Chapter 15. 클래스의 상속 2: 오버라이딩

15-1. 상속을 위한 관계

■ 상속을 위한 기본 조건인 IS-A 관계의 성립

✓ 상속을 위한 기본 조건

• 상속관계에 있는 두 클래스 사이에는 IS-A 관계가 성립해야 한다.
• IS-A 관계가 성립하지 않는 경우에는 상속의 타당성을 면밀히 검토해야 한다.
• IS-A 이외에 HAS-A 관계도 상속으로 표현 가능하다. 그러나 HAS-A를 대신해서 Composition 관계를 유지하는 것이 보다 적절한 경우가 많다.

• 전화기 → 무선 전화기
• 컴퓨터 → 노트북 컴퓨터

• 무선 전화기는 일종의 전화기입니다.
• 노트북 컴퓨터는 일종의 컴퓨터입니다.

• 무선 전화기 is a 전화기.
• 노트북 컴퓨터 is a 컴퓨터.

무선 전화기는 전화기를 상속한다!

노트북 컴퓨터는 컴퓨터를 상속한다!

IS-A 기반 상속의 예

나는 정말 JAVA를 공부한 적이 없다고

```
class Computer
{
    String owner;
    public Computer(String name){   }
    public void calculate() {  }
}
```

일반적으로 IS-A 관계가 성립되면, 불필요한 상속관계는 형성될 수 있으나, 잘못된 상속관계가 형성된다고는 이야기하지 않는다.

노트북 컴퓨터는 컴퓨터이다!

```
class NotebookComp extends Computer
{
    int battary;
    public NotebookComp(String name, int initChag) {  }
    public void charging() {  }
    public void movingCal() {  }
}
```

타블렛은 노트북 컴퓨터이다!

```
class TabletNotebook extends NotebookComp
{
    String regstPenModel;
    public TabletNotebook(String name, int initChag, String pen){ }
    public void write(String penInfo){ }
}
```

HAS-A 관계에 상속을 적용한 경우


```
class Gun
{
    int bullet;     // 장전된 총알의 수
    public Gun(int bnum) { bullet=bnum; }
    public void shut()
    {
        System.out.println("BBANG!");
        bullet--;
    }
}
```

```
class Police extends Gun
{
    int handcuffs;      // 소유한 수갑의 수
    public Police(int bnum, int bcuff)
    {
        super(bnum);
        handcuffs=bcuff;
    }
    public void putHandcuff()
    {
        System.out.println("SNAP!");
        handcuffs--;
    }
}
```

경찰은 총을 소유하고 있다!

경찰 has a 총!

상속은 강한 연결고리를 형성한다. 때문에 총을 소유하지 않는 경찰, 또는 총이 아닌 경찰봉을 소유하는 경찰 등 다양한 표현에 한계를 보인다는 단점이 있다!

HAS-A 관계에 복합관계를 적용한 경우

```
class Gun
{
    int bullet;     // 장전된 총알의 수
    public Gun(int bnum){bullet=bnum;}
    public void shut()
    {
        System.out.println("BBANG!");
        bullet--;
    }
}
```

복합관계는 강한 연결고리를 형성하지 않는다. 따라서 소유하던 대상을 소유하지 않을 수도 있고, 소유의 대상을 늘리는 것도 상속보다 훨씬 간단하다. 때문에 HAS-A 관계는 복합 관계로 표현한다.

```
class Police
{
    int handcuffs;      // 소유한 수갑의 수
    Gun pistol;         // 소유하고 있는 권총
    public Police(int bnum, int bcuff)
    {
        handcuffs=bcuff;
        if(bnum!=0)
            pistol=new Gun(bnum);
        else
            pistol=null;
    }
    public void putHandcuff()
    {
        System.out.println("SNAP!");
        handcuffs--;
    }
    public void shut()
    {
        if(pistol==null)
            System.out.println("Hut BBANG!");
        else
            pistol.shut();
    }
}
```

