Chapter 14. 클래스의 상속 1: 상속의 기본

14-1. 상속은 재활용 + 알파

■ 객체지향에서의 상속에 대한 논의와 CBD

나는 정말 JAVA를 공부한 적이 없다

✓ 상속에 대한 정확한 평가!

- 객체지향 패러다임은 재활용의 관점에서 실패한 패러다임
- 자바의 상속은 클래스의 재활용적 측면에서 바라보는 것은 문제가 있다.
- 재활용할 수 있다는 이유만으로 상속을 하는 경우는 드물다(우리는 한다! 그리고 그 결과는 좋지 못하다).

✓ 상속의 이유!

상속을 통해 연관된 일련의 클래스에 대한 공통적인 규약을 정의하고 적용하는데, 상속의 실질적인 목적이 있다!

상속을 정확히 이해하게 되면, 이 문장이 의미하는바를 이해할 수 있다.

✓ CBD(Component Based Development)

재활용이라는 공학적 측면의 적용을 위해 정의된 새로운 패러다임!

객체지향이 재활용의 관점에서 실패한 이유는?

- 클래스 하나를 재활용하는 것이 새롭게 디자인하는 것보다 더 큰 노력이 든다.
- 재활용을 고려해서 클래스를 디자인할 경우, 설계에 필요한 시간이 몇 배 더 길어진다.
- 실무에서의 현황 다른 동료가 설계한 클래스에 대한 신뢰도 부족!

상속은 재활용이다! 라는 공식을 깨는데서부터 시작해야 상속을 정확히 이해할 수 있다!

14-2. 상속의 기본문법 이해

상속의 가장 기본적인 특성

문법적 측면에서의 상속

상속은 기존에 정의된 클래스에 메소드와 변수를 추가하여 새로운 클래스를 정의하는 것!

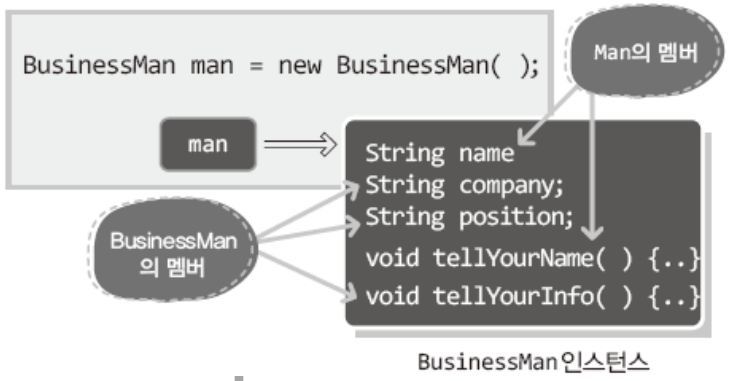
상위 클래스, 기초 클래스

```
class Man
{
    public String name;
    public void tellYourName() {
      System.out.println("My name is "+name); }
}
```

하위 클래스, 유도 클래스

```
class BusinessMan extends Man
{
    public String company;
    public String position;
    public void tellYourInfo()
    {
        System.out.println("My company is "+company);
        System.out.println("My position is "+position);
        tellYourName();
    }
}
```

Man 클래스를 상속했기 때문에 호출가능!



■ 상속 관계에 있는 인스턴스의 생성 예


```
class Man
{
    private String name;
    public Man(String name) { this.name=name;}
    public void tellYourName() { System.out.println("My name is "+name); }
}

class BusinessMan extends Man
{
    private String company;        // 회사이름
    private String position;       // 직급
    public BusinessMan(String name, String company, String position)
    {
        super(name);               // 상위 클래스의 생성자 호출문
        this.company=company;
        this.position=position;
    }
    public void tellYourInfo()
    {
        System.out.println("My compan
        System.out.println("My positi
        tellYourName();            // Man 클래
    }
}
```

외부에서 호출하는 것은 BusinessMan 클래스의 생성자 이니, 이 생성자가 부모 클래스의 인스턴스 변수를 초기화할 의무를 지닌다.

BusinessMan 인스턴스 생성시 초기화해야 할 인스턴스 변수는 다음과 같다.

