제 21 강 : 이너클래스

※ 학습목표
✓ 이너클래스의 특징을 설명할 수 있다.
✓ 이너클래스를 작성할 수 있다.

1. 이너클래스 - 내부클래스 - 중첩클래스
① 이너 클래스의 이해와 특징
✓ 특정 클래스 내에 또 다른 클래스가 정의되는 것을 의미한다.
✓ 이너 클래스가 필요한 이유는 지금까지 작업해 왔던 클래스들과는 다르게 독립적이지는 않지만 하나의 멤버처럼 사용할 수 있는 특징이 있다.

② 이너 클래스를 정의 시 주의사항이자 장점
✓ 이너 클래스는 외부 클래스의 모든 멤버들을 마치 자신의 멤버처럼 사용할 수 있다.
✓ static 이너 클래스는 제외하고는 다른 이너 클래스는 항상 외부 클래스를 통해야 생성이 가능하다
✓ '외부클래스명$내부클래스명.class'으로 class파일이 만들어짐

③ 이너클래스의 종류

member class	외부클래스의 멤버변수 선언위치에 선언하며, 외부클래스의 인스턴스멤버처럼 다루어진다. 주로 외부클래스의 인스턴스멤버들과 관련된 작업에 사용될 목적으로 선언된다.
static class	외부클래스의 멤버변수 선언위치에 선언하며, 외부클래스의 static멤버처럼 다루어진다. 주로 외부클래스의 static멤버, 특히 static메서드에서 사용될 목적으로 선언된다.
local class	외부클래스의 메서드나 초기화블럭 안에 선언하며, 선언된 영역 내부에서만 사용될 수 있다.
anonymous class	클래스의 선언과 객체의 생성을 동시에 하는 이름없는 클래스(일회용)

2. Member 이너 클래스

✓ 말 그대로 객체를 생성해야만 사용할 수 있는 멤버들과 같은 위치에 정의되는 클래스를 말한다.

✓ 이너 클래스를 생성하려면 외부 클래스의 객체를 생성한 후에 생성할 수 있다.

✓ inner class 내부에는 static이 존재할 수 없다.

✓ Outer.Inner in = new Outer().new Inner(); 형태로 사용

✓ 형태

```
class Outer {
        …
        class Inner {

        }
        …
}
```

[실습]

```
1  package tommy.java.exam01;
2
3  class OuterEx {
4          public class InnerEx {
5                  int x = 5;
6          }
7  }
8
9  public class InnerExOne {
10         public static void main(String ar[]) {
11                 // 외부클래스 먼저 생성하고.
12                 OuterEx eo = new OuterEx();
13                 // 외부를 통해서 내부클래스를 생성한다.
14                 OuterEx.InnerEx ei = eo.new InnerEx();
15                 System.out.println("ei.x = " + ei.x);
16         }
17 }
```

[실습]

```
1  package tommy.java.exam02;
2
3  public class MemberInner {
4          int a = 10;
5          private int b = 100;
6          static int c = 200;
7
8          class Inner { // 이너 클래스 정의
9                  public void printData() {
10                         System.out.println("int a : " + a);
11                         System.out.println("private int b : " + b); // 주시하자!
12                         System.out.println("static int c : " + c);
13                 }
14         }
15
16         public static void main(String[] args) {
17                 // MemberInner outer = new MemberInner();
18                 // MemberInner.Inner inner = outer.new Inner();
19                 MemberInner.Inner inner = new MemberInner().new Inner();
20                 inner.printData();
21         }
22 }
```

3. Local 이너 클래스

✓ Local 이너 클래스는 특정 메소드 안에서 정의되는 클래스를 말한다.

✓ 특정 메소드 안에서 선언되는 지역변수와 같은 것이다.

✓ 메소드가 호출될 때 생성할 수 있으며 메소드의 수행력이 끝나면 지역변수와 같이 자동 소멸된다.

✓ 잘 사용되지 않는다.

✓ method 내부에 class가 존재 method 실행 시 실행된다.

