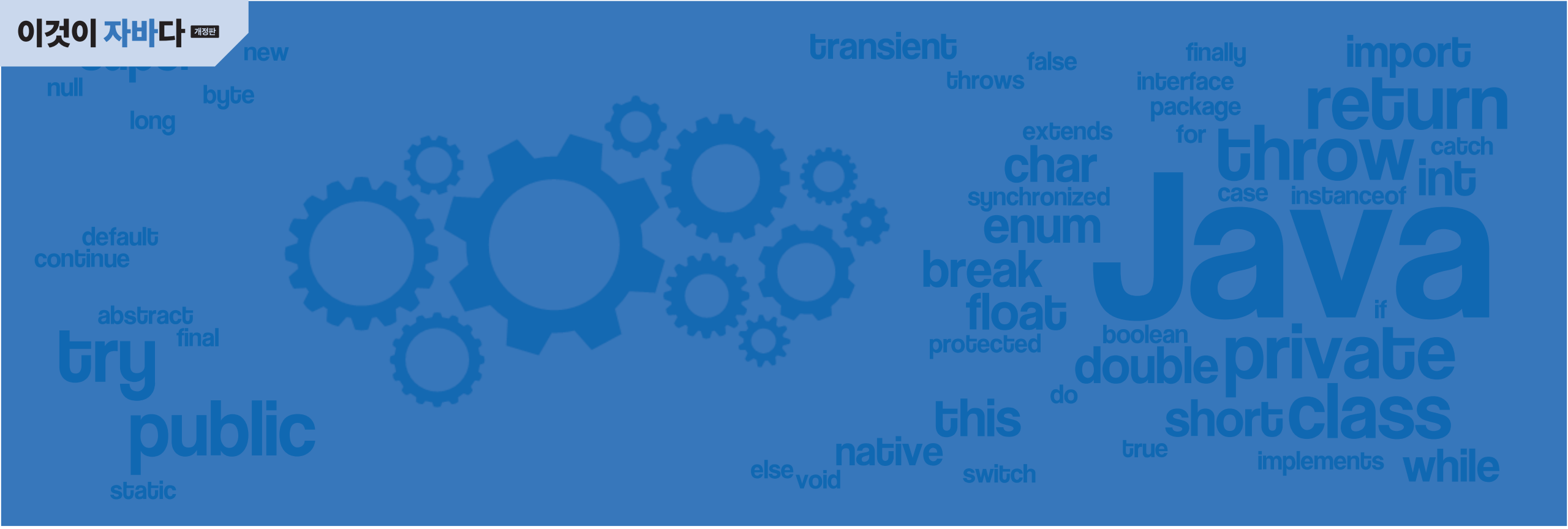
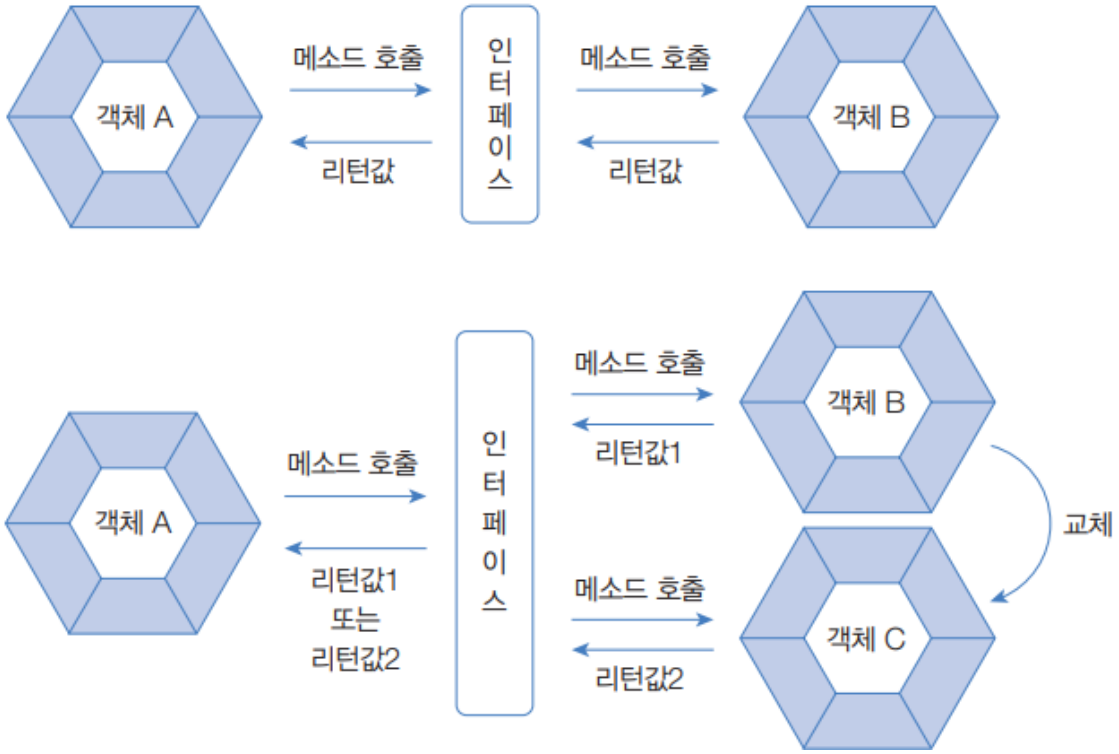


