Interface

경복대학교
소프트웨어융합과
배희호 교수
010-2369-4112
031-570-9600
hhbae@kbu.ac.kr

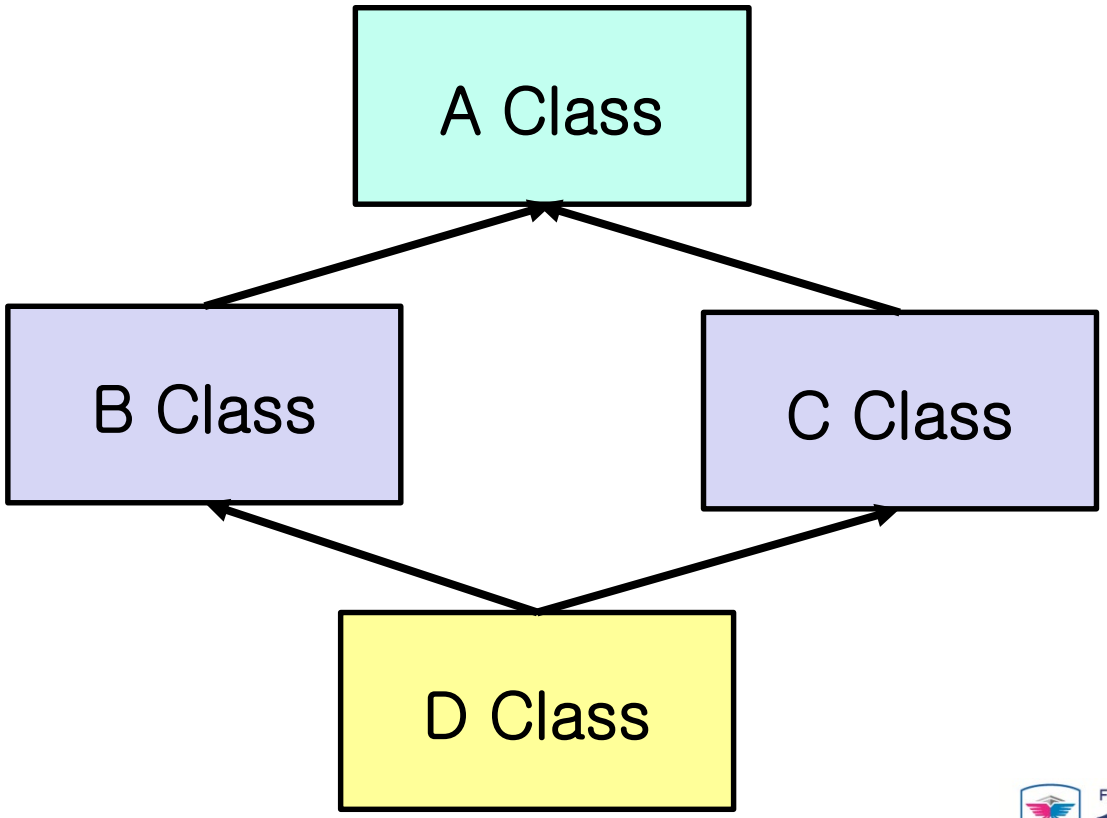
Smart Watch

- Smart Device(예: Smart Phone, Tablet, Smart Watch 등)는 공통적으로 충전(charge()) 기능을 가지고 있고, 블루투스를 지원하는 기기들은 연결(connectBluetooth()) 기능도 제공함
- 새로운 시계 기능이 있는 Smart Watch Class를 만들 때, 이 두 가지 기능을 모두 구현하려면 어떻게 할까 ?

Futuristic Innovator
KBU
京福大學校
KYUNGBOK UNIVERSITY

Diamond Problem

- JAVA에서의 Diamond Problem은 하나의 Class가 두 개의 부모 Class를 상속받고, 그 부모들이 같은 상위 Class를 상속받을 때 발생하는 Ambiguity(모호성) 문제



