

```
for object to mirror_mod.mirror_object
operation == "MIRROR_X":
mirror_mod.use_x = True
mirror_mod.use_y = False
mirror_mod.use_z = False
operation == "MIRROR_Y":
mirror_mod.use_x = False
mirror_mod.use_y = True
mirror_mod.use_z = False
operation == "MIRROR_Z":
mirror_mod.use_x = False
mirror_mod.use_y = False
mirror_mod.use_z = True
```

```
@selection at the end -add
mirror_ob.select= 1
modifier_ob.select=1
context.scene.objects.active
("Selected" + str(modifier_ob.name))
mirror_ob.select = 0
= bpy.context.selected_object
data.objects[one.name].select
```

```
print("please select exactly one object")
-- OPERATOR CLASSES --
```

```
types.Operator):
X mirror to the selected
object.mirror_mirror_x"
mirror X"
```

Java 기초

다형성

다형성

- 다형성이란

- 하나의 참조변수로 여러가지 객체를 참조할 수 있는 것
- 즉, 조상타입의 참조변수로 자손타입의 객체를 다룰 수 있는 것이 다형성.

Ex)

```
public interface TV {  
    public void turnOnOff();  
    public void switchChaner();  
}
```

상속

상속

```
public class TVProduct1 implements TV {  
    @Override  
    public void turnOnOff() {  
        System.out.println("TV 버튼 클릭");  
    }  
    @Override  
    public void switchChaner() {  
        System.out.println("기계식 채널 돌림");  
    }  
}  
  
public class TVProduct2 implements TV{  
    @Override  
    public void turnOnOff() {  
        System.out.println("리모콘 전원 입력");  
    }  
    @Override  
    public void switchChaner() {  
        System.out.println("리모콘 채널 입력");  
    }  
    public void newFunction() {  
        System.out.println("새로운 기능");  
    }  
}
```

다형성

- 다형성예제

Ex)

```
public class TVmain {  
    public static void main(String[] args) {  
        TV tv1 = new TVProduct1(); //TVProduct1의 객체  
        TV tv2 = new TVProduct2(); //TVProduct2의 객체  
        //참조변수는 같으나 출력되는 값이 다르다  
        tv1.turnOnOff();  
        tv2.turnOnOff();  
        tv1.switchChaner();  
        tv2.switchChaner();  
    }  
}
```

Result)

TV 버튼 클릭
리모콘 전원 입력
기계식 채널 돌림
리모콘 채널 입력

다형성

- 다형성의 형변환(캐스팅 연산자)

- 부모 참조객체가 자식의 객체로 선언될 때 자식이 추가로 선언한 기능은 쓸 수가 없다.
- 이럴 경우에 선언한 자식 뒤에 캐스팅 연산자를 사용하여 강제 형변환 후 자식에 선언되어 있는 메서드를 사용할 수 있다.

Ex)

```
public class TVmain {  
    public static void main(String[] args) {  
        TV tv2 = new TVProduct2(); //TVProduct2의 객체  
        //TVProduct2의 newFunction기능을 사용하려면  
        //강제 형변환을 해야한다  
        tv2.newFunction(); //에러  
        TVProduct2 newtv = (TVProduct2)tv2;  
        newtv.newFunction();  
    }  
}
```

Result)

새로운 기능

다형성

- instanceof
 - 참조변수가 참조하는 인스턴스의 실제 타입을 체크하는데 사용.
 - 이항연산자이며 피연산자는 참조형 변수와 타입. 연산결과는 true, false.
 - instanceof의 연산결과가 true이면, 해당 타입으로 형변환이 가능하다.

Ex)

```
public class TVmain {  
    public static void main(String[] args) {  
        TVProduct2 tv2 = new TVProduct2(); //TVProduct2의 객체  
        if(tv2 instanceof TV ){  
            System.out.println("이 객체는 TV의 상속에 포함된 객체입니다.");  
        }  
    }  
}
```

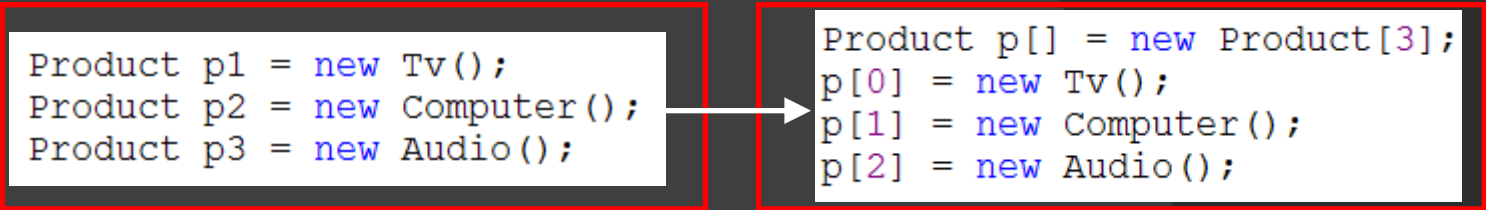
Result)

이 객체는 TV의 상속에 포함된 객체입니다.

다형성

- 다형성을 이용한 배열 선언
 - 각기 다른 인스턴스를 하나의 부모 클래스 배열에 담을 수 있다.

```
Product p1 = new Tv();  
Product p2 = new Computer();  
Product p3 = new Audio();
```



```
Product p[] = new Product[3];  
p[0] = new Tv();  
p[1] = new Computer();  
p[2] = new Audio();
```

* 배열을 이용해 참조형 객체를 담는 방법보다 컬렉션을 이용한 객체 처리가 더 상용화 되어있음.