com.mycomp.inner;
class L1 {
private int a = 0;
void f(final int w, int x) {
int y = 3;
final int z = 4;
class L2 {
public void print() {
System.out.println(a + w + z); // cannot assesses x and y
}
}
new L2().print();
}
}
```

ใน local class จะสามารถอ้างถึง

- members (แม้แต่ private) ของ outer class
- final local variables และ final parameters ของ block นั้น

4. Anonymous Classes คือ คลาสที่ไม่มีชื่อ

Local classes ถูกประกาศขึ้นในระดับของ statement

Anonymous classes ถูกประกาศขึ้นในระดับของ expression ใน statement

ตัวอย่าง `x = y + 1;` เป็น statement ส่วน y + 1 เป็น expression

`a = f(a+b, c);` เป็น statement ส่วน a+b กับ c เป็น expressions

ตอนประกาศ anonymous class จะไม่มีชื่อของคลาส แต่ต้องระบุคลาสแม่ หรือ interface ที่คลาสนั้นจะ implements ตามด้วยวงเล็บแล้วอาจมี arguments ที่ chain ขึ้น constructor ของคลาสแม่ และตามด้วย class body ที่จะ override methods ของคลาสแม่ หรือ implements interface นั้น ดังนี้

```
new <parent-class|interface> (<arguments>) { <body> }
```

คลาสนี้ไม่มีชื่อสำหรับอ้างถึงได้อีก ดังนั้นเมื่อประกาศ anonymous class ที่ใด จะต้อง new instance ขึ้นตรงนั้นเลย และมีได้เพียง instance เดียวเท่านั้น

```
// AnonymousClassTest.java
```

```
package com.mycomp.inner;
class T {
public void print() { System.out.println("T"); }
}
public class AnonymousClassTest {
public static void main(String[] args) {
T t = new T() {
public void print() { System.out.println("A"); }
};
t.print();
}
}
```

กรณีนี้เราสร้าง anonymous class ที่เป็นคลาสลูกของคลาส T และ override method print() ให้พิมท์ออกมาเป็น A เมื่อถูก compile แล้วคลาสที่ไม่มีชื่อนี้จะมีไฟล์เป็น AnonymousClassTest$1.class เนื่องจากถูกกำหนดขึ้นภายในคลาส AnonymousClassTest หากคลาสใดมีไฟล์ .class ที่มีชื่อลงท้ายด้วย $ และตามด้วยตัวเลข แสดงว่าเป็น anonymous class

Anonymous class อาจเป็นคลาสที่ implements interface ก็ได้ เช่น

```
interface I {
public void print();
}
                .....
I i = new I() {
public void print() { System.out.println("X"); }
};
i.print();
```

ตอนกำหนด anonymous class อาจจะมีการ chain ค่าพารามิเตอร์ ขึ้น constructor ของคลาสแม่ เช่น

```
class U {
    private String s;
    U(String s) { this.s = s; }
public void print() { System.out.println(s); }
}
                .....
new U("Hello") { }.print();
```

กฎเกณฑ์เกี่ยวกับ visibility ของ anonymous class จะเหมือนกับ local class