



9장. 중첩 클래스와 중첩 인터페이스

이것이 자바다 (<http://cafe.naver.com/thisjava>)

Contents

- ❖ 1절. 중첩 클래스와 중첩 인터페이스란?
- ❖ 2절. 중첩 클래스
- ❖ 3절. 중첩 클래스의 접근 제한
- ❖ 4절. 중첩 인터페이스
- ❖ 5절. 익명 객체



1절. 중첩 클래스와 중첩 인터페이스란

❖ 중첩 클래스와 중첩 인터페이스란?

- 중첩 클래스: 클래스 멤버로 선언된 클래스

```
class ClassName {  
    class NestedClassName {  
    }  
}
```

중첩 클래스

- 중첩 인터페이스: 클래스 멤버로 선언된 인터페이스
 - UI 컴포넌트 내부 이벤트 처리에 많이 활용

```
class ClassName {  
    interface NestedInterfaceName {  
    }  
}
```

중첩 인터페이스



2절. 중첩 클래스

❖ 중첩 클래스의 분류

선언 위치에 따른 분류		선언 위치	설명
멤버 클래스	인스턴스 멤버 클래스	<pre>class A { class B { ... } }</pre>	A 객체를 생성해야만 사용할 수 있는 B 중첩 클래스
	정적 멤버 클래스	<pre>class A { static class B { ... } }</pre>	A 클래스로 바로 접근할 수 있는 B 중첩 클래스
로컬 클래스		<pre>class A { void method() { class B { ... } } }</pre>	method()가 실행할 때만 사용할 수 있는 B 중첩 클래스

- 클래스 생성시 바이트 코드 따로 생성 (p.391)



2절. 중첩 클래스

❖ 인스턴스 멤버 클래스

```
class A {  
    /**인스턴스 멤버 클래스**/  
    class B {  
        B() { }                -----생성자  
        int field1;             -----인스턴스 필드  
        //static int field2;    -----정적 필드 (x)  
        void method1() { }      -----인스턴스 메소드  
        //static void method2() { } -----정적 메소드 (x)  
    }  
}
```

```
A  a = new A();  
A.B b = a.new B();  
b.field1 = 3;  
b.method1();
```



2절. 중첩 클래스

❖ 정적 멤버 클래스

- **static** 키워드로 선언된 클래스, 모든 종류의 필드, 메소드 선언 가능

```
class A {  
    /**정적 멤버 클래스**/  
    static class C {  
        C() {}                -----생성자  
        int field1;            -----인스턴스 필드  
        static int field2;     -----정적 필드  
        void method1() {}     -----인스턴스 메소드  
        static void method2() {} -----정적 메소드  
    }  
}
```

```
A.C c = new A.C();  
c.field1 = 3;    //인스턴스 필드 사용  
c.method1();    //인스턴스 메소드 호출  
A.C.field2 = 3; //정적 필드 사용  
A.C.method2();  //정적 메소드 호출
```



2절. 중첩 클래스

❖ 로컬 클래스 – 메소드 내에서만 사용

```
void method() {  
    /**로컬 클래스**/  
    class D {  
        D() {}                -----생성자  
        int field1;           -----인스턴스 필드  
        //static int field2;   -----정적 필드(x)  
        void method1() {}     -----인스턴스 메소드  
        //static void method2() {} -----정적 메소드(x)  
    }  
    D d = new D();  
    d.field1 = 3;  
    d.method1();  
}
```

```
void method() {  
    class DownloadThread extends Thread { ... }  
    DownloadThread thread = new DownloadThread();  
    thread.start();  
}
```

3절. 중첩 클래스의 접근 제한

❖ 바깥 필드와 메소드에서 사용 제한 (p.396)

```
public class A {  
    //인스턴스 멤버 클래스  
    class B {}  
  
    //정적 멤버 클래스  
    static class C {}  
}
```

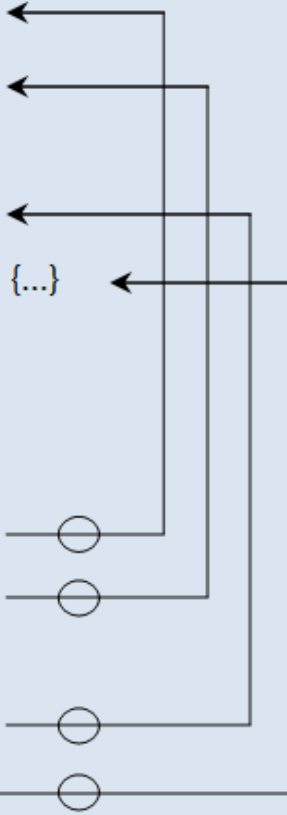
```
public class A {  
    //인스턴스 필드  
    B field1 = new B();      ----- (o)  
    C field2 = new C();      ----- (o)  
  
    //인스턴스 메소드  
    void method1() {  
        B var1 = new B();    ----- (o)  
        C var2 = new C();    ----- (o)  
    }  
}
```

```
//정적 필드 초기화  
//static B field3 = new B();      ----- (x)  
static C field4 = new C();      ----- (o)  
  
//정적 메소드  
static void method2() {  
    //B var1 = new B();          ----- (x)  
    C var2 = new C();          ----- (o)  
}
```