Futuristic Innovator
KBU
KYUNGBOK UNIVERSITY

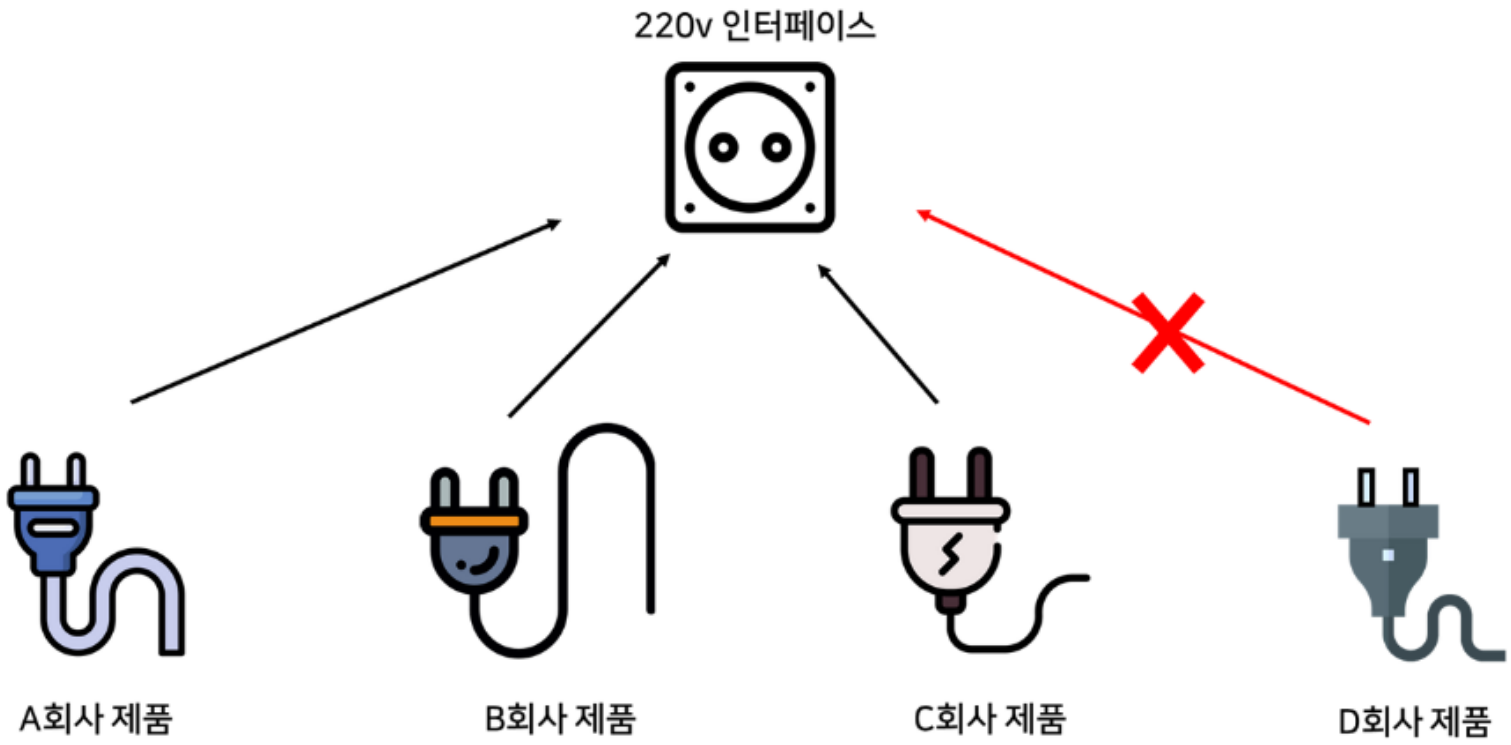
Interface

- 서로 다른 Devices들이 연결되어서 상호 Data를 주고받는 규격을 의미

USB 인터페이스

Interface

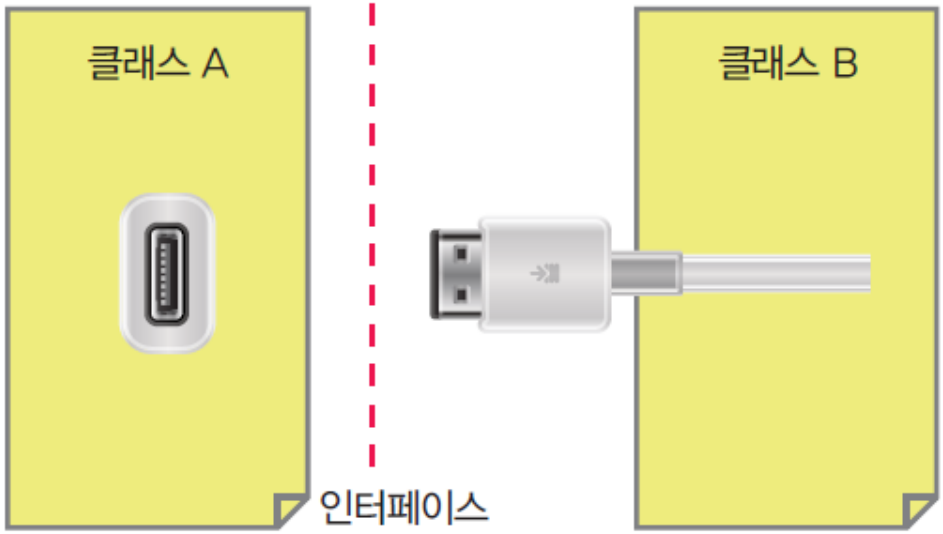
- Interface 개념



Futuristic Innovator
KBU
京福大學校
KYUNGBOK UNIVERSITY

Interface

- Coding의 단위가 Class이므로 Class와 Class사이의 상호작용의 규격을 나타낸 것이 Interface



Interface

- C++ 언어에서는 여러 개의 Super Class로부터 Sub Class가 상속을 받는 다중 상속이 가능하나 JAVA에서는 한 개의 Super Class로부터 상속을 받는 단일 상속만을 허용
  - JAVA에서는 Interface를 사용하여 다중 상속이 가능 함
- Interface는 Abstract Class의 특별한 형태로서 역시 구현과 사용을 분리하며, 상수와 구현하지 않은 Method Header의 선언만으로 구성
- Abstract Class는 Abstract Method외에 일반 Method와 변수를 가질 수 있으나 Interface는 일반 Method와 변수는 사용할 수 없음
- Interface는 Method의 명세만을 갖고 있어서 Class 간에 관련이 없는 Class들이 공통으로 Interface를 사용하기 위한 용도임
- 몸체 없는 Method의 Header만으로 구성된 Interface의 모든 Method는 반드시 구현해야 함

Futuristic Innovator
KBU
KYUNGBOK UNIVERSITY

Interface

- Interface 선언
  - Interface의 접근 제한자는 public이고 Abstract Class이며 일반 Method와 변수는 가질 수 없고 상수와 Method의 Header 선언만이 가능
  - Interface에 선언된 Method 역시 접근 제한자는 public이고 abstract가 생략된 Method 임
  - 변수는 public 접근 제한자이고 수정자는 static final로 지정

```
[접근 제한자] interface 인터페이스 이름 [extends 인터페이스 이름]{
