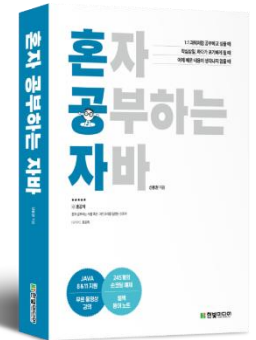


Chapter

08

인터페이스



08-1. 인터페이스

혼자 공부하는 자바 (신용권 저)

시작하기 전에

[핵심 키워드] : 인터페이스, 상수 필드, 추상 메소드, 구현 클래스, 인터페이스 사용

[핵심 포인트]

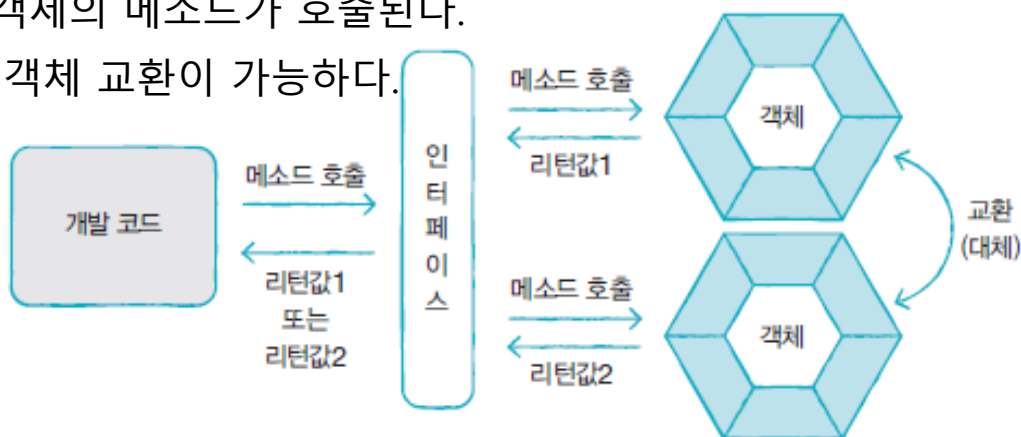
인터페이스란 객체의 사용 방법을 정의한 타입이다.

인터페이스를 통해 다양한 객체를 동일한 사용 방법으로 이용할 수 있다.

인터페이스를 이용해서 다형성을 구현할 수 있다.

❖ 인터페이스 (interface)

- 개발 코드는 인터페이스를 통해서 객체와 서로 통신한다.
- 인터페이스의 메소드 호출하면 객체의 메소드가 호출된다.
- 개발 코드를 수정하지 않으면서 객체 교환이 가능하다.



인터페이스 선언

❖ 인터페이스 선언

- ~.java 형태 소스 파일로 작성 및 컴파일러 통해 ~.class 형태로 컴파일된다.
- 클래스와 물리적 파일 형태는 같으나 소스 작성 내용이 다르다.

```
[public] interface 인터페이스이름 { ... }
```

- 인터페이스는 객체로 생성할 수 없으므로 생성자 가질 수 없다.

```
interface 인터페이스이름 {  
    //상수  
    타입 상수이름 = 값;  
    //추상 메소드  
    타입 메소드이름(매개변수, ...);  
}
```



인터페이스 선언

❖ 상수 필드 (constant field) 선언

- 데이터를 저장할 인스턴스 혹은 정적 필드 선언 불가
- 상수 필드만 선언 가능

```
[public static final] 타입 상수이름 = 값;
```

- 상수 이름은 대문자로 작성하되 서로 다른 단어로 구성되어 있을 경우 언더바(_)로 연결

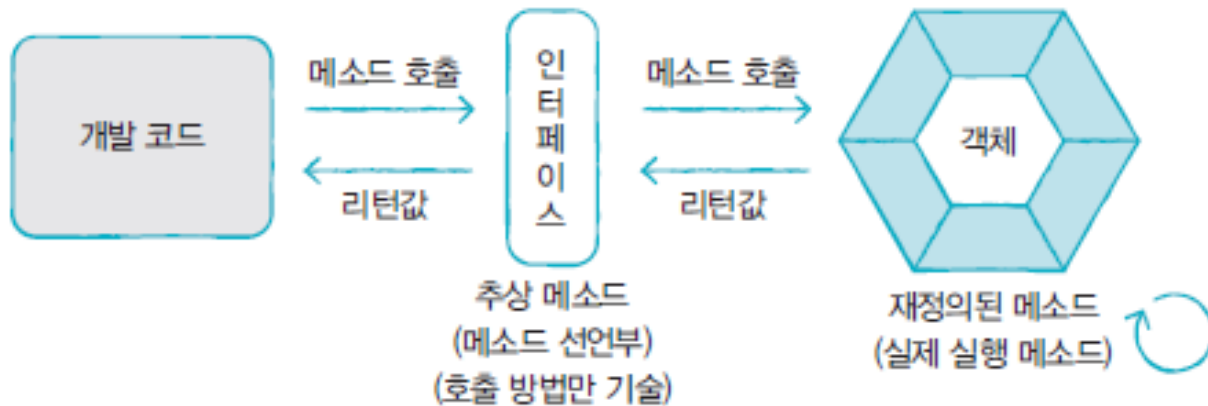
```
public interface RemoteControl {  
    public int MAX_VOLUME = 10;  
    public int MIN_VOLUME = 0;  
}
```