Chapter 08 인터페이스

Contents

인터페이스

- 두 객체를 연결하는 역할
- 다형성 구현에 주된 기술



인터페이스 선언

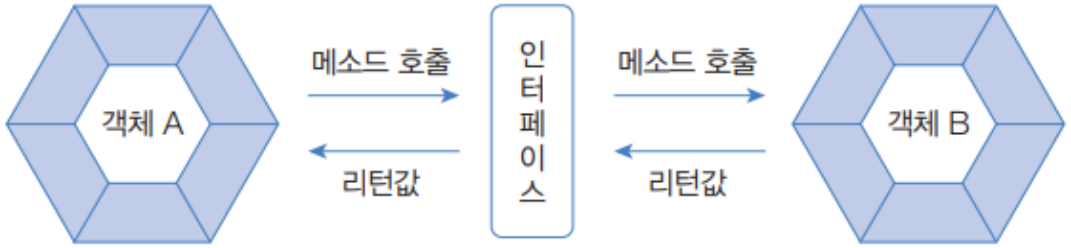
- 인터페이스 선언은 class 키워드 대신 interface 키워드를 사용
- 접근 제한자로는 클래스와 마찬가지로 같은 패키지 내에서만 사용 가능한 default, 패키지와 상관없이 사용하는 public을 붙일 수 있음

```
interface 인터페이스명 { … }          //default 접근 제한
public interface 인터페이스명 { … }   //public 접근 제한
```

```
public interface 인터페이스명 {
  //public 상수 필드
  //public 추상 메소드
  //public 디폴트 메소드
  //public 정적 메소드
  //private 메소드
  //private 정적 메소드
}
```

구현 클래스 선언

- 인터페이스에 정의된 추상 메소드에 대한 실행 내용이 구현



- implements 키워드는 해당 클래스가 인터페이스를 통해 사용할 수 있다는 표시이며, 인터페이스의 추상 메소드를 재정의한 메소드가 있다는 뜻

```
public class B implements 인터페이스명 { … }
```

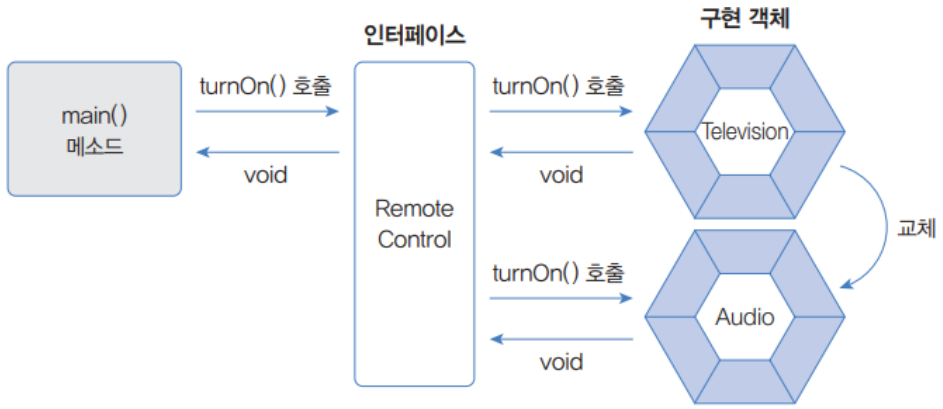
변수 선언과 구현 객체 대입

- 인터페이스는 참조 타입에 속하므로 인터페이스 변수에는 객체를 참조하고 있지 않다는 뜻으로 null을 대입할 수 있음

```
RemoteControl rc;
RemoteControl rc = null;
```

