제 28 강

내부클래스

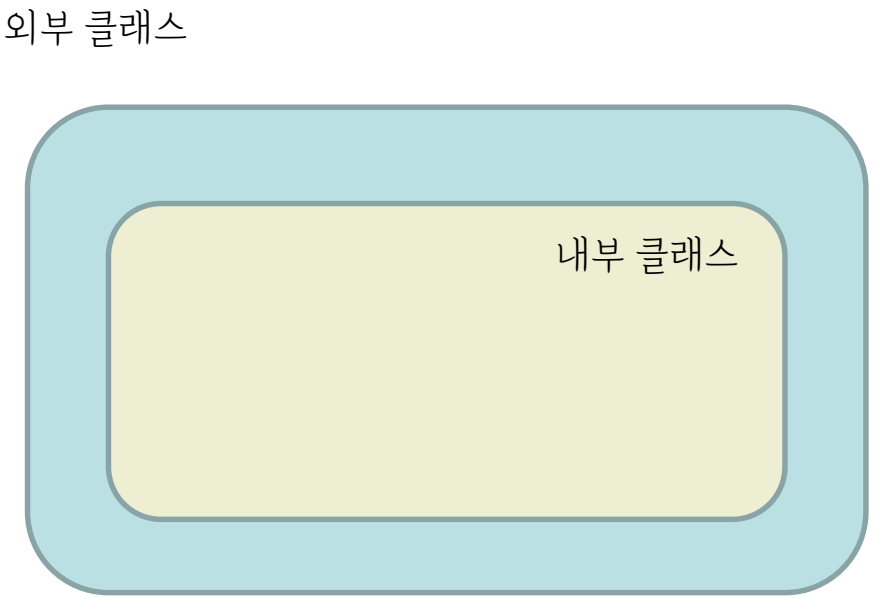
교재:p151~160

목차

1. 내부클래스의 정의

내부클래스(Inner Class)란?

: 클래스 내부에서 생성된 클래스



1. 내부클래스의 정의

내부클래스(Inner Class)의 구조

```
Class OuterClass{

        class InnerClass{
            …
                                    }

}
```

내부클래스(Inner Class)의 사용목적

: 클래스간의 관계가 긴밀할 때 사용하여 코드를 간결하게 하기 위해 사용

2. 내부클래스의 종류

"내부 클래스는 선언된 위치에 따라 인스턴스 클래스, 스태틱 클래스, 지역 클래스, 익명 클래스로 구분"

1. 인스턴스 클래스

- 외부 클래스에서 멤버 변수 위치에 선언
- 클래스의 내부에서 인스턴스(객체)멤버처럼 다뤄지며, 인스턴스 멤버들과 상호작용 가능

<구조>

```
Class OuterClass{

        class InnerClass{
                …
                                        }

}
```

2. 내부클래스의 종류

"내부 클래스는 선언된 위치에 따라 인스턴스 클래스, 스태틱 클래스, 지역 클래스, 익명 클래스로 구분"

2. 스태틱(static) 클래스

- 외부 클래스에서 멤버 변수 위치에 선언
- 클래스 내부에서 static 멤버처럼 다뤄지며 static 멤버들과 상호작용 가능

<구조>

```
Class OuterClass{

        static class InnerClass{
                ...
                                        }

}
```

2. 내부클래스의 종류

"내부 클래스는 선언된 위치에 따라 인스턴스 클래스, 스태틱 클래스, 지역 클래스, 익명 클래스로 구분"

3. 지역 클래스

- 메서드 내부 지역변수 위치에 선언
- 메서드나 초기화 블록의 내부에서 다뤄지며 선언된 영역 내부에서 사용

<구조>

```
Class OuterClass{
        void A(){
                class InnerClass{
                …
                                        }
                }
}
```

3. 내부 클래스의 접근성

“ 내부 클래스도 클래스이기 때문에 선언부에 접근제어자를 사용 할 수 있다.”

<구조>

```
Class OuterClass{
        private class InnerClass{
                ...        }
        protected class InnerClass{
                ...        }

}
```

3. 내부 클래스의 접근성

<실습> Exam-57.java

Static 클래스와 인스턴스 클래스 실습

```
class A{
    static class StaticInner{

    }
    class InstanceInner{

    }
```

Static 클래스

인스턴스 클래스

1) 두 클래스 모두 객체 생성 가능

```
StaticInner st1=new StaticInner();
InstanceInner ii1 = new InstanceInner();
```

2) static 메서드 내 인스턴스 클래스 접근 불가

```
static void StaticMethod() {
    StaticInner st2=new StaticInner();
    InstanceInner ii2=new InstanceInner(); 오류발생
}
```

3) 인스턴스 메서드는 스태틱 클래스, 인스턴스 클래스 모두 접근 가능

```
void InstanceMethod(){
    StaticInner  st3=new StaticInner();
    InstanceInner ii3=new InstanceInner();
}
```

3. 내부 클래스의 접근성

<실습> Exam-58.java

클래스의 종류에 따라 관계 알아보는 실습

```
class OuterClass{
    int a;
    static int b=4;
    class Inner{
        int c=5;
        public void InnerMethod() {
            System.out.println("<Inner Class>");
        }
    }
    static class StaticInner{
        int d=6;
        static int stat=10;

        public static void staticMethod() {
            System.out.println("<Static Inner>");
        }
    }

}
```

