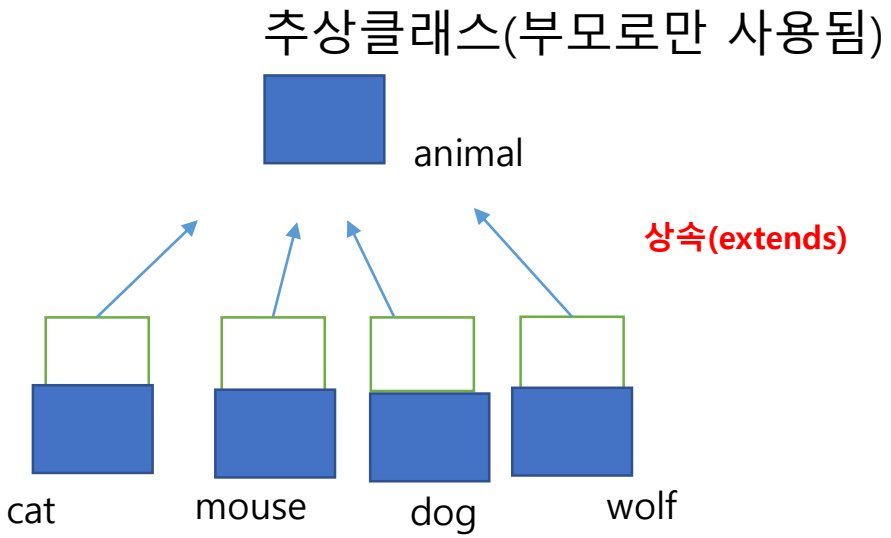
인터페이스(interface)

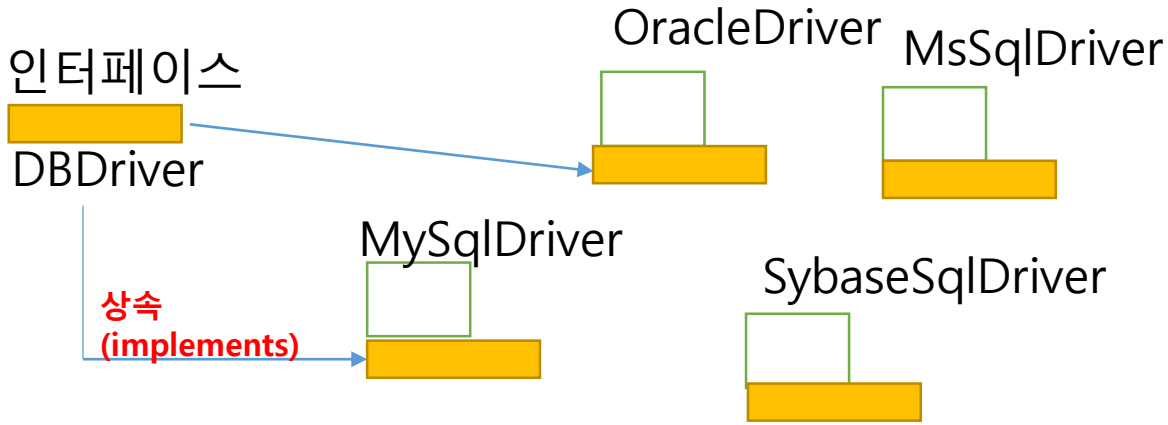
추상클래스
(Abstract)

집중화




인터페이스

약속(규격)



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>

LG바테리

```
class LGBatttery implements Battery {
    //LG만의 기능

    @override
    void getEnergy(){
      System.out.println("lg   에너지얻어옴");
    }

}
```

삼성바테리

```
class SamsungBattery   implements Battery {
     //삼성만의 기능

    @override
     void getEnergy(){
       System.out.println("삼성 에너지 얻어옴");
     }
}
```

# 인터페이스를 이용한 느슨한 결합이 가능해짐

# 계산기 예제

HandPhone


Battery

getEnergy();

SamsungBattery

LgBattery

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

```
public class LgBattery implements Battery {
    public Lg() {
    System.out.println("LG battery입니다.");
    }
    public void getEnergy{
    System.out.println("에너지 얻어옴 " );
    }
}
```

```
public class TestMain {

    public static void main(String[] args) {
        HandPhone cellphone = new HandPhone();
        LgBattery lg = new LgBattery();
        cellphone.setBattery(lg);
        cellphone.powerOn();
    }
}
```

