

인터페이스(interface)

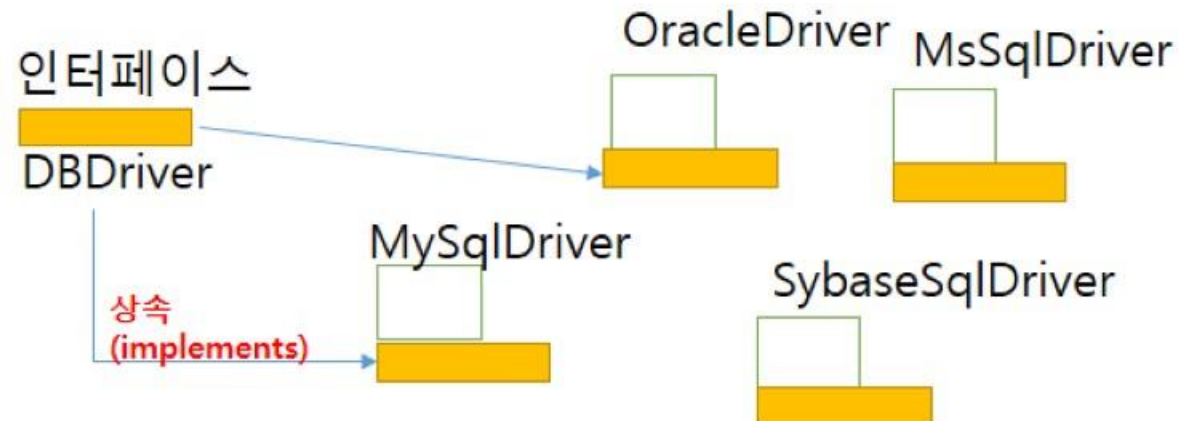
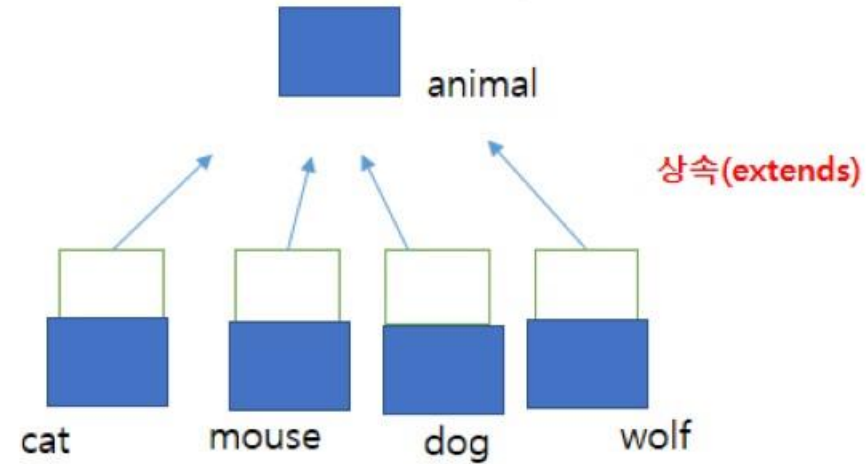
추상클래스
(Abstract)

집중화

인터페이스

약속(규격)

추상클래스(부모로만 사용됨)



추상클래스

```
class Abstract Animal{  
    void sleep(){  
        System.out.println("잔다");  
    }  
    abstract void bark();  
}
```

인터페이스

```
interface Battery{  
    void getEnergy() ;  
}
```

공통점:
객체를 생성할 수 없다.
어떻게 객체 생성하는가?
자식이 부모를 상속받으면서
객체를 생성함

공통된 코드의
집중화가
일어남

```
class Abstract Animal{  
    void sleep(){  
        System.out.println("잔다");  
    }  
    abstract void bark();  
}
```

extends
확장

cat

```
class Cat extends Animal{  
  
    @override  
    void bark(){  
        System.out.println("야옹");  
    }  
  
}
```

dog

```
class Dog extends Animal{  
  
    @override  
    void bark(){  
        System.out.println("멍멍");  
    }  
  
}
```

wolf

```
class Wolf extends Animal{  
  
    @override  
    void bark(){  
        System.out.println("아우~");  
    }  
  
}
```

작업의 명세정의
약속, 분리



```
interface Battery{  
    void getEnergy();  
}
```

implements
<구현>

LG배터리

```
class LGBattery implements Battery {  
    //LG만의 기능  
  
    @override  
    void getEnergy(){  
        System.out.println("lg 에너지얼어옴");  
    }  
}
```

