상속과 다형성

자바의 상속, 다형성, 추상화 등등 이해하기

# 0. 학습 목표

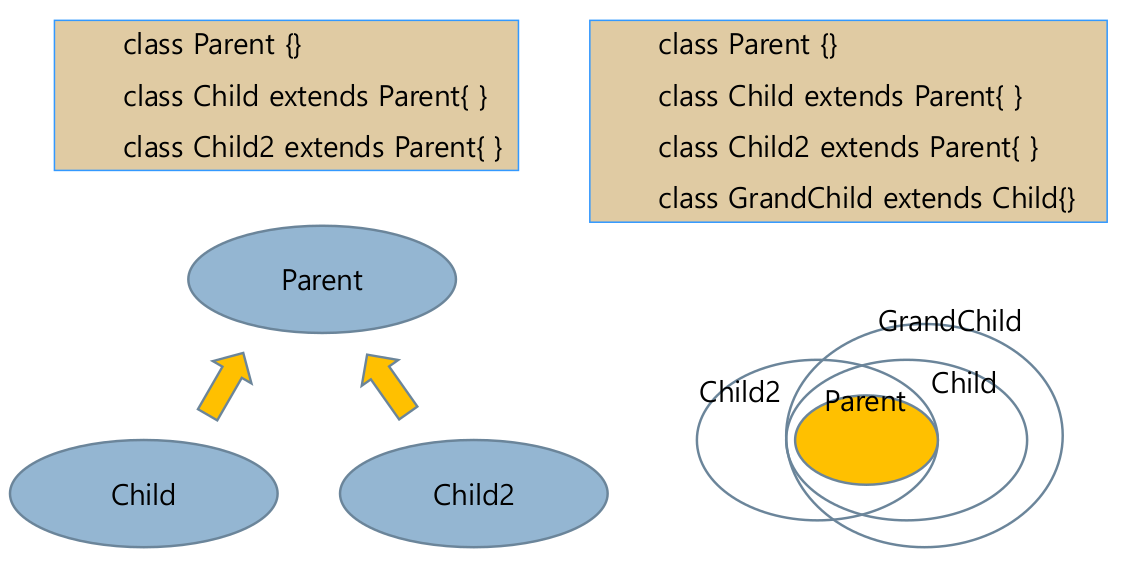
1. 자바 상속의 개념 이해하기
2. protected 접근 지정자
3. 상속 시 생성자의 실행 과정 이해하기
4. 업캐스팅과 inscanceof 연산자
5. 메소드 오버라이딩과 동적 바인딩 이해하기
6. 추상 클래스
7. 인터페이스

# 1. 자바 상속의 개념 이해하기

### 상속이란?

* 상속이란 **미리 만들어 둔 클래스를 다시 이용하는 방법**을 말함.
* 상속은 클래스에서만 통용되는 개념으로, **extends** 키워드를 사용한다.

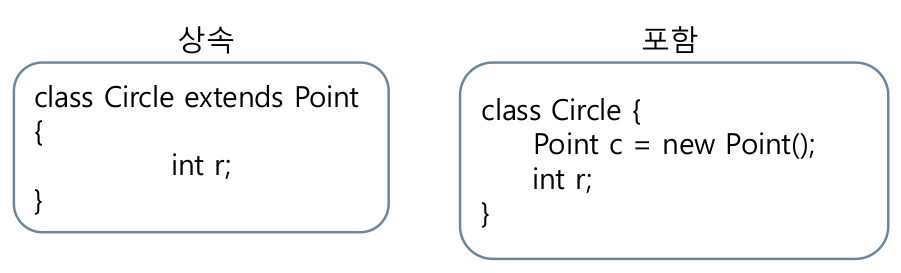
Ex) class Child extends Parent



### 상속의 필요성 : 중복이 제거되고 클래스 구조가 간단해진다!

### 포함 / 상속 관계

1. 상속 관계 : ~은 ~이다 (is-a) 관계 성립.
2. 포함 관계 : ~은 ~을 가지고 있다 (has-a) 관계 성립.



