Inner Class Usage Rules in Java

1. **區域變數若要被內部類別調用，必須具備 final 特性：**
   * 這個描述基本上是針對 JDK 8 之前的規則。在 JDK 8 之前，內部類別只能存取封閉區塊（即它所定義的方法或區塊）中的 final 區域變數。這些變數在初始化後不能再修改，稱為常數。
2. **JDK 8 之後的改變：**
   * 從 JDK 8 開始，內部類別不再要求區域變數必須顯式宣告為 final，但是區域變數的值仍然必須在初始化後保持不變（事實上的 "effectively final"）。也就是說，即使變數未使用 final 關鍵字，內部類別仍可存取該變數，但該變數的值不能在其後進行修改。

### 內部類別的規則：

1. **種類：**
   * **成員內部類別**（Member Inner Class）：在類別的外部不可直接存取，需透過外部類別實例化後調用。
   * **靜態內部類別**（Static Inner Class）：屬於外部類別的靜態成員，可以不依賴外部類別的實例直接存取。
   * **區域內部類別**（Local Inner Class）：定義在方法、建構子或區塊中的類別。
   * **匿名內部類別**（Anonymous Inner Class）：沒有名稱，通常用來即時實作接口或繼承類別。
2. **存取封閉類別的成員：**
   * 成員內部類別可以直接存取外部類別的所有成員，包括 private 成員。
   * 靜態內部類別只能存取外部類別的靜態成員。
3. **區域變數的限制（JDK 8 前後的區別）：**
   * JDK 8 之前，區域內部類別只能存取外部方法的 final 區域變數。
   * JDK 8 之後，區域內部類別可以存取「實際上是 final」的區域變數，即這些變數在初始化後不能再被修改。
4. **實例化規則：**
   * 成員內部類別需要通過外部類別的實例來實例化：OuterClass outer = new OuterClass(); OuterClass.InnerClass inner = outer.new InnerClass();
   * 靜態內部類別則可以直接通過類別名稱實例化：OuterClass.StaticInnerClass inner = new OuterClass.StaticInnerClass();

### 外部類別的規則：

1. **外部類別不能是 private 或 protected：**
   * 外部類別只能有 public 或包內可見的存取修飾符。
2. **外部類別不能宣告為 static：**
   * 只有內部類別才能宣告為 static，外部類別不能直接使用 static 修飾符。
3. **外部類別不能直接存取內部類別的成員：**
   * 外部類別若要存取內部類別的成員，必須先實例化該內部類別，然後透過內部類別的實例來存取其成員。

希望這些規則對您有幫助，並正確地反映了您所提供的資訊。

下面是針對您提到的內部類別規則的對應 Java 程式碼範例，讓您更清楚了解這些規則如何在程式中實現：

### 1. ****區域變數若要被內部類別調用，必須具備 final 特性（JDK 8 之前的規則）****

public class OuterClass {

public void methodWithInnerClass() {

final int number = 100; // 必須是 final 在 JDK 8 之前

class LocalInnerClass {

public void printNumber() {

System.out.println("The number is: " + number);

}

}

LocalInnerClass inner = new LocalInnerClass();

inner.printNumber();

}

public static void main(String[] args) {

OuterClass outer = new OuterClass();

outer.methodWithInnerClass();

}

}

### 2. ****JDK 8 之後的改變：即使變數未使用 final，但它必須是 "effectively final"****

public class OuterClass {

public void methodWithInnerClass() {

int number = 100; // 不需要宣告 final，但必須保持不變

class LocalInnerClass {

public void printNumber() {

System.out.println("The number is: " + number);

}

}

// 不能在這裡修改 number，否則會報錯

LocalInnerClass inner = new LocalInnerClass();

inner.printNumber();

}

public static void main(String[] args) {

OuterClass outer = new OuterClass();

outer.methodWithInnerClass();

}

}

### 3. ****成員內部類別（Member Inner Class）****

public class OuterClass {

private String message = "Hello from OuterClass!";

// 成員內部類別

public class InnerClass {

public void displayMessage() {

System.out.println(message); // 可以存取外部類別的 private 成員

}

}

public static void main(String[] args) {

OuterClass outer = new OuterClass();

InnerClass inner = outer.new InnerClass(); // 外部類別的實例來實例化內部類別

inner.displayMessage();

}

}

### 4. ****靜態內部類別（Static Inner Class）****

public class OuterClass {

private static String staticMessage = "Hello from Static Inner Class!";

// 靜態內部類別

public static class StaticInnerClass {

public void displayStaticMessage() {

System.out.println(staticMessage); // 只能存取外部類別的靜態成員

}

}

public static void main(String[] args) {

OuterClass.StaticInnerClass inner = new OuterClass.StaticInnerClass(); // 直接透過類別名稱實例化

inner.displayStaticMessage();

}

}

### 5. ****區域內部類別（Local Inner Class）****

public class OuterClass {

public void methodWithLocalClass() {

class LocalInnerClass {

public void show() {

System.out.println("Inside Local Inner Class");

}

}

LocalInnerClass inner = new LocalInnerClass();

inner.show();

}

public static void main(String[] args) {

OuterClass outer = new OuterClass();

outer.methodWithLocalClass();

}

}

### 6. ****匿名內部類別（Anonymous Inner Class）****

public class OuterClass {

public void createAnonymousClass() {

// 使用匿名內部類別來即時實作接口

Runnable runnable = new Runnable() {

@Override

public void run() {

System.out.println("Running from Anonymous Inner Class");

}

};

new Thread(runnable).start();

}

public static void main(String[] args) {

OuterClass outer = new OuterClass();

outer.createAnonymousClass();

}

}

### 7. ****實例化規則****

* **成員內部類別需要外部類別實例化：**

OuterClass outer = new OuterClass();

OuterClass.InnerClass inner = outer.new InnerClass(); // 外部實例來創建內部實例

inner.displayMessage();

* **靜態內部類別可以直接實例化：**

// 直接透過類別名稱實例化

OuterClass.StaticInnerClass staticInner = new OuterClass.StaticInnerClass();

staticInner.displayStaticMessage();

### 8. ****外部類別不能是 private 或 protected，不能宣告為 static****

// 無法使用 private 或 protected 修飾外部類別，且外部類別不能是 static

public class OuterClass {

// 外部類別

}

### 9. ****外部類別不能直接存取內部類別的成員****

public class OuterClass {

public class InnerClass {

private String innerMessage = "Message from InnerClass";

}

public static void main(String[] args) {

OuterClass outer = new OuterClass();

OuterClass.InnerClass inner = outer.new InnerClass();

// outer.innerMessage; // 無法直接存取內部類別的成員

System.out.println("Cannot access innerMessage directly from OuterClass.");

}

}