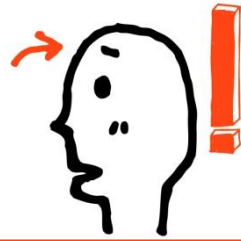




# Android Java

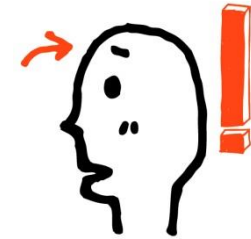
15장

클래스의 상속 3: 상속의 목적



# 클래스의 상속 3: 상속의 목적

- 01** 개인정보 관리 프로그램
- 02** 모든 클래스가 상속하는 Object 클래스
- 03** final 클래스와 final 메소드



# 다음 클래스 정의에서 상속의 이유를 찾아보자.

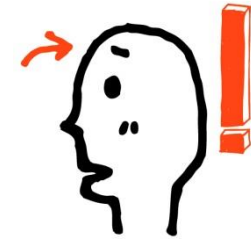


"예제는 소스코드를 직접 참조합니다!"

```
class HighFriend extends Friend
class UnivFriend extends Friend
```

- Friend 클래스는 인스턴스화 되지 않는다.
- 다만 HighFriend 클래스와 UnivFriend 클래스의 상위 클래스로만 의미를 지닌다.
- Friend 클래스의 showBasicInfo 메소드를 하위 클래스에서 각각 오버라이딩 하고 있다.
- Friend 클래스의 showBasicInfo 메소드는 비어있다!

그렇다면! 굳이 Friend 클래스를 정의하면서까지 HighFriend 클래스와 UnivFriend 클래스를 상속의 관계에 두는 이유는 무엇인가! 이 문제에 대한 답은 FriendInfoHandler 클래스에서 찾을 수 있다.



# 다음 클래스 정의에서 상속의 이유를 찾아보자.

```
class FriendInfoHandler
{
    private Friend[] myFriends;
    private int numOfFriends;

    public FriendInfoHandler(int num)
    {
        myFriends=new Friend[num];
        numOfFriends=0;
    }

    private void addFriendInfo(Friend fren)
    {
        myFriends[numOfFriends++]=fren;
    }
    . . . . .
    public void showAllSimpleData()
    {
        for(int i=0; i<numOfFriends; i++)
        {
            myFriends[i].showBasicInfo();
            System.out.println("");
        }
    }
}
```

Friend 클래스를 상속했기 때문에 Friend의 하위 클래스의 인스턴스가 저장 가능하다!

상속이 주는 매우 큰 이점!

- FriendInfoHandler 클래스는 Friend의 하위 클래스의 인스턴스를 저장 및 관리한다.
- FriendInfoHandler 클래스 입장에서는 HighFriend 클래스의 인스턴스도, UnivFriend 클래스의 인스턴스도 모두 Friend 클래스의 인스턴스로 간주한다.

showBasicInfo 메소드를 오버라이딩 했기 때문에 Friend 클래스의 참조변수를 통해서도 하위 클래스의 showBasicInfo 메소드를 호출할 수 있다! 이것이 바로 showBasicInfo 메소드를 오버라이딩의 관계에 든 이유이다!



# FriendInfoHandler 클래스의 관찰

showBasicInfo 메소드를 오버라이딩 관계에 두지 않는다면 FriendInfoHandler 클래스의 showAllSimpleData 메소드는 다음과 같이 변경되어야 한다.

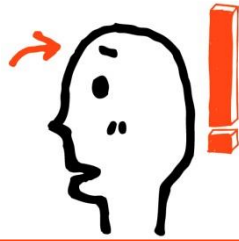
```
void showAllSimpleData( )
{
    for(int i=0; i<numOfFriends; i++)
    {
        if(myFriends[i] instanceof HighFriend)
            ((HighFriend)myFriends[i]).showBasicInfo();
        else
            ((UnivFriend)myFriends[i]).showBasicInfo();

        System.out.println("");
    }
}
```

변경된  
부분

그러나 이것이 전부가 아니다. Friend 클래스를 상속하는 하위 클래스가 하나 더 등장할 때마다 위의 메소드는 엄청나게 복잡해진다.

특히 UnivFriend 클래스와 HighFriend 클래스를 상속의 관계로 묶지 않았다면, FriendInfoHandler 클래스는 지금보다 훨씬 더 복잡해져야 하며, Friend 클래스를 상속하는 하위 클래스의 수가 증가할 때마다 엄청난 코드의 확장이 필요해진다.



