

자바 기초

May. 7, 2007

객체와 클래스 II

학습 목표

■ 학습 목표

- 상속의 개념을 배우고, 구조를 이해한다.
- 다른 객체를 가져다 사용하는 경우를 이해
- 오버라이딩의 개념과 사용법을 알아본다.
- this와 super 객체를 이해하고 사용법을 알아본다.

상속이란?

- Inheritance
- 하나의 클래스에서 이를 확장하여 새로운 클래스를 만들어 가는 과정
- 많은 기능과 특성을 물려받으면 프로그래밍을 쉽게 할 수 있음.
- 상속을 사용하는 방법
 - `class 만들_클래스_이름 extends 부모클래스`

상속을 사용하지 않는 예제

- 상속을 사용하지 않는 예제는 상속을 사용하지 않으므로, 클래스 변수를 생성해야 하고, 변수이름.메서드명 과 같이 실행해야 메서드가 실행되지만
- 상속을 사용하는 예제는 마치 자신의 메서드를 사용하는 것처럼 메서드 이름만 호출하면 됩니다.

부모 클래스

- public class ShapeArea {
- //사각형의 넓이 구하기
- public double calRectArea(double w, double h){
- double area = w*h;
- return area;
- }
- //원의 넓이 구하기
- public double calCircleArea(double r){
- double area = Math.PI*r*r;
- return area;
- }
- }

상속을 사용하지 않은 예제

- public class SubClassShape1 {
 - public static void main(String args[]){
 - ShapeArea sa = new ShapeArea();
 - double rect_area = sa.calRectArea(4,5);
 - double circle_area=sa.calCircleArea(7);
 - System.out.println("가로가 4, 세로가 5인 사각형의 넓이는 "+rect_area+" 입니다.");
 - System.out.println("반지름이 7인 원의 넓이는 "+circle_area+" 입니다.");
 - }
- }

```
General Output
-----Configuration: <Default>-----
가로가 4, 세로가 5인 사각형의 넓이는 20.0 입니다.
반지름이 7인 원의 넓이는 153.93804002589985 입니다.

Process completed.
```

상속을 사용한 예제

- `public class SubClassShape2 extends ShapeArea{`
 - `public static void main(String args[]){`
 - `double rect_area = calRectArea(4.0,5.0);`
 - `double circle_area = calCircleArea(7.0);`
 - `System.out.println("가로가 4, 세로가 5인 사각형의 넓이
는 "+rect_area+" 입니다.");`
 - `System.out.println("반지름이 7인 원의 넓이는 "+circle_area+"
입니다.");`
 - `}`
 - `}`

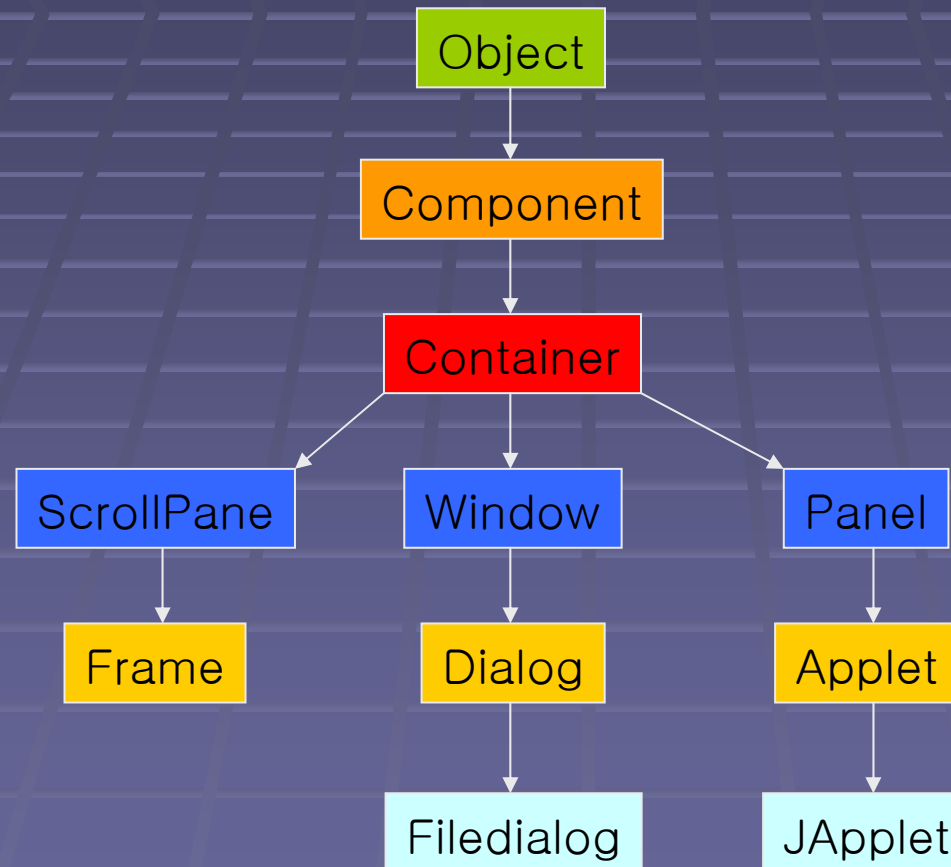
```
General Output
-----Configuration: <Default>-----
가로가 4, 세로가 5인 사각형의 넓이는 20.0 입니다.
반지름이 7인 원의 넓이는 153.93804002589985 입니다.

Process completed.
```

