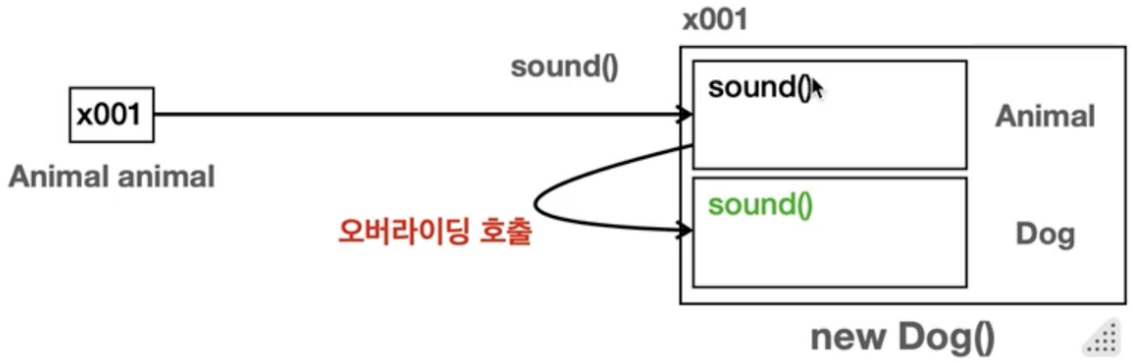
다형성 활용 [Java]

소유자	종수 김
태그	

[다형성 활용]



- 다형적 참조를 통해 Animal객체를 상속받는 Dog, Cat, Caw를 선언함으로써 셋을 같은 타입으로 묶어 함수의 매개변수로 넘기는게 가능
- 함수 오버라이딩을 통해 sound()함수는 Animal객체를 상속받는 메서드에서 구현한 내용이 출력
- Animal 변수타입의 배열을 만들어 상속 받는 객체를 모두 생성가능.

Q. Animal 클래스는 추상적인 개념인데 실체를 new로 생성해도 되는 건가 ?

Q. 만약 Animal을 상속받는 자식 객체가 sound() 오버라이딩 하지 않는다면 ?

[추상 클래스]

동물(Animal)과 같이 부모 클래스는 제공하지만, 실제 생성되면 안되는 클래스

기존 클래스와 완전히 같으나, new AbstractClass()와 같이 직접 인스턴스를 생성하지 못하는 제약이 있다.

추상 클래스는 일반 메서드와, 추상 메서드 둘 모두 가질 수 있다. [인터페이스는 로직을 갖는 일반 메서드를 사용할 수 없음.]
추상 메서드 - 자식 객체에서 '무조건' 오버라이딩 해라.
일반 메서드 - 자식 객체에서 오버라이딩 해도 되고, 안해도 되는 그냥 일반 메서드

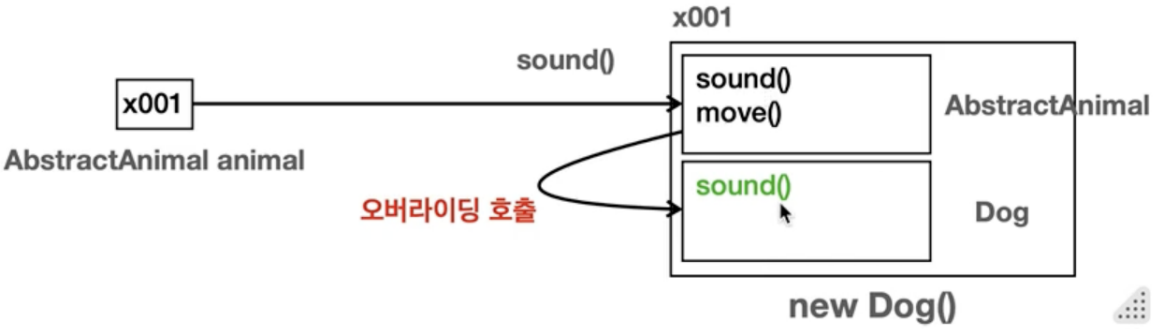
어차피 추상 클래스는 직접 인스턴스를 생성하지 못하므로 일반 메서드는 자식에서만 사용이 가능하다.

[추상 메서드]

부모 클래스를 상속받는 자식 클래스가 반드시 오버라이딩 해야 하는 메서드를 부모 클래스에 정의.
이를 추상 메서드라고 함.

```
public abstract void sound();
```

- 추상 메서드는 선언할 때 메서드 앞에 추상이라는 의미의 abstract 키워드 사용
- 추상 메서드가 하나라도 있는 클래스는 추상 클래스로 선언해야 함.
  - 그렇지 않으면 컴파일 오류 발생
  - 추상 메서드는 바디가 없으므로 작동하지 않는 메서드를 가진 불완전한 클래스, 때문에, 직접 생성하지 못하도록 추상 클래스로 선언해야함.
- 추상 메서드는 상속 받는 자식 클래스가 반드시 오버라이딩 해서 사용 해야함.
  - 그렇지 않으면 컴파일 오류 발생
  - 오버라이딩 하지 않으려면 상속받는 자식 객체도 추상 클래스여야 함.