- 인터페이스를 통해 구현 객체를 사용하려면, 인터페이스 변수에 구현 객체의 번지를 대입해야 함

```
rc = new Television();
```



상수 필드

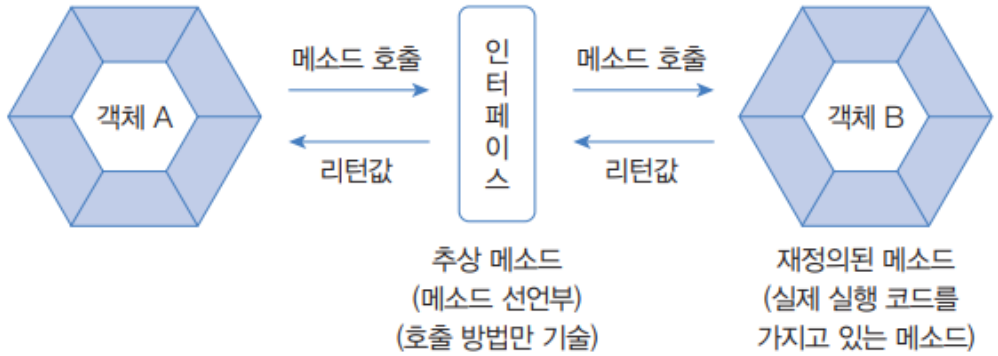
- 인터페이스는 public static final 특성을 갖는 불변의 상수 필드를 멤버로 가질 수 있음

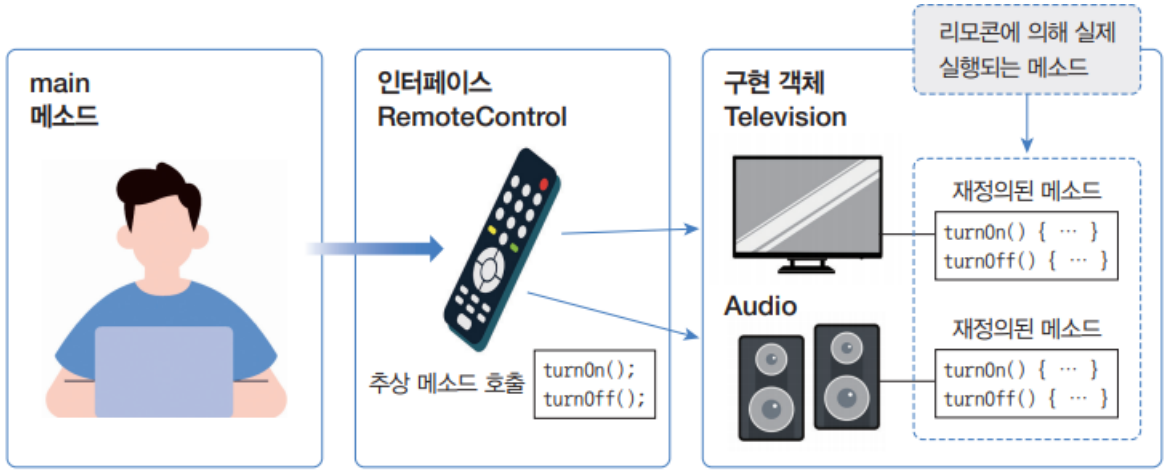
```
[ public static final ] 타입 상수명 = 값;
```

- 인터페이스에 선언된 필드는 모두 public static final 특성을 갖기 때문에 public static final을 생략해도 자동으로 컴파일 과정에서 붙음
- 상수명은 대문자로 작성하되, 서로 다른 단어로 구성되어 있을 경우에는 언더바(_)로 연결

추상 메소드

- 리턴 타입, 메소드명, 매개변수만 기술되고 중괄호 { }를 붙이지 않는 메소드
- public abstract를 생략하더라도 컴파일 과정에서 자동으로 붙음
- 추상 메소드는 객체 A가 인터페이스를 통해 어떻게 메소드를 호출할 수 있는지 방법을 알려주는 역할





디폴트 메소드

- 인터페이스에는 완전한 실행 코드를 가진 디폴트 메소드를 선언할 수 있음
- 추상 메소드는 실행부(중괄호 { })가 없지만 디폴트 메소드는 실행부 있음. default 키워드가 리턴 타입 앞에 붙음

```
[public] default 리턴타입 메소드명(매개변수, …) { … }
```

