




# 01.09 月\_ 상속(Inheritance)

uml - Google Search

Welcome to the Unified Modeling Language™ (UML®) website. Feel free to browse news and articles on UML, success stories, available certification and ...

 <https://www.google.com/search?q=uml&oq=uml&aqs=chrome..69i57j0i131i433i512l2j0i512l7.942j0j7&sourceid=chrome&ie=UTF-8>

## 「classのお浸い！」

class는 객체(Object , Instance)를 찍기 위한 붕어빵틀!



클래스안에 들어갈 수 있는 구성 요소 4가지 (초기화 블록 제외)

- **멤버 변수**
  - class 안의 변수
  - OOP에서의 3가지 변수 종류  
→ 멤버 변수 ,지역 변수, 매개변수 ( 생명주기와, Scop에 따라 봐야함!)
- **멤버 메소드**
  - 함수와 같다
  - 반환형이 필요하다
  - 리턴값은 반드시 1개!
  - **Overloading**가능
- **생성자**
  - class 이름과 동일하게 생성된다

- 객체가 생성될 때 초기화 작업을 한다.
- **Overloading** 가능
- 반환형이 없다 ( 주소값을 반환하기 때문)
- 초기화 블록이라는 개념을 제공  
초기화를 하기 위한 알고리즘이 들어가는데 , 생성자 보다 앞서 호출된다!! ( 공통적으로 생성자들의 앞에서 행해져야 할 것들을 초기화 블록쪽으로 빼놓고 실행 시키기 )
  - 인스턴스 초기화 블록 : 객체가 만들어 질 때 마다 초기화 블록 생성
  - 클래스 초기화 블록 : 클래스가 처음 쓰일 때 1번 호출된다
  -

- 소멸자

- java에서는 가비지 콜렉터가 있어서 소멸자는 없다

```
public class TEST5 {
```

```
public static void main(String[] args) {
```

- 객체가 만들어지면 메모리 영역 중 , heap 에 올라간다.
- 객체가 메모리에 올라가면 가지는 멤버들이 2가지 있다  
⇒ **멤버 변수** and **멤버 메소드**
- 객체를 찍을 때 객체가 소멸될 때 어떤 알고리즘을 집어 넣고 싶을 때 사용하는 것이 “**생성자**” , “**소멸자**”

## 相續 상속 +

1. 뒤를 이음.
2. 일정(一定)한 친족적(親族的) 신분(身分) 관계(關係)가 있는 사람 사이에서, 그 한 쪽이 사망(死亡)하거나 또는 ...



상속 관계가 되면 ?

: 부모 class에 선언된 **멤버** 들을 물려 받는다 (코드의 **재사용성** ↑)

oop 에서는 class라는 단위로 코드를 작성하고

기존에 있던 코드들을 **상속** 으로 재사용하면서 여러 기술들을 걸어 줄 수가 있다.

## 부모 요소의 멤버를 물려 받는다

```
class A {
    int a_x = 10;

    int getAx() {
        return a_x;
    }
}

class B extends A {
    ...
}

public class Main {
    public static void main(String arg[]) {
        B b1 = new B();
        System.out.println(b1.getAx());
    }
}
```



1. 상속이 무엇?

⇒ 상속이란 어떤 클래스를 다른 클래스에게 자신의 멤버를 공유하는 것 ( 부모 개념 ← 자식 개념)

```
class 부모
{

}

class 자식 extends 부모 {
```

```
}
```

## ? 2. 상속을 왜 씬?

⇒

1. A, B 라는 class가 있다면 A에게 있는 멤버가 B에 없을 경우 B가 A의 내용을 참조하려고 사용함
2. 마치 Logic Tree 의 형태로 가지치기 하듯 뺄거나할 수 있다
3. 불필요한 코드의 再사용을 방지하여 메모리 용량을 줄일 수 있다

## ? 3. 자바에서 가지는 상속의 특징은?

⇒ 부모 요소의 멤버를 물려 받는다

### ★ 교수님 질문

연산자를 사용하는 것도 알고리즘!!

1. 멤버 변수를 선언할 때 = 하는 것은 초기값으로 하는 것
2. 미리 만들어진 변수에 값을 집어넣는 것은 알고리즘 !!  
⇒ 생성자 또는 메소드, 초기화블럭에 들어가야 한다.

