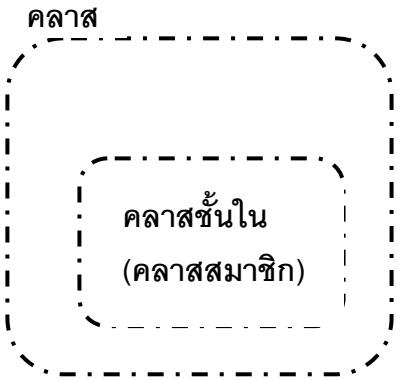
คลาสชั้นใน

คลาสชั้นใน (inner class) เป็นจุดเด่นอีกจุดหนึ่งของภาษาจาวาที่เพิ่งเพิ่มเข้ามาใหม่ โดยเริ่มเผยแพร่ตั้งแต่ JDK รุ่น ๑.๑ เป็นต้นมา คลาสชั้นในหมายถึงคลาสที่บรรจุอยู่ภายในคลาส จึงถือว่าเป็นสมาชิกของคลาสด้วย เดิมทีสมาชิกของคลาสมี ๒ ประเภทคือตัวแปรสมาชิก และ เมธอดสมาชิก เมื่อภาษาจาวาพัฒนามากขึ้นมีคลาสชั้นในเพิ่มขึ้นมาใหม่ จึงนับได้ว่าคลาสชั้นในเป็นสมาชิกประเภทที่ ๓ ของคลาส คลาสชั้นในซึ่งในภาษาอื่นมักเรียกว่าคลาสซ้อน (nested class) ช่วยทำให้โปรแกรมมีความชัดเจนดูง่ายและยังทำให้เขียนโปรแกรมได้กระชับขึ้นอีก
ด้วย



ตั้งแต่รุ่น ๑.๑ เป็นต้นมานักเขียนโปรแกรมสามารถคลาสชั้นในให้เป็นสมาชิกของคลาสอื่น หรือเขียนไว้ภายในบล็อกของข้อความสั่ง(เป็นคลาสเฉพาะบริเวณ - local class) หรือเขียนคลาสไว้ในนิพจน์(เป็นคลาสนิรนาม - anonymous class) ก็ได้ แต่ในเอกสารฉบับนี้จะกล่าวถึงเฉพาะคลาสชั้นในที่เป็นสมาชิกของคลาสอื่นเท่านั้น

สมบัติบางประการที่ทำให้คลาสชั้นในน่าใช้คือ

- ชื่อคลาสชั้นในไม่สามารถใช้ได้นอกขอบเขตของมัน เว้นเสียแต่ว่าจะใช้ชื่อเต็ม (qualified name) การทำเช่นนี้ช่วยให้จัดโครงสร้างของคลาสในแพ็คเกจได้ดีขึ้น
- การเขียนคลาสชั้นในสามารถใช้ชื่ออย่างง่ายภายในขอบเขตของคลาสที่ล้อมรอบมันได้ รวมทั้งตัวแปรสมาชิกของคลาสชั้นนอกที่ล้อมรอบ และตัวแปรเฉพาะบริเวณ(local variable)ของบล็อกภายนอกที่ล้อมรอบตัวมันด้วย

โดยพื้นฐานแล้วคลาสชั้นในเหมือนคลาสอื่นๆทั่วไป แต่ประกาศไว้ในคลาสอื่น ทำให้สามารถเข้าถึงและเรียกใช้ตัวแปรสมาชิกและเมธอดต่างๆ ของคลาสชั้นนอกได้ ดังตัวอย่างต่อไปนี้

ตัวอย่างที่ ๑

```
๑. public class TestInner1 {
๒.     public static void main(String args[]) {
๓.         OuterClass oc = new OuterClass();
๔.         oc.outerMethod();
๕.         oc.ic.innerMethod();
๖.     }
๗. }
๘. class OuterClass {
๙.     int x=1;
๑๐.    InnerClass ic = new InnerClass();
๑๑.    public void outerMethod() {
๑๒.        System.out.println("x is " + x);
๑๓.    }
๑๔.    public class InnerClass {
๑๕.        int y=2;
๑๖.        void innerMethod() {
๑๗.            System.out.println("y is " + y);
๑๘.            System.out.println("inner classes can access member " +
๑๙.                "variables of outer class : x is " + x);
๒๐.            System.out.print("inner classes can call methods of " +
๒๑.                "outer class : ");
๒๒.            outerMethod();
๒๓.        }
๒๔.    }
๒๕.}
```

