



# 接口与实现



邹国兵

上海大学

计算机学院



# 第6章 接口与实现

1. 接口
  2. 接口回调
  3. 面向接口编程
- 

## ■■■ 6.1 接口

- 从语言的角度看，**接口**是Java语言的又一特色
- Java不支持多继承性，即一个类只能有一个父类。单继承性使得Java简单，易于管理和维护
- Java的接口更加符合人的思维方式，一个类能够实现多个接口



## ■■■ 6.1.1 接口的定义与使用

➤ 使用关键字interface定义一个接口。接口的定义和类定义很相似，分为**接口声明**和**接口体**。

### 1) 接口声明

接口通过使用关键字interface来声明

**接口声明：** interface 接口名



## ■■■ 6.1.1 接口的定义与使用

### ➤ 2) 接口体

包含常量声明和方法定义两部分：

- ① 接口体中所有的常量的访问权限一定都是public的  
(允许省略public、final和static修饰符)；
- ② 接口体中只有抽象方法，没有普通的方法。所有的抽象方法的访问权限一定都是public (允许省略public、abstract修饰符)

```
41 interface Printable {  
42     public final int MAX=100; //等价写法: int MAX=100;  
43     public abstract void add(); //等价写法: void add();  
44     public abstract float sum(float x, float y);  
45 }
```

## ■■■ 6.1.1 接口的定义与使用

### ➤ 3) 接口的使用

- Java语言中，接口由类去实现以便使用接口中的方法。  
实现接口的类，重写接口的方法。
- 类使用关键字**implements**声明自己实现一个或多个接口
- 一个类可以实现多个接口，如果实现多个接口，则用逗号隔开接口名

```
public class Apple extends Computer implements printable, addable {  
    .....  
}
```



## 6.1.1 接口的定义与使用

```
1 package shu.ces.java.chap6;
2
3 public interface Computable {
4     int MAX = 100;
5     int compute(int a, int b);
6 }
7
```

```
1 package shu.ces.java.chap6;
2
3 public class ComputerS implements Computable {
4     int number;
5
6     public int compute(int a, int b) {
7         number = a+b;
8
9         return number;
10    }
11
12    public void display() {
13        System.out.println("The result:"+number);
14    }
15 }
16
```

```
1 package shu.ces.java.chap6;
2
3 public class ComputerM implements Computable {
4     int number;
5
6     public int compute(int a, int b) {
7         number = a*b;
8
9         return number;
10    }
11
12    public void display() {
13        System.out.println("The result:"+number);
14    }
15 }
16
```

## 6.1.1 接口的定义与使用

```
1 package shu.ces.java.chap6;
2
3 public class Computer {
4     public static void main(String[] args) {
5         ComputerS cs = new ComputerS();
6         ComputerM cm = new ComputerM();
7
8         int a = 10;
9         int b = 20;
10
11        cs.compute(a, b);
12        cm.compute(a, b);
13
14        cs.display();
15        cm.display();
16    }
17 }
```

Eclipse演示



## ■■■ 6.1.1 接口的定义与使用

- public接口可以被任何类实现，友好接口可以被同一包中的类实现；
- 如果类声明实现一个接口，但没有重写接口中的所有方法，那么这个类必须是abstract类。





## 6.1.1 接口的定义与使用

```
interface Computable {  
    final int MAX=100;  
    void speak(String s);  
    int f(int x);  
    float g(float x,float y);  
}  
  
abstract class A implements Computable {  
    public int f(int x) {  
        int sum=0;  
        for(int i=1;i<=x;i++) {  
            sum=sum+i;  
        }  
        return sum;  
    }  
}
```



## ■■■ 6.1.2 理解接口

- 接口只关心操作，并不关心操作的具体实现
- 不同的类可以实现相同的接口，同一个类也可以实现多个接口
- 当一个类不希望通过继承使得自己具有某些方法时，可以考虑实现接口



## 6.1.2 理解接口

```
1 package shu.ces.java.chap6;  
2  
3 interface MoneyFare {  
4     void charge();  
5 }  
6  
7 interface ControlTemperature {  
8     void controlAirTemperature();  
9 }  
10
```

```
11 class Bus implements MoneyFare {  
12     public void charge() {  
13         System.out.println("公共汽车:一元/张,不计算公里数");  
14     }  
15 }  
16  
17 class Taxi implements MoneyFare,ControlTemperature {  
18     public void charge() {  
19         System.out.println("出租车:2元/公里,起价3公里");  
20     }  
21  
22     public void controlAirTemperature() {  
23         System.out.println("出租车安装了Hair空调");  
24     }  
25 }  
26
```

```
27 class Cinema implements MoneyFare,ControlTemperature {  
28     public void charge() {  
29         System.out.println("电影院:门票,十元/张");  
30     }  
31  
32     public void controlAirTemperature() {  
33         System.out.println("电影院安装了中央空调");  
34     }  
35 }
```