# 상속과 오버라이딩이 가져다 주는 이점

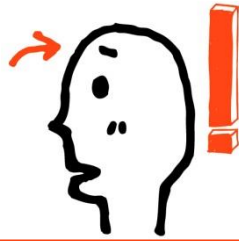
"상속을 통해 연관된 일련의 클래스에 대한 공통적인 규약을 정의할 수 있습니다."



위의 문장은 Friend 관련 예제를 통해서 다음과 같이 이해되어야 한다!



FriendInfoHandler 클래스는, 상속을 통해 연관된 HighFriend, UnivFriend 클래스에 대해(일련의 클래스에 대해) 동일한 방식으로 배열에 저장 및 메소드 호출을 할 수(공통적인 규약을 정의할 수) 있습니다.



# 모든 클래스는 Object 클래스를 상속한다.

```
class MyClass { . . . }
```

```
class MyClass extends Object { . . . }
```

```
Object obj1=new MyClass();
```

```
Object obj2=new int[5];
```

배열도 인스턴스이므로 작성 가능

자바 클래스가 아무것도 상속하지 않으면 java.lang 패키지의 Object 클래스를 자동으로 상속한다. 때문에 모든 자바 클래스는 Object 클래스를 직접적 혹은 간접적으로 상속한다.

모든 클래스가 Object 클래스를 직접 혹은 간접적으로 상속하므로, 다음 두 가지가 가능하다.

- ✓ 자바의 모든 인스턴스는 Object 클래스의 참조변수로 참조 가능
- ✓ 자바의 모든 인스턴스를 대상으로 Object 클래스에 정의된 메소드 호출 가능



# String 클래스와 Object 클래스

모든 클래스가 Object 클래스를 상속하는 것과 관련해서 기억할 것

- ✓ Object 클래스에는 toString 메소드가 다음의 형태로 정의되어 있다.  
`public String toString( ) { ... }`
- ✓ 그리고 우리가 흔히 호출하는 println 메소드는 다음과 같이 정의되어 있다.  
`public void println(Object x) { ... }`
- ✓ 때문에 모든 인스턴스는 println 메소드의 인자로 전달될 수 있다.
- ✓ 인자로 전달되면, toString 메소드가 호출되고, 이 때 반환되는 문자열이 출력된다.
- ✓ 때문에 toString 메소드는 적절한 문자열 정보를 반환하도록 오버라이딩 하는 것이 좋다!

```
class Friend
{
    String myName;
    public Friend(String name)
    {
        myName=name;
    }
    public String toString()
    {
        return "제 이름은 "+myName+"입니다.";
    }
}
```

예제 StringToString.java 다시 보기

```
public static void main(String[] args)
{
    Friend fnd1=new Friend("이종수");
    Friend fnd2=new Friend("현주은");

    System.out.println(fnd1);
    System.out.println(fnd2);
}
```

실행 결과

제 이름은 이종수입니다.  
제 이름은 현주은입니다.





# 클래스의 final 선언과 final 메소드

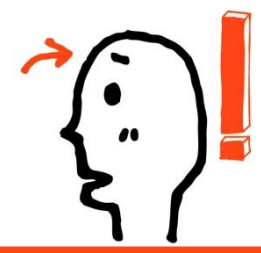
```
final class MyClass
{
    . . . .
}
```

클래스 `MyClass`가 상속되는 것을 허용하지 않는다!

