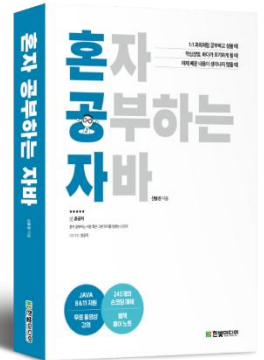


Chapter

08

인터페이스



08-2. 타입 변환과 다형성

혼자 공부하는 자바 (신용권 저)

- 시작하기 전에
- 자동 타입 변환
- 필드의 다형성
- 매개 변수의 다형성
- 강제 타입 변환
- 객체 타입 확인
- 인터페이스 상속
- 키워드로 끝내는 핵심 포인트



시작하기 전에

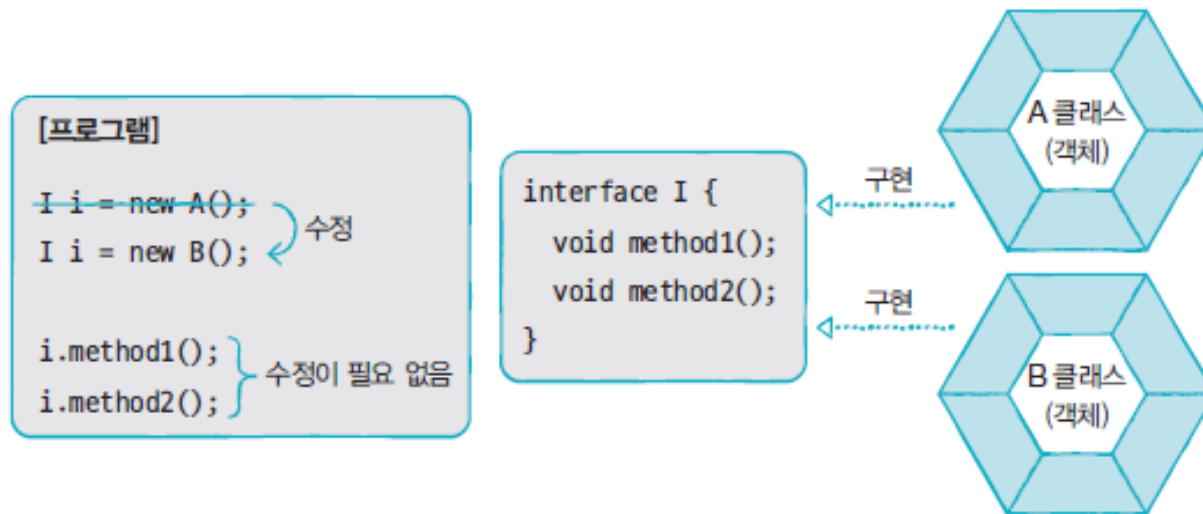
[핵심 키워드] : 자동 타입 변환, 다형성, 강제 타입 변환, instanceof, 인터페이스 상속

[핵심 포인트]

인터페이스도 메소드 재정의와 타입 변환되므로 다형성을 구현할 수 있다.