### 클래스 상속과 객체

* 상속 선언 : extends 키워드로 선언한다.
* 부모 클래스를 물려받아 확장한다는 의미!
* 상속을 하면 서브 클래스 객체는 슈퍼 클래스 멤버를 포함하게 된다.

### 자바 상속의 특징

* **클래스 다중 상속이 불가능**함
* 모든 자바 클래스는 묵시적으로 Object 클래스를 상속받음!

java.lang.Object 클래스는 모든 클래스의 슈퍼 클래스임.

# 2. protected 접근 지정자

### 슈퍼클래스 멤버에 대한 서브클래스의 접근



### protected 멤버

* 같은 패키지의 모든 클래스에게 허용된다.
* **상속되는 서브 클래스(같은 패키지든 다른 패키지든 상관 없음)에게 허용**됨!

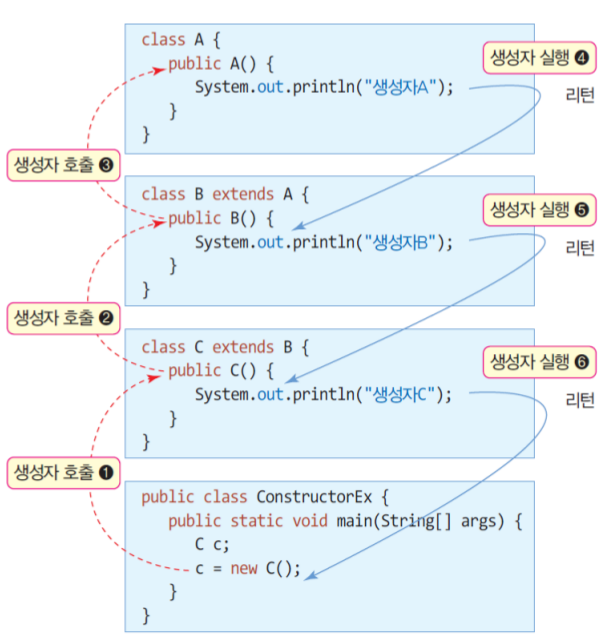
# 

# 

# 3. 상속 시 생성자의 실행 과정 이해하기

### 서브클래스/슈퍼클래스의 생성자 호출과 실행

* 생성자의 목적은 객체 초기화에 있다.
* 서브클래스의 생성자는 서브클래스를 초기화하고, 슈퍼클래스의 생성자는 슈퍼클래스를 초기화한다. **따라서 서브클래스 객체가 생성될 때, 서브클래스의 생성자와 슈퍼클래스의 생성자 모두 실행**된다!
* 주의할 점은, 슈퍼클래스의 생성자가 먼저 실행된 후 서브클래스의 생성자가 실행된다.



### 서브클래스와 슈퍼클래스의 생성자 선택

* 슈퍼 클래스와 서브 클래스는 각각 여러 개의 생성자를 작성할 수 있다.
* 서브클래스의 객체가 생성될 때 슈퍼클래스 생성자 1개와 서브클래스 생성자 1개 실행.
* 서브클래스의 생성자와 슈퍼클래스의 생성자가 결정되는 방식은?

1. 개발자가 명시적으로 선택하는 경우
2. 컴파일러가 기본생성자를 선택하는 경우

### 컴파일러가 기본생성자를 선택하는 경우

* 개발자가 서브클래스의 생성자에 대해 슈퍼클래스의 생성자를 명시적으로 선택하지 않으면!
* **[주의] 이럴 때 슈퍼클래스에 기본 생성자가 없으면 갖다 쓸 게 없어서 오류가 날 수 있음**

### 개발자가 명시적으로 선택하는 경우

* super()로 슈퍼 클래스의 생성자를 명시적으로 선택함!
* **[주의] 반드시 서브 클래스 생성자 코드의 제일 첫 줄에 위치해야 한다!**

# 4. 업캐스팅과 instanceof 연산자

### 업캐스팅

* 서브클래스의 레퍼런스를 슈퍼클래스 레퍼런스에 대입
* 슈퍼클래스 레퍼런스로 서브클래스 객체를 가리키게 되는 현상.