인터페이스 선언

❖ 추상 메소드 선언

- 인터페이스 통해 호출된 메소드는 최종적으로 객체에서 실행
- 인터페이스의 메소드는 실행 블록 필요 없는 추상 메소드로 선언



`[public abstract] 리턴타입 메소드이름(매개변수, ...);`

```
public interface RemoteControl {  
    //추상 메소드  
    public void turnOn();  
    public void turnOff();  
    public void setVolume(int volume);  
}
```

인터페이스 구현

❖ 구현 (implement) 클래스

- 인터페이스에서 정의된 추상 메소드를 재정의해서 실행내용을 가지고 있는 클래스
- 클래스 선언부에 implements 키워드 추가하고 인터페이스 이름 명시

```
public class 구현클래스이름 implements 인터페이스이름 {  
    //인터페이스에 선언된 추상 메소드의 실제 메소드 선언  
}
```

```
public class Television implements RemoteControl {
```

```
    //turnOn() 추상 메소드의 실제 메소드
```

```
    public void turnOn() {  
        System.out.println("TV를 켭니다.");  
    }
```

```
    //turnOff() 추상 메소드의 실제 메소드
```

```
    public void turnOff() {  
        System.out.println("TV를 끕니다.");  
    }
```

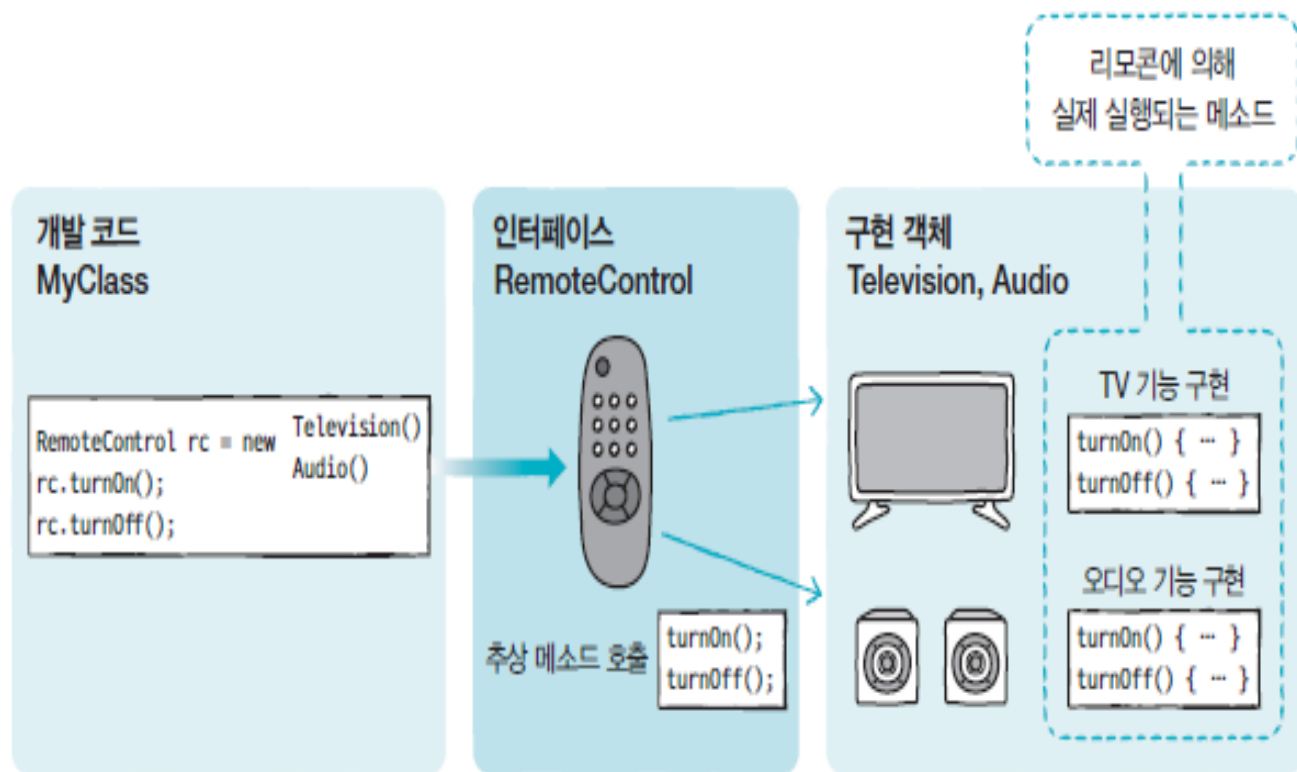


인터페이스 구현

- ❖ 인터페이스와 구현 클래스 사용 방법
 - 인터페이스 변수 선언하고 구현 객체를 대입

```
인터페이스 변수;  
변수 = 구현객체;
```

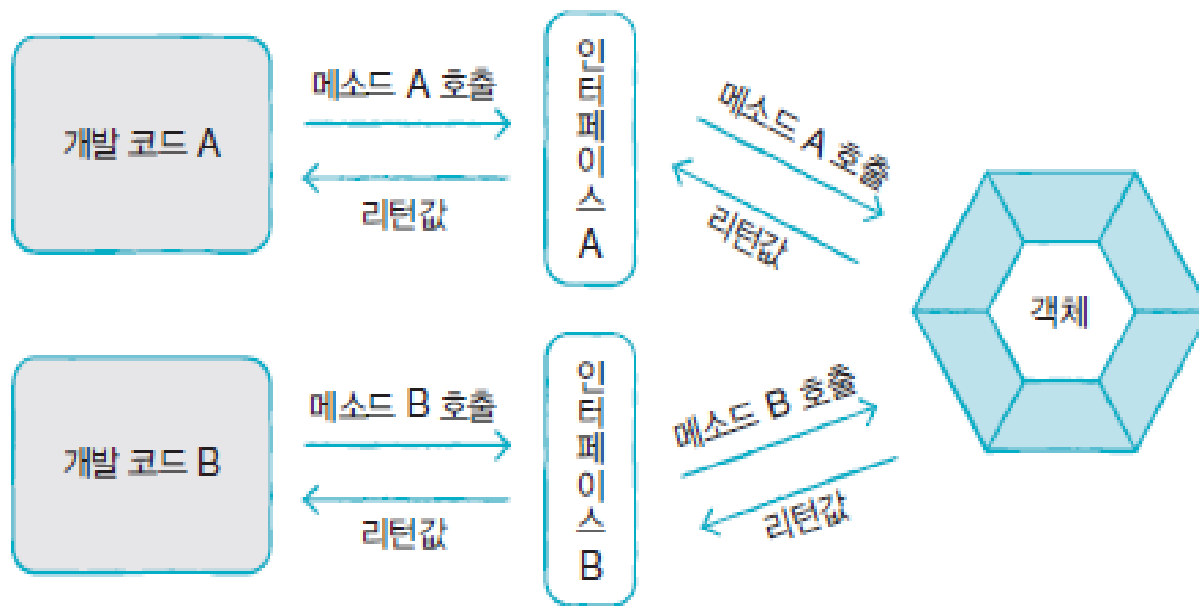
```