삼성배터리

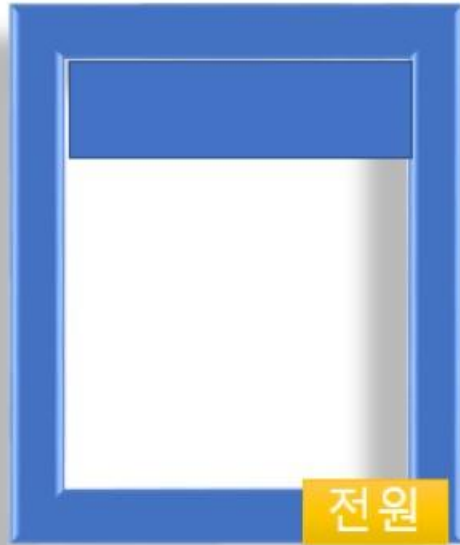
```
class SamsungBattery implements Battery {  
    //삼성만의 기능  
  
    @override  
    void getEnergy(){  
        System.out.println("삼성 에너지 얼어옴");  
    }  
}
```

인터페이스를 이용한 느슨한 결합이 가능해짐

계산기 예제

특정 계산기에 의존적이지 않는 프로그램을 작성할 수 있다.
약속에 기반한 프로그램을 작성할 수 있다.

HandPhone

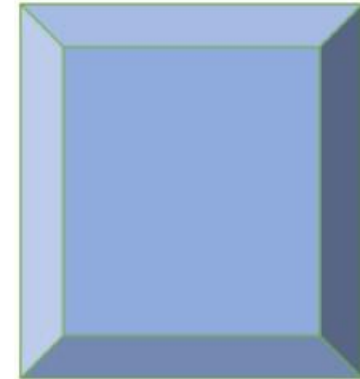
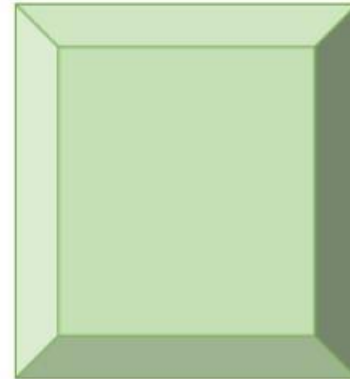


Battery

getEnergy();

SamsungBattery

LgBattery




```

public class HandPhone {

    private Battery battery;

    void setBattery(Battery battery){
        this.battery = battery;
    }

    void powerOn() {
        battery.getEnergy();
        System.out.println("핸드폰이 켜집니다.");
    }

}

```

핸드폰은 배터리를
의존한다.

(has a 관계): 멤버로 가질 때
Has a 관계라고 한다.

```

public class TestMain {

    public static void main(String[] args) {

        HandPhone cellphone = new HandPhone();

        LgBattery lg = new LgBattery();

        cellphone.setBattery( lg );
        cellphone.powerOn();

    }

}

```

배터리객체가 생성되어
Setter로 전달받음

핸드폰의 파워의 기능을 사용
할 수 있다.

Battery가 가져야 할 기능의 명세

```

public interface Battery {

    public void getEnergy();

}

```

```

public class LgBattery implements Battery {

    public LgBattery() {
        System.out.println("LG battery입니다.");
    }

    public void getEnergy{
        System.out.println("에너지 얻어옴 ");
    }

}

```

```

public class SMBattery implements Battery {

    public SMBattery() {
        System.out.println("SM battery입니다.");
    }

    public void getEnergy{
        System.out.println("에너지 얻어옴 ");
    }

}

```

익명클래스로 객체 만들기

주의! 상속으로 클래스 만들 때만
익명으로 클래스를 만들 수 있다.



