# 중첩 클래스

중첩 클래스란 클래스 내부에 다른 클래스를 정의하는 것을 의미하며, 이를 통해 코드 구조를 모듈화하고 객체 지향 프로그래밍을 효과적으로 활용할 수 있게 해줍니다.

중첩 클래스는 다양한 종류로 나뉘며, 각각의 종류는 특정 상황에 적합하게 사용됩니다. 오늘은 인스턴스 멤버 클래스, 정적 멤버 클래스, 로컬 클래스, 중첩 인터페이스, 그리고 익명 객체에 대한 설명과 예시를 통해 중첩 클래스의 활용법을 살펴보겠습니다.

### 멤버 클래스

- 인스턴스 멤버 클래스

> A 객체를 생성해야만 B 객체를 생성할 수 있음

```
class A {
    class B {}
}
```

- 정적 멤버 클래스

> A 객체를 생성하지 않아도 B 객체를 생성할 수 있음

```
class A {
   static class B {}
}
```

### 로컬 클래스

> method가 실행할 때만 B 객체를 생성할 수 있음

```
class A {
    void method() {
        class B {}
    }
}
```

### 인스턴스 멤버 클래스

인스턴스 멤버 클래스(Instance Member Class)는 바깥 클래스의 인스턴스를 생성해야만 객체를 생성할 수 있는 중첩 클래스이다. 인스턴스 멤버 클래스는 클래스 내부에서 사용되므로 private 접근 제한을 갖는 것이 일반적이다. 인스턴스 멤버 B 내부에 필드, 생성자, 메소드 선언이 올 수 있는데 필드와 정적 메소드 선언은 Java 17부터 가능하다.

| 구분 | 접근 범위 |
| --- | --- |
| public class B {} | 다른 패키지에서 B클래스를 사용할 수 있다. |
| class B | 같은 패키지에서만 B클래스를 사용할 수 있다. |
| private class B {} | A 클래스 내부에서만 B 클래스를 사용할 수 있다. |

```
[public] class A {
    [public | private] class B {} // 인스턴스 멤버 클래스
}
```

```
public class Parent
{
    public class Child
    {
        void method() {
            System.out.println("instance member child");
        }
    }

    Parent() {
        Child child = new Child();
    }
}
```

```
public class main
{
    public static void main(String[] args)
    {
        Parent parent = new Parent();

        Parent.Child child = parent.new Child();

        child.method();
    }
}
```

### 정적 멤버 클래스

정적 멤버 클래스(Static Member Class)"에서의 "static" 키워드는 클래스 자체가 정적(Static)으로 선언되었음을 나타내기 위한 것이 아니라, 중첩 클래스의 인스턴스가 바깥 클래스의 인스턴스에 대한 종속성이 없다는 것을 의미한다. 이것은 중첩 클래스가 독립적으로 생성될 수 있고, 바깥 클래스의 인스턴스 생성 없이도 중첩 클래스의 객체를 만들 수 있다.

```
[public] class A {
    [public | private | protected | package-private(default)] class B {} // 인스턴스 멤버 클래스
}
```

```
public class OuterClass {
    private int outerField;

    // public 접근 제한자를 사용한 정적 멤버 클래스
    public static class PublicStaticInnerClass {
        // ...
    }

    // private 접근 제한자를 사용한 정적 멤버 클래스
    private static class PrivateStaticInnerClass {
        // ...
    }

    // protected 접근 제한자를 사용한 정적 멤버 클래스
    protected static class ProtectedStaticInnerClass {
        // ...
    }

    // package-private(default) 접근 제한자를 사용한 정적 멤버 클래스
    static class PackagePrivateStaticInnerClass {
        // ...
    }
}
```

```
public class OuterClass {
    private int outerField;

    public OuterClass(int outerField) {
        this.outerField = outerField;
    }

    // 정적 멤버 클래스
    public static class StaticInnerClass {
        private int innerField;

        public StaticInnerClass(int innerField) {
            this.innerField = innerField;
        }

        public void display() {
            System.out.println("Inner Field: " + innerField);
            // 정적 멤버 클래스에서는 바깥 클래스의 인스턴스 멤버에 직접 접근할 수 없습니다.
```

```
            // System.out.println("Outer Field: " + outerField); // 컴파일 오류
        }
    }
}
```

```
public class main
{
    public static void main(String[] args)
    {
        // 정적 멤버 클래스의 객체 생성
        OuterClass.StaticInnerClass innerObj = new
OuterClass.StaticInnerClass(42);

        // 객체의 메소드 호출
        innerObj.display();
    }
}
```

**로컬 클래스**

로컬 클래스(Local Class)는 메서드 내부에서 정의되고 해당 메서드 내에서만 사용될 수 있는 중첩 클래스이다. 로컬 클래스는 특정 메서드 내부에서만 인스턴스를 생성하고 사용할 수 있으며, 해당 메서드의 로컬 변수와 파라미터에 접근할 수 있다.

```
public class OuterClass {
    public void outerMethod() {
        final int outerVariable = 10;

        // 로컬 클래스
        class LocalInnerClass {
            public void innerMethod() {
                int innerVariable = 5;

                // 로컬 클래스에서 바깥 클래스의 메서드 내 로컬 변수 및 파라미터 접근
가능
                System.out.println("Outer Variable: " + outerVariable);
                System.out.println("Inner Variable: " + innerVariable);
            }
        }

        // 로컬 클래스의 객체 생성 및 메서드 호출
        LocalInnerClass innerObj = new LocalInnerClass();
        innerObj.innerMethod();
    }
}
```

### 바깥 멤버 접근

중첩 클래스는 바깥 클래스와 긴밀한 관계를 맺으면서 바깥 클래스의 멤버(필드, 메소드)에 접근할 수 있다. 하지만 중첩 클래스가 어떻게 선언되었으냐에 따라 접근 제한이 있을 수 있다.

| 구분 | 바깥 클래스의 사용 가능한 멤버 |
|---|---|
| 인스턴스 멤버 클래스 | 바깥 클래스의 모든 필드와 메소드 |
| 정적 멤버 클래스 | 바깥 클래스의 정적 필드와 정적 메소드 |

### 중첩 인터페이스

중첩 인터페이스는 클래스의 멤버로 선언된 인터페이스를 말한다. 인터페이스를 클래스 내부에 선언하는 이유는 해당 클래스와 긴밀한 관계를 맺는 구현 객체를 만들기 위해서이다.

```
class A {
    /* 중첩 인터페이스 */
    [public | private] [static] interface  B {
        // 상수 필드
        // 추상 메소드
        // 디폴트 메소드
        // 정적 메소드
    }
    /* 중첩 인터페이스 */
}
```

### 익명 객체

명시적으로 클래스를 선언하지 않기 때문에 쉽게 객체를 생성할 수 있다. 익명 객체는 필드값, 로컬 변수값, 매개변수값으로 주로 사용된다. 익명 객체는 클래스를 상속하거나 인터페이스를 구현해야만 생성할 수 있다. 클래스를 상속해서 만들 경우 익명 자식 객체라고 하고, 인터페이스를 구현해서 만들 경우 익명 구현 객체라고 한다.

- 익명 자식 객체

```
new 부모 생성자(매개값) {
    // 필드
    // 메소드
}
```

- 익명 구현 객체

```
new 인터페이스() {
    // 필드
    // 메소드
}
```

중첩 클래스는 이렇게 코드를 더 모듈화하며, 개발하는데 있어 유연성을 높여줍니다. 이를 통해 프로그래밍의 복잡성을 줄이고 코드를 관리하기 쉽게 만들 수 있습니다.