Chapter 08

인터페이스

08-1. 인터페이스

혼자 공부하는 자바 (신용권 저)

목차

시작하기 전에

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

[핵심 포인트]

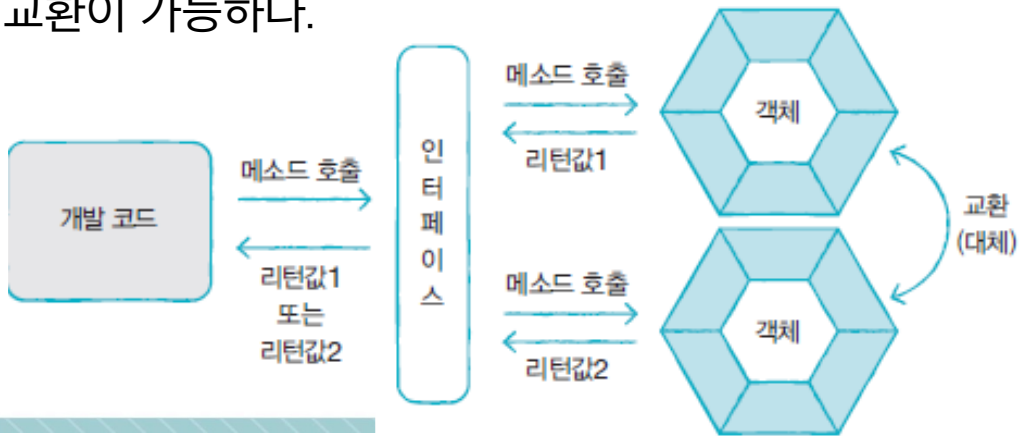
인터페이스란 객체의 사용 방법을 정의한 타입이다.

인터페이스를 통해 다양한 객체를 동일한 사용 방법으로 이용할 수 있다.

인터페이스를 이용해서 다형성을 구현할 수 있다.

❖ 인터페이스 (interface)

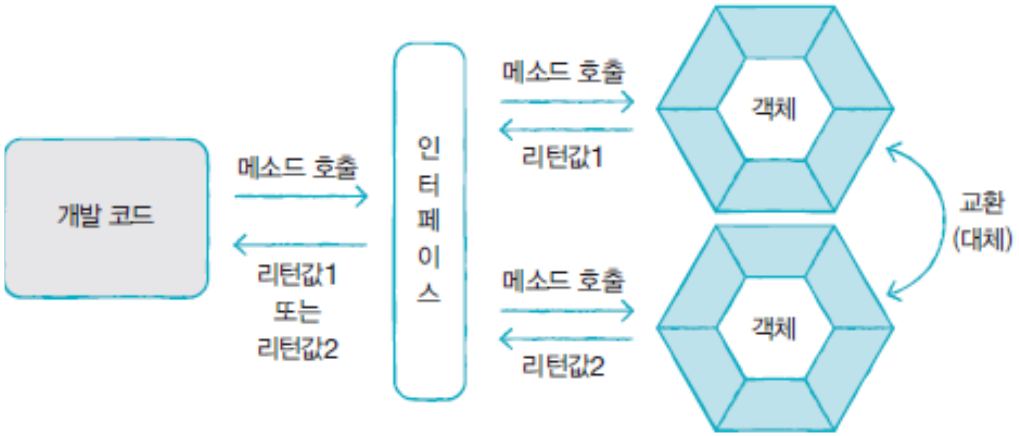
- 개발 코드는 인터페이스를 통해서 객체와 서로 통신한다.
- 인터페이스의 메소드 호출하면 객체의 메소드가 호출된다.
- 개발 코드를 수정하지 않으면서 객체 교환이 가능하다.