ตัวอย่างนี้มีคลาสระดับสูงสุด (top level class) ๒ คลาส คลาสแรกชื่อว่า `TestInner1` มีขอบเขตตั้งแต่บรรทัดที่ ๑ - ๗ คลาสที่สองคือ `OuterClass` มีขอบเขตตั้งแต่บรรทัดที่ ๘ - ๒๕ ภายในคลาส `OuterClass` นอกจากมีการประกาศตัวแปร `x` และเมธอด `outerMehtod` แล้ว ยังมีการประกาศคลาสชั้นในชื่อว่า `InnerClass` มีขอบเขตตั้งแต่บรรทัดที่ ๑๔ - ๒๔ บรรจุอยู่ภายในอีกด้วย ตรงบรรทัดที่ ๑๐ มีการสร้างอ๊อบเจ็กต์ของคลาสนี้ให้เป็นสมาชิกของคลาส `OuterClass` บรรทัดที่ ๕ มีการเรียกเมธอด `innerMethod` ของอ๊อบเจ็กต์ `ic` ซึ่งอยู่ในอ๊อบเจ็กต์ `oc`

<u>หมายเหตุ</u> บรรทัดที่ ๕ เป็นเมธอดเรียกเมธอดของอ็อบเจ็กท์ซึ่งอยู่ในอีกอ๊อบเจ็กต์ ไม่จำเป็นว่าจะต้องเป็นอ๊อบเจ็กต์ที่สร้างมาจากคลาสชั้นใน เราสามารถเรียกใช้อ๊อบเจ็กต์ที่สร้างมาจากคลาสชั้นในเหมือนกับการเรียกใช้อ๊อบเจ็กต์ที่สร้างมาจากคลาสทั่วไป

เมื่อคอมไพล์โปรแกรมนี้แล้วจะได้แฟ้มคลาส ๓ แฟ้มคือ `TestInner1.class` `OuterClass.class` และ `OuterClass$InnerClass.class` คอมไพเลอร์จะตั้งชื่อคลาสชั้นในโดยนำหน้าชื่อคลาสชั้นในด้วยชื่อคลาสชั้นนอกคั่นด้วยเครื่องหมายดอลลาร์ ($) การที่ใช้เครื่องหมายดอลลาร์ไม่ใช้เครื่องหมายจุด (.) ก็เพื่อหลีกเลี่ยงความสับสนกับชื่อ package

ในคลาสชั้นใน `InnerClass` มีตัวแปรสมาชิกคือ `y` และมีเมธอดสมาชิกคือ `innerMethod` ตัวอย่างนี้แสดงให้เห็นว่าเราสามารถใช้ตัวแปรสมาชิกของคลาสชั้นนอกได้ดังในบรรทัดที่ ๑๙ เมธอด `InnerMethod` ซึ่งเป็นเมธอดของคลาสชั้นในจะรู้จักและสามารถใช้ตัวแปร `x` ซึ่งเป็นตัวแปรของคลาส `OuterClass` ได้โดยตรง โดยไม่ต้องมีตัวแปรอ้างอิงนำหน้า บรรทัดที่ ๒๒ ก็เช่นเดียวกัน ภายในเมธอด `innerMethod` สามารถเรียกเมธอด `outerMethod` ของ `OuterClass` ได้โดยตรง

```
x is 1
y is 2
inner classes can access member variables of outer class : x is 1
inner classes can call methods of outer class : x is 1
```

ผลลัพธ์จากโปรแกรมที่ ๑

การใช้คลาสชั้นในให้ข้อดีหลายประการเช่น

- คลาสชั้นในจะถูกซ่อนไว้ในคลาสชั้นนอกทำให้คลาสอื่นมองไม่เห็น ทำให้ไม่ต้องกังวลว่าจะตั้งชื่อคลาสซ้ำกับคลาสอื่น
- คลาสชั้นในสามารถเข้าถึงตัวแปรและเมธอดที่อยู่ภายในขอบเขตของคลาสชั้นนอกได้โดยสะดวก หากเขียนทั้งสองคลาสแยกออกจากกันไม่ทำเป็นคลาสชั้นในและคลาสชั้นนอกแล้ว จะไม่สามารถใช้ตัวแปรและเมธอดของคลาสอื่นได้โดยตรง ต้องเข้าถึงผ่านตัวแปรอ้างอิง หรือชื่อคลาส

