내부클래스와 enum

- 내부 클래스
  - 클래스 안에 정의된 클래스
  - 스윙, 안드로이드 이벤트 처리
- **enum**
  - 열거형을 표현하는 타입
  - 상수들의 집합
  - 만들어지면 값을 바꿀 수 없다.
- 무명 클래스
  - 클래스 몸체는 정의되지만 이름이 없는 클래스
  - 기명 클래스 **vs** 무명 클래스

메서드의 종류와 특징

클래스 종류	특징
일반 메서드	
static 메서드	static 키워드가 붙은 메서드
추상 메서드	abstract 키워드가 붙은 메서드

클래스의 종류와 특징

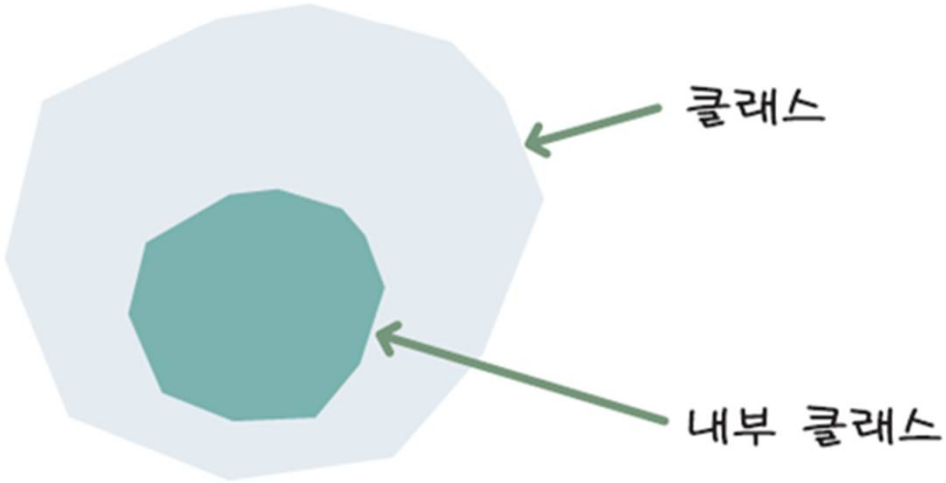
클래스 종류	특징
인스턴스 클래스 (instance class)	**외부 클래스의 필드 위치에 new 이용하여 만들어짐**
static 클래스	static 키워드가 사용되는 클래스 java.util.Math 클래스 ex) Math.sqrt(2);
내부 클래스	**클래스 안에 선언된 클래스**
익명 클래스	클래스의 선언과 인스턴스 생성을 동시에 하는 클래스. 일회용.
추상 클래스	abstract 키워드가 사용되는 클래스 abstract 메서드가 들어있으면 추상 클래스로 만든다. 인스턴스를 만들 수 없다.
인터페이스	**추상(abstract) 메서드만 들어 있는 클래스**

내부 클래스

- 내부 클래스(inner class): 클래스 안에 정의된 클래스
- 이벤트 처리시 주로 사용된다.

```
public class OuterClass {
    // 클래스의 필드와 메소드 정의
    ...
    private class InnerClass {
    // 내부 클래스의 필드와 메소드 정의
        ...
    }
}
```

내부 클래스는 다른 클래스 내부에 정의된 클래스이다.
외부 클래스의 모든 멤버를 자유롭게 사용할 수 있다.



내부 클래스

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

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



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

내부 클래스-예제

```
class OuterClass {
    private String secret = "Time is money";

    public OuterClass() {
        InnerClass obj = new InnerClass();
        obj.print();
    }

    private class InnerClass {
        public InnerClass() {
            System.out.println("내부 클래스 생성자입니다.");
        }

        public void print() {
            System.out.println(secret);
        }
    }
}
public class OuterClassTest {
    public static void main(String args[]) {
        new OuterClass();
    }
}
```

전용 필드 선언

내부 클래스의 객체를 생성하고 method() 호출

클래스의 private 변수인 secret를 자유롭게 사용할 수 있다.

실행결과

```
내부 클래스 생성자입니다.
Time is money
```

왜 내부 클래스를 사용하는가?

- 필드나 메서드를 private로 유지하면서 자유롭게 사용할 수 있다.
- 하나의 장소에서만 사용되는 클래스들을 한곳에 모을 수 있다
- 보다 읽기 쉽고 유지 보수가 쉬운 코드가 된다.