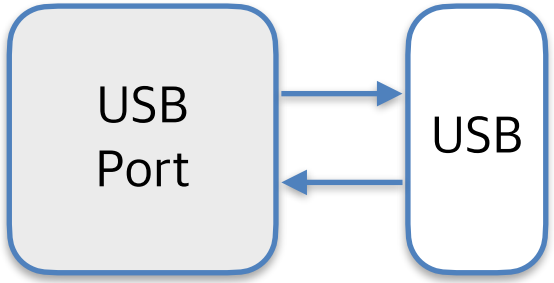


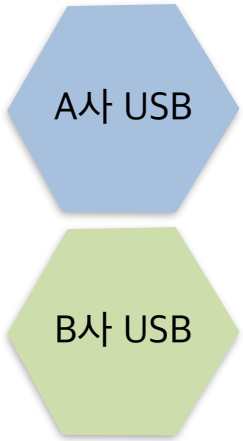
시작하기 전에

❖ 인터페이스 (interface)

- 인터페이스의 규격만 지킨다면 클래스가 달라도 된다.







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 { … }
```

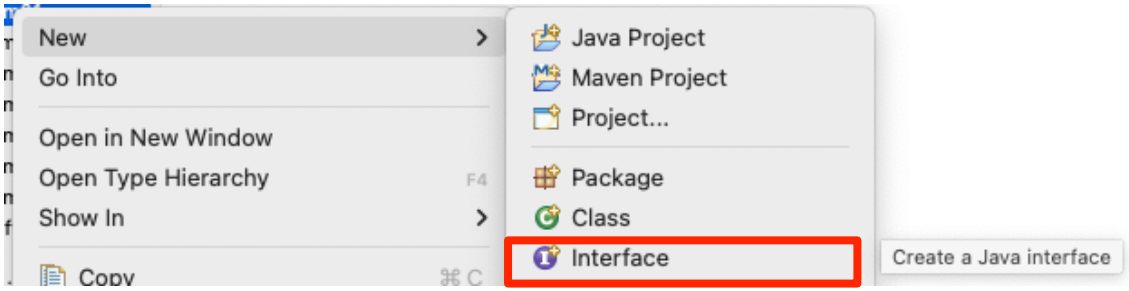
- 인터페이스는 객체로 생성할 수 없으므로 생성자 가질 수 없다.

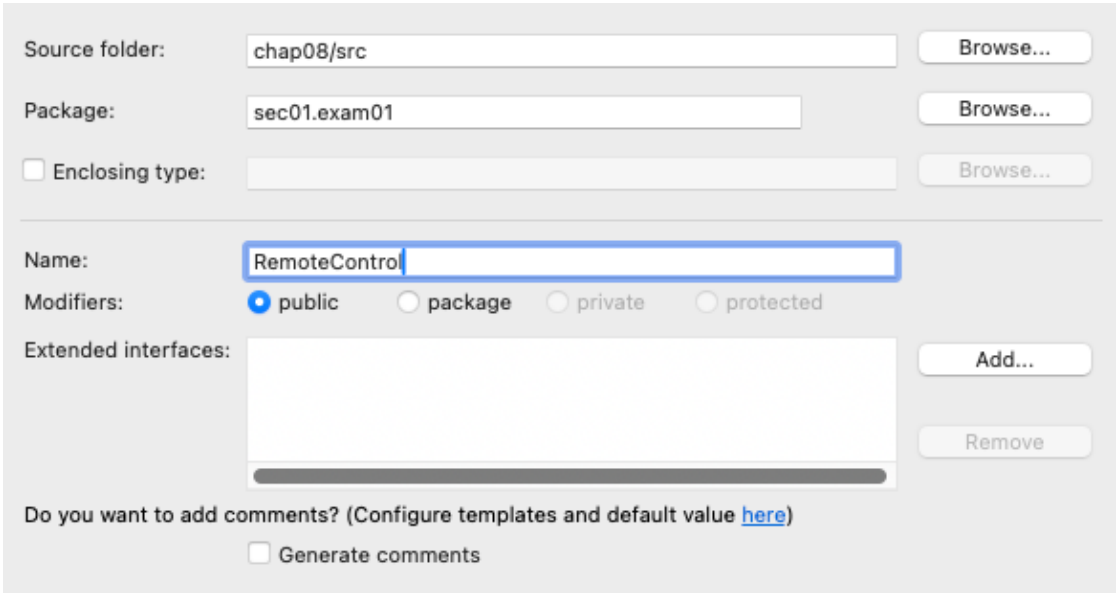
```
interface 인터페이스이름 {
  //상수
  타입 상수이름 = 값;
  //추상 메소드
  타입 메소드이름(매개변수,…);
}
```

인터페이스 선언

❖ 인터페이스 선언

- sec01.exam01.RemoteControl





인터페이스 선언

❖ 인터페이스 선언

393 페이지

- sec01.exam01.RemoteControl