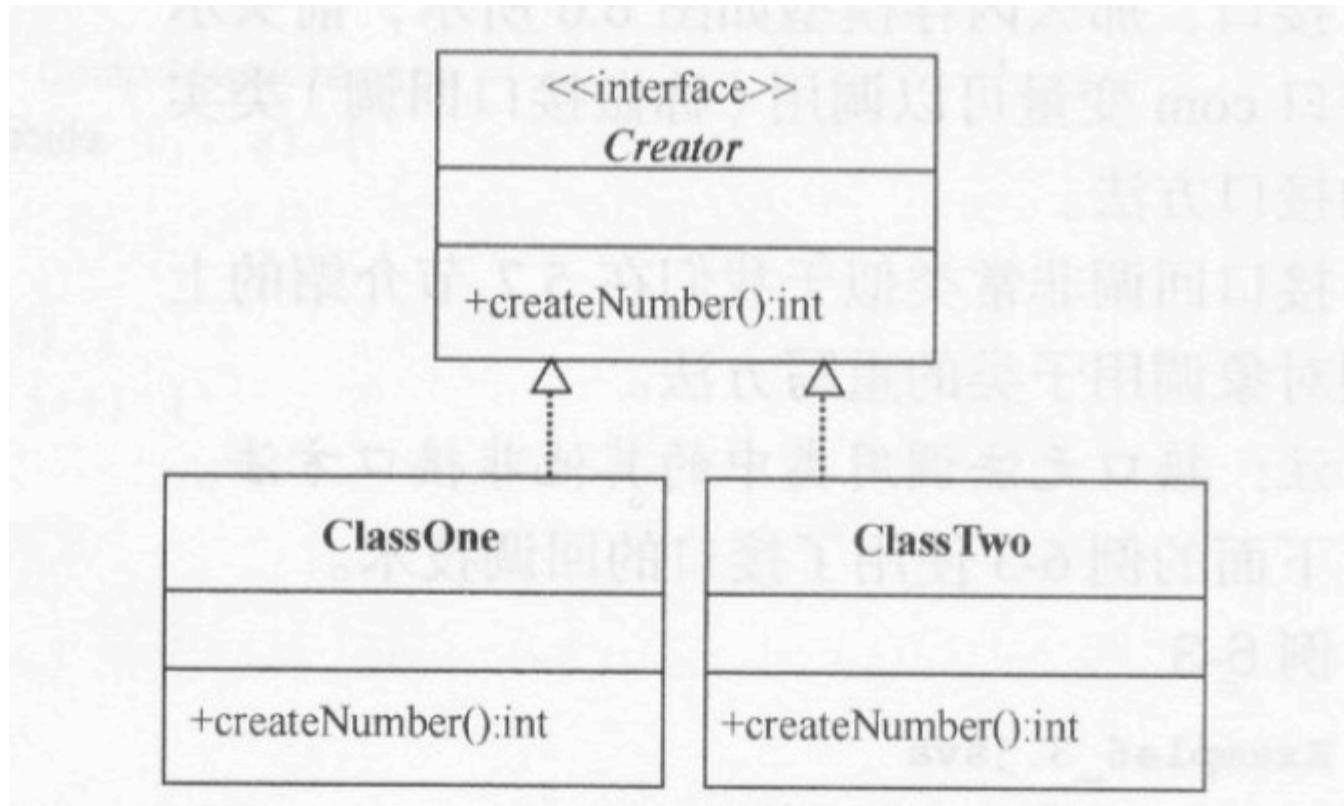
## ■■■ 6.1.2 理解接口

```
37 public class Example6_2 {  
38     public static void main(String args[]) {  
39         Bus bus101 = new Bus();  
40         Taxi buleTaxi = new Taxi();  
41         Cinema redStarCinema = new Cinema();  
42  
43         bus101.charge();  
44  
45         buleTaxi.charge();  
46         buleTaxi.controlAirTemperature();  
47  
48         redStarCinema.charge();  
49         redStarCinema.controlAirTemperature();  
50     }  
51 }
```

