4주차: 상속

https://github.com/whiteship/live-study/issues/6

상속

: 부모 클래스에서 정의된 필드와 메소드를 자식 클래스가 물려받는 것.

그럼 상속이 필요한 이유는 뭘까?

1. 공통된 특징을 가지는 클래스 사이의 멤버(필드, 메소드) 선언이 불필요하다.
2. 부모 클래스의 멤버(필드, 메소드)를 재사용함으로써 자식 클래스가 간결해진다.
3. 클래스간 계층적 분류 및 관리가 쉬워진다.

- 자바 상속의 특징

1. 부모클래스의 생성자**,** 초기화블럭은 상속 안됨.
   - 후손클래스 객체 생성시, 부모클래스 생성자가 먼저 실행되도록 되어 있음.
   - 후손클래스 생성자 안에서 부모클래스 생성자 호출을 명시하고 싶으면 super() 를 입력함
2. 자바에서는 다중상속을 지원하지 않는다. 따라서 **extends** 뒤에는 단 하나의 부모 클래스만 올 수 있다.
3. 자바에서는 상속의 횟수에 제한을 두지 않는다.
4. C++의 경우에는 최상위 클래스가 없지만 자바에서 최상위 클래스는 ***Object***클래스이다. 다시말해 Object 클래스만이 유일하게 **super class**를 가지지 않는다. 클래스를 상속받지 않는 모든 클래스는 Object 를 자동으로 상속받도록 되어있어, toString(), equals() 와 같은 메소드를 바로 사용할 수 있다.

하나씩 살펴보면,

1. 부모클래스의 생성자, 초기화블럭은 상속 안됨
   - 후손클래스 객체 생성시, 부모클래스 생성자가 먼저 실행되도록 되어 있음.
   - 후손클래스 생성자 안에서 부모클래스 생성자 호출을 명시하고 싶으면 super() 를 입력함

초기화 블럭은 어느 정도 이해가 된다. 값에 관계 없이 변수만을 상속받기만 해도 되니까. 그런데 왜 생성자는 안 되는지 궁금해서 찾아보다가,

> 1. 부모 클래스의 생성자는 상속되지 않고, 자식 클래스로 인스턴스를 생성할 때 자동적으로 부모의 기본 생성자가 호출된다.
>
> 2. 부모 생성자가 매개변수를 갖고 있다면 자식 클래스를 인스턴스화할 때 자동으로 호출되지 않는다.
>
> 3. 따라서 자식 생성자에서 명시적으로 부모 생성자를 호출해야 한다. 이때 사용되는 키워드가 **super();** 이다. 단, **super()**를 사용할 때는 자식 생성자의 첫 줄에 위치하여야 한다.
>
> **super**는 부모를 의미하고 괄호**(( ))** 안에 매개값을 전달하여 부모 생성자를 호출하게 된다. 이때 주의해야 할 내용이 있다. **super()**는 반드시 자식 클래스 생성자의 첫 줄에 위치하여야 하며 부모 생성자의 매개변수와 동일한 타입을 매개값으로 작성해야 한다.

라는 글을 보았다. 그러니까 원래는 자동으로 부모의 기본 생성자가 호출되지만 부모 생성자가 기본 생성자가 아닌 사용자가 매개변수를 추가한 생성자라면, 자동으로 호출되지 않고 자식 생성자에서 명시적으로 부모 생성자를 호출해주어야 한다는 것인데...

매개변수가 있는 경우에 super(매개변수)를 통해 부모 클래스의 생성자를 호출해야하는 건 알겠는데, 왜 상속이 아니라 호출의 개념일까**..?**

어차피 자식 생성자에서 실행이 되는 메서드라면 상속할 때 한 번에 받아와서 실행하면 안되는 걸까..? 생성자는 꼭 본인만 소유해야하는 건가?

- **super** 키워드

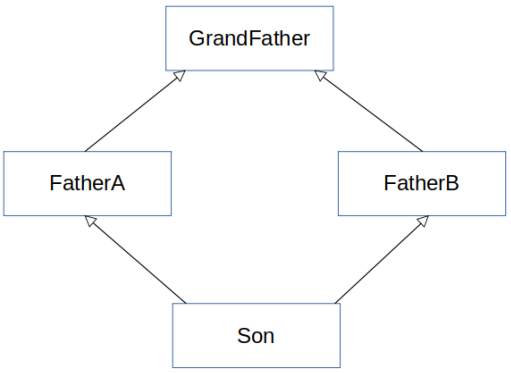
super()가 나온 김에 함께 보고 넘어가자면, **super**는 저번 시간에 한 **this** 와 유사하다.

this 가 자기 참조 변수라면, **super** 는 부모 클래스를 참조하는 변수이다.