OuterClass의 구조

<필드>
- 인스턴스 변수: a
- static 변수: b

<내부 클래스>
- 인스턴스 클래스: Inner
- static 클래스: StaticInner

3. 내부 클래스의 접근성

<실습> Exam-58.java

클래스의 종류에 따라 관계 알아보는 실습

```
public class Inner4 {
    public static void main(String[] args) {
        OuterClass oc = new OuterClass();
        System.out.println("OuterClass의 a 값: "+oc.a);
        System.out.println("OuterClass의 b 값: "+OuterClass.b);

        System.out.println("===inner 클래스 접근하기===");
        OuterClass oc2 = new OuterClass();
        OuterClass.Inner i = oc2.new Inner();
        System.out.println("Inner의 c 값: "+i.c);
        i.InnerMethod();

        OuterClass.StaticInner si = new OuterClass.StaticInner();
        System.out.println("StaticInner의 d 값: "+si.d);

        si.staticMethod();
        OuterClass.StaticInner.staticMethod();
    }
}
```

1. OuterClass 객체 생성 후 멤버에 접근

- 인스턴스 멤버 접근: 인스턴스명.변수명
- static 멤버 접근: 클래스명.변수명;

3. 내부 클래스의 접근성

<실습> Exam-58.java

클래스의 종류에 따라 관계 알아보는 실습

```
public class Inner4 {
    public static void main(String[] args) {
        OuterClass oc = new OuterClass();
        System.out.println("OuterClass의 a 값: "+oc.a);
        System.out.println("OuterClass의 b 값: "+OuterClass.b);

        System.out.println("===inner 클래스 접근하기===");
        OuterClass oc2 = new OuterClass();
        OuterClass.Inner i = oc2.new Inner();
        System.out.println("Inner의 c 값: "+i.c);
        i.InnerMethod();

        OuterClass.StaticInner si = new OuterClass.StaticInner();
        System.out.println("StaticInner의 d 값: "+si.d);

        si.staticMethod();
        OuterClass.StaticInner.staticMethod();
    }
}
```

2. 인스턴스 내부 클래스 접근

- 외부클래스 객체 생성
  OuterClass oc2 = new OuterClass( );

- 외부클래스 객체로 내부 클래스 객체 생성
  OuterClas.inner i = oc2.new Inner();

- 내부 클래스 객체로 메서드 호출
  i.InnnerMethod( );

3. 내부 클래스의 접근성

<실습> Exam-58.java

클래스의 종류에 따라 관계 알아보는 실습

```
public class Inner4 {
    public static void main(String[] args) {
        OuterClass oc = new OuterClass();
        System.out.println("OuterClass의 a 값: "+oc.a);
        System.out.println("OuterClass의 b 값: "+OuterClass.b);

        System.out.println("===inner 클래스 접근하기===");
        OuterClass oc2 = new OuterClass();
        OuterClass.Inner i = oc2.new Inner();
        System.out.println("Inner의 c 값: "+i.c);
        i.InnerMethod();

        OuterClass.StaticInner si = new OuterClass.StaticInner();
        System.out.println("StaticInner의 d 값: "+si.d);

        si.staticMethod();
        OuterClass.StaticInner.staticMethod();
    }
}
```

스태틱 멤버는 객체를 생성하지 않고도 클래스명으로 호출 가능

3. 스태틱 내부 클래스 접근

- 스태틱 클래스 객체 생성
  OuterClass.StaticInner si = …
- 스태틱 클래스로 스태틱 메서드 호출
  si.staticMethod( );

4. 익명 클래스

"익명 클래스 또는 무명 클래스라고도 하며, 이름이 없는 클래스를 의미"

익명클래스(Anonymous)

: 이름이 없는 클래스

기존 클래스를 오버라이딩해서 만든다!

<기존 클래스의 구조>

```
class Some{
        private int a=3;
        int getter( ){
                return this.a;
}
        void setter(int a){
                this.a=a;
                        }
}
```

<익명 클래스의 구조>

```
Some annony=new Some(){
        private int a=3;
        int getter( ){
                return this.a;
}
        void setter(int a){
                this.a=a;
                        }
};
```

4. 익명 클래스

<실습> Exam-59.java

<기존 클래스의 구조>

```
class OuterClass1{
    void a() {
        System.out.println("method a");
    }
    void b(){    }
}
```

<기존 클래스를 오버라이딩한 익명 클래스>

```
public class Anonymous2 {
    public static void main(String[] args) {
        OuterClass1 o = new OuterClass1() {
            void a() {
                System.out.println("새롭게 정의한 익명클래스 메서드");
            }
        };
        o.a();
        OuterClass1 ok=new OuterClass1();
        ok.a();
    }
}
```

익명클래스는 일회성 이므로, 객체를 다시 생성하고 호출하면

OuterClass1에 정의된 메서드가 호출

→

<출력결과>

```
새롭게 정의한 익명클래스 메서드
method a
```

4. 익명 클래스

"익명클래스는 인터페이스 혹은 추상클래스를 일회성으로 구현할 때 많이 사용하므로
인터페이스와 추상클래스를 구현하는 익명클래스를 생성해보기"

<실습> Annonymous3.java

```
public class Anonymous3 {
    public static void main(String[] args) {
//      Inter1 i1=new Inter1(); 인터페이스 객체 생성 불가
        Inter1 i1= new Inter1() {
            public void f1() {
                System.out.println("f1()");
            }
        };
        i1.f1();
    }
}

interface Inter1{
    public void f1();
}
```

4. 익명 클래스

"익명클래스는 인터페이스 혹은 추상클래스를 일회성으로 구현할 때 많이 사용하므로
인터페이스와 추상클래스를 구현하는 익명클래스를 생성해보기"

<실습> Annonymous4.java

```
public class Anonymous4 {
    public static void main(String[] args) {
//      Abstract1 ab1 = new Abstract1();  추상클래스 객체 생성 불가
        Abstract1 ab1 = new Abstract1() {
            void f2() {
                System.out.println("f2( )");
            }};
    }
}

abstract class Abstract1{
    abstract void f2();
}
```