8장. 인터페이스

8.1. 인터페이스 개념

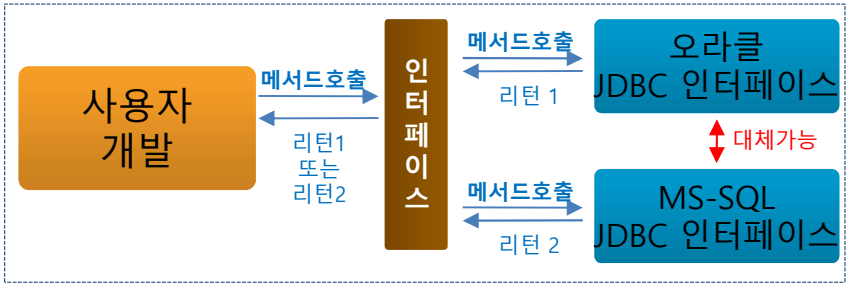
- 인터페이스는 객체의 사용방법을 정의한 타입임.
- 개발코드를 수정하지 않고, 객체를 변경하면서 사용 가능함. (환경변화에 대응이 쉬움)

인터페이스 개념

인터페이스 개념

❖ 인터페이스는 사용자 개발코드와 통신하는 접점임.
개발코드는 인터페이스 구현객체의 내용을 알지 못해도 사용 가능함. (인터페이스에 명시된 메서드만 알아도 됨.)

❖ 주요역할
- 개발코드가 객체에 종속되지 않으므로 객체교체 용이함.
- 개발코드 변경없이 리턴 및 실행내용 변경가능 (다형성 지원)



인터페이스 선언

❖ 인터페이스 사용법
- public interface 인터페이스 명 { ... }
- 인터페이스명은 클래스명 규칙과 동일함.

❖ 인터페이스 구성 : 상수, 메서드
- 상수
. 상수 필드만 선언가능 : static final의 형태로 사용.
. 상수명은 대문자로 작성하는 것이 관례임.
. 선언과 동시에 초기값 지정 필요함
(static 블럭이 없음)

- 메서드 : 추상메서드 선언, default 메서드, static 메서드
. 메서드 구현없이 선언만으로 처리함.
(설계자가 구현해야 주요기능에 대해 선언함)
. JAVA8 버전 이 후 지원사항 (예외적인 사항)
- Default 메서드 / 정적 메서드로 선언된 것은
실행내용까지 있음.

8.2. 인터페이스 구현

- 구현 객체 : 인터페이스의 추상메소드에 대한 실체 메소드를 가진 객체를 의미함.
- 구현 클래스 : 구현 객체를 생성하는 클래스

인터페이스 구현

인터페이스 문법

인터페이스

```
public interface 인터페이스명{

  public static int 상수명 = 값;     // 상수선언
  void method 1( );                  // 추상메서드 선언
  default void method2( ) {          // default 메서드 정의
    /* 실행문 */
  }
  static void method3( ) {           // 정적 메서드 정의
    /* 실행문 */
  }
}
```

구현 클래스

```
public class 클래스명 implements 인터페이스명 {
  void method 1( ) {          // 구현 메서드
     /* 실행문 */             // 구현 내용
  }
 인터페이스명 구현객체 = new 클래스명;
}
```

인터페이스 사용예제

인터페이스

```
public interface RemoteContol{

  public static int MIN_VOL = 0;   // 상수선언
  void turnOn( );                  // 추상메서드 선언
  default void setMute() {         // default 메서드 정의
    System.out.print ("볼륨 : 0");
  }
  static void changeBatt( ) {      // 정적 메서드 정의
    System.out.print ("배터리 교체");
  }
}
```

구현 클래스

```
public class Television implements RemoteContol{
  int volume = 10;
  void turnOn( ) {                 // 구현 메서드
     System.out.print ("TV를 켜다.);
  }
  RemoteContol rc = new Television ();
  rc.turnOn( );                    // "TV를 켜다" 출력됨.
  rc.setMute() ;                   // volumn 값이 0이 됨.
  RemoteContol.changeBatt();       // "배터리 교체" 출력됨.
}
```

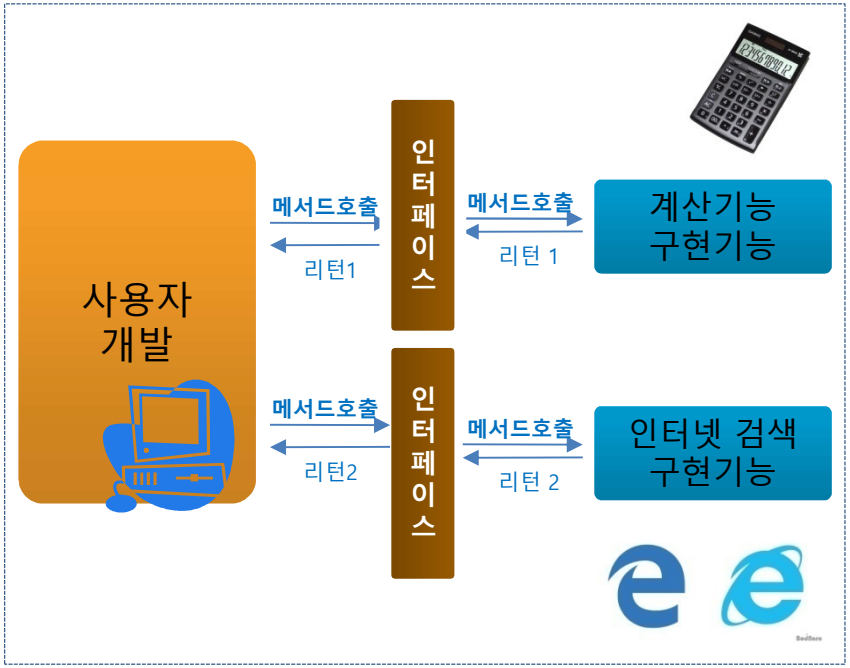
8.3. 인터페이스 활용 (다중 인터페이스)

- 객체는 다양한 인터페이스를 구현할 수 있음
- 2개 이상의 인터페이스를 implement하여 실제 구현해야 함.

