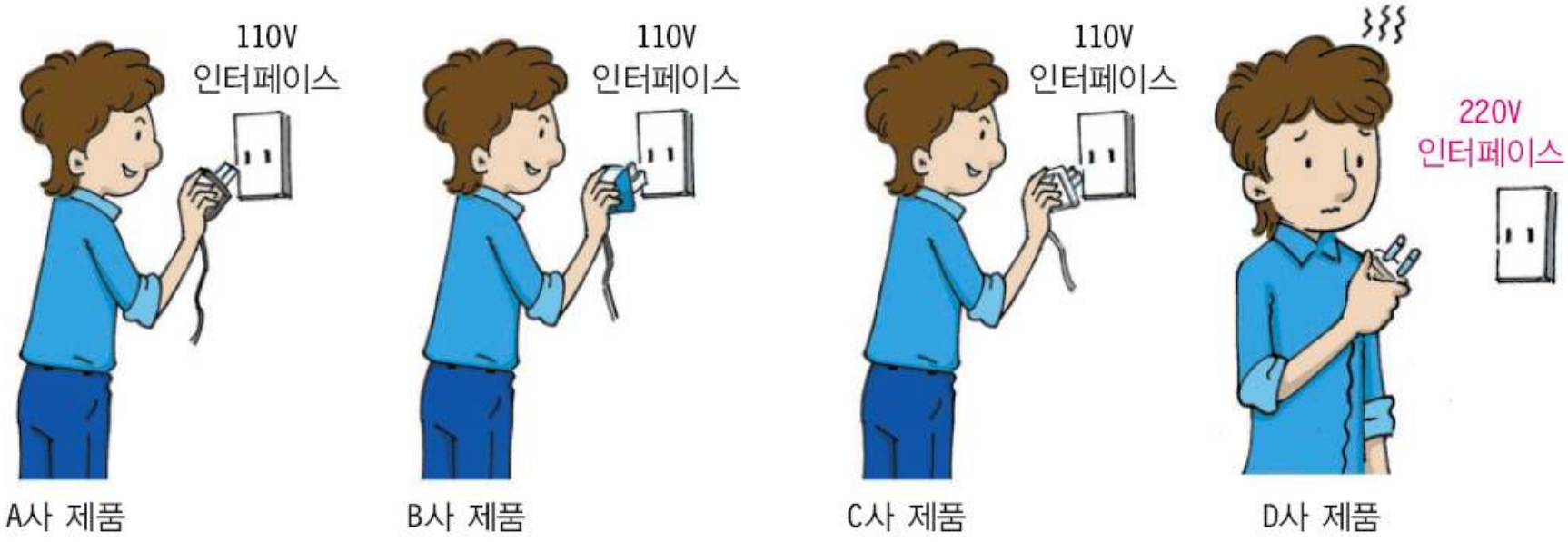
8장. 인터페이스

이것이 자바다(http://cafe.naver.com/thisjava)

한빛미디어 Hanbit Media, Inc.

Contents

인터페이스의 필요성



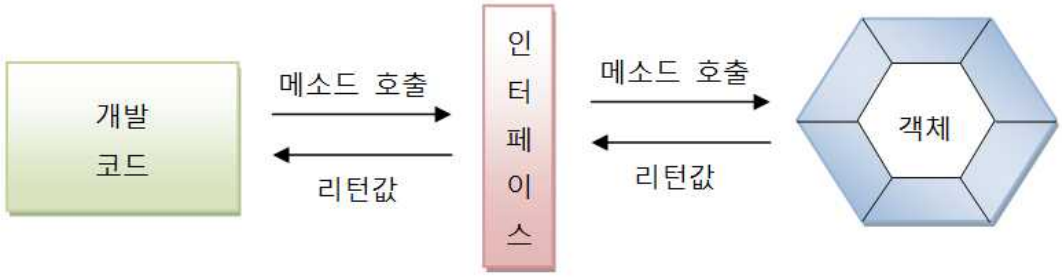
정해진 규격(인터페이스)에
맞기만 하면 연결 가능.
각 회사마다 구현 방법은 다름