    [접근 제한자] 데이터형 상수 이름 = 상수 값;
    [접근 제한자] 반환형 메소드 이름([매개변수 ....]);
}
```

Futuristic Innovator
KBU 京福大學校
KYUNGBOK UNIVERSITY

Interface

- Interface 선언 형식

```
interface 인터페이스 이름 {
    속성 변수 정의;
    Method 정의;
 }
```

- Class에서 Interface에 정의된 Method 구현

```
interface D {
    클래스 B의 상수와 Method 정의;
}

class C extends A implements D {
    인터페이스 D에서 정의된 Method 구현;
}
```

Futuristic Innovator
KBU 東福大學校
KYUNGBOK UNIVERSITY

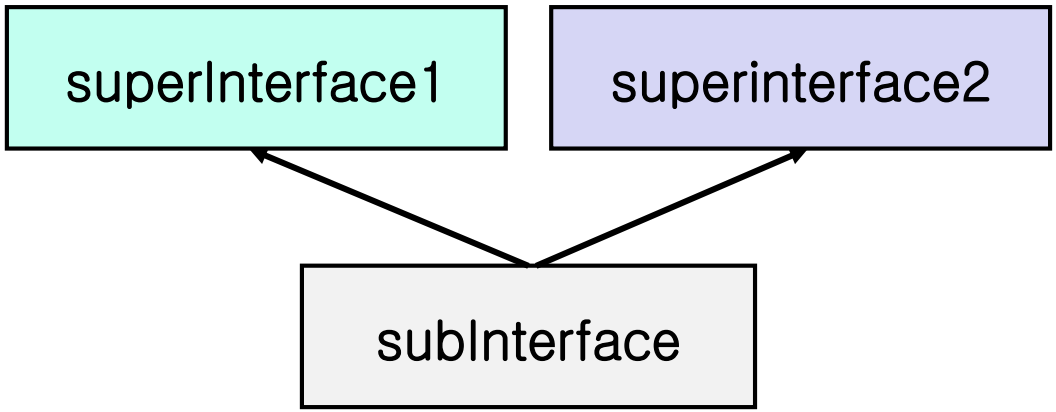
Interface

- 정의
  - Methods 선언들의 모임
    - Interface Method는 이를 상속 받는 Class에서 구현해야 함
    - public abstract
  - Variables은 상수 만을 갖을 수 있음
    - public final static (Class 상수 선언)

Futuristic Innovator
KBU
KYUNGBOK UNIVERSITY

Interface

- Interface에서의 계층 구조
  - Interface 간의 다중 상속을 허용
  - JAVA는 Class 다중 상속을 허용치 않음
  - Class는 여러 개의 Interface를 구현할 수 있음
    - JAVA에서는 Interface를 통해 다중 상속 가능



Futuristic Innovator
KBU
京福大學校
KYUNGBOK UNIVERSITY

Interface

- “extends”라는 예약어를 통해 여러 개의 다른 Interface로부터 상속받을 수 있음
  - Class는 단일 상속을 전제로 하지만 Interface 그 자체는 다중 상속을 받을 수 있음

- 확장 Interface 선언 형식

```
interface  확장 인터페이스 이름
                                extends 확장 대상 인터페이스 이름들 {
    멤버 변수 정의;
    메소드 정의;
}
```