- 디폴트 메소드의 실행부에는 상수 필드를 읽거나 추상 메소드를 호출하는 코드를 작성할 수 있음

정적 메소드

- 구현 객체가 없어도 인터페이스만으로 호출할 수 있음
- 선언 시 public을 생략하더라도 자동으로 컴파일 과정에서 붙음

```
[public | private] static 리턴타입 메소드명(매개변수, …) { … }
```

- 정적 실행부를 작성할 때 상수 필드를 제외한 추상 메소드, 디폴트 메소드, private 메소드 등을 호출할 수 없음

private 메소드

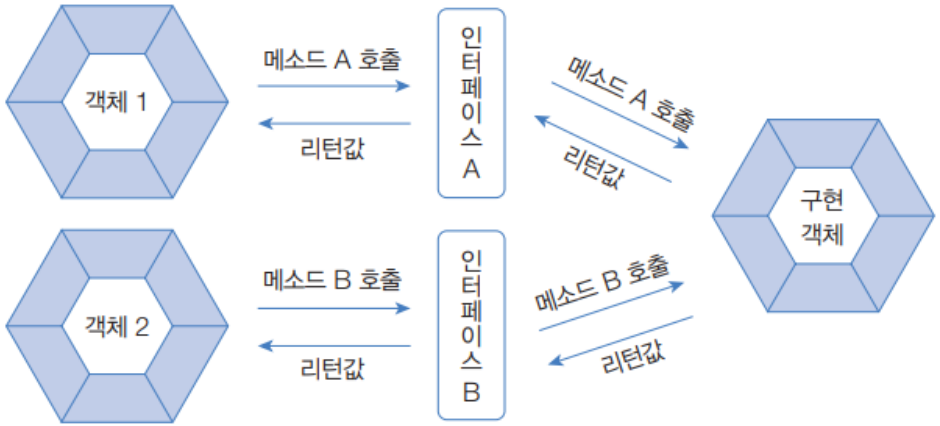
- 인터페이스의 상수 필드, 추상 메소드, 디폴트 메소드, 정적 메소드는 모두 public 접근 제한을 가짐
  public을 생략하더라도 항상 외부에서 접근이 가능
- 인터페이스에 외부에서 접근할 수 없는 private 메소드 선언도 가능

구분	설명
private 메소드	구현 객체가 필요한 메소드
private 정적 메소드	구현 객체가 필요 없는 메소드

- private 메소드는 디폴트 메소드 안에서만 호출이 가능
- private 정적 메소드는 정적 메소드 안에서도 호출이 가능

다중 인터페이스

- 구현 객체는 여러 개의 인터페이스를 통해 구현 객체를 사용할 수 있음



- 구현 클래스는 인터페이스 A와 인터페이스 B를 implements 뒤에 쉼표로 구분해서 작성해, 모든 인터페이스가 가진 추상 메소드를 재정의

```
public class 구현클래스명 implements 인터페이스A, 인터페이스B {
  //모든 추상 메소드 재정의
}
```

인터페이스 상속

- 인터페이스도 다른 인터페이스를 상속할 수 있음. 다중 상속을 허용
- extends 키워드 뒤에 상속할 인터페이스들을 나열

```
public interface 자식인터페이스 extends 부모인터페이스1, 부모인터페이스2 { … }
```

- 자식 인터페이스의 구현 클래스는 자식 인터페이스의 메소드뿐만 아니라 부모 인터페이스의 모든 추상 메소드를 재정의
- 구현 객체는 다음과 같이 자식 및 부모 인터페이스 변수에 대입될 수 있음

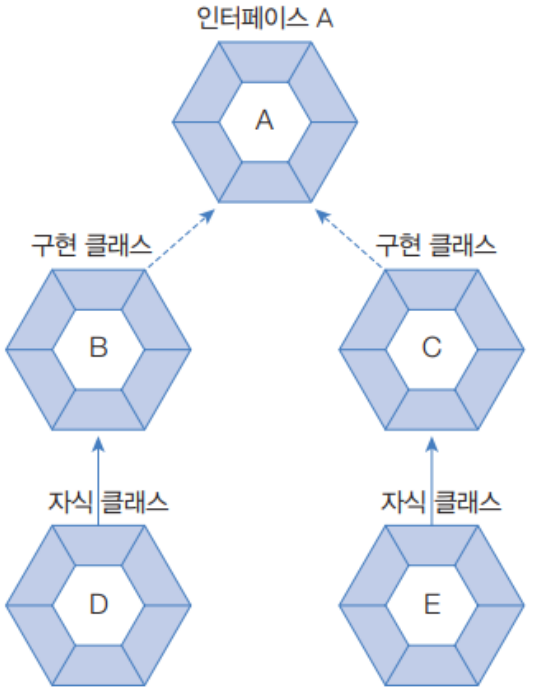
```
자식인터페이스 변수 = new 구현클래스(…);
부모인터페이스1 변수 = new 구현클래스(…);
부모인터페이스2 변수 = new 구현클래스(…);
```

