

# 열혈 Java 프로그래밍

Chapter 14. 클래스의 상속 1: 상속의 기본

## 14-1. 상속의 기본 문법 이해

# 상속의 매우 치명적인 오해

---

“코드의 재활용을 위한 문법입니다.” ( X )

“연관된 일련의 클래스들에 대해 공통적인 규약을 정의할 수 있습니다.” ( O )

# 상속의 가장 기본적인 특성

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

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

```
BusinessMan man = new BusinessMan( );
```

man  
참조변수

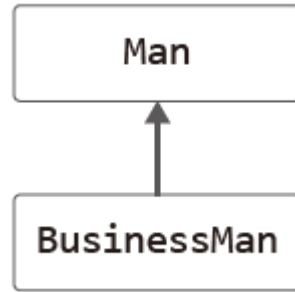
String name : Man의 멤버  
String company;  
String position;  
void tellYourName( ) {..} : Man의 멤버  
void tellYourInfo( ) {..}

BusinessMan 인스턴스

# 상속 관련 용어의 정리와 상속의 UML 표현

---

```
class Man {  
    . . .  
}  
  
class BusinessMan extends Man {  
    . . .  
}
```



상속의 대상이 되는 클래스 상위 클래스, 기초 클래스, 부모 클래스

ex) Man 클래스

상속을 하는 클래스 하위 클래스, 유도 클래스, 자식 클래스

ex) BusinessMan 클래스

# 상속과 생성자1

---

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

BusinessMan 인스턴스 생성시 문제점은?

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

# 상속과 생성자2

---

```
class BusinessMan extends Man {
    String company;
    String position;

    public BusinessMan(String name, String company, String position) {
        // 상위 클래스 Man의 멤버 초기화
        this.name = name;

        // 클래스 BusinessMan의 멤버 초기화
        this.company = company;
        this.position = position;
    }

    public void tellYourInfo() { . . . }
}
```

```
class Man {
    String name;

    public Man(String name) {
        this.name = name;
    }
    . . .
}
```

모든 멤버의 초기화는 이루어진다. 그러나  
생성자를 통한 초기화 원칙에는 어긋남!

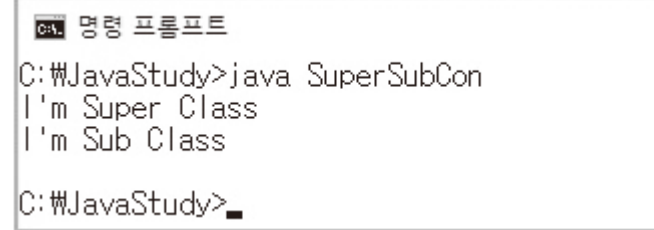
```
BusinessMan man =
    new BusinessMan("YOON", "Hybrid ELD", "Staff Eng.");
```

# 상속과 생성자3: 생성자 호출 관계 파악하기

```
class SuperCLS {  
    public SuperCLS() {  
        System.out.println("I'm Super Class");  
    }  
}
```

```
class SubCLS extends SuperCLS {  
    public SubCLS() {  
        System.out.println("I'm Sub Class");  
    }  
}
```

```
class SuperSubCon {  
    public static void main(String[] args) {  
        new SubCLS();  
    }  
}
```



```
명령 프롬프트  
C:\JavaStudy>java SuperSubCon  
I'm Super Class  
I'm Sub Class  
C:\JavaStudy>
```

상위 클래스의 생성자 실행 후  
하위 클래스의 생성자 실행 됨

호출할 상위 클래스의 생성자 명시하지 않으면 void 생성자 호출 됨



# 상속과 생성자4: 상위 클래스의 생성자 호출 지정

---

```
class SuperCLS {  
    public SuperCLS() {  
        System.out.println("...");  
    }  
  
    public SuperCLS(int i) {  
        System.out.println("...");  
    }  
  
    public SuperCLS(int i, int j) {  
        System.out.println("...");  
    }  
}
```

```
class SubCLS extends SuperCLS {  
    public SubCLS() {  
        System.out.println("...");  
    }  
  
    public SubCLS(int i) {  
        super(i);  
        System.out.println("...");  
    }  
  
    public SubCLS(int i, int j) {  
        super(i, j);  
        System.out.println("...");  
    }  
}
```

키워드 `super`를 통해 상위 클래스의 생성자 호출을 명시할 수 있음

# 적절한 생성자 정의의 예

---

```
class Man {
    String name;

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

class BusinessMan extends Man {
    String company;
    String position;

    public BusinessMan(String name, String company, String position) {
        super(name);
        this.company = company;
        this.position = position;
    }
    public void tellYourInfo() {
        System.out.println("My company is " + company);
        System.out.println("My position is " + position);
        tellYourName();
    }
}
```

# 단일 상속만 지원하는 자바

---

```
class AAA {...}
```

```
class MMM extends AAA {...}
```

```
class ZZZ extends MMM {...}
```

자바는 다중 상속을 지원하지 않는다.

한 클래스에서 상속할 수 있는 최대 클래스의 수는 한 개이다.

## 14-2. 클래스 변수, 클래스 메소드와 상속

# 클래스 변수, 메소드는 상속이 되는가?

---

```
class SuperCLS {  
    static int count = 0;    // 클래스 변수  
  
    public SuperCLS() {  
        count++;    // 클래스 내에서는 직접 접근이 가능  
    }  
}
```

} 프로그램 전체에서 딱 하나만 존재하는데 상속의 대상이 되겠는가?

```
class SubCLS extends SuperCLS {  
    public void showCount() {  
        System.out.println(count);    // 상위 클래스에 위치하는 클래스 변수에 접근  
    }  
}
```

} 그러나 하위 클래스에서 이름만으로 접근 가능하다!  
접근 수준 지시자에서 허용한다면!

*The End*

Chapter 14의 강의를 마칩니다.