+) 슈퍼클래스 레퍼런스이기 때문에 객체 내의 슈퍼클래스 멤버만 접근할 수 있다.

### 다운캐스팅

* 슈퍼클래스 레퍼런스를 서브클래스 레퍼런스에 대입
* 업캐스팅된 것을 다시 원래대로 되돌리는 것
* 업캐스팅과 다르게 반드시 명시적 타입 변환을 지정해줘야 한다.

Ex) Student s = (Student)p;

### insctanceof 연산자

* 업캐스팅된 레퍼런스로는 객체의 실제 타입을 구분하기 어렵다!
* 슈퍼클래스가 여러 서브클래스에 상속되기 때문
* **instanceof** : 레퍼런스가 가리키는 객체의 타입을 식별하기 위해 사용한다. 

# 5. 메소드 오버라이딩과 동적 바인딩 이해하기

### 메소드 오버라이딩의 개념

* 서브클래스에서 슈퍼클래스의 메소드를 중복해서 작성함
* 슈퍼클래스의 메소드가 무력화됨. 항상 서브클래스에 오버라이딩한 메소드가 실행됨!

### 메소드 오버라이딩의 조건

* 슈퍼클래스 메소드의 원형과 동일하게 작성 **(메소드 이름, 인자 타입 및 개수, 리턴 타입!)**

### 다형성

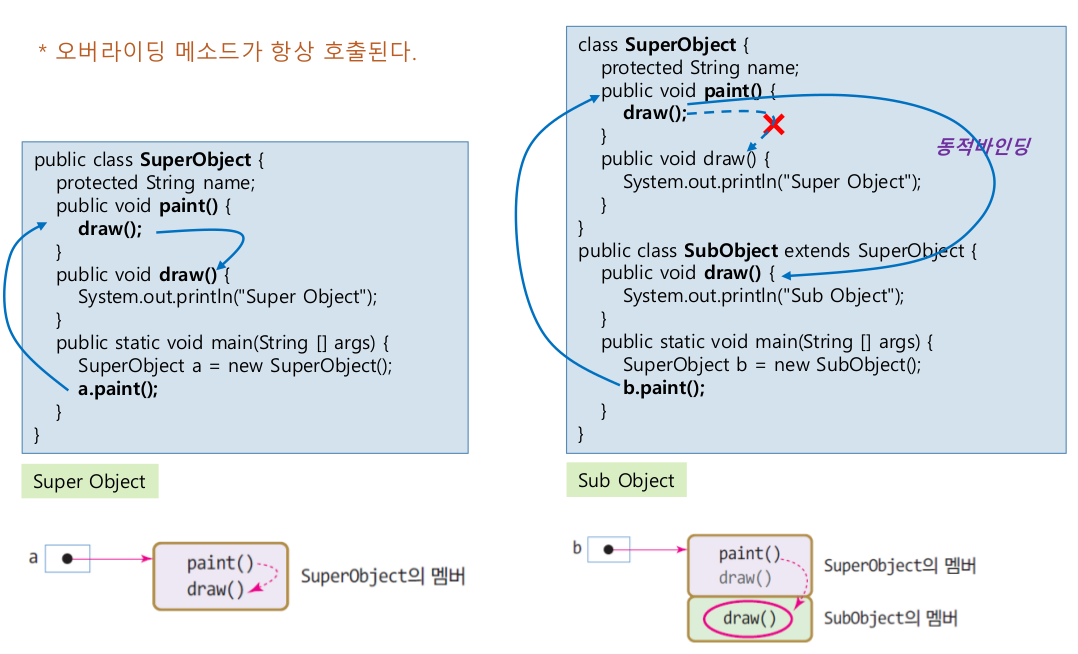
* 여러 가지 형태를 가질 수 있는 능력을 의미한다.
* 오버라이딩은 다형성을 실현하기 위한 목적을 가지고 있다.