자동 타입 변환

- 자동으로 타입 변환이 일어나는 것

```
자동 타입 변환
인터페이스 변수 = 구현객체;
```

- 부모 클래스가 인터페이스를 구현하고 있다면 자식 클래스도 인터페이스 타입으로 자동 타입 변환될 수 있음

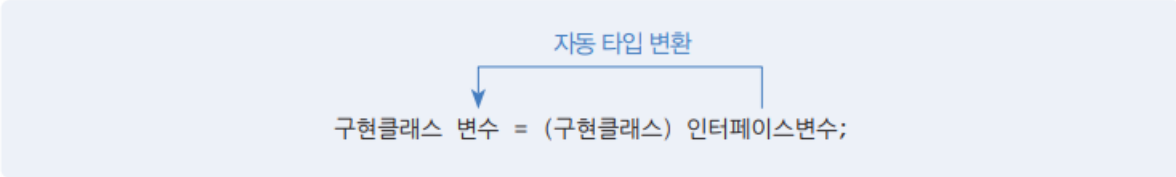



```
B b = new B();
C c = new C();
D d = new D();
E e = new E();
```

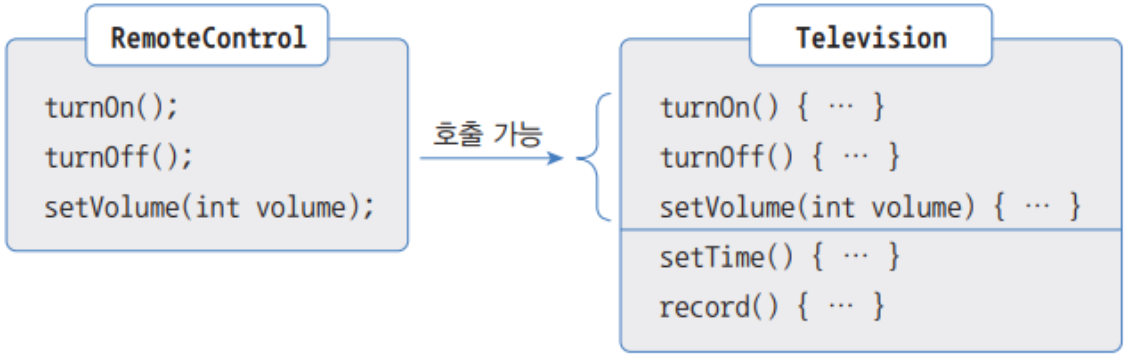
```
A a;
a = b;  (가능)
a = c;  (가능)
a = d;  (가능)
a = e;  (가능)
```