❖ 인터페이스의 다형성

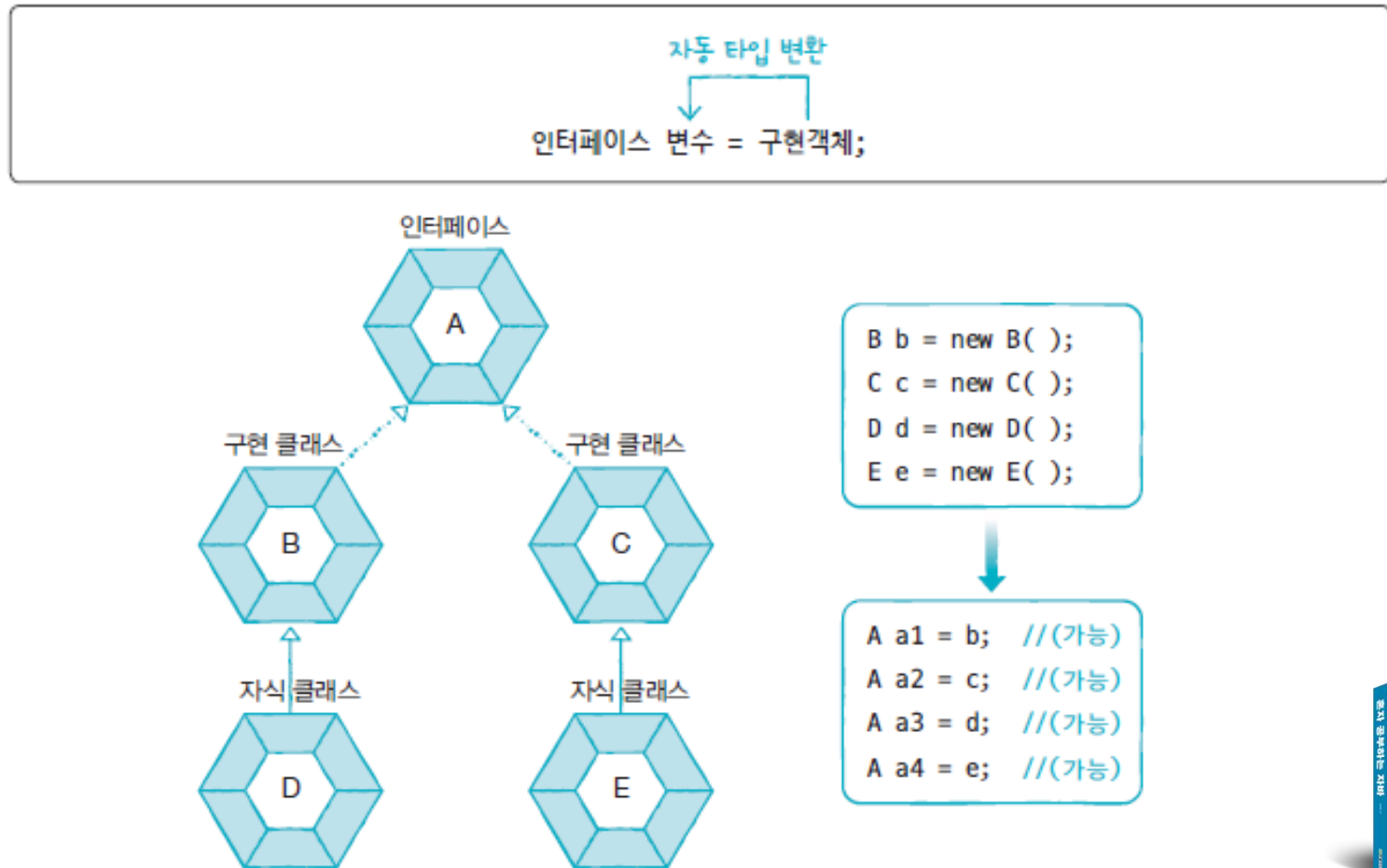
- 인터페이스 사용 방법은 동일하지만 구현 객체 교체하여 프로그램 실행 결과를 다양화



자동 타입 변환

❖ 자동 타입 변환 (promotion)

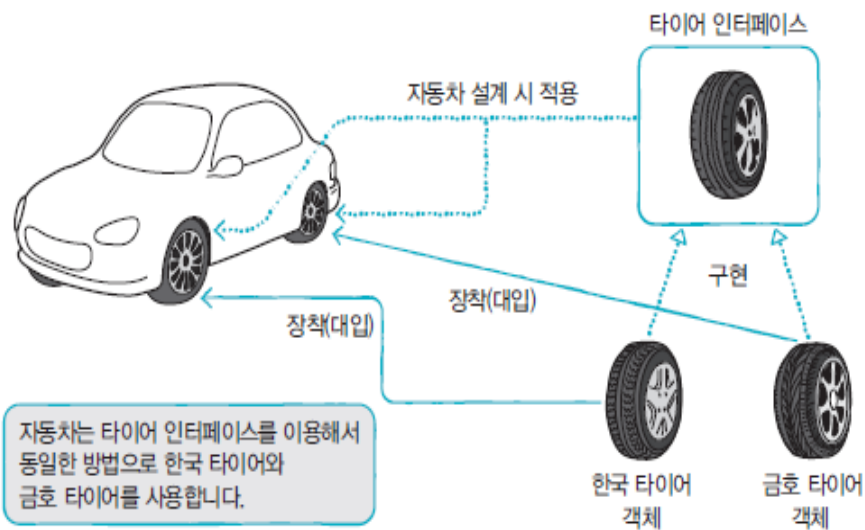
- 구현 객체와 자식 객체는 인터페이스 타입으로 자동 타입 변환 된다.