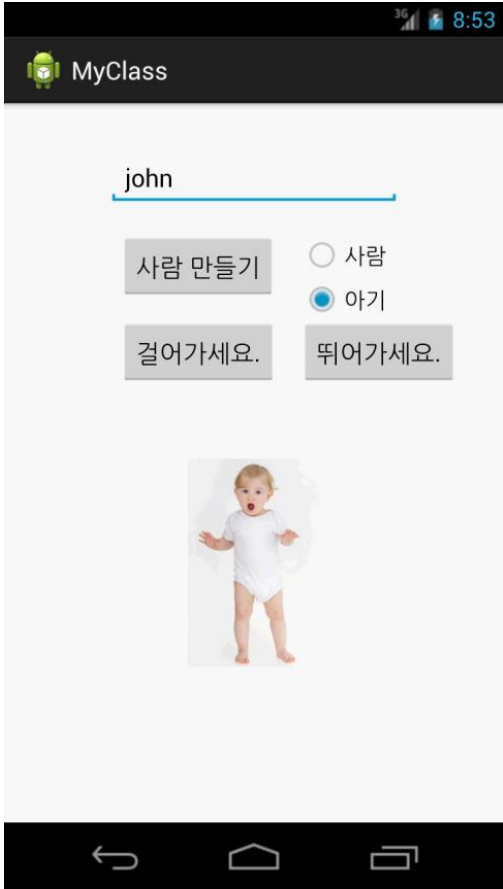
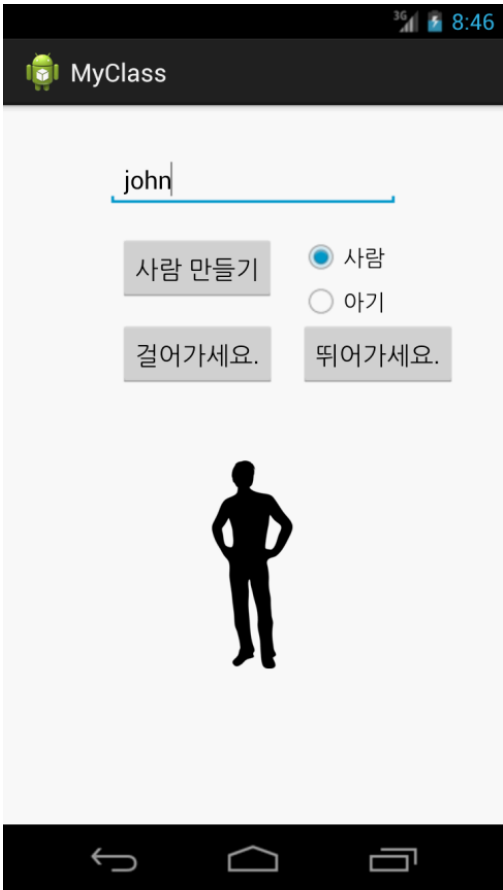
```
class YourClass
{
    final void yourFunc(int n) { . . . }
    . . . .
}
```

클래스 `YourClass`가 상속되는 것을 허용은 하되, 메소드 `yourFunc`의 오버라이딩은 허용하지 않는다!

대표적인 final 메소드로는 Object 클래스의 `wait`, `notify`, `notifyall` 메소드 등이 있으며, 이들은 실제로 오버라이딩이 바람직하지 않은 메소드들이다.



# 클래스에 따라 다른 이미지 보여주기





# 클래스에 따라 다른 이미지 보여주기

MyClass2

alignLeft: button2  
centerVertical: true

사람 만들기

☒ 사람  
☐ 아기

걸어가세요.

뛰어가세요.

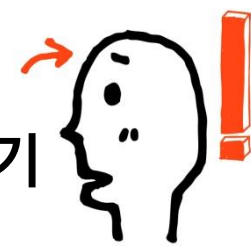
울어요.

RelativeLayout

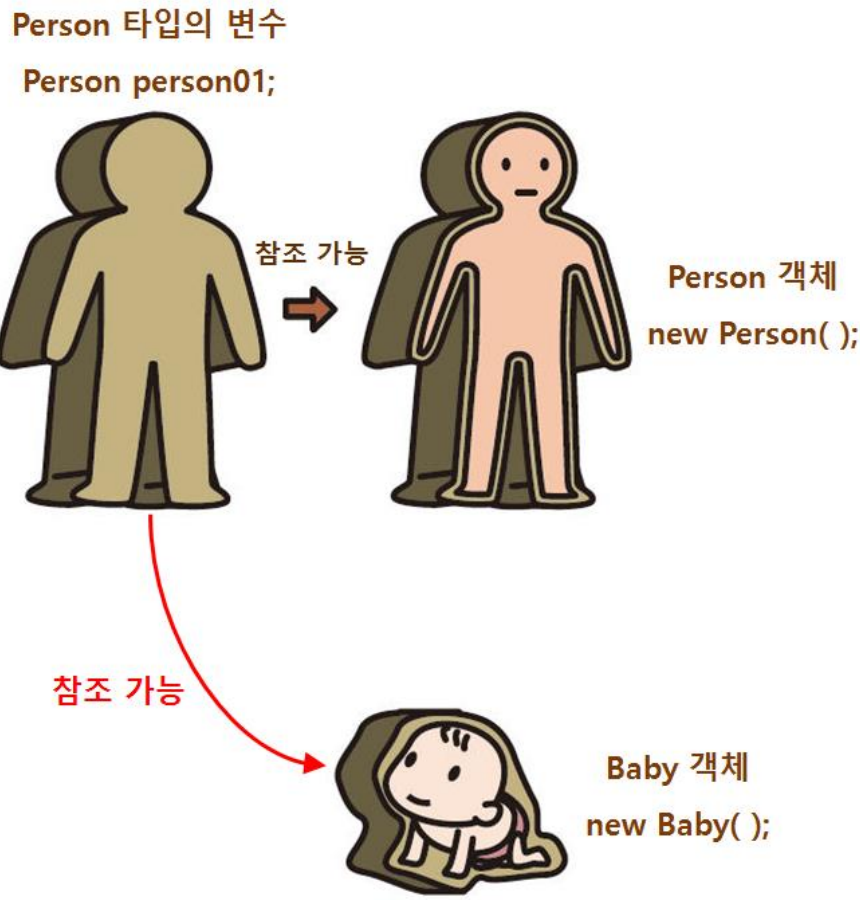
- editText1
- button2 - "걸어가세요."
- button3 - "뛰어가세요."
- imageView1 - person
- button1 - "사람 만들기"
- radioGroup1
  - radio0 (RadioButton) - "사람"
  - radio1 (RadioButton) - "아기"
- button4 - "울어요."