- **이벤트 핸들러에서 주로 사용.** **스윙, 안드로이드, JavaFX**

enum

- enum이란?
  - 상수들의 집합
- 왜 클래스를 상수들의 집합으로 만드나요?
  - 코드성 데이터

- 열거형을 표현하는 방식에는
  - 클래스 방식
  - enum 방식

- enum은 무엇인가?
  - 클래스와 같은 타입이고 enum은 상수로 정의해 사용할 수 있다.
  - enum은 상수에 static과 final이 내재되어 있고,
  - enum은 만들어지면 값을 바꿀 수 없다.

- 자바의 Enumeration(Enum)은 JDK 1.5 부터 사용되기 시작

예제 - enum

- 클래스 방식의 열거형

```
package java15.enumeration

public class PhoneHeaderClass {
    public static final String P010 = "010";
    public static final String P011 = "011";
    public static final String P016 = "016";
    public static final String P017 = "017";
    public static final String P018 = "018";
    public static final String P019 = "019";
}
```

PhoneHeaderClass.java

Blah.valueOf("A") method is case sensitive and doesn't tolerate extraneous whitespace

- enum 방식의 열거형

```
package java15.enumeration

public enum PhoneHeaderEnum {
    P010("010"),
    P011("011"),
    P016("016"),
    P017("017"),
    P018("018"),
    P019("019");

    private final String value;
    PhoneHeaderEnum(String value) {
        this.value = value;
    }
    public String getValue() {
        return value;
    }
}
```

PhoneHeaderEnum.java

예제 - enum

```
package java15.enumeration
public class PhoneHeaderTest {

    public static void main(String [] args) {
        // 키포드로 폰번호를 입력받는다.

        // 입력 받은 폰번호에서 앞으로부터 3자리 추출
        String header = phone.substring(0,3);

        // PhoneHeaderClass를 사용하여 아래를 출력되게 하시오.
        // header 가 010이면 general, 011이면 sk, 016이면 kt, 019이면 lg.

        // PhoneHeaderEnum을 사용하여 아래를 출력되게 하시오.
        // header 가 010이면 general, 011이면 sk, 016이면 kt, 019이면 lg.

    }
}
```

예제 - enum #2

```
public enum Day {
    SUN(1), MON(2), TUE(3), WED(4), THU(5), FRI(6), STA(7);

    private final int value;
    public int getValue() { return value; }

    Day(int value) { this.value= value; }

    public static Day fromValue(int id) {
        for (Day type : Day.values()) {
            if (type.getValue() == id) {
                return type;
            }
        }
        return null;
    }
}
```

예제 - enum #2

```
package java15.enumeration;

import java.util.Calendar;

public class DayTest {

    public static void main( String[] args){

        Calendar date = Calendar.getInstance();

        int i = date.get(Calendar.DAY_OF_WEEK) +1;

        Day week = Day.fromValue(i);

        if ( week == Day.SUN ) {

        }
    }
}
```

무명 클래스(anonymous class)

- 클래스의 몸체는 있고 이름이 없는 일회용 클래스
- 하나의 객체만을 생성할 수 있다.

```
new 조상클래스이름() {
    // 멤버 선언
}

    또는

new 구현인터페이스이름() {
    // 멤버 선언
}
```

무명 클래스(anonymous class)

- 이름이 있는 클래스의 경우

```
class TV implements RemoteControl {
    ...
}
RemoteControl obj = new TV();
```

- 무명 클래스의 경우

```
RemoteControl obj = new RemoteControl() {
    ...
};
```

무명 클래스-예제

AnonymousClassTest.java

```
interface RemoteControl {
    void turnOn();
    void turnOff();
}

public class AnonymousClassTest {
    public static void main(String args[]) {
        RemoteControl ac = new RemoteControl() {     // 무명 클래스 정의
            public void turnOn() {
                System.out.println("TV turnOn()");
            }
            public void turnOff() {
                System.out.println("TV turnOff()");
            }
        };
        ac.turnOn();
        ac.turnOff();
    }
}
```

무명 클래스가 정의되면서
동시에 객체도 생성된다.

무명 클래스(anonymous class)

- 클래스의 몸체는 있지만 이름이 없는 일회용 클래스
- **쓰레드 호출시 주로 이용됨**

- 기명 클래스의 경우 : 이름이 있는

```
class TV implements RemoteControl {
    ...
}
RemoteControl obj = new TV();
```

- 무명 클래스의 경우

```
RemoteControl obj = new RemoteControl() {
    ...
};
```