자바 클래스의 상속 관계

- C++과 자바 클래스의 상속 관계 차이
 - 자바는 여러 개의 클래스로부터 상속 받는 것을 허용하지 않는다.
 - C++은 여러 개의 클래스로부터 상속 받는 것을 허용한다.
- 상속 관계를 잘 활용하면 부모 클래스가 많은 만큼 자식 클래스에서 구현해야 하는 기능도 많이 줄어들어 프로그램 개발 작업이 한결 수월함.

자바 클래스의 상속 관계

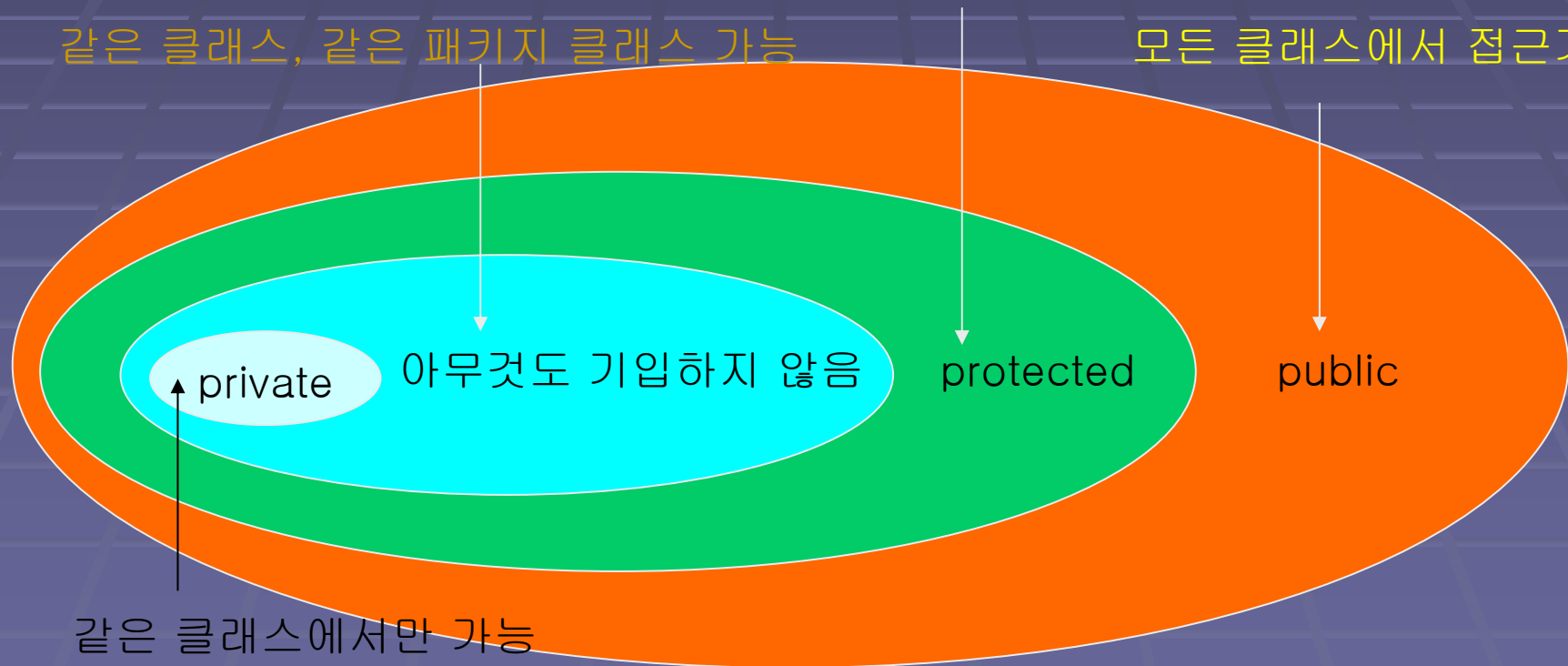


접근 제어

- 한정자(Modifier)와 은닉(Encapsulation)등을 사용함을 말한다.

같은 클래스, 같은 패키지 클래스 가능

모든 클래스에서 접근가능



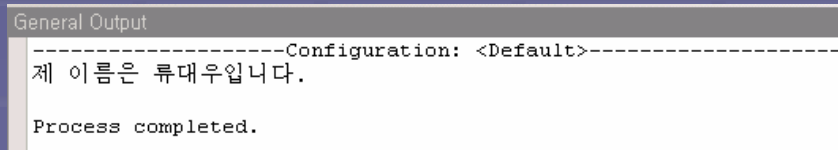
한정자 사용의 예

- AccessTest.java 파일

- public class AccessTest{
 - **public** void getName(){
 - System.out.println("제 이름은 류대우입니다.");
 - }
- }

- AccessTestMain.java 파일

- public class AccessTestMain{
 - public static void main(String args[]){
 - AccessTest at = new AccessTest();
 - at.getName();
 - }
- }



General Output

-----Configuration: <Default>-----

제 이름은 류대우입니다.

Process completed.

