Java의 정석

제 10 장

내부 클래스

1. 내부 클래스(inner class)

1.1 내부 클래스(inner class)란?

1.2 내부 클래스의 종류와 특징

1.3 내부 클래스의 제어자와 접근성

1.4 익명 클래스(anonymous class)

1. 내부 클래스(inner class)

1.1 내부 클래스(inner class)란?

- 클래스 안에 선언된 클래스
- 특정 클래스 내에서만 주로 사용되는 클래스를 내부 클래스로 선언한다.
- GUI어플리케이션(AWT, Swing)의 이벤트처리에 주로 사용된다.

```
class A {
    //...
}

class B {
    //...
}
```



```
class A {  // 외부 클래스
    //...
    class B { // 내부 클래스
        //...
    }
    //...
}
```

▶ 내부 클래스의 장점

- 내부 클래스에서 외부 클래스의 멤버들을 쉽게 접근할 수 있다.
- 코드의 복잡성을 줄일 수 있다.(캡슐화)

1.2 내부 클래스의 종류와 특징

- 내부 클래스의 종류는 변수의 선언위치에 따른 종류와 동일하다.
- 유효범위와 성질도 변수와 유사하므로 비교해보면 이해하기 쉽다.

내부 클래스	특 징
인스턴스 클래스 (instance class)	외부 클래스의 멤버변수 선언위치에 선언하며, 외부 클래스의 인스턴스멤버처럼 다루어진다. 주로 외부 클래스의 인스턴스멤버들과 관련된 작업에 사용될 목적으로 선언된다.
스태틱 클래스 (static class)	외부 클래스의 멤버변수 선언위치에 선언하며, 외부 클래스의 static멤버처럼 다루어진다. 주로 외부 클래스의 static멤버, 특히 static메서드에서 사용될 목적으로 선언된다.
지역 클래스 (local class)	외부 클래스의 메서드나 초기화블럭 안에 선언하며, 선언된 영역 내부에서만 사용될 수 있다.
익명 클래스 (anonymous class)	클래스의 선언과 객체의 생성을 동시에 하는 이름없는 클래스(일회용)

```
class Outer {
    int iv=0;
    static int cv=0;

    void myMethod() {
        int lv=0;
    }
}
```

```
class Outer {
    class InstanceInner {}
    static class StaticInner {}

    void myMethod() {
        class LocalInner {}
    }
}
```

1.3 내부 클래스의 제어자와 접근성(1/5)

- 내부 클래스의 접근제어자는 변수에 사용할 수 있는 접근제어자와 동일하다.

```
class Outer {
    private int iv=0;
    protected static int cv=0;

    void myMethod() {
        int lv=0;
    }
}
```



```
class Outer {
    private class InstanceInner {}
    protected static class StaticInner {}

    void myMethod() {
        class LocalInner {}
    }
}
```

- static클래스만 static멤버를 정의할 수 있다.

```
class InnerEx1 {
    class InstanceInner {
        int iv = 100;
//      static int cv = 100;          // 에러! static변수를 선언할 수 없다.
        final static int CONST = 100; // final static은 상수이므로 허용한다.
    }

    static class StaticInner {
        int iv = 200;
        static int cv = 200;
    }

    void myMethod() {
        class LocalInner {
            int iv = 300;
//          static int cv = 300;          // 에러! static변수를 선언할 수 없다.
            final static int CONST = 300; // final static은 상수이므로 허용
        }
    } // void myMethod() {
}
```

```
class InnerTest {
    public static void main(String args[]) {
        System.out.println(InnerEx1.InstanceInner.CONST);
        System.out.println(InnerEx1.StaticInner.cv);
    }
}
```

1.3 내부 클래스의 제어자와 접근성(2/5)

- 내부 클래스도 외부 클래스의 멤버로 간주되며, 동일한 접근성을 갖는다.

```
class InnerEx2 {
   class InstanceInner {}
   static class StaticInner {}

   InstanceInner iv = new InstanceInner();    // 인스턴스멤버 간에는 서로 직접 접근이 가능하다.
   static StaticInner cv = new StaticInner(); // static 멤버 간에는 서로 직접 접근이 가능하다.

   static void staticMethod() {
//     InstanceInner obj1 = new InstanceInner(); // static멤버는 인스턴스멤버에 직접 접근할 수 없다.
       StaticInner obj2 = new StaticInner();

       // 굳이 접근하려면 아래와 같이 객체를 생성해야한다.
       InnerEx2 outer = new InnerEx2();
       InstanceInner obj1 = outer.new InstanceInner();
   }

   void instanceMethod() {
       InstanceInner obj1 = new InstanceInner();
       StaticInner obj2 = new StaticInner();
//     LocalInner lv = new LocalInner();
   }

   void myMethod() {
       class LocalInner {}
       LocalInner lv = new LocalInner();
   }
}
```

인스턴스클래스는 외부 클래스를 먼저 생성해야만 생성할 수 있다.