강제 타입 변환

- 캐스팅 기호를 사용해서 인터페이스 타입을 구현 클래스 타입으로 변환시키는 것

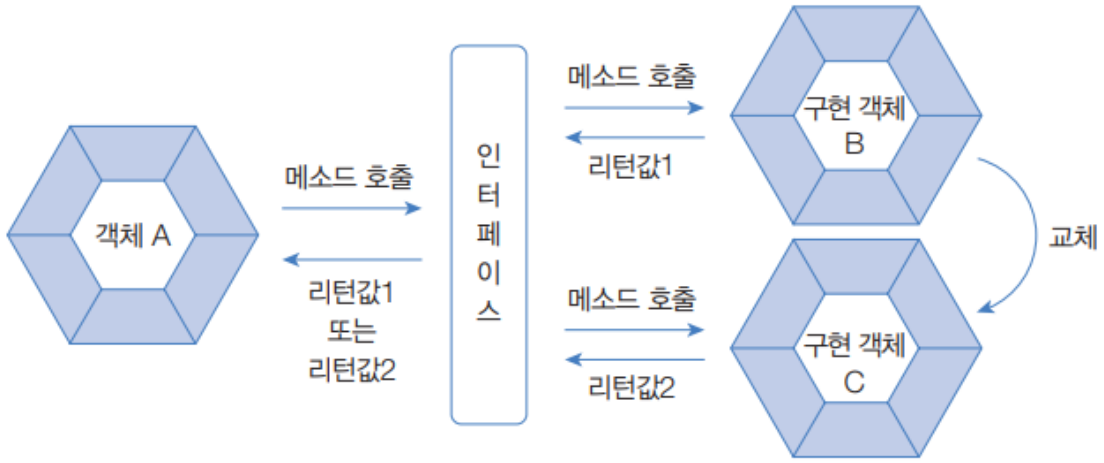


- 구현 객체가 인터페이스 타입으로 자동 변환되면, 인터페이스에 선언된 메소드만 사용 가능

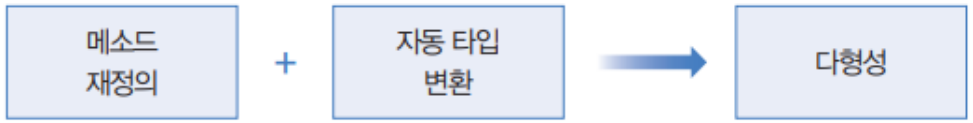


다형성

- 사용 방법은 동일하지만 다양한 결과가 나오는 성질



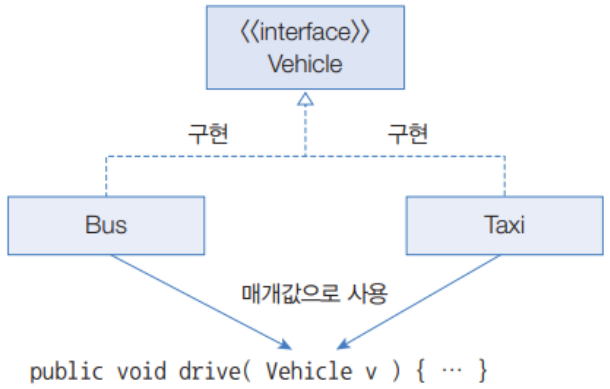
- 인터페이스 역시 다형성을 구현하기 위해 재정의와 자동 타입 변환 기능을 이용

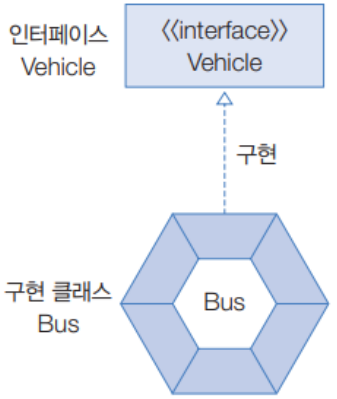


필드의 다형성

매개변수의 다형성

- 매개변수 타입을 인터페이스로 선언하면 메소드 호출 시 다양한 구현 객체를 대입할 수 있음






```
Driver driver = new Dirver();

Bus bus = new Bus();

driver.drive( bus );
```

자동 타입 변환 발생
Vehicle vehicle = bus;

instanceof 연산자

- 인터페이스에서도 객체 타입을 확인하기 위해 instanceof 연산자를 사용 가능

```
if( vehicle instanceof Bus ) {
  //vehicle에 대입된 객체가 Bus일 경우 실행
}
```

- Java 12부터는 instanceof 연산의 결과가 true일 경우, 우측 타입 변수를 사용할 수 있기 때문에 강제 타입 변환이 필요 없음

```
if(vehicle instanceof Bus bus) {
  //bus 변수 사용
}
```

sealed 인터페이스

- Java 15부터 무분별한 자식 인터페이스 생성을 방지하기 위해 봉인된 인터페이스 사용

```
public sealed interface InterfaceA permits InterfaceB { ··· }
```

- sealed 키워드를 사용하면 permits 키워드 뒤에 상속 가능한 자식 인터페이스를 지정. non-sealed는 봉인을 해제한다는 뜻

```
public non-sealed interface InterfaceB extends InterfaceA { ··· }
```


transient
false
finally
import
new
byte
null
long
throws
interface
return
package
extends
for
throw
catch
char
int
synchronized
case
instanceof
enum
default
continue
abstract
final
try
boolean
double
private
public
do
this
short
class
native
true
implements
while
else
void
switch
static

Thank you!