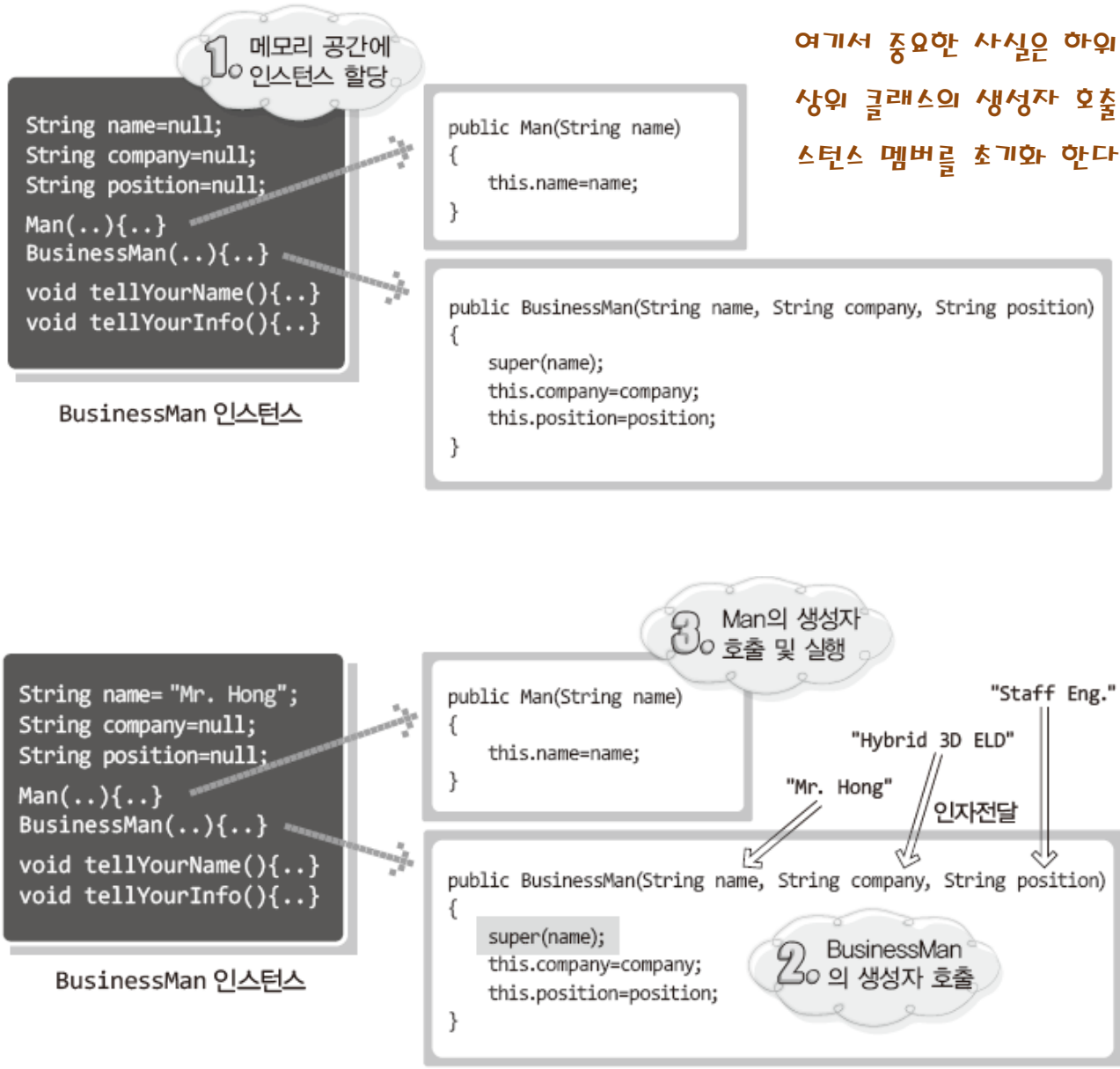
name, company, position

클래스 Man　　클래스 BusinessMan

name을 인자로 전달받는 상위 클래스의 생성자를 호출하겠다.

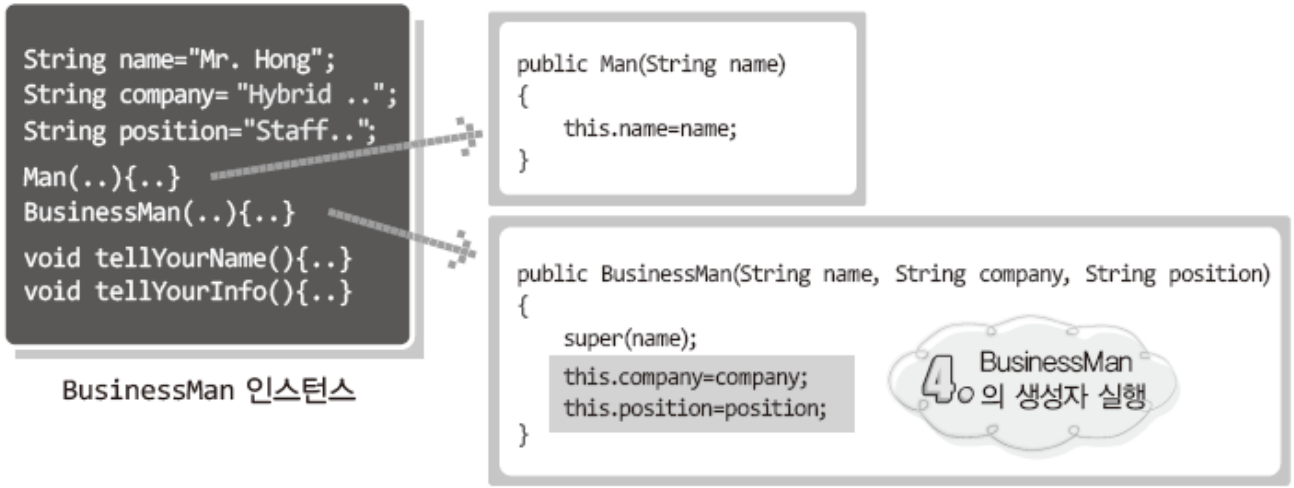
```
public static void main(String[] args)
{
    BusinessMan man1
        =new BusinessMan("Mr. Hong", "Hybrid 3D ELD", "Staff Eng.");
    BusinessMan man2
        =new BusinessMan("Mr. Lee", "Hybrid 3D ELD", "Assist Eng.");

    System.out.println("First man info.........");
    man1.tellYourName();
    man1.tellYourInfo();
    System.out.println("Second man info.........");
    man2.tellYourInfo();
}
```