인터페이스 변수 = 구현객체;
```



인터페이스 구현

❖ 다중 인터페이스 구현 클래스

- 객체는 다수의 인터페이스 타입으로 사용 가능



```
public class 구현클래스이름 implements 인터페이스A, 인터페이스B {  
    //인터페이스 A에 선언된 추상 메소드의 실제 메소드 선언  
    //인터페이스 B에 선언된 추상 메소드의 실제 메소드 선언  
}
```


인터페이스 사용

❖ 인터페이스 사용

- 인터페이스는 필드, 매개 변수, 로컬 변수의 타입으로 선언가능

```
public class MyClass {  
    //필드  
    ① RemoteControl rc = new Television();  
  
    //생성자  
    ② MyClass( RemoteControl rc ) {  
        this.rc = rc;  
    }  
  
    //메소드  
    void methodA() {  
        //로컬 변수  
        ③ RemoteControl rc = new Audio();  
    }  
    ④ void methodB( RemoteControl rc ) { ... }  
}
```

생성자의 매개값으로 구현 객체 대입
MyClass mc = new MyClass(new Television());

생성자의 매개값으로 구현 객체 대입
mc.methodB(new Audio());



예제

```
RemoteControl.java  ⌕
1  package sec01.exam06;
2
3  public interface RemoteControl {
4      //상수
5      int MAX_VOLUME = 10;
6      int MIN_VOLUME = 0;
7
8      //추상 메소드
9      void turnOn();
10     void turnOff();
11     void setVolume(int volume);
12 }
13
```