정해진 규격(인터페이스)에 맞지
않으면 연결 불가

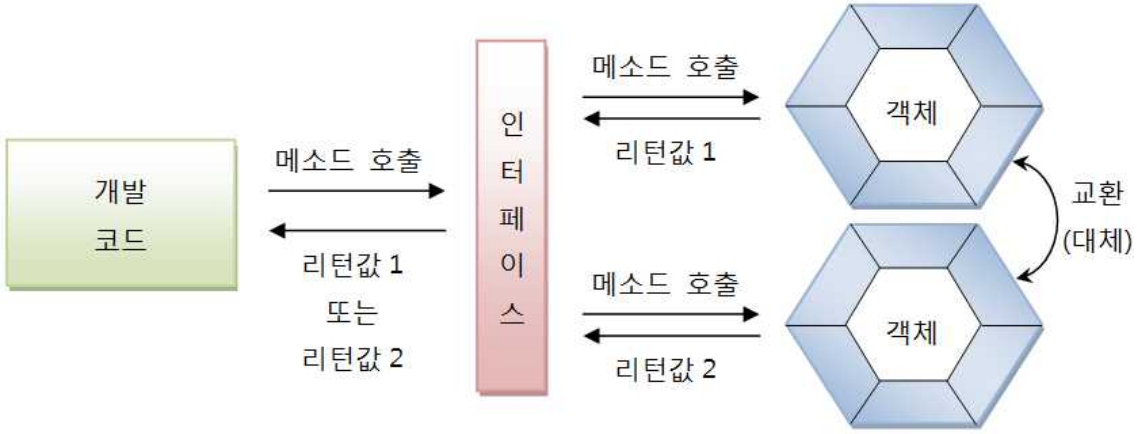
1절. 인터페이스의 역할

❖ 인터페이스란?

- 개발 코드와 객체가 서로 통신하는 접점(형식을 약속)
  - 개발 코드는 **인터페이스의 메소드만 알고 있으면 OK**



- 인터페이스의 역할
  - 개발 코드가 객체에 종속되지 않게 -> 객체 교체할 수 있도록 하는 역할
  - 개발 코드 변경 없이 리턴값 또는 실행 내용이 다양해 질 수 있음 (다형성)



인터페이스 장점

- 기능에 대한 선언과 구현분리
  - 표준화 가능
  - 독립적인 프로그래밍이 가능
  - 개발시간 단축 가능

- Abstract 클래스와의 차이점
  - 모두 abstract 메소드로 구현되어 있음
    - 구현부분이 존재하지 않음

2절. 인터페이스 선언

❖ 인터페이스 선언

- 인터페이스 이름 – 자바 식별자 작성 규칙에 따라 작성

- 소스 파일 생성
  - 인터페이스 이름과 대소문자가 동일한 소스 파일 생성

- 인터페이스 선언

[public] interface 인터페이스명 { ... }

public class 클래스명{...}

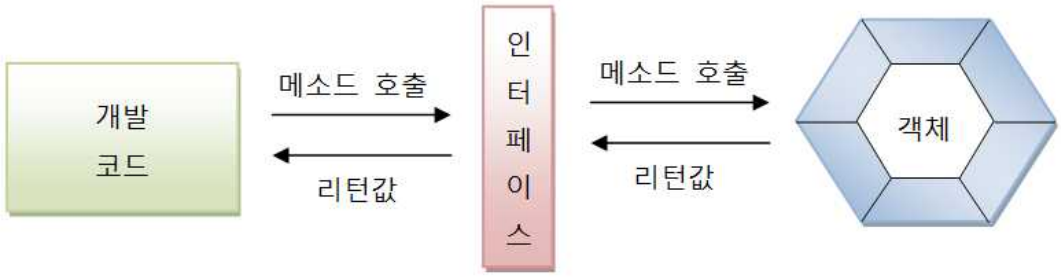
public abstract class 클래스명{...}

```
public interface RemoteControl {
}
public class RemoteControl{
}
public atstract class RemoteControl{
}
```