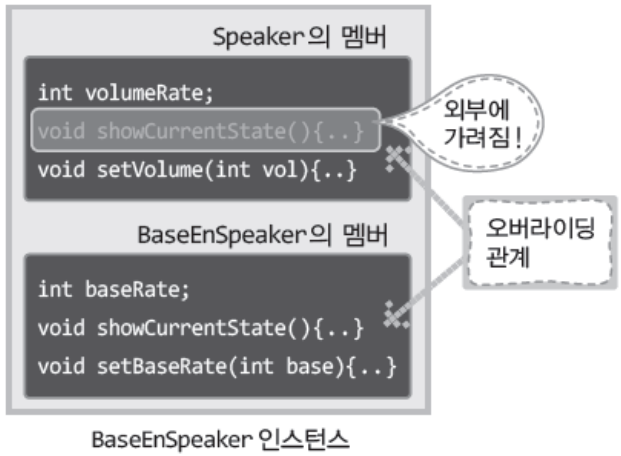
15-2. 하위 클래스에서 메소드를 다시 정의한다면?

메소드 오버라이딩

- 상위 클래스에 정의된 메소드의 이름, 반환형, 매개변수 선언까지 완전히 동일한 메소드를 하위 클래스에서 다시 정의하는 것!
- 하위 클래스에 정의된 메소드에 의해 상위 클래스의 메소드는 가리워진다.

```
class Speaker
{
    private int volumeRate;
    public void showCurrentState()
    {
        System.out.println("볼륨 크기 : "+ volumeRate);
    }
    public void setVolume(int vol)
    {
        volumeRate=vol;
    }
}
```

```
class BaseEnSpeaker extends Speaker
{
    private int baseRate;
    public void showCurrentState()
    {
        super.showCurrentState();
        System.out.println("베이스 크기 : "+baseRate);
    }
    public void setBaseRate(int base)
    {
        baseRate=base;
    }
}
```



■ 상위 클래스의 참조변수로 하위 클래스의 인스턴스 참조

나 정말 JAVA를 공부한 적 없는데

- 중 저음 보강 스피커는 (일종의) 스피커이다. (O)
- BaseEnSpeaker is a Speaker. (O)

- 스피커는 (일종의) 중 저음 보강 스피커이다. (X)
- Speaker is a BaseEnSpeaker. (X)

자바 컴파일러의 실제 관점

```
public static void main(String[] args)
{
    Speaker bs=new BaseEnSpeaker();
    bs.setVolume(10);
    bs.setBaseRate(20);   // 컴파일 에러
    bs.showCurrentState();
}
```

BaseEnSpeaker도 Speark의 인스턴스이므로 성립한다!

bs가 참조하는것은 Speaker의 인스턴스로 인식하기 때문에 BaseEnSpeaker의 멤버에 접근 불가!

위의 내용을 정확히 이해하는 것이 중요하다. 특히 상위 클래스의 참조변수가 하위 클래스의 인스턴스를 참조할 수 있는 이유를 잘 이해하자!

■ 참조변수의 참조 가능성에 대한 일반화


```
class AAA { . . . }
class BBB extends AAA { . . . }
class CCC extends BBB { . . . }
```

아래의 문제 제시를 위한 클래스의 상속관계

```
AAA ref1 = new BBB();
AAA ref2 = new CCC();
BBB ref3 = new CCC();
```

```
CCC ref1 = . . .     // 컴파일 완료
BBB ref2 = ref1;
AAA ref3 = ref1;
```

```
AAA ref1 = new CCC();
BBB ref2 = ref1;
CCC ref3 = ref1;
```

참조변수의 자료형에 따라서 대입연산의 허용여부가 결정된다.

이 사실을 바탕으로 왼쪽 문장 중에서 컴파일 에러가 발생하는 문장들을 모두 고르면?

■ 오버라이딩 관계에서의 메소드 호출


```
class AAA
{
    public void rideMethod() { System.out.println("AAA's Method"); }
    public void loadMethod() { System.out.println("void Method"); }
}

class BBB extends AAA
{
    public void rideMethod() { System.out.println("BBB's Method"); }
    public void loadMethod(int num) { System.out.println("int Method"); }
}

class CCC extends BBB
{
    public void rideMethod() { System.out.println("CCC's Method"); }
    public void loadMethod(double num) { System.out.println("double Method"); }
}
```

참조변수의 자료형에 상관없이 오버라이딩 된 메소드는 외부로부터 가려지므로, 마지막으로 오버라이딩 한 메소드가 호출된다!