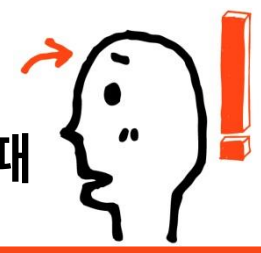
Properties

<b>Id</b>	@+id/button4
<b>Layout Para...</b>	[]
<b>Style</b>	android:buttonStyle
<b>Text</b>	울어요.
<b>Hint</b>	
<b>Content Desc...</b>	
<b>TextView</b>	[]
Text	울어요.
Hint	
Text Color	@android:color/prima...
Text Color H...	@android:color/hint...
Text Appear...	?android:attr/textApp...
Text Size	
Typeface	

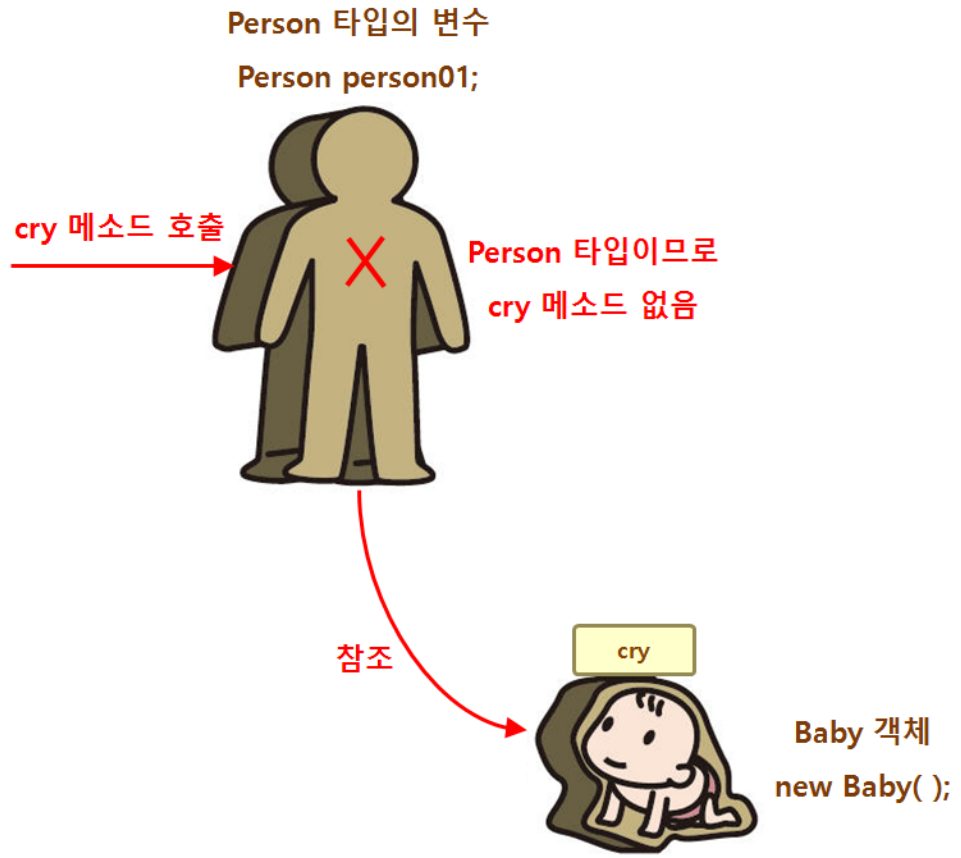


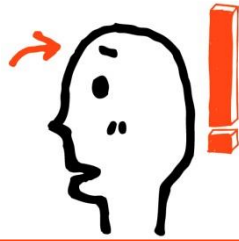
# 자식 객체를 부모 클래스의 타입으로 만든 변수 상자로 가리키기





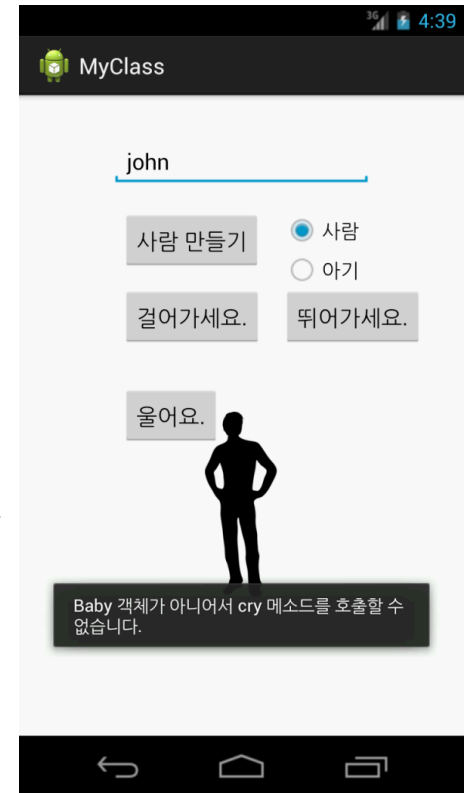
부모 클래스의 타입으로 만들어진 변수에 메소드가 들어있지 않을 때

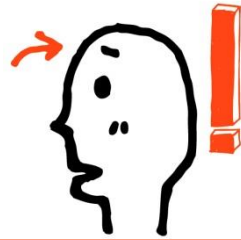




# 클래스에 따라 다른 이미지 보여주기

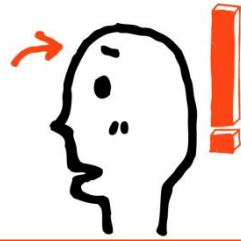
```
public void onButton4Clicked(View v) {  
    if (person01 != null) {  
        if (person01 instanceof Baby) {  
            Baby baby01 = (Baby) person01;  
            baby01.cry();  
        } else {  
            Toast.makeText(getApplicationContext(),  
                "Baby 객체가 아니어서 cry 메소드를 호출할 수 없습니다.",  