2절. 인터페이스 선언

❖ 추상 메소드 선언

- 메소드 선언하는 형식에 **abstract 키워드 추가, 구현부 없음**
- **인터페이스 통해 호출된 메소드는 최종적으로 객체에서 실행**
  - **인터페이스의 메소드는 기본적으로 실행 블록이 없는 추상 메소드로 선언**



추상메소드
(메소드 선언부)
(호출 방법만 기술)

재정의된 메소드
(실제 실행 메소드)

[public abstract] 리턴타입 메소드명(매개변수, ..);

public 리턴타입 매소드명(매개변수, ...) {...}

자바 인터페이스

❖ 자바 인터페이스

- 상수와 추상 메소드로만 구성 : **변수 필드 없음**
- 인터페이스 선언
  - interface 키워드로 선언

```
interface PhoneInterface {
    int BUTTONS = 20; // 상수 필드 선언
    void sendCall(); // 추상 메소드
    void receiveCall(); // 추상 메소드
}
```

public interface로서 public 생략 가능

public static final로서 public static final 생략 가능

abstract public 으로서 abstract public 생략 가능

❖ 자바 인터페이스의 특징

- 상수와 추상 메소드로만 구성
  - 메소드 : public abstract 타입으로 생략 가능
  - 상수 : public static final 타입으로 생략 가능
- 인터페이스의 객체 생성 불가

```
~~new~~ PhoneInterface(); // 오류. 인터페이스의 객체를 생성할 수 없다.
```

오류

예제

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

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

실습

- package : com.interf.decare
- 다음 인터페이스를 구현하세요
- PhoneInterface
  - int call(String num);
  - int disConnectCall();

3절. 인터페이스 구현

❖ 구현 클래스 선언

- 자신의 객체가 인터페이스 타입으로 사용할 수 있음
  - implements 키워드로 명시

```
public class 구현클래스명 implements 인터페이스명 {
    //인터페이스에 선언된 추상 메소드의 실체 메소드 선언
}
```

❖ 추상 메소드의 실체 메소드를 작성하는 방법

- 메소드의 선언부가 정확히 일치해야
- 인터페이스의 모든 추상 메소드를 재정의하는 실체 메소드 작성해야
  - 일부만 재정의할 경우, 추상 클래스로 선언 + abstract 키워드 붙임

예제

```
public interface RemoteControl {

    void turnOn();
    void turnOff();
    void setVolume(int volume);

}
```

```
public class RemoteContolImpl 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.");
    }
    //setVolume() 추상 메소드의 실체 메소드
    public void setVolume(int volume) {

        System.out.println("현재 TV 볼륨: " + volume);
    }
}
```