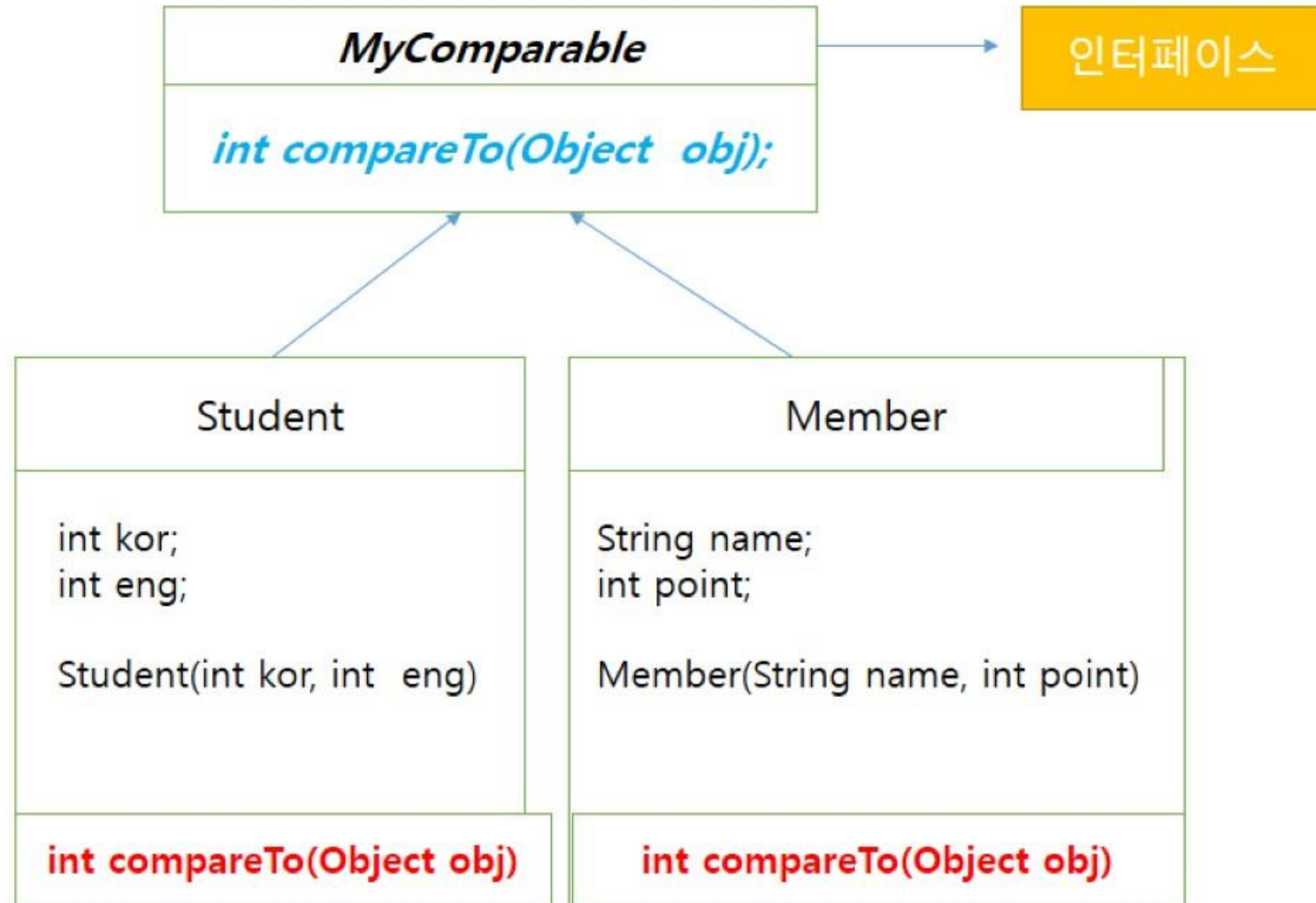
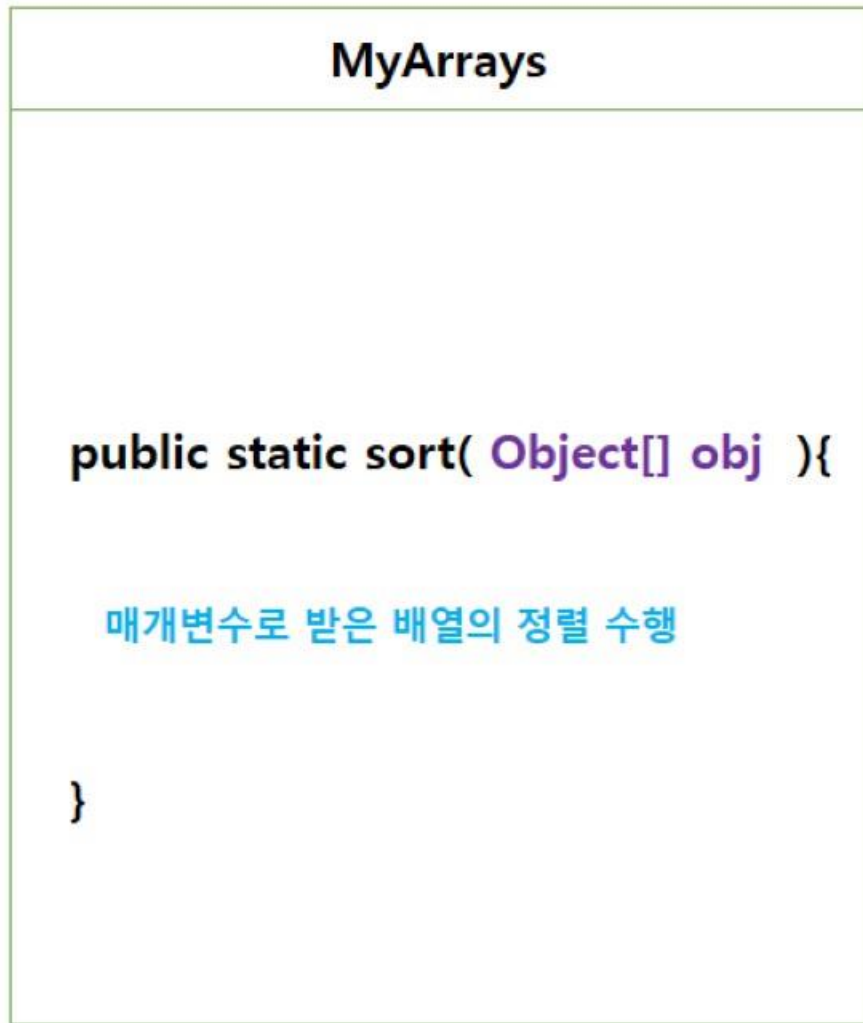
```
public class AnimalTest {  
    public static void main(String[] args) {  
        Animal c = new Cat();    //고양이 클래스 만들고 객체 생성  
        c.bark();  
  
        new Animal() {           //익명으로 클래스 만들고 객체 생성  
            @Override  
            public void bark() {  
                System.out.println("멍멍");  
            }  
        }.bark();  
    } //main  
}
```

```
public abstract class Animal {  
    public abstract void bark();  
}
```

```
class Cat extends Animal{    // 고양이 클래스 만듦  
    @Override  
    public void bark() {  
        System.out.println("야옹");  
    }  
}
```



```
interface MyComparable{  
    int compareTo(Object obj);  
}
```



```

public static void sort(Object[] arr) {

    for(int i=0 ; i< arr.length-1 ; i++ ) {
        for(int j=i+1; j< arr.length; j++ ) {

            if( arr[i] instanceof MyComparable) {
                MyComparable obj = (MyComparable) arr[i];
                if(obj.compareTo( arr[j] ) >0 ){ // 앞의 내용이 클 때 자리 바꿈 일어남(오름차순)
                    Object tmp = arr[i];
                    arr[i]= arr[j];
                    arr[j]= tmp;
                }
            }
        }
    }
}