Futuristic Innovator
KBU 京福大學校
KYUNGBOK UNIVERSITY

Interface

- Interface 구현
  - Interface의 Method를 구현하는 Class는 Interface에서 선언한 모든 Method의 Body를 구현해야 하며 선언된 Method를 모두 구현하지 않으면 Error가 발생
  - Interface의 Method를 구현하는 Class는 Class 이름과 함께 다음의 형식에 따라 implements를 사용하여 Interface의 관계를 나타냄

Futuristic Innovator
KBU 京福大學校
KYUNGBOK UNIVERSITY

Interface

- Interface 구현

```
[접근 제한자] class 클래스 이름 [extends 슈퍼 클래스 이름]
                    implements 인터페이스 이름, [인터페이스 이름, ... ] {
   변수 선언

   메소드

   [접근 제한자] 반환형 메소드 이름([매개변수....]) {
      // 오버라이딩된 메소드 구현
   }
   ..............
}
```

Futuristic Innovator
KBU 京福大學校 KYUNGBOK UNIVERSITY

Interface

- Interface 특징
  - Interface는 상수와 Body가 없는 Method Header만을 갖는 특수한 Abstract Class
  - 반드시 Sub Class에서 Interface에 선언된 Method를 모두 구현해야 함
  - 이와 같이 함으로써 새로운 Sub Type을 정의하여 사용할 수 있음
  - Abstract Class의 상속은 단일 상속임에 비하여 Interface는 다중 상속이 가능

Futuristic Innovator
KBU 京福大學校
KYUNGBOK UNIVERSITY

Interface

- Interface 특징

항목	Abstract Class	Interface
상속	단일 상속	다중 상속
Method	추상 메소드, 일반 메소드	추상 메소드
Method 구현	추상 메소드를 선택하여 구현	모든 추상 메소드를 구현
변수	일반 변수	final static 변수(상수)
Object 생성	생성할 수 없음	생성할 수 없음
형식	extends	implements
Class 관계	Is – a 관계	has – a 관계

Futuristic Innovator
KBU 京福大學校
KYUNGBOK UNIVERSITY

Interface

- Sub Class가 여러 Super Class를 상속받을 수 있다면, 다양한 동작을 수행할 수 있다는 장점을 가지게 될 것임
- Class를 이용하여 다중 상속을 할 경우 Method 출처의 모호성 등 여러 가지 문제가 발생할 수 있어 JAVA에서는 Class를 통한 다중 상속은 지원하지 않음
- 다중 상속의 이점을 버릴 수는 없기에 JAVA에서는 Interface라는 것을 통해 다중 상속을 지원함
- Interface란 다른 Class를 작성할 때 기본이 되는 틀(template)을 제공하면서, 다른 Class 사이의 중간 매개 역할까지 담당하는 일종의 Abstract Class를 의미함
- Abstract Class는 Abstract Method뿐만 아니라 생성자, Field, 일반 Method도 포함할 수 있음
- Interface는 오로지 추상 Method와 상수 만을 포함할 수 있음

