

상속(Inherit)





상속

상속이란?

다른 클래스가 가지고 있는 멤버(필드와 메소드)들을 새로 작성할 클래스에서 직접 만들지 않고, 상속을 받음으로써 새 클래스가 자신의 멤버처럼 사용할 수 있는 기능

상속의 목적

클래스의 재사용, 연관된 일련의 클래스들에 대해 공통적인 규약을 정의

상속의 장점

1. 보다 적은 양의 코드로 새로운 클래스를 작성 가능
2. 코드를 공통적으로 관리하기 때문에 코드의 추가 및 변경이 용이함
3. 코드의 중복을 제거하여 프로그램의 생산성과 유지보수에 크게 기여함



상속

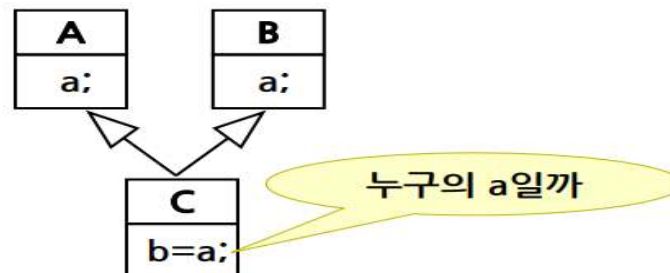
상속의 방법

단일상속(Single inheritance)

클래스간의 관계가 다중 상속보다 명확하고 신뢰성 있는 코드를 작성
자바에서는 다중상속을 지원하지 않음 -> **단일상속만을 지원**함

다중상속(Multiple inheritance)

여러 클래스로부터 상속을 받음 (C++에서는 가능함)
복합적인 기능을 가진 클래스를 쉽게 작성 가능함
서로 다른 클래스로부터 상속받은 멤버간의 이름이 같은 경우 문제점 발생





상속

방법

클래스간의 상속시에는 **extends** 키워드 사용함

표현식

[접근제한자] **class** 클래스명 **extends** 클래스명{ }

상속받는 클래스
후손클래스
자식클래스
파생클래스
서브(sub)클래스

상속하는 클래스
상위클래스
부모클래스
선조클래스
슈퍼클래스





상속

상속의 특징

부모클래스의 생성자, 초기화블럭은 상속 안됨

- 자식클래스 객체 생성시, 부모클래스 생성자가 먼저 실행
- 자식클래스 생성자 안에서 부모클래스 생성자 호출을 명시하고 싶으면 `super()` 활용

부모의 **private** 멤버는 상속은 되지만 접근 불가능

- 자식 객체 생성시에 부모의 필드값도 전달받은 경우, 자식 생성자 안에서 부모의 **private** 필드에 직접 초기값 대입 못함
- `super()` 이용하여 전달받은 부모 필드값을 부모생성자 쪽으로 넘겨서 생성
- `setter`, `getter` 매소드를 이용한 접근

모든 클래스는 **Object** 클래스의 후손

- **Object**클래스가 제공하는 메소드를 오버라이딩해서 메소드 재구현 가능.
예시) `java.lang.String` 클래스의 `equals()`과 `toString()`





상속

클래스와 메소드의 final 키워드

final class – 상속 X

예시) `public final class FinalClass{}` //상속 불가

final method – 오버라이딩 X

예시) `public final void method(){}` //오버라이딩 불가





제어자 조합 유의사항

대상	사용 가능한 제어자
클래스	public, (default), final, abstract
메소드	모든 접근 제어자, final, abstract, static
변수	모든 접근 제어자, final, static
지역변수	final

- 메소드에 static과 abstract를 함께 사용할 수 없다.
- 클래스에 abstract와 final을 동시에 사용할 수 없다.
- abstract 메소드의 접근제어자가 private일 수 없다.





super()

- 부모객체의 생성자를 호출하는 메소드이다. 후손 생성자에 기본적으로 부모생성자가 포함되어 있으며, 명시적으로 부모객체의 생성자를 호출할 부모 생성자가 가장 먼저 실행이 되어야 하기 때문에 **반드시 첫 줄에만 작성**할 수 있다.
- 매개변수가 있는 부모생성자 호출은 **super(매개변수,매개변수)**를 넣으면 된다.

super.

부모객체의 주소가 있는 참조변수로 생각하면 된다. 자식 클래스 내에서 부모클래스 객체에 접근하여 필드나 메소드를 호출할 때 사용





오버라이딩(overriding)

자식 클래스가 상속받은 부모 메소드를 재작성하는 것을 말한다.
부모가 제공하는 기능을 후손이 일부 고쳐 사용하겠다는 의미로,
자식객체를 통한 실행 시 후손 것이 우선권을 가진다.(동적바인딩)

- 메소드 헤드라인 위에 반드시 Annotation 표시

Annotation : 자바 컴파일러에게 알리는 주석문

예시) @Override

public 반환형 메소드명([자료형 매개변수]){}

- 부모메소드의 접근제어자 수정 가능

부모의 것보다 같거나 **넓은 범위**로만 변경 가능하다.

- 부모메소드의 예외처리 클래스 개수 수정 가능

부모 메소드의 예외처리 클래스 개수보다 크거나 처리범위가 넓으면 안됨





오버라이딩(overriding) 성립요건

부모 클래스의 메소드와 자식 클래스의 메소드의

- 이름이 동일해야 한다.
- 매개변수의 개수와 타입이 동일해야 한다.
- 리턴 타입이 동일해야 한다.
- private 메소드의 오버라이딩은 불가하다.





바인딩이란?

실제 실행할 메소드 코드와 호출하는 코드를 연결시키는 것을 바인딩이라고 한다. 프로그램이 실행되기 전에 컴파일이 되면서 모든 메소드는 정적 바인딩 된다.

동적바인딩이란?

컴파일시 정적바인딩된 메소드를 실행할 당시의 객체 타입을 기준으로 바인딩 되는 것을 동적 바인딩이라고 한다.





상속

동적바인딩 성립 요건

상속 관계로 이루어져 다형성이 적용된 경우에, 메소드 오버라이딩이 되어 있으면 정적으로 바인딩 된 메소드 코드보다 오버라이딩 된 메소드 코드를 우선적으로 수행하게 된다.





오버라이딩(overriding) VS 오버로딩(overloading)

Overriding(재정의)	Overloading(다중정의)
<ul style="list-style-type: none">메소드를 하위 클래스에서 정의	<ul style="list-style-type: none">메소드를 같은 클래스에서 정의
<ul style="list-style-type: none">메소드 이름 동일매개변수(개수 및 데이터 타입) 동일리턴 타입 동일	<ul style="list-style-type: none">메소드 이름 동일매개변수(개수나 데이터 타입) 다름리턴 타입 다를 수 있음
<ul style="list-style-type: none">접근 제어자 : 하위 메소드의 접근 범위가 상위 메소드의 접근 범위 보다 넓거나 같아야 함	<ul style="list-style-type: none">접근 제어자 : 관계 없음
<ul style="list-style-type: none">예외 처리 : 예외 발생시 같은 예외 형식이거나, 더 구체적인 예외 형식이어야 함	<ul style="list-style-type: none">예외 처리 : 관계없음