상속관계에 있는 인스턴스의 생성과정 1~3



여기서 중요한 사실은 하위 클래스의 생성자 내에서 상위 클래스의 생성자 호출을 통해서 상위 클래스의 인스턴스 멤버를 초기화 한다는 점이다!

상속관계에 있는 인스턴스의 생성과정 4



결론 1! 하위 클래스의 생성자는 상위 클래스의 인스턴스 변수를 초기화 할 데이터까지 인자로 전달받아야 한다!

결론 2! 하위 클래스의 생성자는 상위 클래스의 생성자 호출을 통해서 상위 클래스의 인스턴스 변수를 초기화 한다!

결론 3! 키워드 super는 상위 클래스의 생성자 호출에 사용된다. super와 함께 표시된 전달되는 인자의 수와 자료형을 참조하여 호출할 생성자가 결정된다!

■ 반드시 호출되어야 하는 상위 클래스의 생성자

```
class AAA
{
    int num1;
}

class BBB extends AAA
{
    int num2;
}
```

AAA() { }

자동으로 삽입되는 디폴트 생성자의 형태

BBB() { super(); }

```
class AAA
{
    int num1;
}

class BBB extends AAA
{
    int num2;
    BBB() { num2=0; }
}
```

결과적으로 어떠한 형태로건(프로그래머가 직접 삽입하건, 컴파일러가 삽입하건) 상위 클래스의 생성자는 반드시 호출이 이뤄진다!

자동으로 삽입되는 상위 클래스의 생성자 호출문!

super();

14-3. 상속과 접근제어 지시자

protected 지시자

지시자	클래스 내부	동일 패키지	상속받은 클래스	이외의 영역
private	●	×	×	×
default	●	●	×	×
protected	●	●	●	×
public	●	●	●	●

default 패키지로 묶인 두 클래스

```
class AAA
{
    int num1;
    protected int num2;
}

class BBB extends AAA
{
    BBB()
    {
        num1=10;     // AAA 클래스의 default 멤버에 접근
        num2=20;     // BBB 클래스의 protected 멤버에 접근
    }
}
```

이 둘은 상속관계에 앞서 동일 패키지로 묶인 BBB 클래스에 의해 접근 가능!

private 멤버도 상속은 된다. 다만 접근만 불가능!


```
class Accumulator
{
    private int val;
    Accumulator(int init){ val=init; }
    protected void accumulate(int num)
    {
        if(num<0)
            return;
        val+=num;
    }
    protected int getAccVal(){return val;}
}
```

```
class SavingAccount extends Accumulator
{
    public SavingAccount(int initDep)
    {
        super(initDep);
    }
    public void saveMoney(int money)
    {
        accumulate(money);
    }
    public void showSavedMoney()
    {
        System.out.print("지금까지의 누적금액 : ");
        System.out.println(getAccVal());
    }
}
```

private 멤버도 상속이 된다! 다만, 함께 상속된 다른 메소드를 통해서 간접 접근을 해야만 한다!

나는 정말 JAVA를 공부한 적이 없다고

14-4. static 변수(메소드)의 상속과 생성자의 상속에 대한 논의

static 변수도 상속이 되나요?

static 변수는 접근의 허용여부와 관계가 있다. 따라서 다음과 같이 질문을 해야 옳다!

"상위 클래스의 static 변수에 하위 클래스도 그냥 이름만으로 접근이 가능한가요?"

이 질문에 대한 답은 YES!

```
class Adder
{
    public static int val=0;
    public void add(int num) { val+=num; }
}

class AdderFriend extends Adder
{
    public void friendAdd(int num) { val+=num; }
    public void showVal() { System.out.println(val); }
}
```

상위 클래스의 static 변수에 이름으로 직접 접근이 가능!