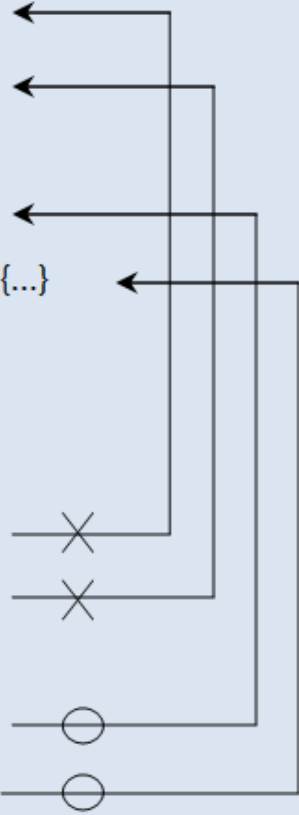

3절. 중첩 클래스의 접근 제한

❖ 멤버 클래스에서 사용 제한 (p.397)

```
class A {  
    int field1;  
    void method1() { ...}  
  
    static int field2;  
    static void method2() {...}  
  
    class B {  
        void method() {  
            field1 = 10;  
            method1();  
  
            field2 = 10;  
            method2();  
        }  
    }  
}
```



```
class A {  
    int field1;  
    void method1() { ...}  
  
    static int field2;  
    static void method2() {...}  
  
    static class C {  
        void method() {  
            field1 = 10;  
            methodA1();  
  
            fieldA2 = 10;  
            methodA2();  
        }  
    }  
}
```



3절. 중첩 클래스의 접근 제한

❖ 로컬 클래스에서 사용 제한 (p.398~400)

```
public class Outer {  
    //자바7 이전  
    public void method1(final int arg) {  
        final int localVariable = 1;  
        //arg = 100; (x)  
        //localVariable = 100; (x)  
        class Inner {  
            public void method() {  
                int result = arg + localVariable;  
            }  
        }  
    }  
}
```

final 매개변수와 로컬 변수는
로컬 클래스의 메소드의 로컬변수로 복사
(final 붙이지 않으면 컴파일 오류 발생)

```
//자바8 이후  
public void method2(int arg) {  
    int localVariable = 1;  
    //arg = 100; (x)  
    //localVariable = 100; (x)  
    class Inner {  
        public void method() {  
            int result = arg + localVariable;  
        }  
    }  
}
```

매개변수와 로컬 변수는 final 특성을 가지며,
로컬 클래스의 필드로

3절. 중첩 클래스의 접근 제한

❖ 중첩 클래스에서 바깥 클래스 참조 얻기

```
public class Outter {  
    String field = "Outter-field";  
    void method() {  
        System.out.println("Outter-method");  
    }  
  
    class Nested {  
        String field = "Nested-field";  
        void method() {  
            System.out.println("Nested-method");  
        }  
        void print() {  
            System.out.println(this.field);  
            this.method();  
            System.out.println(Outter.this.field);  
            Outter.this.method();  
        }  
    }  
}
```

● ----- 중첩 객체 참조

● ----- 바깥 객체 참조



4절. 중첩 인터페이스

❖ 중첩 인터페이스 선언 (p.401~403)

```
public class Button {
```

```
    OnClickListener listener;
```

인터페이스 타입 필드

```
    void setOnClickListener(OnClickListener listener) {
```

매개변수의 다형성

```
        this.listener = listener;
```

```
    }
```

```
    void touch() {
```

```
        listener.onClick();
```

구현 객체의 onClick() 메서드 호출

```
    }
```

```
    interface OnClickListener {
```

중첩 인터페이스

```
        void onClick();
```

```
    }
```

```
}
```



5절. 익명 객체

❖ 익명 객체: 이름이 없는 객체

- 익명 객체는 단독 생성 불가

- 클래스 상속하거나 인터페이스 구현해야만 생성 가능

- 사용 위치

- 필드의 초기값, 로컬 변수의 초기값, 매개변수의 매개값으로 주로 대입
- UI 이벤트 처리 객체나, 스레드 객체를 간편하게 생성할 목적으로 주로 활용

❖ 익명 자식 객체 생성 - 초기값 설정에 주목

```
class A {  
    Parent field = new Parent() {  
        int childField;  
        void childMethod() { }  
        @Override  
        void parentMethod() { }  
    };  
}
```

A 클래스의 필드 선언

Parent 의 메소드를 오버라이딩

5절. 익명 객체

❖ 익명 객체에 새롭게 정의된 필드와 메소드

- 익명 객체 내부에서만 사용
- 외부에서는 익명 객체의 필드와 메소드에 접근할 수 없음
 - 이유: 익명 객체는 부모 타입 변수에 대입되므로 부모 타입에 선언된 것만 사용 가능

```
class A {  
    Parent field = new Parent() {  
        int childField; ←  
        void childMethod() { } ←  
        @Override  
        void parentMethod() { ←  
            childField = 3;  
            childMethod();  
        }  
    };  
  
    void method() {  
        field.childField = 3; ←  
        field.childMethod(); ←  
        field.parentMethod(); ←  
    }  
}
```



5절. 익명 객체

❖ 익명 구현 객체 생성

```
인터페이스 [필드|변수] = new 인터페이스() {  
    //인터페이스에 선언된 추상 메소드의 실제 메소드 선언  
    //필드  
    //메소드  
};
```

- 초기값 설정에 대해서는 p. 409~ 416 참고