마저 자바 상속의 특징을 살펴보면,

2. 자바에서는 다중상속을 지원하지 않는다.
   따라서 **extends** 뒤에는 단 하나의 부모 클래스만 올 수 있다.

1) 왜 다중 상속이 안될까? = 다이아몬드 문제



간단히 예를 들자면, 옆의 다이어그램의 경우 만약 Father A 와 Father B 가 GrandFather 에게 Hello() 라는 메서드를 받았고, 각자 메서드를 오버라이딩해서 Father A 는 **Hello, A** 를 출력하고 Father B는 **Hello, B**를 출력하도록 하였다.

이 경우 Son 이 Father A와 Father B를 모두 상속할 때, Son 이 Hello() 를 실행하면 이 Hello()는 어떤 걸 실행해야하는가의 문제가 생긴다.

2) 인터페이스는 되던데?

인터페이스는 실질적인 구현이 이루어지지 않고 메소드에 대한 선언만 하고 있기 때문에 위의 Hello()가 각자 다른 방식으로 오버라이딩 되지 않기 때문에 다중 상속이 가능하다.
애초에 interface 자체가 자바의 클래스가 다중 상속이 되지 않음으로 생기는 불편함을 보완하기 위해 생겼다!

3) interface도 다중 상속이 안될 수 있다?

default Method 를 사용하면 interface 도 다중 상속이 불가능하다. default method 가 뭔지 간단하게 보자면! ~~(인터페이스 너무 길게 하면 뇌절이니까...ㅠㅠ)~~

Java8 이상부터 나온 기능으로 메소드 선언 시에 default를 명시하게 되면 인터페이스 내부에서도 로직이 포함된 메소드를 선언할 수 있다...! ㄴㅇㅁㅇㄱ

```
public interface defaulInterface {
    default void defaultMethod() {
        System.out.println("default Method 실행")
    }
}
```

사실상 이렇게 되면 오버라이딩 시 객체마다 다른 결과가 나타나는 메서드가 실행될 것이고, 결국 다이아몬드 문제가 다시 나타나게 된다.

그럼 굳이 이걸 왜 쓰나... 싶다. 구현을 해둘 거면 추상 메서드랑 다른게 뭐냐.

Default Method가 나온 이유는 하위 호환성 때문이다.

기존 인터페이스를 이용해서 구현된 클래스들을 만들고 사용하고 있는 중, 인터페이스를 보완하는 과정에서 추가적으로 구현해야할 필수적인 메소드가 생긴다면, 이미 이 인터페이스를 구현한 클래스와는 호환성이 떨어 지게 되기 때문에 default 메소드를 추가하여 필수적으로 구현해야 할 메소드를 정의하면 하위 호환성은 유지시키며 인터페이스를 보완할 수 있다.

더 자세한 내용은 인터페이스 때 알아보도록 하자!

4) 도대체 자바 8~ 자바9에서는 무슨일이 벌어졌길래 변화가 많나

이번주 TMI)
JDK 1.8은 2014년도에 출시되었고 다음과 같은 기능과 특징들이 있다.

- 람다식(Lambda expressions)
- 함수형 인터페이스 (Functional Interface)
- 디폴트 메서드 (Default Method)
- JVM의 변화
- 병렬 배열 정렬(Parallel Array Sorting)
- 컬렉션을 위한 대용량 데이터 처리 (스트림)
- Optional
- Base64 인코딩과 디코딩을 위한 표준 API
- 새로운 날짜, 시간 API (Date & Time API

- 세부 설명1 : [자바] Java 8 버전 특징
- 세부 설명2 : https://bbubbush.tistory.com/23

여기서 다루기엔 너무 많고 길어서... 한 번 확인만 하고, 다음에 추가적으로 학습을 해보는걸로...

3. 자바에서는 상속의 횟수에 제한을 두지 않는다..

1) 위 문장을 보고 바로 들었던 의문이 들었다.

그럼 다른 언어에서는 횟수 제한을 두나?
그렇다면 다른 언어는 횟수를 두고 자바에서는 안 두는 이유는?