```

```
public static void sort(Object[] arr , MyComparator c) {  
  
    for(int i=0 ; i< arr.length-1 ; i++ ) {  
        for(int j=i+1; j< arr.length; j++ ) {  
  
            if( c.compare( arr[i], arr[j] ) >0 )  
                Object tmp = arr[i];  
                arr[i]= arr[j];  
                arr[j]= tmp;  
        }  
    }  
}  
}
```

```

public class Student implements MyComparable{
    int kor;
    int eng;

    public Student(int kor, int eng) {
        this.kor = kor;
        this.eng = eng;
    }

    @Override
    public int compareTo(Object obj) {
        if(obj instanceof Student) {
            Student another = (Student)obj;
            return this.kor - another.kor ;
        }
        return 0;
    }
}

```

```

public class Member implements MyComparable{

    int point;
    String name;

    public Member(String name, int point) {
        this.name = name;
        this.point = point;
    }

    @Override
    public int compareTo(Object obj) {
        if(obj instanceof Member) {
            Member another = (Member)obj;
            return this.point - another.point ;
        }
        return 0;
    }
}

```



```
public class TestMain {
```

```
    public static void main(String[] args) {
```

```
        Student[] arr = new Student[3];  
        arr[0]= new Student(30,50);  
        arr[1]= new Student(10,90);  
        arr[2]= new Student(60,50);
```

```
        MyArrays.sort(arr);           // 각 클래스는 정렬기준 인터페이스를 구현해야 한다.  
                                       // MyArrays.sort에서는 각 객체를  
                                       // MyComparable형으로 형변환하고 compareTo매서드를 호출하는 코드가 있다.
```

```
        for(int i=0 ; i< arr.length ;i++)  
            System.out.println( arr[i].kor + " " + arr[i].eng);
```

```
        Member[] arr2 = new Member[3];  
        arr2[0]= new Member("hong", 9000);  
        arr2[1]= new Member("kim", 2000);  
        arr2[2]= new Member("park", 3000);
```

```
        MyArrays.sort(arr2);
```

```
        for(int i=0 ; i< arr2.length ;i++)  
            System.out.println( arr2[i].name + " " + arr2[i].point );
```

```
    }
```

```
}
```

Arrays.sort를 사용하려면
클래스는 반드시 약속을 지켜야 한다.
인터페이스를 구현해야한다.
Comparable 인터페이스
- 특정 매서드를 구현해야 한다