```
package sec01.exam01;

public interface RemoteControl {

}
```

- 클래스로 만든 다음 class 키워드를 interface 로 수정해도 된다.

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

❖ 상수 필드 (constant field) 선언

394 페이지

- sec01.exam02.RemoteControl

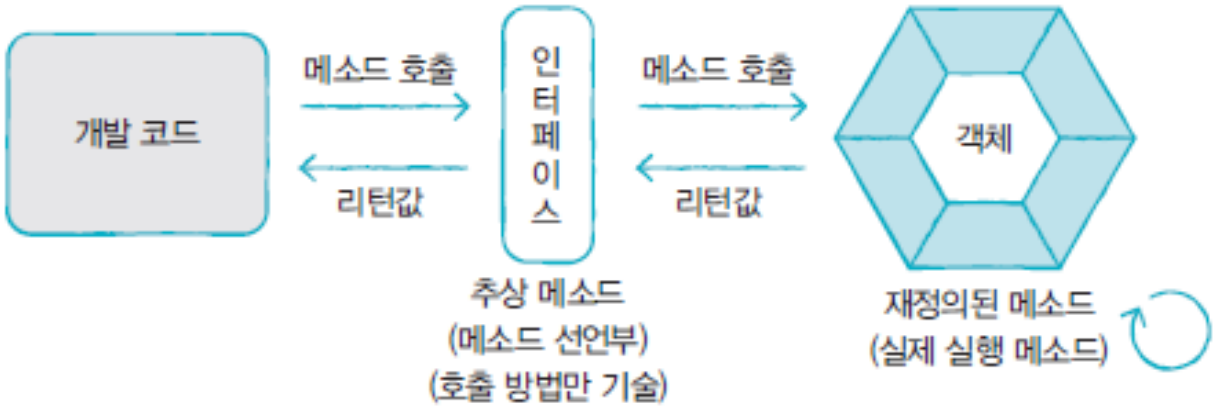
```
package sec01.exam02;

public interface RemoteControl {
    //static final 을 생략해도 interface 의 필드는 static final 로 생성된다.
    int MAX_VOLUME = 10;
    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 선언

❖ 추상 메소드 선언

- sec01.exam03.RemoteControl

395 페이지

```
package sec01.exam03;

public interface RemoteControl {
    //상수
    int MAX_VOLUME = 10;
    int MIN_VOLUME = 0;

    //추상 메소드
    void turnOn();
    void turnOff();
    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 실체 메소드 선언
}
```

```
public class Television implements RemoteControl {
  //turnOn() 추상 메소드의 실체 메소드
  public void turnOn() {
    System.out.println("TV를 켭니다.");
  }
  //turnOff() 추상 메소드의 실체 메소드
  public void turnOff() {
    System.out.println("TV를 끕니다.");
  }
```