                Toast.LENGTH_LONG).show();  
        }  
    }  
}
```





# Self-Check

1. Singer 클래스를 만들고 Singer 클래스를 상속한 GirlGroupSinger 클래스도 만듭니다
2. MainActivity.java 파일을 열고 Singer 자료형으로 된 두 개의 변수를 선언합니다. 첫 번째 입력상자에 'IU' 라는 글자를 입력하고 두 번째 입력 상자에 '29'라는 숫자를 입력한 후 [가수 추가] 버튼을 누르면 Singer 자료형으로 된 인스턴스를 생성한 후 첫 번째 변수에 대입합니다. 그 다음 토스트로 "가수 추가됨 : IU, 가수의 나이 : 29"과 같은 메시지를 보여주도록 코드를 입력합니다
3. 첫 번째 입력상자에 'Twice'라는 글자를 입력하고 두 번째 입력상자에 '24'라는 글자를 입력한 후 [걸그룹 추가] 버튼을 누르면, GirlGroupSinger 자료형으로 된 인스턴수 객체를 생성한 후 두 번째 변수에 대입합니다. 그 다음 토스트로 "걸그룹 추가됨 : Twice, 걸그룹 나이 : 24"와 같은 메시지를 보여주도록 코드를 입력합니다.



# Self-Check

4. Singer.java 파일을 열고 Singer 클래스에 total이라는 클래스 변수를 추가한 후, 이 속성의 값을 가져오거나 설정하기 위한 Getter와 Setter 메서드를 만듭니다. Total 변수에는 새로 만들어진 모든 인스턴스의 합계를 저장할 것입니다.
5. Singer 클래스의 생성자 안에 인스턴스가 만들어질 때 total 변수의 값이 1 증가하도록 코드를 입력합니다.
6. MainActivity.java 파일을 열고 [가수 추가] 버튼 또는 [걸그룹 추가] 버튼을 누를 때마다 지금까지 만든 가수나 걸그룹의 합을 화면의 텍스트에 보여주도록 합니다.
- 예) 추가된 가수의 총 수 : 5명





# THANK YOU

---

실무에서 알아야 할 기술은 따로 있다! 자바를 다루는 기술