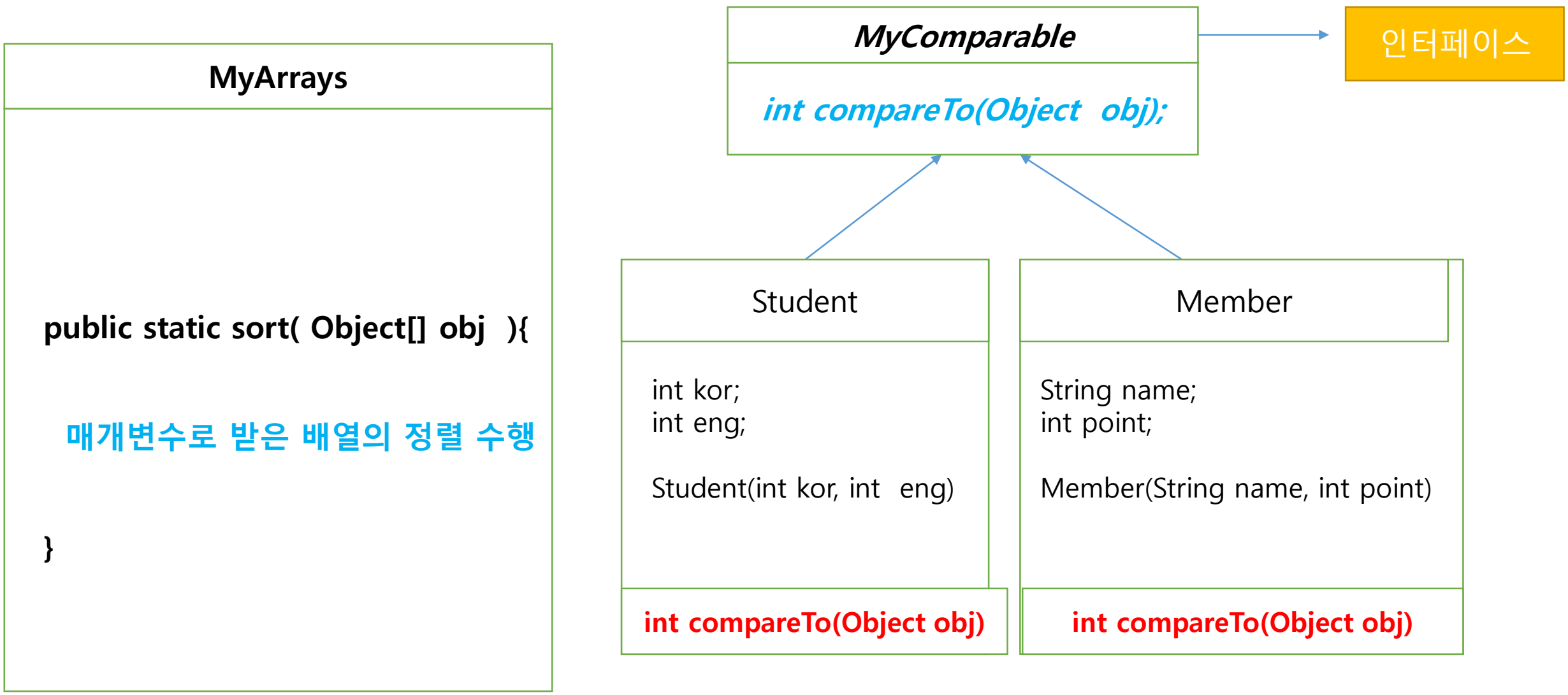
```
public class HandPhone {

    private Battery battery;

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

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

}
```

## 익명클래스로 객체 만들기

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 객체생성
    c.bark();


  new Animal() {      //익명으로 클래스 만들고 객체 생성
      @Override
      public void bark() {
          System.out.println("멍멍");
       }
    }.bark();

  }  //main

}  //클래스


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

```
public abstract class Animal {

      public abstract void bark();

}
```


MyArrays
public static sort( Object[] obj ){
매개변수로 받은 배열의 정렬 수행
}
MyComparable
int compareTo(Object obj);
인터페이스
Student
int kor;
int eng;
Student(int kor, int eng)
int compareTo(Object obj)
Member
String name;
int point;
Member(String name, int point)
int compareTo(Object obj)

```
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 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 tmp = (Student)obj;
                return (this.kor - tmp.kor)>0 ?1:-1;
       }
      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 tmp = (Member)obj;
        return (this.point - tmp.point)>0 ?1:-1;
     }
    return 0;
  }

}
```

```
package com.daesin.sort;
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);

         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 );

    }
}
```