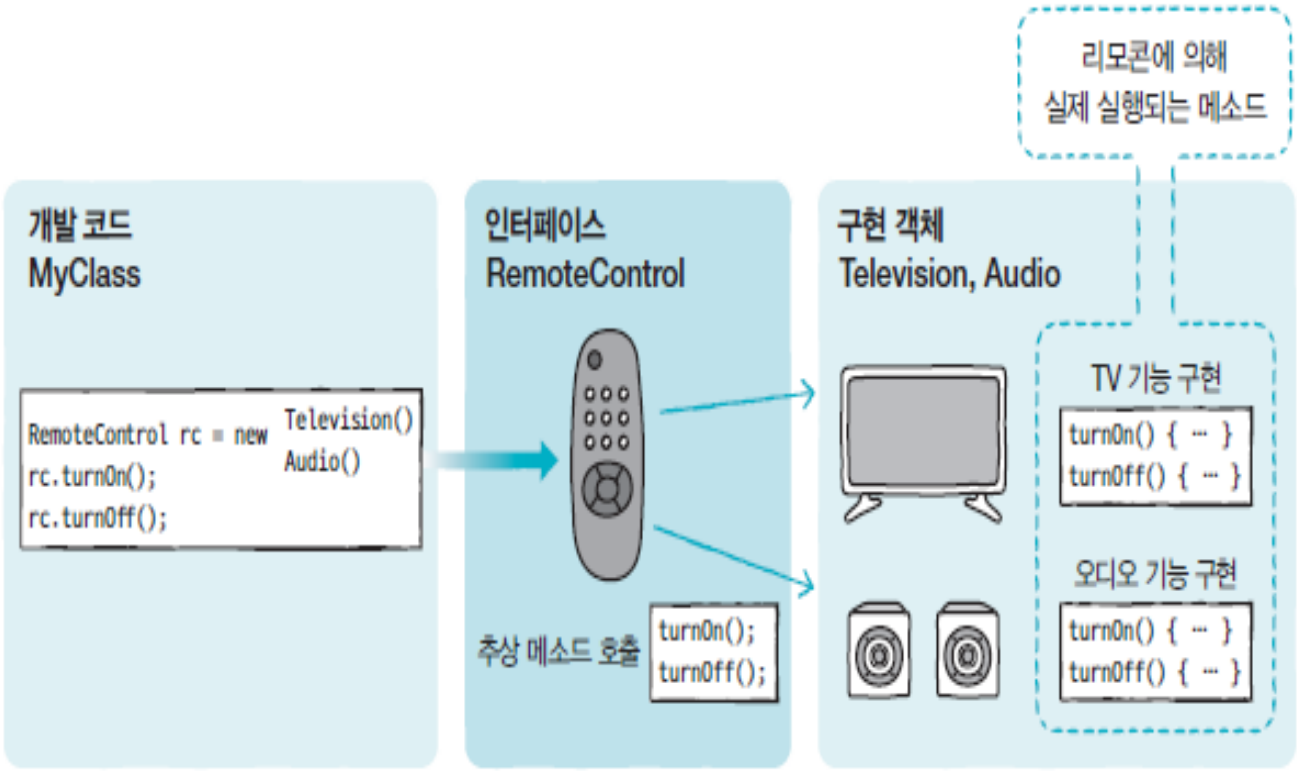
인터페이스 구현

❖ 인터페이스와 구현 클래스 사용 방법

- 인터페이스 변수 선언하고 구현 객체를 대입

```
인터페이스 변수;
변수 = 구현객체;
```

```
인터페이스 변수 = 구현객체;
```



인터페이스 구현

❖ 실습

- sec01.exam04.RemoteControl

```
package sec01.exam03;

public interface RemoteControl {
    //상수
    int MAX_VOLUME = 10;
    int MIN_VOLUME = 0;

    //추상 메소드
    void turnOn();
    void turnOff();
    void setVolume(int volume);
}
```

인터페이스 구현

❖ 실습

- sec01.exam04.Television

396 페이지

```
package sec01.exam04;

public class Television implements RemoteControl {
	//필드
	private int volume;

	//turnOn() 추상 메소드의 실체 메소드
	public void turnOn() {
		System.out.println("TV를 켭니다.");
	}
	//turnOff() 추상 메소드의 실체 메소드
	public void turnOff() {
		System.out.println("TV를 끕니다.");
	}
	//setVolume() 추상 메소드의 실체 메소드
	public void setVolume(int volume) {
		if(volume>RemoteControl.MAX_VOLUME) {
			this.volume = RemoteControl.MAX_VOLUME;
		} else if(volume<RemoteControl.MIN_VOLUME) {
			this.volume = RemoteControl.MIN_VOLUME;
		} else {
			this.volume = volume;
		}
		System.out.println("현재 TV 볼륨: " + this.volume);
	}
}
```