ความสามารถในการเข้าถึงสมาชิกของคลาสที่ปิดล้อม (enclosing class) ได้โดยง่ายโดยไม่ต้องระบุตัวแปรอ้างอิงนั้นเป็นประโยชน์อย่างมาก การที่ทำเช่นนี้ได้นั้น เป็นเพราะคลาสชั้นในมีตัวแปรอ้างอิงไปยังอ๊อบเจ็กต์ของคลาสชั้นนอกซ่อนอยู่ภายใน ก่อนที่จะสร้างอ๊อบเจ็กต์ของคลาสชั้นในนั้น จำเป็นต้องมีอ๊อบเจ็กต์ของคลาสชั้นนอกอยู่ก่อนเสมอ เมื่อมีการสร้างอ๊อบเจ็กต์ของคลาสชั้นในขึ้นมาอ๊อบเจ็กต์ที่สร้างขึ้นมาใหม่นี้จะเก็บที่อยู่ของอ๊อบเจ็กต์ของคลาสชั้นนอกซึ่งเป็นผู้สร้างมันขึ้นมาไว้ใช้ในการอ้างอิงสมาชิกของอ๊อบเจ็กต์ของคลาสชั้นนอกนั่นเอง ผลของมันก็เพื่อให้มั่นใจว่าอ๊อบเจ็กต์คลาสชั้นในและคลาสชั้นนอกต่างก็ขึ้นแก่กันและกัน ไม่ใช่ว่าอ๊อบเจ็กต์ของคลาสชั้นในเป็นเพียงแค่สมาชิกหนึ่งของอ๊อบเจ็กต์ของคลาสชั้นนอกเท่านั้น

ในบางครั้งเราอาจต้องการเขียนข้อความสั่งสำหรับสร้างอ๊อบเจ็กต์ของคลาสชั้นในไว้ภายในเมธอดสถิต (static method) หรือในบางสถานการณ์ที่ไม่มีตัวอ้างอิง `this` ให้ใช้ เช่นในเมธอด `main()` หรือถ้าจำเป็นต้อง

สร้างอ๊อบเจ็กต์ของคลาสชั้นในจากเมธอดของบางอ๊อบเจ็กต์ของคลาสที่ไม่เกี่ยวข้องกัน เราสามารถทำได้โดยใช้ตัวคำสำคัญ new โดยคิดเสมือนว่ามันเป็นเมธอดสมาชิกของคลาสชั้นนอก ทั้งนี้ยังจำเป็นต้องมีอ๊อบเจ็กต์ของคลาสชั้นนอกอยู่ก่อนแล้วจึงจะสร้างอ๊อบเจ็กต์ของคลาสชั้นในได้ ตัวอย่างที่ ๒ มีการสร้างอ๊อบเจ็กต์ของคลาสชั้นในอยู่ในเมธอด `main()`

ตัวอย่างที่ ๒

```
๑. public class TestInner2 {
๒.     public static void main(String args[]) {
๓.         OuterClass.InnerClass i =
                                    new OuterClass().new InnerClass();
๔.         i.innerMethod();
๕.     }
๖. }
๗.
๘. class OuterClass {
๙.     int x=1;
๑๐.    public void outerMethod() {
๑๑.        System.out.println("x is " + x);
๑๒.    }
๑๓.    public class InnerClass {
๑๔.        int y=2;
๑๕.        void innerMethod() {
๑๖.            System.out.println("enclosing x is " + x);
๑๗.            System.out.println("y is " + y);
๑๘.        }
๑๙.    }
๒๐.}
```

