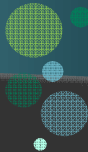


제13장

중첩 클래스



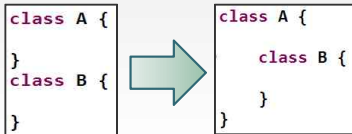
1. 중첩(내부) 클래스 – inner(nested) class

▣ 중첩 클래스의 개념

- 클래스 안에 **멤버로써 선언되어진 클래스**를 칭한다.
- 해당 클래스 내에서만 사용되어질 클래스를 중첩클래스로 선언한다.(외부에서 사용 안함)
- **GUI애플리케이션(AWT, SWING, 안드로이드 프로그래밍 등)의 이벤트 처리에서 주로 사용된다. 또한 웹애플리케이션에서 많이 사용된다.**

▣ 중첩 클래스의 장점

- 중첩클래스에서 외부클래스의 멤버에 쉽게 접근이 가능하다.
- 코드의 복잡성을 줄일 수 있다.(캡슐화)



2. 중첩클래스의 종류와 특징

■ 중첩클래스의 종류는 변수의 선언위치에 따른 종류와 동일하다.

■ 유효범위 및 성질도 변수와 유사하다.

선언 위치에 따른 분류		선언 위치	설명
멤버 클래스	인스턴스 멤버 클래스	class A { class B { ... } }	A 객체를 생성해야만 사용할 수 있는 B 중첩 클래스
	정적 멤버 클래스	class A { static class B { ... } }	A 클래스로 바로 접근할 수 있는 B 중첩 클래스 (인스턴스 생성 없이도 접근 가능)
로컬 클래스 (메서드내 선언)		class A { void method() { class B { ... } } }	method()가 실행할 때만 사용할 수 있는 B 중첩 클래스 (static 절대 못붙임!)

■ 중첩 클래스의 바이트 코드 파일

A \$ B . class → 외부클래스 \$ 멤버 클래스, A \$1 B .class → 외부 클래스 \$1 로컬 클래스
(멤버클래스와 로컬 클래스는 순번으로 구분을 짓는다.)

3. 인스턴스 멤버 클래스

```
public class A {  
    //인스턴스 멤버 클래스  
    class B {  
        int iv = 100;  
        static int cv = 500;  
  
        //인스턴스 생성과 무관함 하여, 멤버로 사용가능  
        static final int CONST = 1;  
  
        public B() {}  
  
        public void method1() {}  
  
        public static void method2() {}  
    }  
}
```

```
public static void main(String[] args) {  
    //외부클래스 생성함.  
    A a = new A();  
    //인스턴스 멤버클래스 생성함.  
    A.B ic = a.new B();  
    System.out.println(ic.iv);  
    ic.method1();  
}
```

외부 클래스가 생성되어야, 내부클래스를 사용할 수가 있다.

*** static은 인스턴스 생성 없이도 접근이 가능해야 하므로, 인스턴스 멤버클래스에는 static 멤버변수나 메서드로 선언이 불가능에 주목하자.

언제 인스턴스가 생성될지는 아무도 모른다.==> 인스턴스 멤버 클래스에는 static 멤버가 올 수가 없다. (상식)

항상 A의 인스턴스가 생성되어야 함.

A.B b = a.new B()로 생성한다.

4. 정적 멤버 클래스

```
public class A {  
    //정적멤버클래스  
    static class C{  
        //인스턴스 멤버변수 사용가능  
        int iv = 100;  
        static int cv = 500;  
  
        public C() {}  
        //인스턴스 멤버메서드 사용가능  
        public void method1() {}  
  
        public static void method2() {}  
    }  
}
```

```
public static void main(String[] args) {  
    //외부클래스 생성하지 않고 접근 가능함.  
    System.out.println(A.C.cv);  
    A.C.method2();  
    A.C c = new C();  
    //인스턴스 생성 후, 인스턴스멤버 접근 가능함  
    System.out.println(c.iv);  
    c.method1();  
}
```

외부 클래스 A가 생성되지 않아도
접근 가능하다.