인터페이스 구현

❖ 실습

- sec01.exam04.Audio

397 페이지

```
package sec01.exam04;

public class Audio implements RemoteControl {
	//필드
	private int volume;

	//turnOn()  추상 메소드의 실체 메소드
	public void turnOn() {
		System.out.println("Audio를 켭니다.");
	}
	//turnOff() 추상 메소드의 실체 메소드
	public void turnOff() {
		System.out.println("Audio를 끕니다.");
	}
	//setVolume() 추상 메소드의 실체 메소드
	public void setVolume(int volume) {
		if(volume>RemoteControl.MAX_VOLUME) {
			this.volume = RemoteControl.MAX_VOLUME;
		} else if(volume<RemoteControl.MIN_VOLUME) {
			this.volume = RemoteControl.MIN_VOLUME;
		} else {
			this.volume = volume;
		}
		System.out.println("현재 Audio 볼륨: " + this.volume);
	}
}
```

인터페이스 구현

❖ 실습

- sec01.exam04.RemoteExample

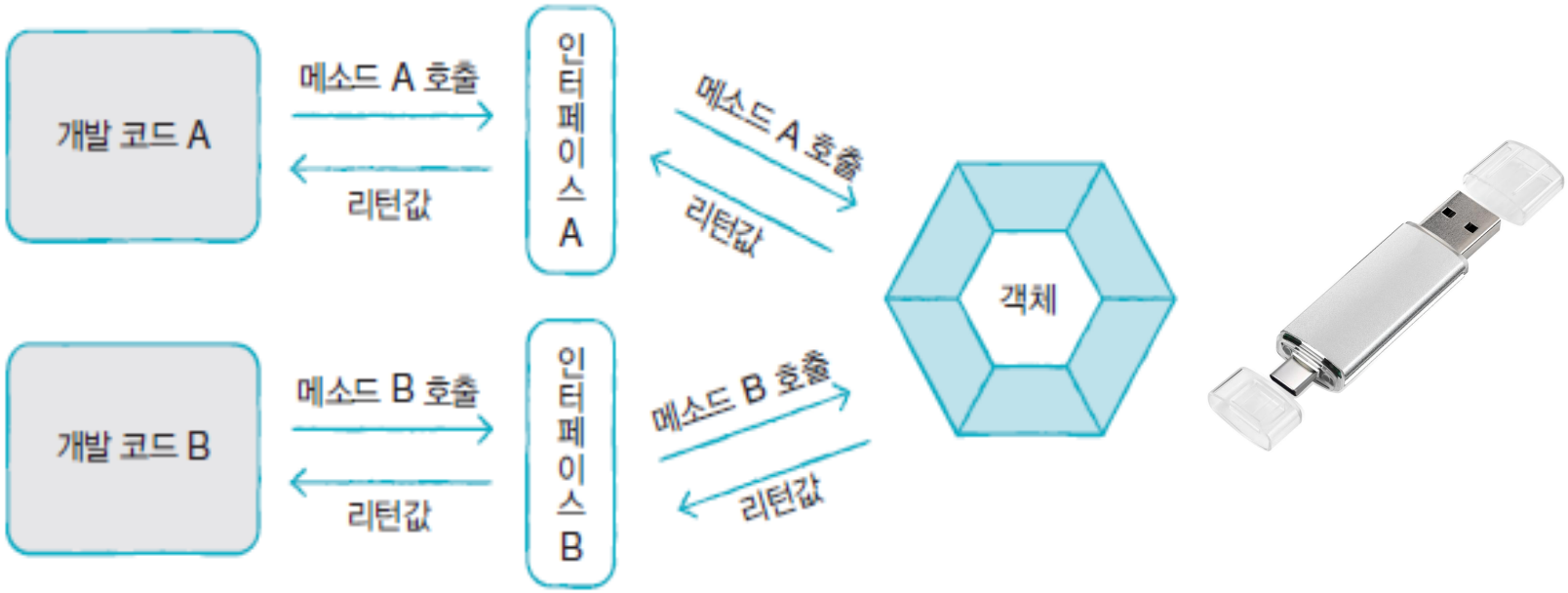
399 페이지

```
package sec01.exam04;

public class RemoteControlExample {
     public static void main(String[] args) {
          RemoteControl rc;
          rc = new Television();
          rc.turnOn();
          rc.turnOff();
          rc = new Audio();
          rc.turnOn();
          rc.turnOff();
     }
}
```

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
}
```

https://www.usbmemorydirect.com/media/images/products/ko_dual/ko_dual_flash_drive-4.jpg

인터페이스 구현

400페이지

- 실습
  - sec01.exam05.RemoteControl
    - sec01.exam04.RemoteControl 과 동일한 내용.