이에 대한 답은, C++ 도 자바와 마찬가지로 상속 횟수에 제한이 없고, 위에서 언급되었던 다중 상속을 제한하는 것도 C++에서는 제한하지 않았지만, C#은 다중상속을 제한해뒀음을 확인할 수 있었다.

+ 그럼 C++은 왜 다중 상속을 지원할까?
: 다중 상속 자체가 꽤 논란이 되는 문법이라고 한다. 많은 프로그래머들이 다중 상속 사용을 지양해야한다고 하고 있다. (저번에 java의 소멸자 같은 취급인 듯 하다.)
득보다 실이 더 많으니, 절대 사용하지 말아야하고 그냥 C++ 기본 문법에서 제외해야한다는 의견과 가급적 사용하지말자는 건 동의하는데 예외적으로 매우 제한적인 사용까지 부정할 필요는 없지 않을까...? 정도인 것 같다.

심지어 다중 상속으로만 해결할 수 없는 문제는 존재하지 않는다고 하니 legacy 같은 느낌이지 않을까...
tmi) C++ 에서 다중상속으로 인해 부모 클래스가 다르지만, 이름은 같은 메서드들은 부모클래스명::메서드명과 같은 형식으로 사용한다. (네임스페이스와 유사!)

횟수와 다중 상속 외에 상속에서 가장 직접적으로 제한되는 것은 **final** 키워드이다.

- **final** 키워드

처음 final 을 접하는 건 아마 상수 선언을 할 때가 아닌가 싶다. 변수명에 final 을 붙이면 변경할 수 없는 상수가 되는데, 클래스에도 이를 붙일 수 있다.

간단하게 정리해보자면,

- fianl 은 클래스, 메서드, 변수에 적용되며, abstract(추상화)와 동시에 사용될 수 없음.

애초에 추상화는 추후에 완성하라고 빈칸을 남겨뒀다의 느낌인데 같이 쓸 수가 없는게 당연하다.

final 클래스	상속 안됨 = 서브 클래스 가질 수 없음.	상속 금지

final 메서드	재정의가 안됨.	재정의 금지
final 필드	값을 변경할 수 없음.	변경 금지

- final 사용하는 이유

  : 중요한 클래스의 서브클래스가 시스템을 파괴하지 않게 하기 위해 중요한 클래스는 final 클래스로 선언한다. ex) String 클래스

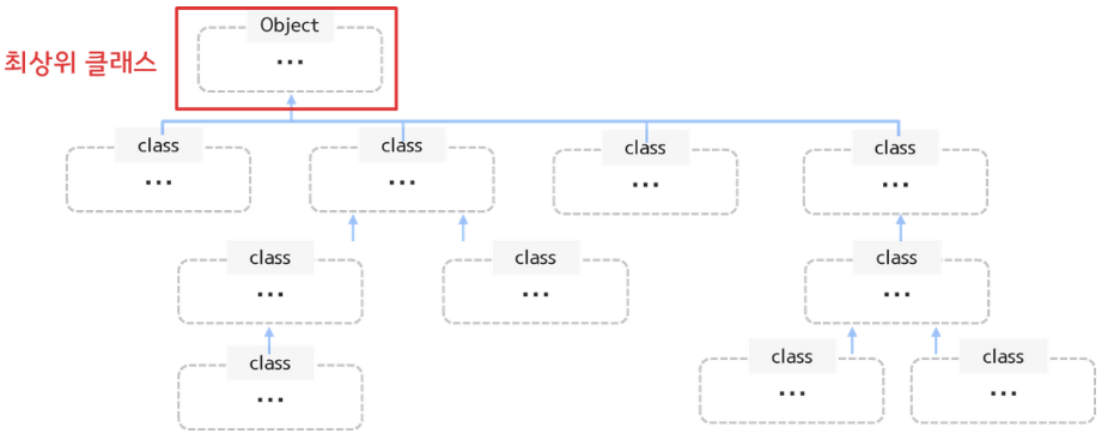
  당연히 상속이 안되기 때문에 내부의 모든 메서드는 오버라이딩이 될 수 없다.

  final 메소드는 부모 클래스에 있는 메소드를 자식 클래스가 그대로 쓰게 하고 싶을 때 사용한다.

  final 필드는 앞에서 언급했던 대로 상수화를 위해 사용하는데, 상수는 모든 클래스에서 공통적으로 사용될 가능성이 있고, 여러개를 생성하면 오히려 메모리 낭비이기 때문에 `final static 변수명`과 같이 **static** 키워드를 붙여 모든 클래스가 공유하도록 한다.