Futuristic Innovator
KBU 京福大學校
KYUNGBOK UNIVERSITY

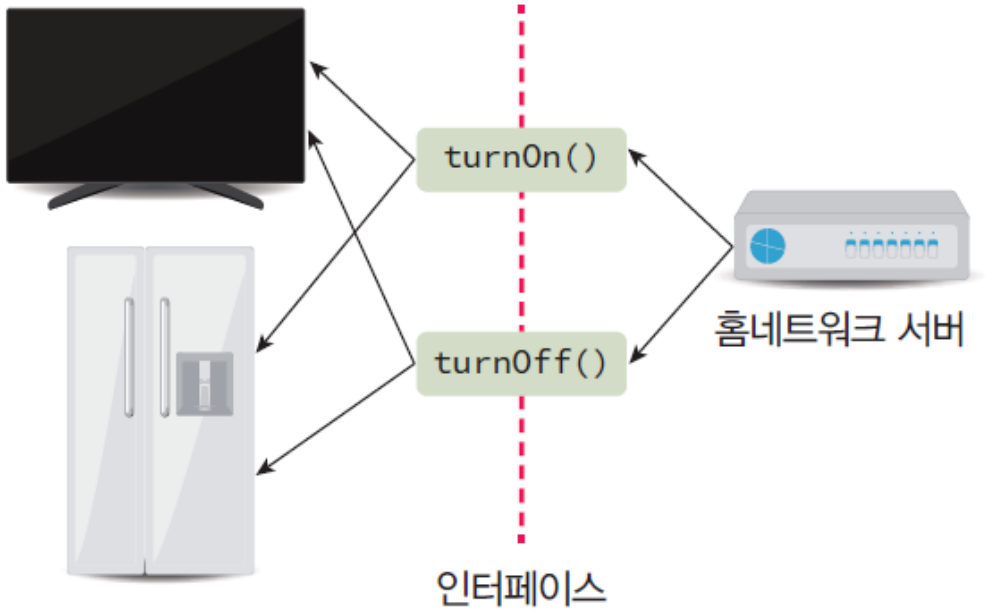
Interface

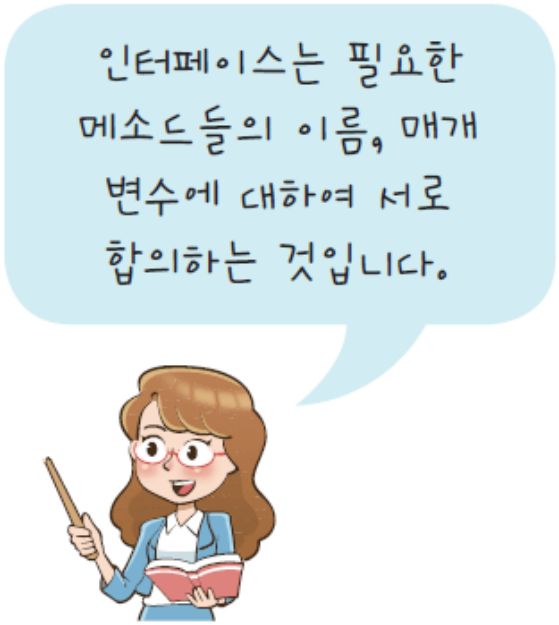
- Abstract Class는 일반 Method와 Abstract Method를 둘 다 가질 수 있는 반면에, Interface는 오로지 Abstract Method와 상수 만을 가짐. 즉 Logic은 작성할 수 없음
- Interface 내에 존재하는 Method는 무조건 public abstract 로 선언
- Interface 내에 존재하는 변수는 무조건 public static final 로 선언
- Interface는 다중 상속과 비슷한 기능을 제공

Futuristic Innovator
KBU
京福大學校
KYUNGBOK UNIVERSITY

Interface

- 예)스마트 홈 시스템(Smart Home System)





Futuristic Innovator
KBU 京福大學校
KYUNGBOK UNIVERSITY

Interface

- Interface 선언
  - Interface를 정의하는 것은 Class를 정의하는 것과 유사

형식

```
public interface 인터페이스_이름 {
    반환형      추상메소드1(...);
    반환형      추상메소드2(...);
    ...
}
```


```
public interface RemoteControl {
        // 추상 메소드 정의
        public void turnOn();    // 가전 제품을 켠다.
        public void turnOff();   // 가전 제품을 끈다.
}
```

KBU 慶福大學校 KYUNGBOK UNIVERSITY

Interface

- Interface 선언
  - Interface를 선언할 때에는 접근 제어자와 함께 interface Keyword를 사용하면 됨

```
접근 제어자 interface 인터페이스  이름 {
    public static final 타입 상수이름 = 값;
            …
    public abstract 메소드 이름(매개변수 목록);
            …
}
```

  - 단, Class와는 달리 Interface의 모든 Field는 public static final이어야 하며, 모든 Method는 public abstract 이어야 함

Futuristic Innovator
KBU 京福大學校
KYUNGBOK UNIVERSITY

Interface

- Interface 선언

```
접근 제어자 interface 인터페이스  이름 {
    public static final 타입 상수이름 = 값;
        …
    public abstract 메소드 이름(매개변수 목록);
        …
}
```

  - 이 부분은 모든 Interface에 공통으로 적용되는 부분이므로 이 제어자는 생략할 수 있음
  - 이렇게 생략된 제어자는 Compile 시 JAVA Compiler가 자동으로 추가해 줌

Futuristic Innovator
KBU 京福大學校 KYUNGBOK UNIVERSITY

Interface

- 모든 Method가 Abstract Method인 Class
- Interface는 상수와 Abstract Method로만 구성되어 있음
- 변수 Field 선언 불 필요
- Interface 선언
  - interface Keyword로 선언
  - 예) public interface SerialDriver {…}
- Interface 특징
  - Interface의 Method
    - public abstract Type으로 생략 가능
  - Interface의 상수
    - public static final Type으로 생략 가능
  - Interface의 Object 생성 불가
  - Interface에 대한 Reference 변수는 선언 가능