❖ 필드의 다형성

```
public class Car {  
    Tire frontLeftTire = new HankookTire();  
    Tire frontRightTire = new HankookTire();  
    Tire backLeftTire = new HankookTire();  
    Tire backRightTire = new HankookTire();  
  
    void run() {  
        frontLeftTire.roll();  
        frontRightTire.roll();  
        backLeftTire.roll();  
        backRightTire.roll();  
    }  
}
```

```
Car myCar = new Car();  
myCar.frontLeftTire = new KumhoTire();  
myCar.frontRightTire = new KumhoTire();
```



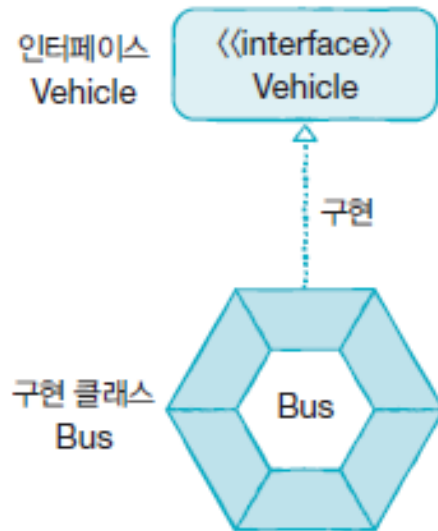
매개 변수의 다형성

❖ 매개변수의 다형성

```
public interface Vehicle {  
    public void run();  
}
```

```
public class Driver {  
    public void drive(Vehicle vehicle) {  
        vehicle.run();  
    }  
}
```

구현 객체
구현 객체의 run() 메소드가 실행됨



```
Driver driver = new Driver();  
Bus bus = new Bus();  
driver.drive( bus );
```

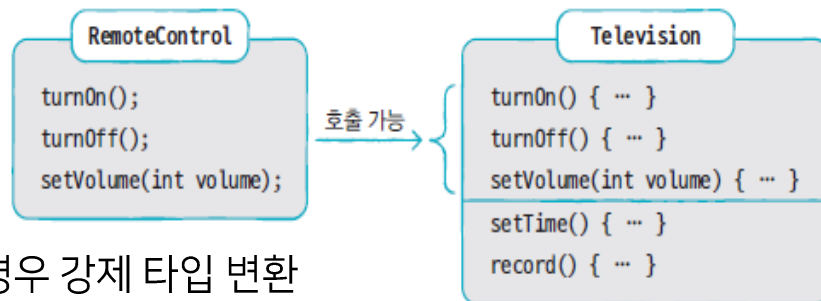
자동 타입 변환 발생
Vehicle vehicle = bus;



강제 타입 변환

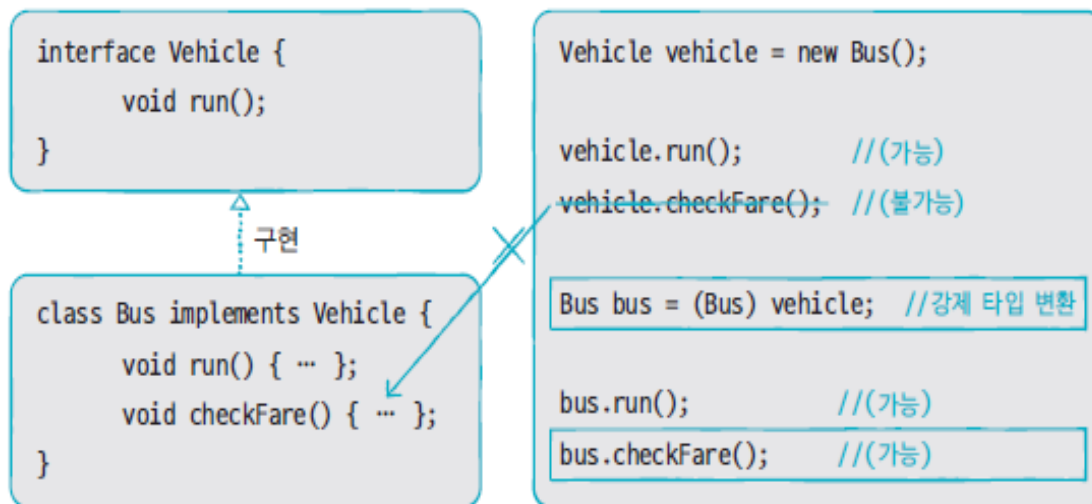
❖ 강제 타입 변환 (casting)

- 구현 객체가 인터페이스 타입으로 자동 변환하면
인터페이스에 선언된 메소드만 사용 가능
- 구현 클래스에만 선언된 필드나 메소드를 사용할 경우 강제 타입 변환