4. C++의 경우에는 최상위 클래스가 없지만 자바에서 최상위 클래스는 Object클래스이다. 다시말해 Object 클래스만이 유일하게 super class를 가지지 않는다. 클래스를 상속받지 않는 모든 클래스는 Object 를 자동으로 상속받도록 되어있어, toString(), equals() 와 같은 메소드를 바로 사용할 수 있다.

마지막 자바 상속의 특징은 **Object** 클래스이다.



- **Object** 클래스

왜 Object 클래스라는 걸 만들어서 최상위 클래스로 두었을까? 이에 대한 답을 찾다가 이런 질문을 보게 되었다. 2011년 포스팅인데 면접 질문으로 나왔다고 한다.

"왜 java.lnag.Object 는 interface로 구현하지 않고 클래스로 구현하였나요?"

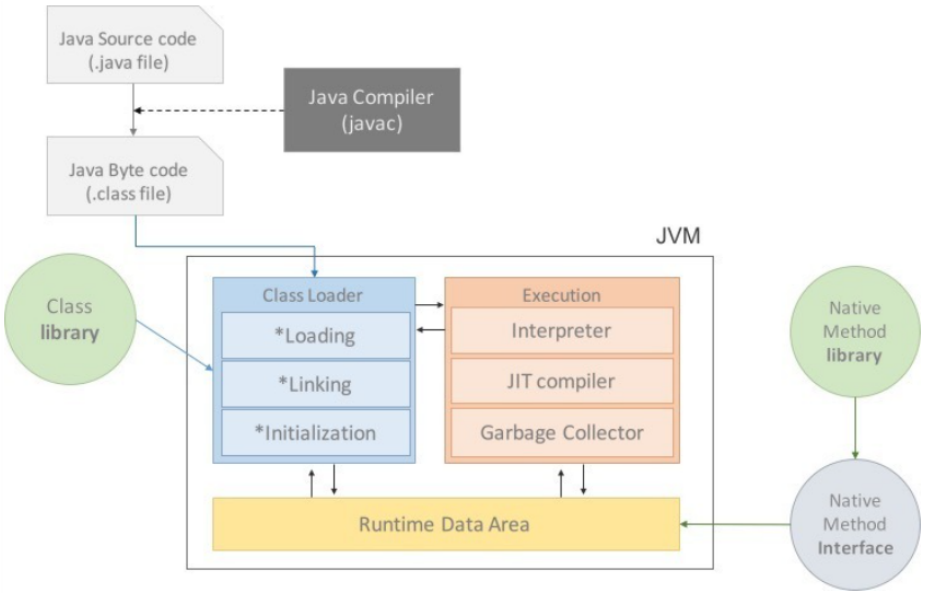
이 질문에 대한 답이 Object 라는 최상위 클래스를 만든 이유와 동일하다고 생각한다. Object 클래스의 메소드를 살펴보자!

- 객체를 shallow copy 해주는 Object : clone()
- 객체와 등가 되는 객체인지 boolean : equals(Object obj)
- 객체가 참조되는 무엇이 없으면 가비지컬렉터에 수집되는 void : finalize()
- 객체의 클래스 타입을 돌려주는 Class<?> : getClass()
- 객체의 해시 코드 값을 리턴하는 int : hashCode()

이 외에도 여러 메소드들이 있다.
이 메소드들의 특징은 어떤 객체든 지원되어야하는 메소드라는 점이다. interface 로 Object 가 구현이 되었다면, 이 모든 메소드들을 하나하나 구현해야하기 때문에 Object 를 만들 필요가 없지 않을까.

+ java.lang 패키지는 자바 컴파일러가 자동으로 추가하는 작업.
  1주차에 JVM에서 Execution 파트에 대해 공부할 때,
  > 클래스 로더를 통해 읽어들여 자바 *API*와 함께 실행하는 파트가 *Execution* 이다.

  라고 했다. 이 자바 API 중 하나가 java.lang 패키지이다. import 구문으로 호출해야 사용할 수 있는 다른 패키지들과는 달리 import 구문 없이도 자동으로 프로그램에 포함된다.