인스턴스메서드에서는 인스턴스멤버와 static멤버 모두 접근 가능하다.

메서드 내에 지역적으로 선언된 내부 클래스는 외부에서 접근할 수 없다.

1.3 내부 클래스의 제어자와 접근성(3/5)

- 외부 클래스의 지역변수는 final이 붙은 변수(상수)만 접근가능하다.
  지역 클래스의 인스턴스가 소멸된 지역변수를 참조할 수 있기 때문이다.

```
class InnerEx3 {
   private int outerIv = 0;
   static int outerCv = 0;

   class InstanceInner {
        int iiv = outerIv;  // 외부 클래스의 private멤버도 접근가능하다.
        int iiv2 = outerCv;
   }

   static class StaticInner {
// 스태틱 클래스는 외부 클래스의 인스턴스멤버에 접근할 수 없다.
//      int siv = outerIv;
        static int scv = outerCv;
   }

   void myMethod() {
        int lv = 0;
        final int LV = 0;

        class LocalInner {
             int liv = outerIv;
             int liv2 = outerCv;
// 외부 클래스의 지역변수는 final이 붙은 변수(상수)만 접근가능하다.
//           int liv3 = lv;    // 에러!!!
             int liv4 = LV;    // OK
        }
   }
}
```

1.3 내부 클래스의 제어자와 접근성(4/5)

```
class Outer {
   class InstanceInner {
        int iv=100;
   }

   static class StaticInner {
        int iv=200;
        static int cv=300;
   }

   void myMethod() {
        class LocalInner {
             int iv=400;
        }
   }
}
```

```
InnerEx4.class
Outer.class
Outer$InstanceInner.class
Outer$StaticInner.class
Outer$1LocalInner.class
```

```
class InnerEx4 {
   public static void main(String[] args) {
        // 인스턴스클래스의 인스턴스를 생성하려면
        // 외부 클래스의 인스턴스를 먼저 생성해야한다.
        Outer oc = new Outer();
        Outer.InstanceInner ii = oc.new InstanceInner();

        System.out.println("ii.iv : "+ ii.iv);
        System.out.println("Outer.StaticInner.cv : "+ Outer.StaticInner.cv);

        // 스태틱 내부 클래스의 인스턴스는 외부 클래스를 먼저 생성하지 않아도 된다.
        Outer.StaticInner si = new Outer.StaticInner();
        System.out.println("si.iv : "+ si.iv);
   }
}
```

1.3 내부 클래스의 제어자와 접근성(5/5)

```
class Outer {
   int value=10;    // Outer.this.value

   class Inner {
        int value=20;    // this.value
        void method1() {
            int value=30;
            System.out.println("           value :" + value);
            System.out.println("      this.value :" + this.value);
            System.out.println("Outer.this.value :" + Outer.this.value);
        }
   } // Inner클래스의 끝
} // Outer클래스의 끝

class InnerEx5 {
   public static void main(String args[]) {
        Outer outer = new Outer();
        Outer.Inner inner = outer.new Inner();
        inner.method1();
   }
} // InnerEx5 끝
```

[실행결과]

```
           value :30
      this.value :20
Outer.this.value :10
```

1.4 익명 클래스(anonymous class)

- 이름이 없는 일회용 클래스. 단 하나의 객체만을 생성할 수 있다.

```
new 조상클래스이름() {
   // 멤버 선언
}

   또는

new 구현인터페이스이름() {
   // 멤버 선언
}
```

[예제10-6]/ch10/InnerEx6.java

```
class InnerEx6 {
   Object iv = new Object(){ void method(){} };         // 익명클래스
   static Object cv = new Object(){ void method(){} };// 익명클래스

   void myMethod() {
       Object lv = new Object(){ void method(){} };    // 익명클래스
   }
}
```

```
InnerEx6.class
InnerEx6$1.class ← 익명클래스
InnerEx6$2.class ← 익명클래스
InnerEx6$3.class ← 익명클래스
```

1.4 익명 클래스(anonymous class) - 예제

```
import java.awt.*;
import java.awt.event.*;

class  InnerEx7{
   public static void main(String[] args) {
        Button b = new Button("Start");
        b.addActionListener(new EventHandler());
   }
}

class EventHandler implements ActionListener {
   public void actionPerformed(ActionEvent e) {
        System.out.println("ActionEvent occurred!!!");
   }
}
```

```
import java.awt.*;
import java.awt.event.*;

class  InnerEx8 {
   public static void main(String[] args) {
        Button b = new Button("Start");
        b.addActionListener(new ActionListener() {
                public void actionPerformed(ActionEvent e) {
                    System.out.println("ActionEvent occurred!!!");
                }
            } // 익명 클래스의 끝
        );
   } // main메서드의 끝
} // InnerEx8클래스의 끝
```