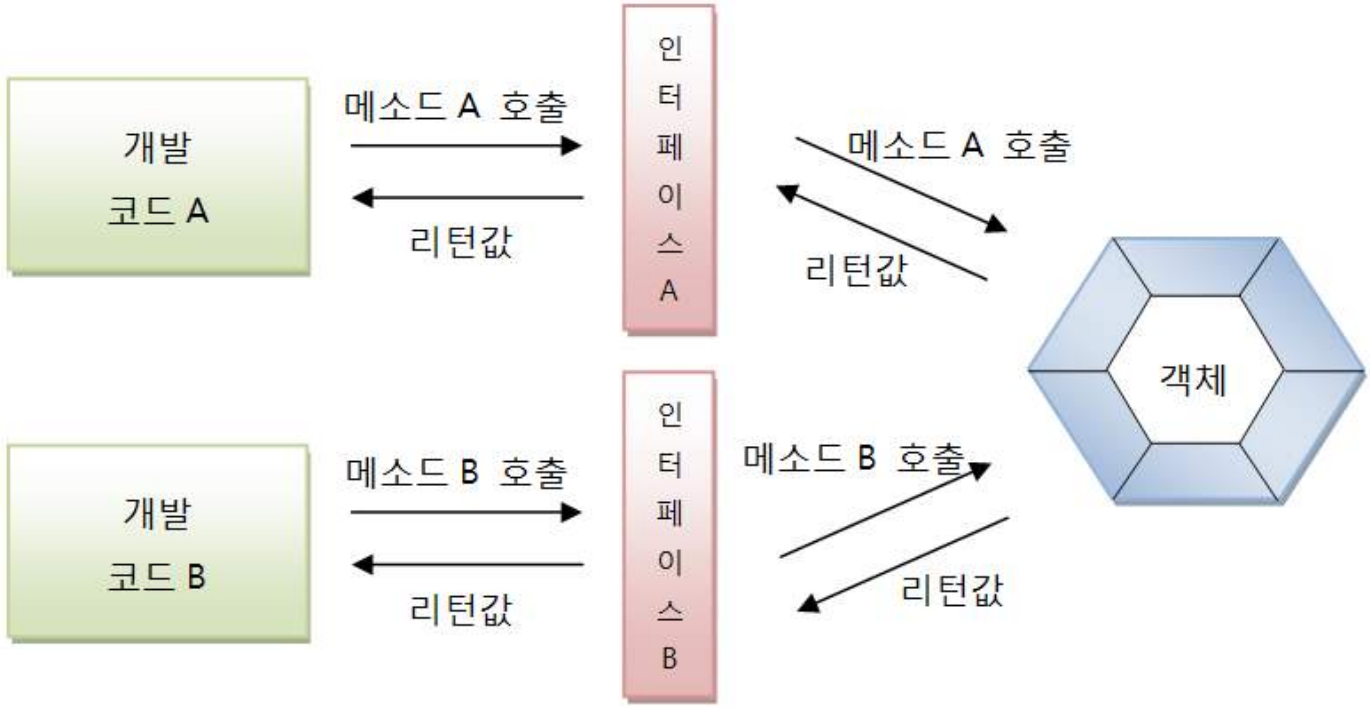
```
public class RemoteControlExample {
    public static void main(String[] args) {
        RemoteControl rc;
        rc = new RemoteContolImpl();
        rc.turnOn();
        rc.turnOff();
        rc.setVolume(1);
    }
}
```

실습

- package : com.interf.impl
- PhoneInterface
  - int call(String num);
  - int disConnectCall();
- PhoneCallImpl : 구현 클래스
- PhoneCallImpl2 : 구현클래스

3절. 인터페이스 구현

❖ 다중 인터페이스 구현 클래스




```
public class 구현클래스명 implements 인터페이스 A, 인터페이스 B {
    //인터페이스 A 에 선언된 추상 메소드의 실체 메소드 선언
    //인터페이스 B 에 선언된 추상 메소드의 실체 메소드 선언
}
```

4절. 인터페이스 사용

❖ 추상 메소드 사용

```
RemoteControl rc = new Television();
rc.turnOn();    → Television 의 turnOn() 실행
rc.turnOff();   → Television 의 turnOff() 실행
```

6절. 인터페이스 상속

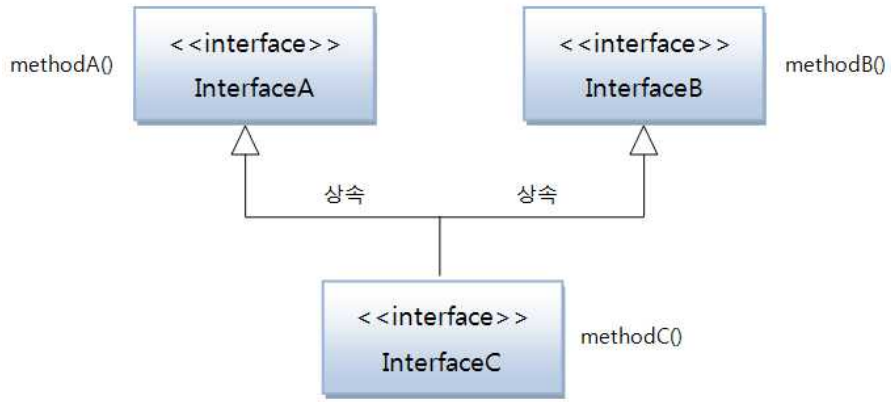
❖ 인터페이스간 상속 가능

```
public interface 하위인터페이스 extends 상위인터페이스1, 상위인터페이스2 { ... }
```