출처: 자바 튜토리얼 11 - 1 최상위 클래스 Object, String

- 메소드 오버라이딩

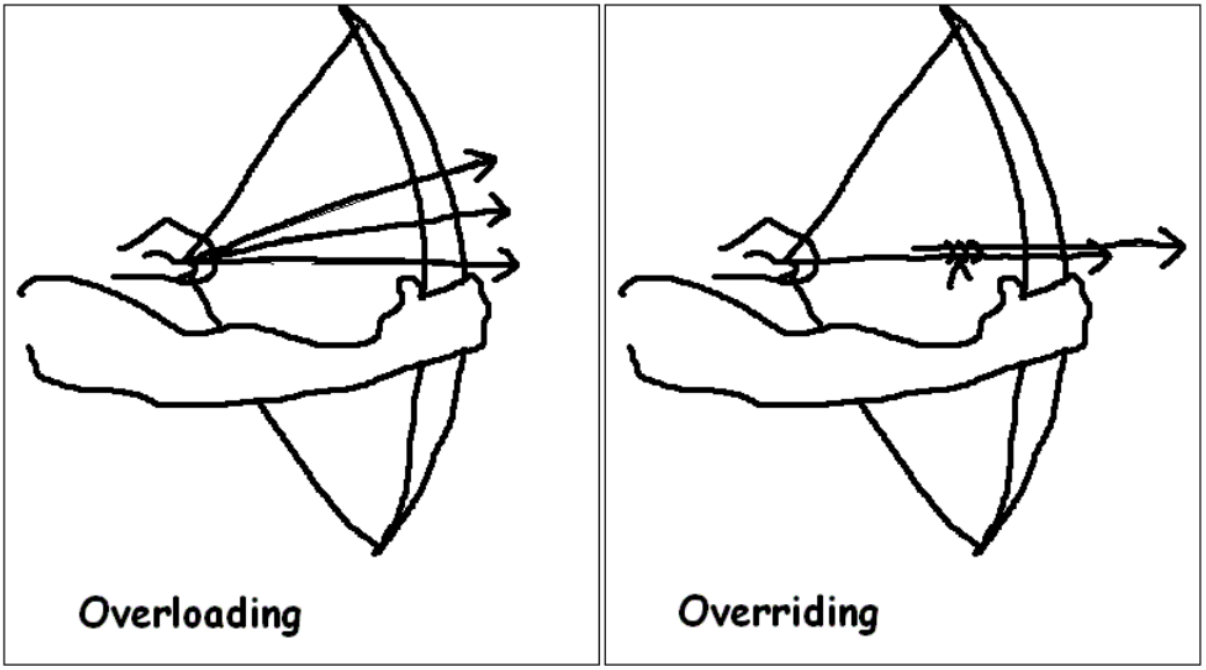
오버라이딩은 다 잘 아는 개념이라 생각한다. '자식 클래스가 부모 클래스의 메소드를 상속 받을 때, 자식이 부모 클래스의 메소드를 재작성하는 작업' 이라고 정리할 수 있을 것 같은데, 다형성과 같이 알아보려니 꽤 혼란스러웠다.

우선, 오버라이딩과 함께 계속 나오는 개념이 오버로딩이다. 오버로딩은 같은 이름의 함수(메소드)를 매개변수의 타입과 개수를 다르게 하여 여러 개를 정의할 수 있는 것인데, 오버라이딩과 오버로딩의 차이는 아래와 같다.

오버로딩(Overloading)과 오버라이딩(Overriding) 성립조건

구분	오버로딩	오버라이딩
메서드 이름	동일	동일
매개변수, 타입	다름	동일
리턴 타입	상관없음	동일

출처: https://private.tistory.com/25



출처: https://gbsb.tistory.com/235

이에서 알 수 있듯, 오버라이딩은 상속이 전제가 된 상태에서 실행되고, 오버로딩은 한 클래스 내에서 작동한다. 이까진 괜찮다. 하지만 다형성을 붙이면서 헷갈리기 시작한다.

이유는 어떤 곳에서는 다형성은 상속을 전제로 한다라고 하고, 어떤 곳에서는 자바에서 다형성을 지원하는 방법으로 메서드 오버로딩과 오버라이딩이 있다 라고 하기 때문이다...

먼말인가... 하면서 멘탈 부서지다가 얼추 답을....내린게..... 다형성을 정의하는 방법이 다르다는 것이다..ㅠㅠ

1) 오버로딩과 오버라이딩이 다형성을 지원하는 방식이다.

이렇게 말하는 사람들은 다형성을 하나의 객체가 여러 가지 타입을 가질 수 있는 것, 여러 가지 형태를 가질 수 있는 능력으로 보았다. 즉, 하나의 객체가 유연하게 다른 코드에 반응할 수 있다는 것을 다형성이라 본 것이다. 이를 자바에 적용하면 한 타입의 참조 변수로 여러 타입의 객체를 참조할 수 있도록 함으로써 다형성을 구현했다가 된다.