*** 인스턴스 멤버변수인 iv와
method1()은 new연산자를 통해
인스턴스가 만들어져야만 접근 가능하다
라는 점에 주목을 하자.

* **static** 키워드로 선언된 클래스,
모든 종류의 필드, 메서드 선언 가능

5. 로컬 클래스 – 메서드 내 작성 후 사용

```
public void method() {  
    class D{  
        public D() {+  
            int lv;  
            static int lcv;  
            public void method1() {}  
            public static method2() {}  
        }  
        D d = new D();  
        d.lv = 3;  
        d.method1();  
    }
```

thread클래스내 run이라는 메서드가 있어서 자동 실행됨. thread에 자주 로컬 클래스 개념이 등장한다.

ex) 다운로드, 윈도우 업데이트 등

메서드내에 선언된 클래스를 로컬클래스라 한다. 외부에서는 접근 불가.

메서드내에서 잠깐 사용할 용도로 만듬.
(일회성 용도)

*** static 멤버는 추가할 수 없음.
그 이유는 인스턴스를 생성해야만 접근할 수 있으므로 static을 선언할 수 없는 것이다.

```
public void download() {  
    class Download extends Thread{  
        /* 다운로드 코드 */  
    }  
    Download d = new Download();  
    d.start();  
}
```

6. 중첩 클래스의 접근 제한-1

▣ 외부 클래스의 멤버의 사용 제한

- 외부 클래스의 멤버인 인스턴스 멤버들은 인스턴스나 static멤버를 new연산자로 생성 가능하나, static은 static만 허락하므로 인스턴스 멤버는 생성하는 것이 불가능하다.
- 이유 : 인스턴스 멤버는 항상 외부클래스의 인스턴스가 존재해야 하기 때문이다.

(수 차례 언급!)

```
public class A {  
    //인스턴스 멤버클래스  
    class B{}  
  
    //정적멤버클래스  
    static class C{}  
}
```

```
public class A {  
    //인스턴스 필드  
    B b = new B(); //멤버변수  
    C c = new C(); //멤버변수  
  
    public void method() {  
        B lb = new B(); //지역변수  
        C lc = new C(); //지역변수  
    }  
}
```

```
static B b = new B(); //멤버변수  
static C c = new C(); //멤버변수  
  
public static void method() {  
    B lb = new B(); //지역변수  
    C lc = new C(); //지역변수  
}
```

6. 중첩 클래스의 접근 제한-2

▣ 멤버 클래스에서의 사용 제한

```
public class A {  
    int iv;  
    static int cv;  
  
    public void imethod() {}  
    public static void smethod() {}
```

```
//인스턴스 멤버클래스  
class B{           인스턴스 멤버클래스 B
```

```
    public void method() {  
        //외부 멤버들 전부 접근가능함  
        iv = 10;  
        imethod();  
  
        cv = 20;  
        smethod();  
    }  
}
```

**중첩클래스 B가 사용하려고 할 시
점에는 이미 외부클래스 A가 인스
턴스로 생성되어 있을 것이다.**

```
public class A {  
    int iv;  
    static int cv;  
  
    public void imethod() {}  
    public static void smethod() {}
```

```
//static 멤버클래스  
static class C{ static 멤버클래스 C
```

```
    public void method() {  
        //static은 static만 접근 가능  
        iv = 10;  
        imethod();  
  
        cv = 20;  
        smethod();  
    }  
}
```

**static은 반드시 static만 접근가
능하다.**

**외부클래스의 멤버들을
자유롭게 접근할 수
있는 이유로 중첩클래스
를 자주 사용하곤 한다.**

6. 중첩 클래스의 접근 제한-3

▣ 내부 클래스에서 외부 클래스 참조하는 방법(UI이벤트 처리, 안드로이드 P/G에서 많이 등장)

```
public class Outside {
    String str = "Outside";

    public void method() {
        System.out.println("Outside-method");
    }
    class Inner{
        String str = "Inner";

        public void method() {
            System.out.println("Inner-method");
        }
        public void show() {
            //중첩클래스 참조
            System.out.println(this.str);
            this.method();
            //외부클래스 참조
            System.out.println(Outside.this.str);
            Outside.this.method();
        }
    }
}
```

1. **this**는 자기자신을 가리킨다는 것은 너무나 잘 알고 있다.
2. 외부클래스의 참조를 얻으려면 무조건 명시적으로 외부클래스명.this.멤버명으로 접근해야 한다.

감사합니다.

