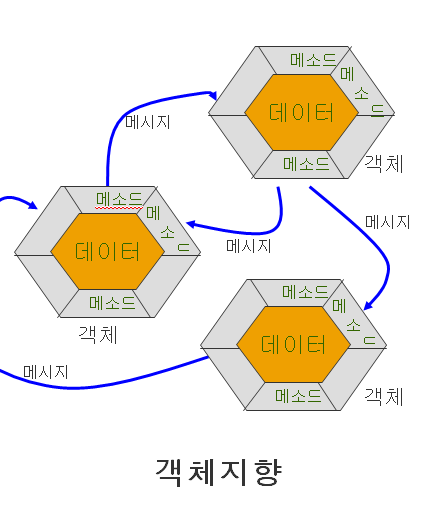
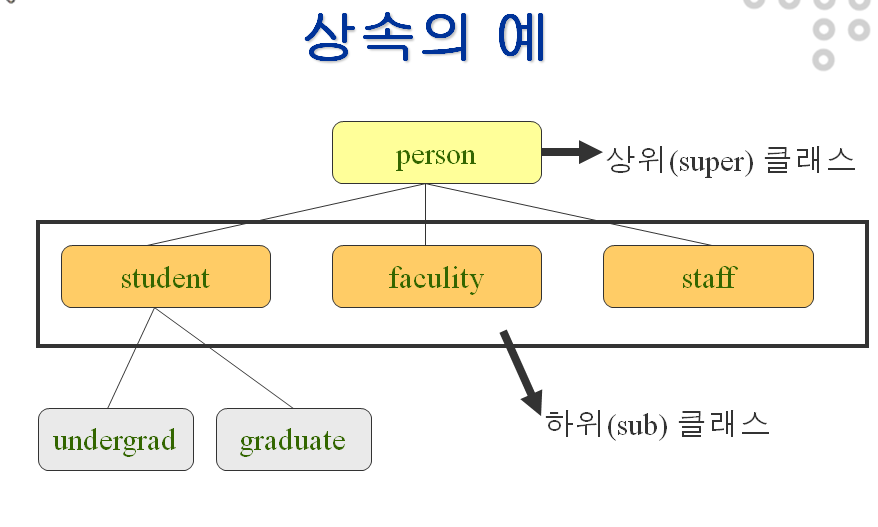
**객체지향 언어의 주요개념 (원칙) 정리**

* **객체지향의 배경** 
  + 소프트웨어 모듈의 재사용과 독립성을 강조
* **객체** 
  + 효율적인 정보관리를 위하여 의미를 부여하고 분류하는 개념적인 단위

****

* **객체의 구성**
  + 속성의 값을 나타내는 데이터(data)
  + 데이터를 변경하거나 조작할 수 있는 메소드로 구성
* **상속**

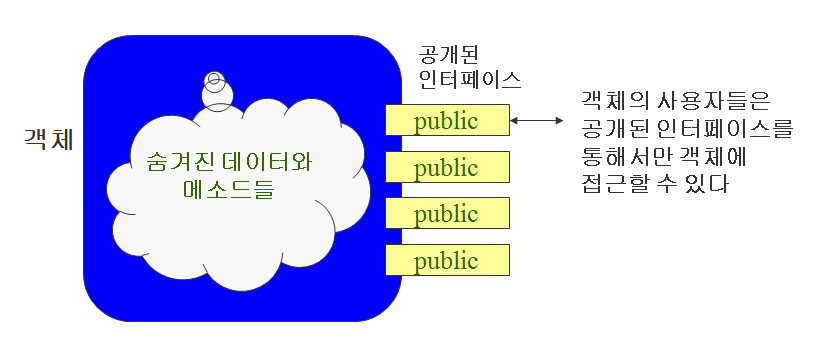
1. 상속관계의 클래스들은 **계층구조**를 구성할 수 있다
2. 하위 계층의 클래스는 상위 계층의 모든 요소를 상속 받고 추가적으로 필요로 되는 새로운 자료구조와 메소드를 더 가진다
3. 하위 클래스는 상위 클래스를 확장한 개념
4. 상속의 개념을 이용하여 소프트웨어의 재사용(reusing)을 지원

****

**Ex. Sds.icto55.oop.example.inheritance**

* **캡슐화(정보은닉)**

1. 객체를 캡슐화 하여 What만 보여주고 How는 감춘다.
2. 객체를 작성할 때 숨겨야 하는 정보(private)와 공개해야 하는 정보(public)를 구분하여 작성
3. 객체의 사용자는 기능만 알고 사용하며 어떻게 처리되는지는 은폐된다(Information Hiding)

****

**장점**

* 객체에 포함된 정보의 손상과 오용을 막을 수 있다.
* 객체 조작 방법이 바뀌어도 사용방법은 바뀌지 않는다.
* 데이터가 바뀌어도 다른 객체에 영향을 주지 않아 독립성이 유지된다.
* 처리된 결과만 사용하므로 객체의 이식성이 좋다.
* 객체를 부품화 할 수 있어 새로운 시스템의 구성에 부품처럼 사용할 수 있다.

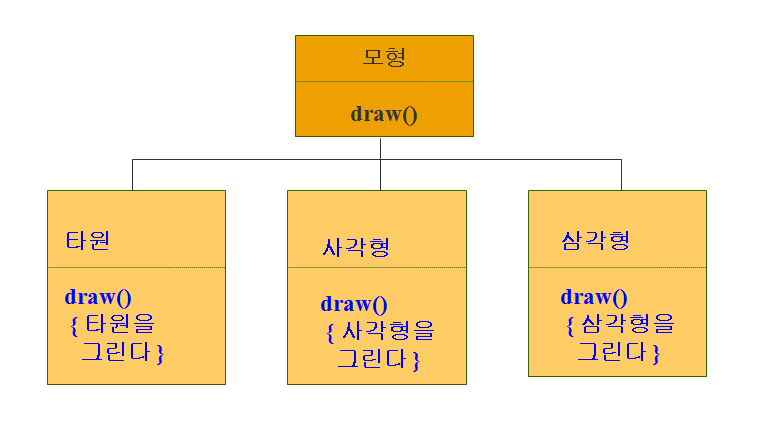
**Ex. Sds.icto55.oop.example.encapsulation**

* **다형성**

“one interface, multiple implementation”

하나의 인터페이스를 사용하여 다양한 구현 방법을 제공

Polymorphism = 다양한(poly) + 변신(morphism)



모형 A ; // 상위 클래스 타입의 객체 변수 A 선언

A = new 타원();

// 상위 클래스 타입의 객체 변수 A에 타원 클래스의 객체를 생성하여 할당

A.draw();

// 타원 클래스에 기술된 draw() 메소드를 수행하여 선언된 타원을 그린다

A = new 사각형();

// 상위 클래스 타입의 객체 변수 A에 사각형 클래스의 객체를 생성하여 할당

A.draw();

// 사각형 클래스에 기술된 draw() 메소드를 수행하여 선언된 타원을 그린다

…………………..

* 모형에서 선언된 draw() 메소드는 할당되는 하위 클래스의 객체에 따라 다양한 변신을 시도하여 서로 다른 결과를 나타낸다
* 메시지에서 요구한 메소드(draw())의 매핑을 동적으로(실행시간) 수행

**Ex. Sds.icto55.oop.example.polymorphism**