또한, 상속을 다형성을 달성할 수 있도록 도와준다고 보고 메소드 오버라이딩의 역할을 한 객체가 여러 기능을 다른 형태로 수행할 수 있도록 하는 것 이라고 하는 의견도 보았다.

2) 다형성을 구현하기 위해서는 상속이 필수 전제가 된다.

지난 주에 객체지향의 배경에 대해 공부하며 메시징 이라는 개념을 배웠다. 다형성에 상속이 필수 전제가 된다고 주장하는 사람들은 다형성을 같은 메시징에 대하여 서로 다른 처리를 할 수있는 것이라고 정의한다. 즉, 서로 다른 객체(여러 개)가 같은 메시지 혹은 코드에 대하여 서로 다른 방법으로 응답할 수 있는 기능을 이야기하는 것이다. *(메시징이 곧 메소드 호출과 같은 개념은 아니라고 한다... 이 부분은 이 링크 참고!)*

그러니, 오버로딩은 제외되고 오버라이딩 및 업캐스팅에 대한 부분만 가정하기 때문에 상속이 필수 전제가 되어야한다고 한 것이다.

뭔가... 둘 다 맞는 것 같은데 애초에 다형성이라는 개념의 정의가 확실히 정해진 게 아니라 사람들이 공부하면서 나온 개념이라 그런 것 같다.. 뭔가 더 알아보고는 싶은데 너무 불필요하게 집착한 것 같기도하고... 다음에 더 알아보는 걸로...

+ 참고하면 좋을 것 같은 링크! [Java] 메소드 오버라이딩/ 메소드 오버로딩을 통한 상속 다형성에 대한 이해와 Self 참조 - MangKyu's Diary

  사유: self 참조 변수와 이를 메시징과 엮어서 설명해줘서 색다르게 다가왔다. 추가로 덧붙이자면 여기서 오버로딩도 상속에서도 사용 가능하다고 한다.

- 다이나믹 메소드 디스패치 **(Dynamic Method Dispatch)**

https://velog.io/@maigumi/Dynamic-Method-Dispatch

다이나믹 = 동적으로

메소드 = 메소드를

디스패치 = 발송하다

인터페이스, 추상 클래스에 정의된 추상 메소드를 호출하는 경우가 될 수도 있고, 부모 클래스의 객체가 자식 클래스를 대입하여 업캐스팅(2주차였나 형변환 때 했었다!)이 될 때, 메소드를 호출하는 경우 어느 메소드를 호출할 지, 런타임 시점에 결정해주는 것이다!

- 추상 클래스

  3주차 클래스에서 잠시 맛보기처럼 나온 '추상' 이란 개념이다!

  추상 클래스는 추상 메서드가 하나라도 있으면 추상 클래스이고, **abstract** 를 사용하여 선언할 수 있다.

  > 추상
  > 여러가지 사물이나 개념에 공통되는 특성이나 속성을 추출해 파악하는 행위
  >
  > 추상화
  > 클래스간의 공통점을 찾아내서 공통의 부모를 설계하는 작업

  보통 특정 기능(메소드)을 당장 결정하지 못하고 상속될 클래스에서 결정해야하는 경우에 추상 메소드로 선언하여 자식 클래스에서 구현할 수 있도록 남겨두는 작업을 할 때 사용된다.

  그렇기 때문에 추상 클래스를 `미완성 설계도`로 비유하는 경우가 많다.

-

출처

- https://jyoel.tistory.com/39
- https://danmilife.tistory.com/21
- https://backenddeveloper.tistory.com/7
- https://siyoon210.tistory.com/125
- [Java, 다이아몬드 문제] Java에서 다중 상속을 지원 못하는 이유
- https://codechacha.com/ko/java8-default-methods/
- https://ssdragon.tistory.com/13
- https://m.blog.naver.com/PostView.naver?isHttpsRedirect=true&blogId=heartflow89&logNo=220961980579
- https://koey.tistory.com/92
- https://shrtorznzl.tistory.com/44
- https://creator1022.tistory.com/115
- http://www.incodom.kr/Java/java.lang
- https://osc131.tistory.com/163
- https://dgtl.tistory.com/43
- https://doompok.tistory.com/21
- https://terry9611.tistory.com/63
-