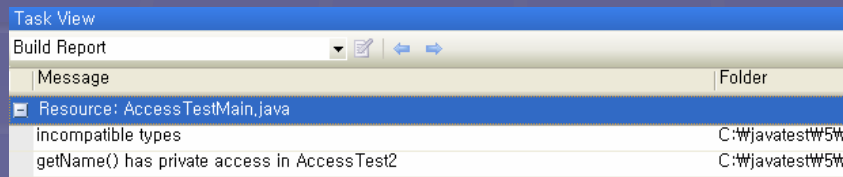
한정자 사용의 예

- AccessTest2.java 파일

- public class AccessTest2{
 - **private** void getName(){
 - System.out.println("제 이름은 류대우입니다.");
 - }
- }

- AccessTestMain.java 파일

- public class AccessTestMain{
 - public static void main(String args[]){
 - AccessTest at = new AccessTest();
 - at.getName();
 - }
- }



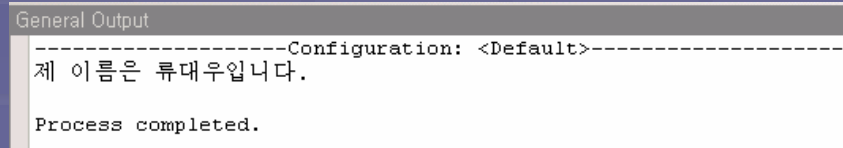
상속 시 한정자 사용의 예

- AccessTest2.java 파일

- `public class AccessTest2{`
 - `public void getName(){`
 - `System.out.println("제 이름은 류대우입니다.");`
 - `}`
- `}`

- AccessTestMain.java 파일

- `public class AccessTestMain extends AccessTest2{`
 - `public static void main(String args[]){`
 - `getName();`
 - `}`
- `}`



```
General Output
-----Configuration: <Default>-----
제 이름은 류대우입니다.
Process completed.
```