인터페이스 활용 - 다중인터페이스

다중 인터페이스 개념

❖ 컴퓨터라는 객체는 계산기능 뿐만 아니라
인터넷 검색 기능까지도 할 수 있음



다중 인터페이스 사용예제

계산기 인터페이스

```
public interface Calc{
  int sum(int a, int b);        // 덧셈기능
}
```

인터넷 인터페이스

```
public interface Internet{
  void search( );               // 검색기능
}
```

구현 클래스

```
public class Computer implements Calc, Internnet {
  int sum(int a, int b) {
     return a+b;
  }
  void search( ) {
     System.out.println ("인터넷 검색");
  }
  Computer com = new Computer();
  int sum = com.sum(2,3);
  com.search( );
}
```

8.4. 다형성과 타입변환

- 다형성 : 하나의 타입에 여러가지 객체를 대입해서 다양한 실행결과를 얻는 것.
- 타입변환 : 타입에 대입되는 객체의 형태로 변환되는 것을 의미함.

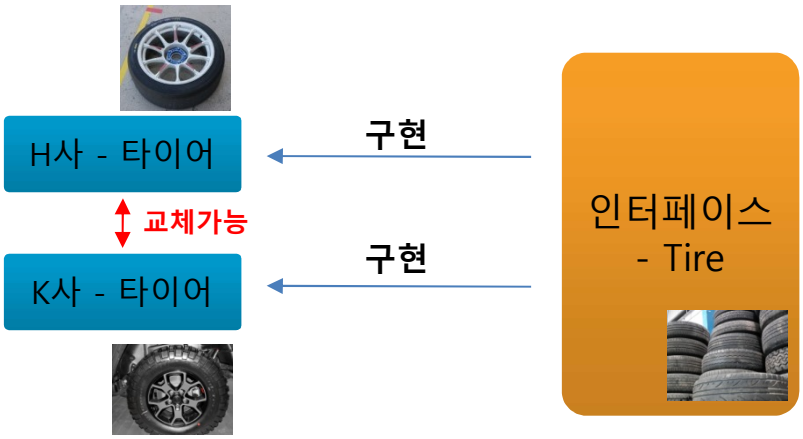
다형성 및 타입변환

다형성

❖ 상속 또는 인터페이스의 자동타입변환
❖ 오버라이딩

효과

❖ 다양한 실행결과를 얻을 수 있음
❖ 객체를 부품화할 수 있어 유지보수가 용이함.



다형성 및 타입변환 예제

TV 인터페이스

```
public interface Tire{
  public void roll( );          // 타이어가 돌아가는 기능
}
```

H사 타이어 구현

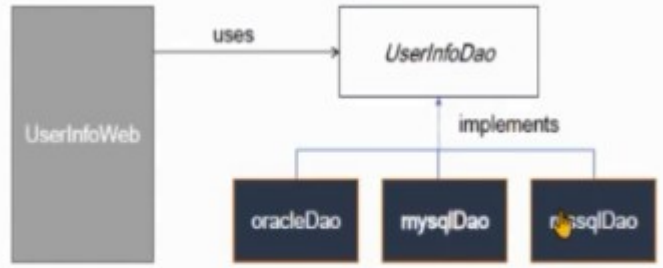
```
public class Htire implements Tire {
  void roll( ) {
    System.out.print ("H사 타이어가 돌아갑니다.");
  }
}
```

K사 타이어 구현

```
public class Ktire implements Tire {
  void roll( ) {
    System.out.print ("K사 타이어가 돌아갑니다.");
  }
}
```

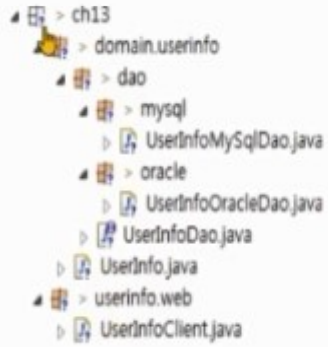
```
public class Car implements Tire {
  Tire tire = new Htire( );  // H사 타이어 객체 생성
  tire.roll( );
}
```

타사 타이어로 교체시 객체만 바꾸면 됨.
Tire tire = new Ktire( );
* 또한 구현객체는 Tire Class의 객체로 자동 타입변환됨.



인터페이스를 활용한 dao 구현하기

- DB에 회원 정보를 넣는 dao(data access object)를 여러 DB 제품이 지원될 수 있게 구현함
- 환경파일(db.properties) 에서 database의 종류에 대한 정보를 읽고 그 정보에 맞게 dao 인스턴스를 생성하여 실행될 수 있게 함
- source hierachy



Exam4

12. 인터페이스는 왜 쓰는가?

인터페이스가 하는 일

- 클래스나 프로그램이 제공하는 기능을 명시적으로 선언
- 일종의 클라이언트 코드와의 약속이며 클래스나 프로그램이 제공하는 명세(specification)
- 클라이언트 프로그램은 인터페이스에 선언된 메서드 명세만 보고 이를 구현한 클래스를 사용할 수 있음
- 어떤 객체가 하나의 인터페이스 타입이라는 것은 그 인터페이스가 제공하는 모든 메서드를 구현했다는 의미임
- 인터페이스를 구현한 다양한 객체를 사용함 - 다형성
- 예) JDBC 인터페이스