Futuristic Innovator
KBU 京福大學校
KYUNGBOK UNIVERSITY

Interface

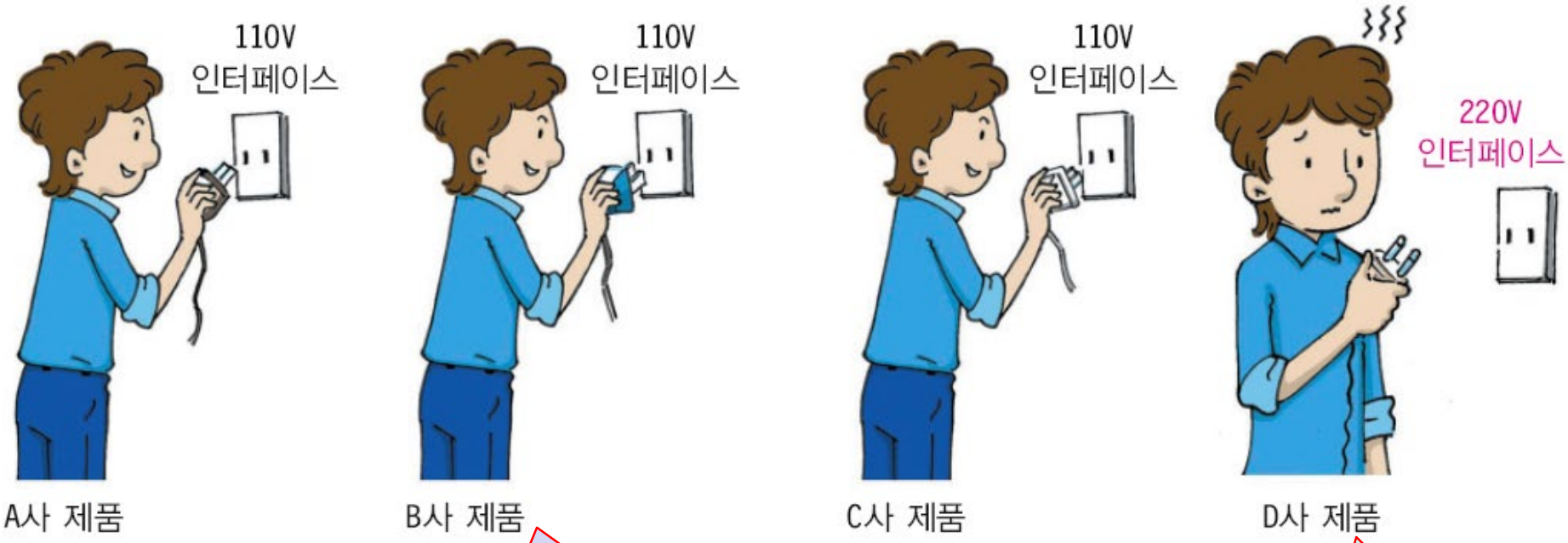
```
public interface 인터페이스_이름 {
    반환형 추상 메소드1(……);
    반환형 추상 메소드2(……);
        ……
}
```

인터페이스 내부에
추상 메소드 정의

```
public class 클래스_이름 implements 인터페이스_이름 {
    반환형 추상 메소드1(……) {
            ……
    }
    반환형 추상 메소드2(……) {
        ……
    }
}
```

인터페이스를 구현한
클래스는
추상 메소드의 몸체를
구현 해야 함

Interface의 필요성



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

정해진 규격 (인터페이스)에 맞지 않으면 연결 불가

Futuristic Innovator
KBU
東福大學校
KYUNGBOK UNIVERSITY

Interface의 필요성

- Interface를 이용하여 다중 상속 구현
  - JAVA에서 Class 다중 상속 불가

- Interface는 명세서와 같음
  - Interface만 선언하고 구현을 분리하여, 작업자마다 다양한 구현을 할 수 있음
  - 사용자는 구현의 내용은 모르지만, Interface에 선언된 Method가 구현되어있기 때문에 호출하여 사용하기만 하면 됨
    - 110v 전원 아울렛처럼 규격에 맞기만 하면, 어떻게 만들어졌는지 알 필요 없이 전원 연결에 사용하기만 하면 됨

Futuristic Innovator
KBU 京福大學校
KYUNGBOK UNIVERSITY

Interface의 구현

- Interface는 Abstract Class와 마찬가지로 자신이 직접 Interface를 생성할 수는 없음
- 따라서 Interface가 포함하고 있는 Abstract Method를 구현해 줄 Class를 작성해야만 함

```
class 클래스 이름 implements 인터페이스 이름 {
        …
}
```

- 만약 모든 Abstract Method를 구현하지 않는다면, abstract Keyword를 사용하여 Abstract Class로 선언해야 함

Futuristic Innovator
KBU 京福大學校 KYUNGBOK UNIVERSITY

Interface의 장점

- Interface를 사용하면 다중 상속이 가능할 뿐만 아니라 다음과 같은 장점을 가질 수 있음
  - 대규모 Project 개발 시 일관되고 정형화된 개발을 위한 표준화가 가능
  - Class의 작성과 Interface의 구현을 동시에 진행할 수 있으므로, 개발 시간을 단축할 수 있음
  - Class와 Class 간의 관계를 Interface로 연결하면, Class마다 독립적인 Programming이 가능

Futuristic Innovator
KBU 京福大學校
KYUNGBOK UNIVERSITY

Interface 특징

- Interface를 상속(구현)하는 Class
  - Interface와 상위 Interface의 모든 상수를 상속 받고 public method를 구현해야 함
  - Class 상속과 Interface들의 구현을 동시에 선언 가능
    - extends class_name
    - implements interface_names_list
  - 완벽한 추상화를 위해 Interface에 선언된 public method들을 통해서만 접근 가능하도록 Modeling 가능
  - 같은 Interface를 상속받아, Class마다 다르게 구현 가능