- 추상 클래스는 일반 클래스와 메모리 구조, 생성 등 모두 동일

[순수 추상 클래스]

모든 메서드가 추상 메서드인 추상 클래스

메서드 선언만 있기에, 메서드 바디가 전혀 존재하지 않음.
다형성을 위해, 껍데기만 있는 역할

- 인스턴스를 생성할 수 없다.
- 상속을 받는 자식은 부모의 모든 메서드를 오버라이딩 해야함.

- 주로 다형성을 위해 사용.

자식 객체는 부모가 선언해 놓은 메서드를 모두 구현해야만 함.
이는 해당 추상 클래스를 상속 받는 자식 객체에게 '제약'을 건 것.

Java에서는 이런 규약을 interface라는 형태로 제공

[interface]

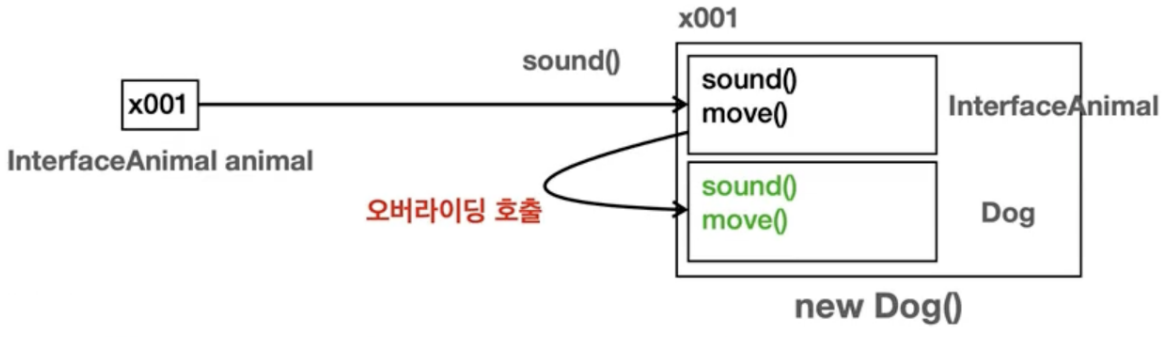
인터페이스는 순수 추상 클래스와 같음. 상속받는 자식은 모든 메서드를 구현해야 함.

- 인터페이스의 메서드는 모두 public, abstract
- public abstract 생략 가능, 생략 권장
- 만약 자식이 구현하지 않으면, default로 메서드를 사용할 수 있음.
  - default 키워드가 필요.
  - default가 없는 함수는 무조건 구현해야 함.
- prviate으로 메서드 선언시, 바디가 있어야함.
  - protected는 선언 불가
  - default 접근 제어자는 public이므로 default는 불가능
- 인터페이스는 다중 구현(다중 상속)지원.
  - 인터페이스는 상속이 아닌, 구현이라고 표현

[인터페이스와 멤버 변수]

인터페이스에서 멤버 변수는 public, static, final이 모두 포함되었다고 간주.

[활용]



클래스, 추상 클래스, 인터페이스는 모두 프로그램 코드, 메모리 구조, .class파일 등 모든 것이 동일

[상속, 구현]

부모 클래스의 기능을 자식 클래스가 상속 받을 때, 클래스는 상속 받는다고 표현
부모 인터페이스의 기능을 자식이 상속 받을 때는 인터페이스를 구현한다고 표현.

상속은 부모의 기능을 물려 받는 것이 목적인데, 인터페이스는 모든 메서드가 추상 메서드로 물려받을 수 있는 기능이 없고, 오히려 인터페이스에 정의한 모든 메서드를 자식이 오버라이딩해서 구현해야함.

인터페이스는 이름만 있는 설계도, 이 설계도가 실제 어떻게 작동하는지는 하위 클래스에서 구현해야함.

- 표현하는 단어만 다를 뿐, 자바는 일반 상속 구조와 동일하게 작동

[인터페이스를 사용하는 이유]

Q. 전부 Abstract를 사용하면 되는데 왜 인터페이스를 쓸까?

1. 제약
   인터페이스는 규격에 맞추어 오버라이딩하고 개발을 진행해야하는데,
   만약 순수 추상 클래스를 사용하다보면, 후에 미래에 누군가가 실행 가능한 메서드를 끼워넣을 수 있음.
   그럼 설계도로의 의미가 없는것, 이런 문제를 인터페이스는 원천 차단 가능(default는,,, 후에 설명)
2. 인터페이스는 다중 구현이 가능.
   자바에서 상속은 하나만 가능하지만, 인터페이스는 다중으로 구현이 가능함.

   다중 상속은 기능을 상속받는 자식이 다중의 부모로부터 동일한 이름의 메서드를 상속받은 후 오버라이딩 하지 않으면, 부모의 메서드를 사용할 때 어느 부모의 메서드를 사용하는지가 애매해짐.
   (근데 C++은 다중 상속 허용함)

   인터페이스를 구현한 객체는 어느 인터페이스로 부터 메서드를 가져와도, 본인이 오버라이딩해서 사용하므로 문제가 되지 않음. - 만약 이름이 같은 메서드가 있으면 구현은 하나만 하면 됨.