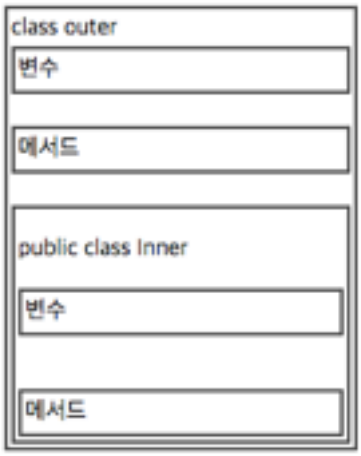
내부 클래스(Inner Class)

안쪽 클래스를 인스턴스 변수처럼 사용하기 위하여 사용한다.



바깥 클래스에서 안쪽 클래스를 사용하려면 바깥 클래스의 객체를 사용하여 안쪽 클래스를 객체화 하고 그 객체로 안쪽 클래스의 자원을 사용한다.

안쪽 클래스에서 바깥쪽 클래스의 자원 직접 사용 가능

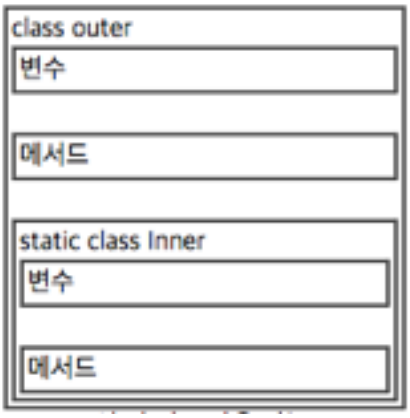
바깥쪽 클래스에서 안쪽 클래스의 자원 사용하려면 반드시 바깥 클래스에서 안쪽 클래스를 객체화해야함

Outer out = new Outer();
Outer.Inner in = out.new Inner();

```java
package day0226;
/**
* 안쪽 클래스를 인스턴스변수처럼 사용할 수 있는 InnerClass의 사용
*/
public class TestInner {
	int i;
	public TestInner() {
		System.out.println("바깥클래스 생성자");
	}// TestInner
	public void outMethod() {
		System.out.println("바깥클래스의 메서드");
	}
	// ///////////Inner Class의 시작//////////////
	public class Inner {
		int j;
		public Inner() {
			System.out.println("안쪽클래스의 생성자");
		}// Inner
		public void inMethod() {
			System.out.println("안쪽클래스의 메서드");
			System.out.println("안쪽메서드 바깥변수 i="+i);
		}
	}// class
	// ///////////Inner Class의 끝//////////////
	public static void main(String[] args) {
		TestInner ti = new TestInner();
		ti.i = 10;
		// ti.j=10;//안쪽 클래스의 변수는 직접 사용 불가능
		// ti.inMethod(); //안쪽 클래스의 메서드는 직접 사용 불가능
		// 바깥클래스에서 안쪽클래스의 자원을 사용하려면 안쪽클래스를
		// 객체화하여 사용
		TestInner.Inner in = ti.new Inner();
		// 안쪽 클래스의 자원 사용
		in.j = 100;
		in.inMethod();
	}// main
}// class
```

중첩 클래스 (Nested Class)

안쪽 클래스를 static 변수처럼 사용
안쪽 클래스 정의시 static 붙여서 정의
바깥 클래스 instance 영역 사용 불가
안쪽 클래스 객체화 하지 않고 사용 가능 (static)




```java
package day0226;
/**
* 안쪽 클래스를 static 변수처럼 사용하는 중첩 클래스의 사용
*
*/
public class TestNested {
	int i;
	static int j;
	public TestNested(){
		System.out.println("바깥클래스의 생성자");
	}//TestNested
	public void outMethod(){
		System.out.println("인스턴스메서드");
	}//outMethod
	public static void outStaticMethod(){
		System.out.println("바깥 클래스의 스태틱메서드");
	}//outstaticMethod
	////////////중첩클래스 시작/////////////
	static class Nested{
	static int k;
	public Nested(){
		System.out.println("안쪽생성자");
	}//Nested
	public static void inMethod(){
		//i=10; //static영역에서는 instance 영역의 자원을 직접 사용할 수 없음.
		System.out.println("안쪽 메서드");
		TestNested.j=10; // = "j=10;"
		//outMethod(); //static영역에서는 instance 영역의 자원을 직접 사용할 수 없음.
		TestNested.outStaticMethod(); // = outStaticMethod();
	}//inMethod
	}//class
	////////////중첩클래스 끝/////////////
	public static void main(String[] args) {
		//안쪽클래스의 자원 사용
		Nested.k=10;
		Nested.inMethod();
	}
}
```