Futuristic Innovator
KBU 京福大學校
KYUNGBOK UNIVERSITY

Interface 상속

- Class간의 상속과 동일한 방법으로 Interface간에 상속을 지원
- 새로운 Interface를 정의할 때, 기존의 Interface를 상속 받을 수 있음
- 기존의 Interface를 상속 받아 새로운 Interface를 정의할 때, Class와 마찬가지로 extends라는 Keyword를 사용함
- 기존의 상위 Interface를 Super Interface라고 부름
- Interface를 상속하는 형식은

```
public interface interface_name
                    extends inteface_name [, interface_name … ] {
   // constants declarations
   // method declarations
}
```

Futuristic Innovator
KBU 京福大學校
KYUNGBOK UNIVERSITY

Interface 상속

- Class가 계층 관계를 가질 수 있듯이, Interface도 Interface간에 상속 관계에 따라 계층 구조를 가질 수 있음
- Class의 계층 구조에서 Class는 하나의 상위 Class로부터 상속 받을 수 있지만, Interface는 여러 개의 Interface로부터 상속 받을 수 있음
- Class 계층에서와 마찬가지로 하위 Interface에서 상위 Interface와 동일한 이름을 가진 변수를 정의하면, 상위 Interface의 변수가 가려짐

Futuristic Innovator
KBU
京福大學校
KYUNGBOK UNIVERSITY

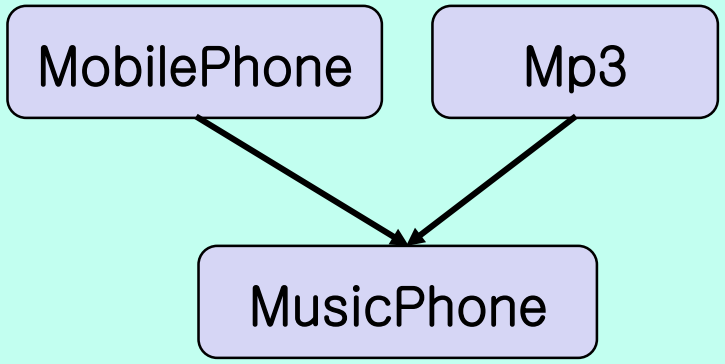
Interface 상속

- Interface 간에도 상속 가능
  - Interface를 상속하여 확장된 Interface 작성 가능
- Interface 다중 상속 허용

```
interface MobilePhone {
        boolean sendCall();
        boolean receiveCall();
        boolean sendSMS();
        boolean receiveSMS();
}

interface MP3 {
        void play();
        void stop();
}

interface MusicPhone extends MobilePhone, MP3 {
        void playMP3RingTone();
}
```



Interface 상속

```
public interface Number {
    int  Lucky = 7;
    int  Bad = 13;
}

interface ExtNumber extends Number {
    int Perfect=12;
    int Death=4;
    int Lucky=16;    // Number의 Lucky와 동일한 이름
}
```

Futuristic Innovator
KBU 京福大學校
KYUNGBOK UNIVERSITY

Interface 상속

```
public class Main implements ExtNumber {

    public static void main(String[] args) {
        System.out.println("Lucky = " + Lucky );
        System.out.println("Bad = " + Bad );
        System.out.println("Perfect = " + Perfect );
        System.out.println("Death = " + Death);
    }
}
```

```
Lucky = 16
Bad = 13
Perfect = 12
Death = 4
```

Futuristic Innovator
KBU 京福大學校
KYUNGBOK UNIVERSITY

Interface와 Abstract Class

- Abstract Class
  - Class 구현부 내부에 Abstract Method가 하나 이상 포함되거나 abstract로 정의된 경우를 말함
  - Abstract Class는 new 연산자를 사용하여 Object를 생성할 수 없음
  - Abstract Class(부모)와 일반 Class(자식)는 상속의 관계에 놓여 있음
  - Abstract Class는 새로운 일반 Class를 위한 부모 Class의 용도로만 사용
  - 일반 Class들의 Field와 Method를 통일하여 일반 Class 작성 시 시간을 절약할 수 있음
  - Abstract Class는 단일 상속만 가능하며 일반 변수를 가질 수 있음
  - Abstract Class는 동일한 부모를 가지는 Class를 묶는 개념으로 상속을 받아서 기능을 확장시키는 것이 목적

Futuristic Innovator
KBU
慶福大學校
KYUNGBOK UNIVERSITY

Interface와 Abstract Class

- Interface
  - Interface는 모든 Method가 Abstract Method인 경우를 말함
  - Interface스는 Abstract Class보다 한 단계 더 추상화된 Class라고 볼 수 있음
  - Interface에 적는 모든 Method들은 Abstract Method로 간주되기 때문에 abstract을 적지 않음
  - Interface도 Abstract Class와 마찬가지로 new 연산자를 사용하여 Object를 생성할 수 없음
  - Interface는 구현 Object가 같은 동작을 한다는 것을 보장하는 것이 목적
  - Interface는 Abstract Class와 반대로 다중 상속이 가능함