- **하위 인터페이스 구현 클래스는 아래 추상 메소드를 모두 재정의해야**
  - **하위 인터페이스의 추상 메소드**
  - **상위 인터페이스1의 추상 메소드**
  - **상위 인터페이스2의 추상 메소드**

```
하위인터페이스 변수 = new 구현클래스(...);
상위인터페이스1 변수 = new 구현클래스(...);
상위인터페이스2 변수 = new 구현클래스(...);
```

- **인터페이스 자동 타입 변환**
  - **해당 타입의 인터페이스에 선언된 메소드만 호출 가능**



인터페이스 상속

❖ **인터페이스 간에 상속 가능**
- **인터페이스를 상속하여 확장된 인터페이스 작성 가능**
- **extends 키워드로 상속 선언**
  - 예)

```
Interface Phone
{
   void call();
   void disConnectCall();
}
interface MobilePhone extends  Phone {
   void sendSMS();        // 새로운 추상 메소드 추가
   void receiveSMS();     // 새로운 추상 메소드 추가
}
```

❖ **인터페이스 다중 상속 허용**
  - 예)

```
interface SmartPhone extends  MobilePhone, AppFunction {
                .....
}
```

예제 (상속)

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

```
public interface SettopRemoteControl extends RemoteControl {

    public void viewVod(String name);
}
```

```
public class SettopRemoteControlImpl implements SettopRemoteControl{

    public void turnOn() {
        System.out.println("TV를 켭니다.");
    }
    public void turnOff() {
        System.out.println("TV를 끕니다.");
    }
    public void setVolume(int volume) {

        System.out.println("현재 TV 볼륨: " + volume);
    }

    public void viewVod(String name) {
        System.out.println(name + "를 시청합니다.");
    }

}
```

```
public class RemotecontrolClient {
    public static void main(String[] args)
    {

        SettopRemoteControl      brc = new  SettopRemoteControlImpl();
        brc.turnOn();
        brc.turnOff();
        brc.setVolume(5);
        brc.viewVod("Tor");
    }
}
```

예제

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

```
public interface SmartRemoteControl {
    void search(String url);
    void viewVod(String name);
}
```

```
public class SmartRemoteControlImpl implements RemoteControl, SmartRemoteControl {

    public void turnOn() {
        System.out.println("TV를 켭니다.");
    }
    public void turnOff() {
        System.out.println("TV를 끕니다.");
    }
    public void setVolume(int volume) {

        System.out.println("현재 TV 볼륨: " + volume);
    }
    public void search(String url) {
        System.out.println(url + "를 검색합니다.");
    }
    public void viewVod(String name) {
        System.out.println(name + "를 시청합니다.");
    }
}
```

```
public class BtbRemoteControl {
    public static void main(String[] args)
    {
        SmartRemoteControlImpl scr = new SmartRemoteControlImpl();
        scr.turnOn();
        scr.turnOff();
        scr.setVolume(5);
        scr.search("www.google.com");
        scr.viewVod("Tor");
    }

}
```

실습

- package : com.interf.inherit
- PhoneCallInterface : interface
  - int call(String num);
  - int disConnectCall();

- AppInterface: interface
  - public void appRun();
  - public void appClose();

- SmartPhone
  - PhoneCallInterface, AppInterface
  - 위 두개의 인터페이스를 하나의 클래스에서 구현

실습

- package : com.interf.inherit
- PaymentInterface : interface
  - int pay(int money);
  - int checkBalance();
- Spay: 구현 class
  - int pay(int money);
  - int checkBalance(int accountNum);
- Kpay : 구현 class
  - int pay(int money);
  - int checkBalance(int accountNum);
- PayManager : class
  - PayInterface 객체 생성
  - int buy(int money, int num)
    - PaymentInterface pay() 호출
  - int checkBalance(int accountNum)
    - PaymentInterface checkBalance() 호출
- PaymentClient : class
  - PayManager 객체 생성
  - 객체 생성 및 메소드 호출

Thank You !

이것이 자바다(http://cafe.naver.com/thisjava)

한빛미디어 Hanbit Media, Inc.