하나의 인터페이스 (같은 이름)에 서로 다른 구현으로!

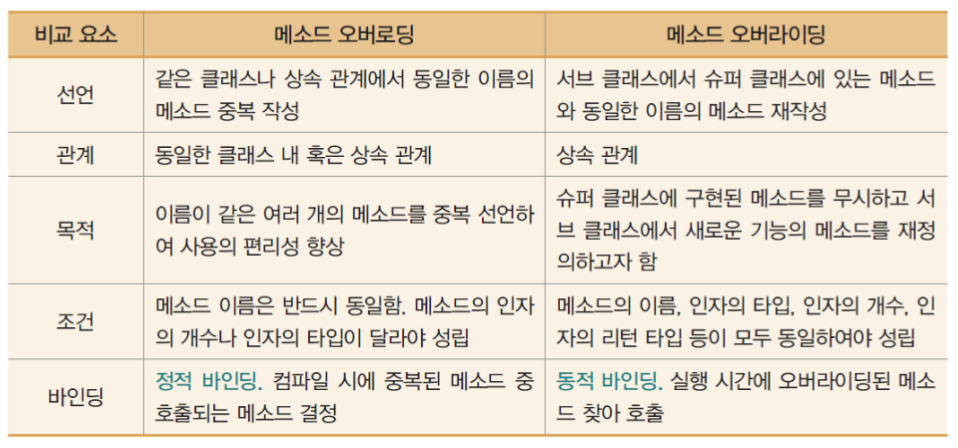
슈퍼클래스의 메소드를 서브클래스에서 각각 목적에 맞게, 다르게 구현한다.

### 동적 바인딩

* 오버라이딩 된 메소드가 항상 호출되는 것!



### 오버로딩과 오버라이딩



# 6. 추상 클래스

### 추상 클래스 : abstract로 선언된 클래스

* 추상 메소드를 가질 수도, 가지지 않을 수도 있다.
* **[주의] 추상 메소드를 가지고 있으면 반드시 abstract로 선언되어야 한다!**
* 추상 메소드 : abstract로 선언된 메소드로, 코드는 없고 원형만 선언되어 있다.
* 추상 클래스는 온전한 클래스가 아니기 때문에 인스턴스를 생성할 수 없다!

### 추상 클래스의 상속

* 추상 클래스를 상속받으면 추상 클래스가 된다.
* 서브 클래스 또한 abstract로 선언해야 한다!

### 추상 클래스의 구현

* 서브클래스에서 슈퍼클래스의 추상 메소드를 구현한다. (오버라이딩)
* 추상 클래스를 구현한 서브클래스는 추상 클래스가 아니다.

### 추상 클래스의 목적

* 상속을 위한 슈퍼클래스로 활용하기 위해서
* 서브클래스에서 추상 메소드를 구현하기 위해서
* 다형성을 실현하기 위해서

# 7. 인터페이스

### 자바 인터페이스 : 상수와 추상 메소드로 구성되어 변수 필드가 없다.

### 자바 인터페이스의 특징

* 상수와 추상 메소드로만 구성된다.

메소드 : public abstract 타입으로 생략할 수 있다.

상수 : public static final 타입으로 생략할 수 있다.

* 인터페이스로는 객체를 생성할 수 없다!

### 인터페이스 상속

* 인터페이스 간 상속이 가능하다. 인터페이스를 상속해 더 확장된 인터페이스를 만들 수 있음.
* extends 키워드로 상속을 선언한다.
* **다중 상속도 가능**하다!

### 인터페이스 구현

* 인터페이스를 상속받아, 모든 추상 메소드를 구현한 클래스를 선언한다.
* **implements** 키워드로 인터페이스를 구현한다
* 여러 개의 인터페이스를 동시에 구현할 수도 있다.
* 클래스 상속과 인터페이스 구현 또한 동시에 할 수도 있음!