การสร้างอ๊อบเจ็กต์ของคลาสชั้นในไว้ในคลาสอื่นที่ไม่ใช่คลาสชั้นนอกจำเป็นต้องบอกชื่อคลาสชั้นในให้สมบูรณ์ กล่าวคือต้องมีชื่อคลาสชั้นนอกกำกับอยู่ด้วยโดยใช้จุดเป็นตัวเชื่อม ดังตัวอย่างที่ ๒ ในบรรทัดที่ ๓ เราต้องเขียนชื่อคลาสของอ๊อบเจ็กต์ที่จะสร้างใหม่เป็น `OuterClass.InnnerClass` ในการสร้างอ๊อบเจ็กต์ใหม่นั้นเราใช้คำสำคัญ new ตามด้วยการเรียก constructor สำหรับการสร้างอ๊อบเจ็กต์ของคลาสชั้นในจำเป็นต้องมีอ๊อบเจ็กต์ของคลาสชั้นนอกก่อน ดังนั้นเราจึงต้องสร้างอ๊อบเจ็กต์ของ `OuterClass` โดยการระบุ new `OuterClass()` ซึ่งจะคืนค่ากลับมาเป็นที่อยู่ของอ๊อบเจ็กต์ของคลาส `OuterClass` ที่สร้างขึ้นใหม่ หลังจากนั้นจึงมาสร้างอ๊อบเจ็กต์ของ `InnerClass` โดยด้วยนิพจน์ new `InnerClass()` ทั้งนี้ต้องมีจุดเป็นตัวเชื่อม ซึ่งเป็นการส่งที่อยู่ของอ๊อบเจ็กต์ของคลาส `OuterClass` ที่สร้างขึ้นมาก่อนไปให้อ๊อบเจ็กต์ของคลาส `InnerClass` ที่กำลังจะสร้างขึ้น

ตัวอย่างที่ ๒ เราไม่มีความจำเป็นต้องใช้อ๊อบเจ็กต์ของคลาส `OuterClass` โดยตรง แต่จำเป็นต้องสร้างขึ้นมาก่อนจึงจะสร้างอ๊อบเจ็กต์ของคลาสชั้นในได้ ถ้าเราจำเป็นต้องใช้อ๊อบเจ็กต์ของคลาสชั้นนอกอยู่แล้ว เราสามารถสร้างอ๊อบเจ็กต์ของคลาสชั้นในหลังจากสร้างอ๊อบเจ็กต์ของคลาสชั้นนอกดังแสดงในตัวอย่างที่ ๓ บรรทัดที่ ๔ เราต้องใส่ตัวแปรอ้างอิงไว้หน้าคำสำคัญ new เชื่อมด้วยจุด เพื่อส่งที่อยู่ของอ๊อบเจ็กต์ของคลาสชั้นนอกไปให้อ๊อบเจ็กต์ใหม่

ตัวอย่างที่ ๓

```
๑. public class TestInner3 {
๒.     public static void main(String args[]) {
๓.         OuterClass o = new OuterClass();
๔.         OuterClass.InnerClass i = o.new InnerClass();
๕.         i.innerMethod();
๖.     }
๗. }
๘. class OuterClass {
๙.     int x=1;
๑๐.    public void outerMethod() {
๑๑.        System.out.println("x is " + x);
๑๒.    }
๑๓.    public class InnerClass {
๑๔.        int y=2;
๑๕.        void innerMethod() {
๑๖.            System.out.println("enclosing x is " + x);
๑๗.            System.out.println("y is " + y);
๑๘.        }
๑๙.    }
๒๐.}
```

ข้อควรระวัง

ถึงแม้ว่าจะใช้จุดเป็นตัวเชื่อมชื่อคลาสชั้นในกับชื่อคลาสชั้นนอกในการประกาศประเภทของตัวแปรอ้างอิงก็ตาม ไม่ได้หมายความว่าชื่อคลาสที่ใช้จริงจะเป็นเช่นนั้น ชื่อคลาสที่แท้จริงจะเชื่อมด้วยเครื่องหมายดอลลาร์ เช่น `OuterClass$InnerClass` เมื่อคอมไพล์โปรแกรมแล้ว คลาสชั้นในยังคงเก็บเป็นแฟ้มอิสระแยกจากคลาสชั้นนอกไม่ได้เก็บรวมกันเป็นแฟ้มเดียวกับคลาสชั้นนอก ดังนั้นในการเคลื่อนย้ายแฟ้มคลาสหรือนำไปใส่ในเว็บไซต์เพื่อการเผยแพร่จึงควรระมัดระวังนำไปให้ครบทุกแฟ้ม