알고리즘은 전부 **생성자** or **메서드** or **초기화 블럭** 안에만 들어가야 한다!!

```
package test;

class Player{
    int num ;
}
class Center extends Player {
    // 알고리즘은 무조건 메소드나 생성자에 들어가야 한다
    {num = 5;}
}
```

```

class Forward extends Player {
    int num = 7;
}

class PF extends Forward {
    int num = 14;
}

class SF extends Forward {
}

class Guard extends Player {
}

class PG extends Guard {
}

class SG extends Guard {
}

public class TEST3 {

    public static void main(String[] args) {
        Player player1 = new Player();
        Center player2 = new Center();
        System.out.println(player2.num);

    }

}

```

```

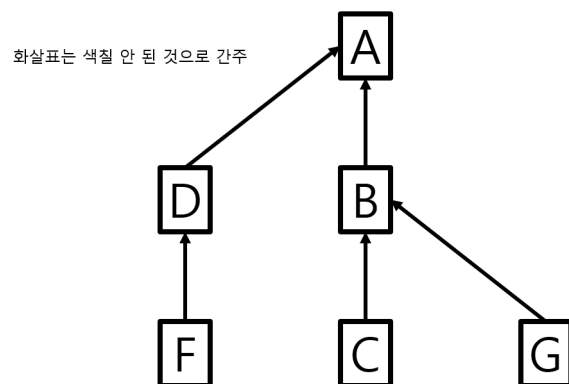
package test;

class A{
}
class B extends A {
}

class C extends B {
}
class D extends A {
}

class F extends D {
}

```



```

}
class G extends B {

}

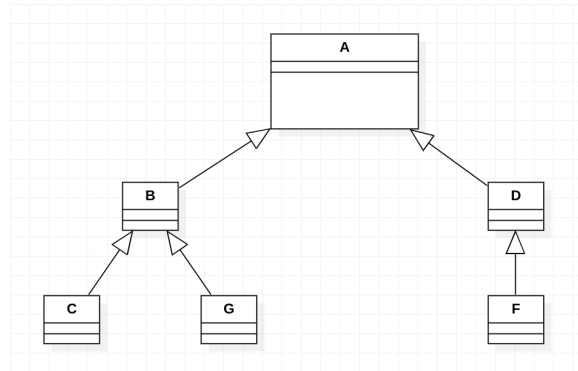
public class TEST3 {

    public static void main(String[] args) {
        // TODO Auto-generated method stub

    }

}

```



```

package test;

class Player{

}
class Center extends Player {

}

class Forward extends Player {

}
class PF extends Forward {

}
class SF extends Forward {

}
class Guard extends Player {

}
class PG extends Guard {

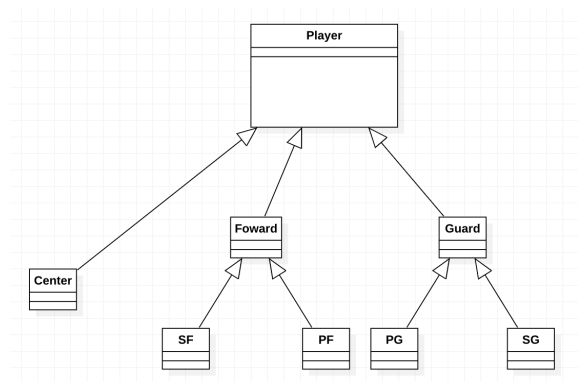
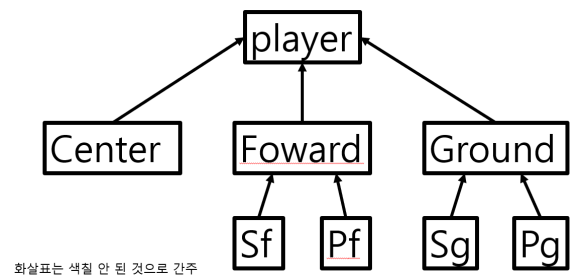
}
class SG extends Guard {

}

public class TEST3 {

    public static void main(String[] args) {

```



```
}  
  
}
```

★ ★같은 속성 ( 멤버 변수 , 멤버 메소드 ) 을 가지는 공통 분모가 보이기 시작한다! ★  
⇒ ★상속을 고민해야 한다★

클래스를 설계하다보면 여러 클래스들이 있다.

그 클래스들의 공통적인 속성을 가지는 것이 보이면 **상속**을 도입한다

1. class 설계
2. 만든 class 들이 같은 부류고, 같은 레벨이면 공통적인 속성을 가질 것이다.
3. 상속을 고려해본다 ( 같은 속성들을 부모로 다 올린다)

**필요하고 변경할게 있으면 부모에서 값을 추가하고, 변경하면 된다**

```
class Player {  
    String name;  
    int num;  
  
    void shoot() {  
        System.out.println("Player 슛");    "Overridng" 의 적용 대상은 메소드  
        // => 부모로부터 물려받은 메소드를 내가 새롭게 재정의 하는 것  
    }  
}  
  
class SG extends Player {    // => 상속이 되어서 , name 과 num 을 상속받는다  
    SG(String argName, int argNum) {  
        name = argName;  
        num = argNum;  
    }  
    void shoot() {  
        System.out.println("SG스�");  
    }  
}
```

```

}

class PF extends Player {                                // => 상속이 되어서 , name 과 num 을 상속받는다
    PF(String argName, int argNum) {
        name = argName;
        num = argNum;
    }
}

public class TEST5 {

    public static void main(String[] args) {
        new SG("", 1);
        new PF("", 1);
    }
}

```

오버라이딩 - 동적 바인딩

참조변수 ← 다형성 (많은 형태의 성질을 품을 수 있다는 )  
 참조변수가 부모의 자료형으로 자식들을 가르킬 수 있다

추상