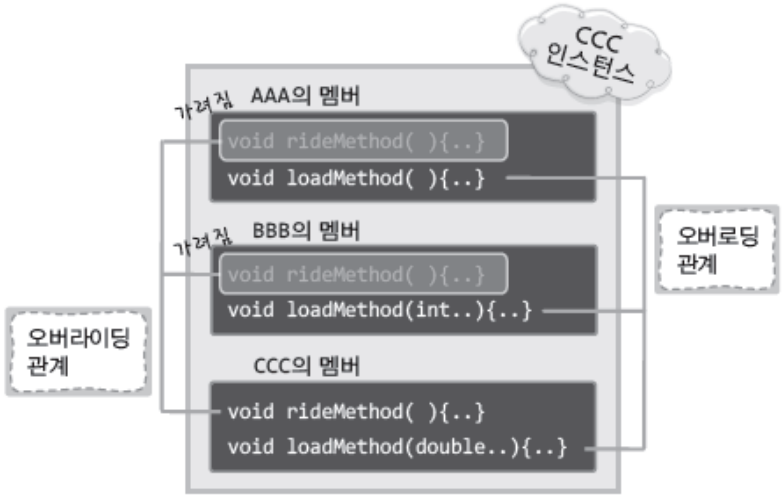
```
public static void main(String[] args)
{
    AAA ref1=new CCC();
    BBB ref2=new CCC();
    CCC ref3=new CCC();

    ref1.rideMethod();
    ref2.rideMethod();
    ref3.rideMethod();

    ref3.loadMethod();
    ref3.loadMethod(1);
    ref3.loadMethod(1.2);
}
```

실행결과

```
CCC's Method
CCC's Method
CCC's Method
void Method
int Method
double Method
```

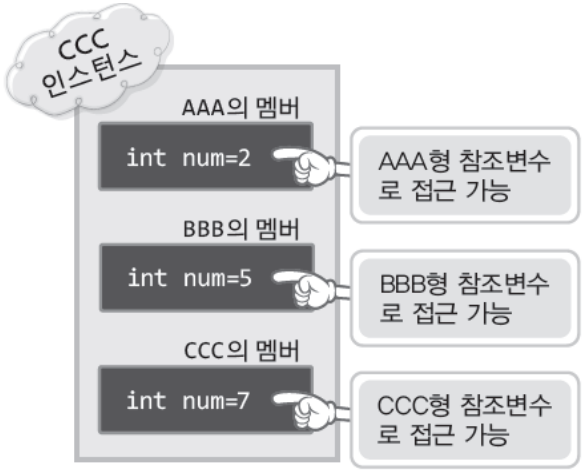


■ 인스턴스 변수도 오버라이딩 되나요?

```
class AAA
{
    public int num=2;
}

class BBB extends AAA
{
    public int num=5;
}

class CCC extends BBB
{
    public int num=7;
}
```



인스턴스 변수는 오버라이딩 관계에 놓이지 않는다.
따라서 참조변수의 자료형에 따라서 접근대상이 결정된다.

```
public static void main(String[] args)
{
    CCC ref1=new CCC();
    BBB ref2=ref1;
    AAA ref3=ref2;

    System.out.println("CCC's ref : "+ref1.num);
    System.out.println("BBB's ref : "+ref2.num);
    System.out.println("AAA's ref : "+ref3.num);
}
```

실행결과

```
CCC's ref : 7
BBB's ref : 5
AAA's ref : 2
```

나는 정말 JAVA를 공부한 적이 없다고

15-3. 참조변수의 인스턴스 참조와 instanceof 연산자

instanceof 연산자

- 형변환이 가능한지를 묻는 연산자이다.
- 형변환이 가능하면 true를 가능하지 않으면 false를 반환.

```
class Box
{
    public void simpleWrap() { . . . }
}

class PaperBox extends Box
{
    public void paperWrap() { . . . }
}

class GoldPaperBox extends PaperBox
{
    public void goldWrap() { . . . }
}
```

```
public static void wrapBox(Box box)
{
    if(box가 GoldPaperBox로 형변환 가능하다면)
        ((GoldPaperBox)box).goldWrap();
    else if(box가 PaperBox로 형변환 가능하다면)
        ((PaperBox)box).paperWrap();
    else
        box.simpleWrap();
}
```

```
public static void wrapBox(Box box)
{
    if(box instanceof GoldPaperBox)
        ((GoldPaperBox)box).goldWrap();
    else if(box instanceof PaperBox)
        ((PaperBox)box).paperWrap();
    else
        box.simpleWrap();
}
```