✓ 형식

```
class Outer {
        …
        public void methodA() { // 멤버 메소드
                class Inner {

                }
        }
        …
}
```

[실습]

```
1  package tommy.java.exam03;
2
3  public class LocalInner {
4          int a = 100; // 멤버 변수
5
6          public void innerTest(int k) {
7                  int b = 200; // 지역변수
8                  final int c = k; // 상수
9                  class Inner {
10                         public void getData() {
11                                 System.out.println("int a : " + a);
12                                 System.out.println("int b : "+b);
13                                 System.out.println("final int c : " + c);// 상수 사용
14                         }
15                 }
16                 Inner i = new Inner(); // 메소드 내에서 Local 이너 클래스 생성
17                 i.getData(); // 생성된 reference를 통해 메소드 호출
18         }
19
20         public static void main(String[] args) {
21                 LocalInner outer = new LocalInner();
22                 outer.innerTest(1000);
23         }
24 }
```

4. static 이너 클래스

✓ static 이너 클래스로 어쩔 수 없이 정의하는 경우가 있는데 그것은 바로 이너 클래스 안에 static변수를 가지고 있다면 어쩔 수 없이 해당 이너 클래스는 static으로 선언하여야 한다.

✓ inner class 의 지정어로 static을 기재한다. 따라서 class 내부에선 static 변수 사용 가능

✓ static 이너 클래스는 외부 클래스를 생성하지 않고도 [외부_클래스명.내부_클래스_생성자()]로 생성이 가능하다

✓ Outer.Inner in = new Outer.Inner(); [또는 new Outer().new Inner()] 사용가능

✓ 만약 inner class 가 main 안에 존재할 때는 Outer$Inner.class 를 실행시킴

✓ 형식

```
class Outer {
        …
        static class Inner {

        }
        …
}
```

[실습]

```
1  package tommy.java.exam04;
2
3  public class StaticInner {
4          int a = 10;
5          private int b = 100;
6          static int c = 200;
7
8          static class Inner {
9                  // 어쩔 수 없이 이너 클래스를 static으로 선언해야 할 경우가 있다.
10                 // 그건 바로 이너 클래스의 멤버들 중 하나라도 static멤버가 있을 때이다.
11                 static int d = 1000;
12
13                 public void printData() {
14                         // System.out.println("int a : "+a); //오류
15                         // System.out.println("private int b : "+b); //오류
16                         System.out.println("static int c : " + c);
```

```
17                      }
18              }
19
20              public static void main(String[] args) {
21                      // 또 다른 독립된 객체에서 static 이너 클래스 생성시
22                      StaticInner.Inner inner = new StaticInner.Inner();
23                      inner.printData();
24              }
25      }
```

[실습]

```
1   package tommy.java.exam05;
2
3   public class OuterEx {
4           public static class InnerEx {
5                   static int x = 10;
6
7                   public static void main(String args[]) {
8                           OuterEx.InnerEx ei = new OuterEx.InnerEx();
9                           System.out.println("ei.x = " + ei.x);
10                          System.out.println("ei.x = " + OuterEx.InnerEx.x);
11                  }
12          }
13  }
```

5. Anonymous 이너 클래스

✓ 익명이란? 이름이 없는 것을 의미한다. 이것을 자바의 프로그램 적으로 해석하면 정의된 클래스의 이름이 없다는 것이 된다.

✓ Event 와 관련이 있다.

✓ interface 구현이 필요 없다.

✓ 일반 메소드 내부에서 정의부를 가질 수 있다.

✓ abstract를 상속 받을 수 있다

✓ 반드시 final로 선언해야 한다.

✓ implements를 할 때는 한 개만 implement한다

✓ 형식

```
class Outer {
        …
        Inner inner = new Inner(){
                …;
        };
        public void methodA() { // 멤버 메소드
                new Inner() {
                        …;
                };
        }
        …
}
```

[실습]

```
1  package tommy.java.exam06;
2
3  abstract class TestAbst {
4          int data = 10000;
5
6          public abstract void printData(); // 추상메소드
7  }
8
9  public class AnonyInner {
10         TestAbst inn = new TestAbst() {
11                 public void printData() { // 미완성된 것을 완성한다.
12                         System.out.println("data : " + data);
13                 }
14         };
15
16         public static void main(String[] args) {
17                 AnonyInner ai = new AnonyInner();
18                 ai.inn.printData();
19         }
20 }
```