```
package sec01.exam05;

public interface RemoteControl {
	//상수
	int MAX_VOLUME = 10;
	int MIN_VOLUME = 0;

	//추상 메소드
	void turnOn();
	void turnOff();
	void setVolume(int volume);
}
```

  - sec01.exam05.Searchable

```
package sec01.exam05;

public interface Searchable {
	void search(String url);
}
```

인터페이스 구현

실습

400페이지

- sec01.exam05.SmartTelevision

```
package sec01.exam05;

public class SmartTelevision implements RemoteControl, Searchable {
	private int volume;

	public void turnOn() {
		System.out.println("TV를 켭니다.");
	}
	public void turnOff() {
		System.out.println("TV를 끕니다.");
	}
	public void setVolume(int volume) {
		if(volume>RemoteControl.MAX_VOLUME) {
			this.volume = RemoteControl.MAX_VOLUME;
		} else if(volume<RemoteControl.MIN_VOLUME) {
			this.volume = RemoteControl.MIN_VOLUME;
		} else {
			this.volume = volume;
		}
		System.out.println("현재 TV 볼륨: " + this.volume);
	}

	public void search(String url) {
		System.out.println(url + "을 검색합니다.");
	}
}
```

인터페이스 구현

❖ 실습

- sec01.exam05.SmartTelevisionExample

401페이지

```
package sec01.exam05;

public class SmartTelevisionExample {
    public static void main(String[] args) {
        SmartTelevision tv = new SmartTelevision();

        // SmartTelevision 객체는 두 인터페이스를 구현했으므로
        // RemoteControl 타입이라 할 수도 있고
        // Searchable 타입이라 할 수도 있다.
        RemoteControl rc = tv;
        rc.turnOn();
        Searchable searchable = tv;
        searchable.search("interface");
    }
}
```

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());
```

❖ 인터페이스 사용

404페이지

- sec01.exam06.MyClass

```
package sec01.exam06;

import sec01.exam04.RemoteControl;
import sec01.exam04.Audio;
import sec01.exam04.Television;

public class MyClass {
	// 필드
	RemoteControl rc = new Television();

	// 생성자
	MyClass() {
	}
	MyClass(RemoteControl rc) {
		this.rc = rc;
		rc.turnOn();
		rc.setVolume(5);
	}

	// 메소드
	void methodA() {
		RemoteControl rc = new Audio();
		rc.turnOn();
		rc.setVolume(5);
	}

	void methodB(RemoteControl rc) {
		rc.turnOn();
		rc.setVolume(5);
	}
}
```

❖ 인터페이스 사용

405페이지

- sec01.exam06.MyClassExample

```
package sec01.exam06;

import sec01.exam04.Audio;
import sec01.exam04.Television;

public class MyClassExample {
	public static void main(String[] args) {
		System.out.println("1)---------------");

		MyClass myClass1 = new MyClass();
		myClass1.rc.turnOn();
		myClass1.rc.setVolume(5);

		System.out.println("2)---------------");

		MyClass myClass2 = new MyClass(new Audio());

		System.out.println("3)---------------");

		MyClass myClass3 = new MyClass();
		myClass3.methodA();

		System.out.println("4)---------------");

		MyClass myClass4 = new MyClass();
		myClass4.methodB(new Television());
	}
}
```

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
- implments : 구현 클래스에는 어떤 인터페이스로 사용 가능한지 기술하기 위해 사용
- 인터페이스 사용: 클래스 선언 시 필드, 매개 변수, 로컬 변수로 선언 가능. 구현 객체를 대입.


혼자 공부하는 자바
혼자 공부하는 자바
1:1 과외처럼 공부하고 싶을 때
작심삼일, 하다가 포기하게 될 때
어제 배운 내용이 생각나지 않을 때
신용권 지음
JAVA 8&11 지원
245개의 손코딩 예제
무료 동영상 강의
별책 용어 노트
한빛미디어

Thank You!