예제

```
MyClass.java
1 package sec01.exam06;
2
3 public class MyClass {
4     // 필드
5     RemoteControl rc = new Television();
6
7     // 생성자
8     MyClass() {
9     }
10
11     MyClass(RemoteControl rc) {
12         this.rc = rc;
13         rc.turnOn();
14         rc.setVolume(5);
15     }
16
17     // 메소드
18     void methodA() {
19         RemoteControl rc = new Audio();
20         rc.turnOn();
21         rc.setVolume(5);
22     }
23
24     void methodB(RemoteControl rc) {
25         rc.turnOn();
26         rc.setVolume(5);
27     }
28 }
```

```
MyClassExample.java
1 package sec01.exam06;
2
3 public class MyClassExample {
4     public static void main(String[] args) {
5         System.out.println("1-----");
6
7         MyClass myClass1 = new MyClass();
8         myClass1.rc.turnOn();
9         myClass1.rc.setVolume(5);
10
11         System.out.println("2-----");
12
13         MyClass myClass2 = new MyClass(new Audio());
14
15         System.out.println("3-----");
16
17         MyClass myClass3 = new MyClass();
18         myClass3.methodA();
19
20         System.out.println("4-----");
21
22         MyClass myClass4 = new MyClass();
23         myClass4.methodB(new Television());
24     }
25 }
26
```

예제

Television.java

```
1 package sec01.exam06;
2
3 public class Television implements RemoteControl {
4     //필드
5     private int volume;
6
7     //turnOn() 추상 메소드의 실제 메소드
8     public void turnOn() {
9         System.out.println("TV를 켭니다.");
10    }
11    //turnOff() 추상 메소드의 실제 메소드
12    public void turnOff() {
13        System.out.println("TV를 끕니다.");
14    }
15    //setVolume() 추상 메소드의 실제 메소드
16    public void setVolume(int volume) {
17        if(volume>RemoteControl.MAX_VOLUME) {
18            this.volume = RemoteControl.MAX_VOLUME;
19        } else if(volume<RemoteControl.MIN_VOLUME) {
20            this.volume = RemoteControl.MIN_VOLUME;
21        } else {
22            this.volume = volume;
23        }
24        System.out.println("현재 TV 볼륨: " + this.volume);
25    }
26 }
27
```

Audio.java

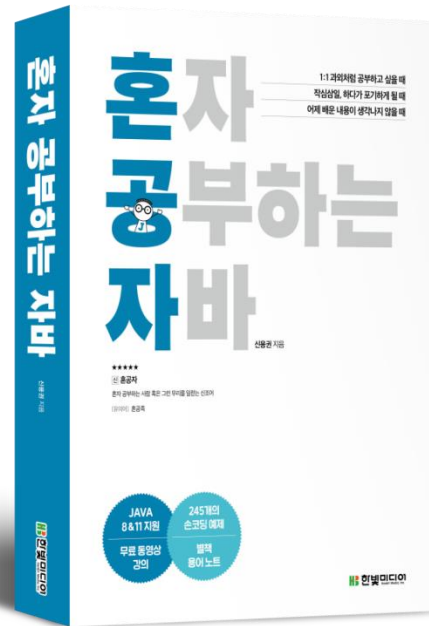
```
1 package sec01.exam06;
2
3 public class Audio implements RemoteControl {
4     //필드
5     private int volume;
6
7     //turnOn() 추상 메소드의 실제 메소드
8     public void turnOn() {
9         System.out.println("Audio를 켭니다.");
10    }
11    //turnOff() 추상 메소드의 실제 메소드
12    public void turnOff() {
13        System.out.println("Audio를 끕니다.");
14    }
15    //setVolume() 추상 메소드의 실제 메소드
16    public void setVolume(int volume) {
17        if(volume>RemoteControl.MAX_VOLUME) {
18            this.volume = RemoteControl.MAX_VOLUME;
19        } else if(volume<RemoteControl.MIN_VOLUME) {
20            this.volume = RemoteControl.MIN_VOLUME;
21        } else {
22            this.volume = volume;
23        }
24        System.out.println("현재 Audio 볼륨: " + this.volume);
25    }
26 }
27
```



키워드로 끝내는 핵심 포인트

- **인터페이스**: 객체의 사용 방법 정의한 타입
- **상수 필드** : 인터페이스의 필드는 기본적으로 `public static final` 특성 가짐
- **추상 메소드** : 인터페이스의 메소드는 `public abstract` 생략되고 메소드 선언부만 있는 추상 메소드
- **implements** : 구현 클래스에는 어떤 인터페이스로 사용 가능한지 기술하기 위해 사용
- **인터페이스 사용**: 클래스 선언 시 필드, 매개 변수, 로컬 변수로 선언 가능. 구현 객체를 대입.





Thank You!