Futuristic Innovator
KBU 京福大學校
KYUNGBOK UNIVERSITY

Interface와 Abstract Class

- Interface
  - 기존에는 Interface에 일반 Method를 구현할 수 없었지만, JAVA 8 버전부터 default 예약어를 통해 일반 Method 구현이 가능
  - Interface는 static final Field만 가질 수 있음
    - Field를 선언할 때는 public static final이 생략되었다고 생각 함
    - public static final을 사용하는 이유
      - 구현 객체의 같은 동작을 보장하기 위한 목적
      - Interface의 변수는 스스로 초기화 될 권한이 없음
      - 어떤 Instance도 존재하지 않는 시점이기 때문

Futuristic Innovator
KBU 京福大學校
KYUNGBOK UNIVERSITY

Interface와 Abstract Class

- Interface와 Abstract Class의 공통점
  - Abstract Class와 Interface는 선언만 있고 구현 내용이 없음
  - 그래서 자기 자신이 new를 통해 Object를 생성할 수 없으며, 상속받은 자식만이 Object를 생성할 수 있음
  - 상속받은 자식이 구현을 반드시 하도록 해야 할 때 사용
  - JAVA에서는 type이 지정되어있기 때문에 선언된 type과 자식의 type이 같아야만 함

Futuristic Innovator
KBU 京福大學校
KYUNGBOK UNIVERSITY

Interface와 Abstract Class

비교	내용
Abstract 클래스	✓ 일반 Method 포함 가능 ✓ 상수, 변수 Field 포함 가능 ✓ 모든 Sub Class에 공통된 Method가 있는 경우에는 Abstract Class가 적합
Interface	✓ 모든 Method가 Abstract Method ✓ 상수 필드만 포함 가능 ✓ 다중 상속 지원

Futuristic Innovator
KBU
京福大學校
KYUNGBOK UNIVERSITY

Interface와 Abstract Class

- Interface와 Abstract Class의 차이점

Interface	Abstract Class
구현 Object의 같은 동작을 보장하기 위함	Abstract Class를 상속받아 기능을 이용하고, 확장시키기 위함
구현된 것이 하나도 없는 밑그림만 그려진 “기본 설계도”	부분적으로 완성된 “미완성 설계도”
다중 상속 가능	다중 상속 불가능
Abstract Method만 가능	일반 Method + Abstract Method 가능
상수 + 추상 메소드 형태	일반 변수(가능) + 일반 메소드(가능) + 추상 메소드 형태
생성자와 일반 변수를 가질 수 없음	생성자와 일반 변수 모두 가질 수 있음
implments	extends
Method 선언만 가능	Method의 부분적인 구현이 가능

Futuristic Innovator
KBU
京福大學校
KYUNGBOK UNIVERSITY

Interface와 Abstract Class

- Interface와 Abstract Class의 차이점

Interface	Abstract Class
클래스가 아님(interface로 선언)	클래스임 (abstract class로 선언)
public abstract 추상 메소드만 가질 수 있음 (일반 메소드는 가질 수 없음)	일반 메소드와 추상 메소드를 모두 가질 수 있음
구현을 하며 implements로 서브 클래스 정의	확장을 하며 extends로 서브 클래스로 정의
다중 상속도 가능	단일 상속만 가능
static final 형태의 상수만 가질 수 있음	변수와 상수를 모두 가질 수 있음
모든 추상 메소드는 객체 생성을 위한 서브 클래스에서 반드시 구현되어야 함 업 캐스팅이 가능	

Futuristic Innovator
KBU
KYUNGBOK UNIVERSITY

Interface

- 상수형 변수와 Abstract Method들로만 구성된 Program의 단위
  - 선언된 모든 변수
    - 묵시적으로 public, static, final의 속성을 갖고, 반드시 초기화해야 함
  - Method
    - public, abstract의 속성을 갖음
- Abstract Method를 가지므로 직접 Object 생성할 수 없음
  - 자료형으로 사용은 가능
- 여러 개의 Interface로부터 다중 상속이 가능
- 해당 Interface를 구현해주는 Class에 의해 간접적으로 사용

Futuristic Innovator
KBU 京福大學校
KYUNGBOK UNIVERSITY

default Method

- JAVA 8부터 Interface에서 default Method를 사용할 수 있음
- 이전에는 Interface에 Abstract Method만 선언할 수 있었지만, default Keyword를 사용하면 구현이 포함된 Method를 선언할 수 있음

- 기능
  - Interface에서도 Method의 기본 구현 제공 가능
  - 기존 Interface를 변경해도 하위 호환성 유지
  - 다중 Interface 구현 시 모호성 해결 가능

Futuristic Innovator
KBU
京福大學校
KYUNGBOK UNIVERSITY