강제타입 변환

구현클래스 변수 = (구현클래스) 인터페이스변수;



객체 타입 확인

❖ 객체 타입 확인 instanceof

- 구현 객체가 변환되어 있는지 알 수 없는 상태에서 강제 타입 변환할 경우 ClassCastException 발생

```
Vehicle vehicle = new Taxi();  
Bus bus = (Bus) vehicle;
```

```
public void drive(Vehicle vehicle) {  
    Bus bus = (Bus) vehicle;  
    bus.checkFare();  
}
```

- instanceof 연산자로 확인 후 안전하게 강제 타입 변환

```
public class Driver {  
    public void drive(Vehicle vehicle) {  
        if(vehicle instanceof Bus) {  
            Bus bus = (Bus) vehicle;  
            bus.checkFare();  
        }  
        vehicle.run();  
    }  
}
```

Bus 객체 Taxi 객체
 ↓ ↓
if(vehicle instanceof Bus) { ← vehicle 매개 변수가 참조하는 객체가 Bus인지 조사
 Bus bus = (Bus) vehicle; ← Bus 객체일 경우 안전하게 강제타입 변환
 bus.checkFare(); ← Bus 타입으로 강제 타입 변환을 하는 이유
}

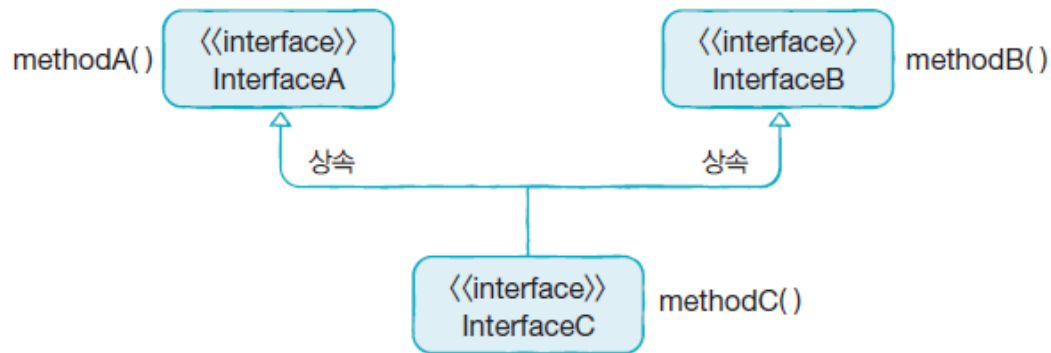


인터페이스 상속

❖ 인터페이스 상속

- 인터페이스는 다중 상속을 할 수 있다.

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



```
public interface InterfaceC extends InterfaceA, InterfaceB {
```

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

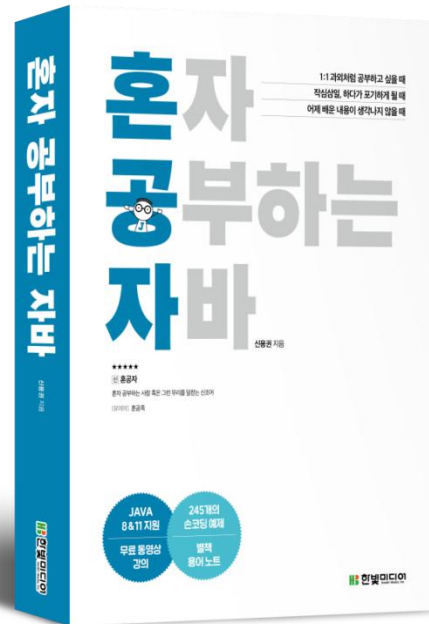
```
public class ImplementationC implements InterfaceC {  
  
    ImplementationC impl = new ImplementationC();  
  
    InterfaceC ic = impl;  
    InterfaceA ia = impl;  
    InterfaceB ib = impl;
```



키워드로 끝내는 핵심 포인트

- **자동 타입 변환**: 구현 객체는 인터페이스 변수로 자동 타입 변환된다.
- **다형성**: 인터페이스도 재정의와 타입 변환 기능 제공하므로 다형성을 구현할 수 있다.
- **강제 타입 변환**: 인터페이스에 대입된 구현 객체를 다시 원래 타입으로 변환하는 것을 말한다.
- **instanceof**: 객체가 어떤 타입인지 조사할 때 사용한다. 강제 타입 변환 전에 사용.
- **인터페이스 상속**: 인터페이스는 다중 상속 허용한다.





Thank You!