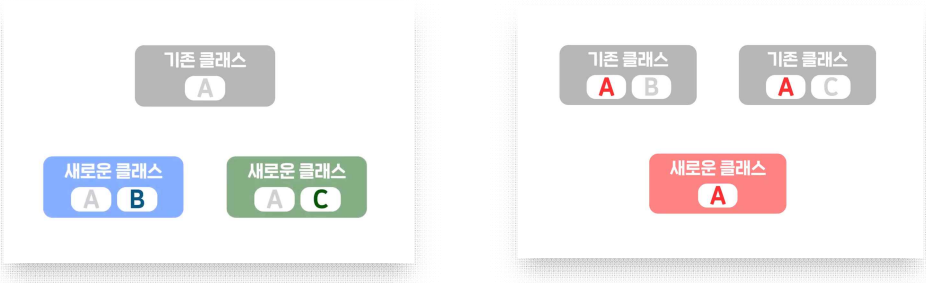
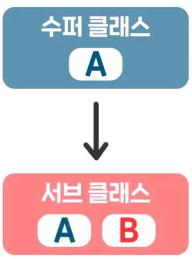
왕초보 코틀린 #10 클래스의 상속

✔ 상속 inheritance이 필요한 이유

실무자 관점에서 기존 클래스를 확장하여 새로운 클래스를 만들 때, 여러 개의 클래스를 만들었는데 클래스들의 공통점을 뽑아 코드 관리를 편하게 해야 할 때 사용한다.



✔ 상속의 구성



속성과 함수를 물려주는 쪽

속성과 함수를 물려받는 쪽

✔ 상속 적용해보기

▪ 애완동물을 관리하는 animal 클래스 만들기

```
fun main(){
}
class Animal(var name:String, var age:Int, var type:String){
    fun introduce(){
        println("저는 ${type} ${name}이고, ${age}살입니다.")    — ①
    }
}
```

① 동물의 정보를 알 수 있는 **introduce** 함수 추가하기 - 타입, 이름과 나이 출력

* 클래스 자신의 속성임이 확실할 때는 this를 사용하지 않아도 된다.

▪ animal 클래스를 상속받아 Dog 클래스로 확장하기

```
fun main(){
    var a = Animal("별이", 5, "개")
    var b = Dog("별이", 5)
```

```
}
open class Animal(var name:String, var age:Int, var type:String){
    fun introduce(){
        println("저는 ${type} ${name}이고, ${age}살입니다.")   — ①
    }
}
class Dog (name: String, age: Int) : Animal(name, age, "개")
}
```

① 현재 Animal 클래스는 'open' 상태가 아니기 때문에 상속을 받을 수 없다. 따라서 open을 Animal 클래스 앞에 붙여준다. **open(**클래스가 상속될 수 있도록 클래스 선언 시 붙여줄 수 있는 키워드**)**

* 코틀린은 상속 금지가 기본값이다.

✔ 상속에 대한 규칙

- 서브클래스는 수퍼클래스에 존재하는 속성과 '같은 이름'의 속성을 가질 수 없다.
- 서브클래스가 생성될 때에는 반드시 수퍼클래스의 생성자까지 호출되어야 한다.

② 생성자에서 이름과 나이를 받긴 하지만 클래스의 자체 속성으로 만들어주는 var을 붙이지 말고 일반 패러미터로 받아 Animal 클래스의 생성자에 직접 넘겨주도록 한다.

③ 클래스 선언 뒤에 콜론을 붙이고 수퍼클래스의 생성자를 호출할 수 있도록 해주면 된다.
생성자의 패러미터로 받은 이름과 나이, 그리고 "개"라는 종류는 Dog 클래스라면 공통된 값이므로 고정된 값으로 넘기도록 한다.

④ 이렇게 하면 Animal 클래스에 이름과 나이, 그리고 "개"라는 종류를 넣어 생성한 a 인스턴스와 Dog 클래스에 이름과 나이만 넣어 생성한 b 인스턴스가 같은 속성과 함수의 기능을 갖게 된다.
► 같은 결과가 나온다

▪ Dog 만의 함수 추가하기

```
fun main(){
    var a = Animal("별이", 5, "개")
    var b = Dog("별이", 5)
    b.bark()
}
open class Animal(var name:String, var age:Int, var type:String){
    fun introduce(){
        println("저는 ${type} ${name}이고, ${age}살입니다.")   — ①
    }
}
class Dog (name: String, age: Int) : Animal(name, age, "개")
{
    fun bark(){
```

```
        println("멍멍")
    }
}
```

① Dog 안에 짖는 함수인 bark를 추가해서 멍멍 짖는 내용을 출력하기
② 이 함수는 Dog 클래스에서만 사용할 수 있기 때문에 b 인스턴스에 bark를 시켜보면 강아지가 멍멍 짖는 것을 확인할 수 있다.

▪ Cat 서브클래스 만들기

```
fun main(){
    var a = Animal("별이", 5, "개")
    var b = Dog("별이", 5)
    b.bark()

    var c = Cat("냐옹이", 4)
    c.introduce()
    c.meow()                          —③
}
open class Animal(var name:String, var age:Int, var type:String){
    fun introduce(){
        println("저는 ${type} ${name}이고, ${age}살입니다.")
    }
}
class Dog (name: String, age: Int) : Animal(name, age, "개")
{
    fun bark(){
        println("멍멍")
    }
Class Cat (name: String, age: Int) : Animal(name, age, "고양이")
{
    fun meow(){
        println("야옹야옹")
    }
}
```

① Dog랑 똑같이 이름과 나이를 받고 이름, 나이, "고양이"라는 값을 수퍼클래스에 넘겨준다.
② 그리고 meow라는 함수를 넣어 야옹을 시킨다.
③ Cat의 인스턴스를 만들고 자기소개와 meow를 시켜본다.

상속은 클래스를 더 구조적으로 만들어준다는 장점이 있지만 지나친 상속구조는 코드를 더 어렵게 만든다는 점도 기억하자.