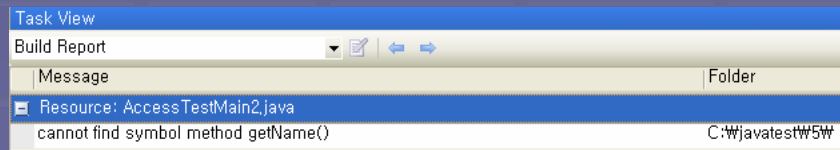
상속 시 한정자 사용의 예

- AccessTest3.java 파일

- public class AccessTest3{
 - **private** void getName(){
 - System.out.println("제 이름은 류대우입니다.");
 - }
- }

- AccessTestMain.java 파일

- public class AccessTestMain extends AccessTest2{
 - public static void main(String args[]){
 - getName();
 - }
- }



오버라이딩

- Overriding
- 자식 클래스가 부모 클래스에 있는 메서드를 재정의하는 것을 뜻합니다.
- ```
public class OverridingEX extends ShapeArea{
 ■ public static double calCircleArea(double r){
 ■ double area = 3.14*r*r;
 ■ return area;
 }
 ■ public static void main(String args[]){
 ■ double rect_area = calRectArea(4,5);
 ■ double circle_area= calCircleArea(7);
 ■ System.out.println("가로가 4, 세로가 5인 사각형의 넓이는 "+rect_area+"
 입니다.");
 ■ System.out.println("반지름이 7인 원의 넓이는 "+circle_area+" 입니다.");
 }
}
```

```
General Output
-----Configuration: <Default>-----
가로가 4, 세로가 5인 사각형의 넓이는 20.0 입니다.
반지름이 7인 원의 넓이는 153.86 입니다.

Process completed.
```

# 오버라이딩 관련 호출 메서드 구별

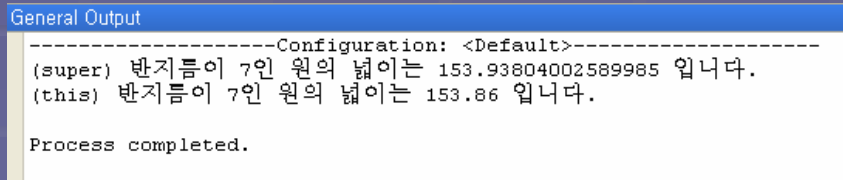
- 오버라이딩을 하지 않은 경우
  - 부모 클래스의 메서드를 호출
- 오버라이딩을 한 경우
  - 자신의 메서드를 호출

# this와 super

- 둘 다 static 영역에서 사용할 수 없다. 예) main
- this
  - 자신(자식 클래스)의 메서드나 멤버 변수를 호출할 때 사용한다.
  - 사용법
    - this.메서드\_이름();
- super
  - 부모 클래스의 메서드나 멤버 변수를 호출 할 때 사용한다.
  - 사용법
    - super.메서드\_이름();

# this와 super 사용 예제

- ```
public class SuperThisEX extends ShapeArea {  
    ■ public static double calCircleArea(double r){  
        ■ double area = 3.14*r*r;  
        ■ return area;  
    }  
    ■ public void printArea(){  
        ■ double super_circle_area = super.calCircleArea(7);  
        ■ double this_circle_area = this.calCircleArea(7);  
        ■ System.out.println("(super) 반지름이 7인 원의 넓이는 "+super_circle_area+" 입니  
        다.");  
        ■ System.out.println("(this) 반지름이 7인 원의 넓이는 "+this_circle_area+" 입니다.");  
    }  
    ■ public static void main(String args[]){  
        ■ SuperThisEX ste = new SuperThisEX();  
        ■ ste.printArea();  
    }  
}
```



```
General Output  
-----Configuration: <Default>-----  
(super) 반지름이 7인 원의 넓이는 153.93804002589985 입니다.  
(this) 반지름이 7인 원의 넓이는 153.86 입니다.  
  
Process completed.
```

this와 super로 생성자 호출하기

- this나 super를 이용하면 상위 클래스나 자기 클래스의 생성자를 호출할 수 있다.
- 교재의 예제 참조
 - this를 사용할 경우, 사용된 매개변수 개수에 맞게 오버로딩 된 자기 클래스의 생성자를 찾아서 실행하게 된다.
- 반드시 생성자 내부의 첫번째 줄에 this나 super를 사용해야 한다.