Eclipse演示



## 6.1.3 接口的UML图





## 第6章 接口与实现

- 1. 接口
  - 2. 接口回调
  - 3. 面向接口编程
- 

## ■■■ 6.2 接口回调

- **接口回调是指：**把实现某一接口的类创建的对象的引用，赋给该接口声明的变量。
- **当接口变量调用被类实现的接口方法时，就是通知相应的对象调用这个方法。**

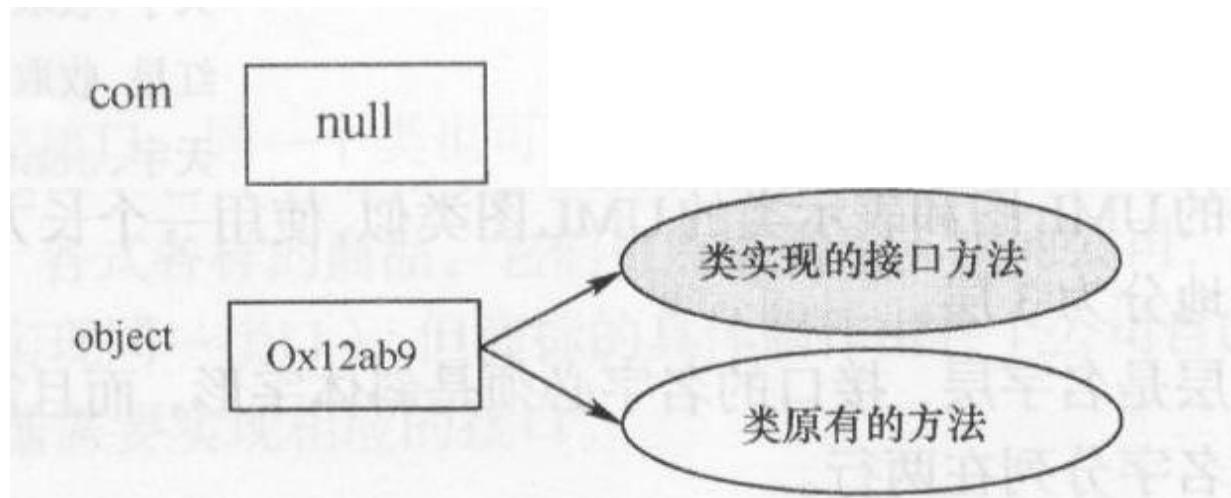


## 6.2.1 接口变量与回调机制

- 接口变量属于**引用型**变量，可存放实现该接口的类创建的对象引用。

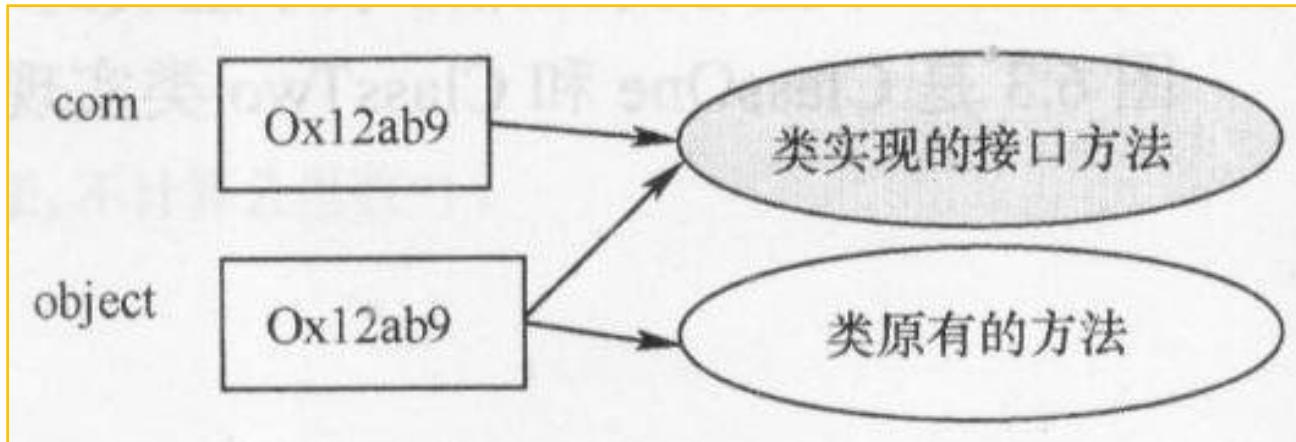
```
Com com;
```

```
ImplCom object = new ImplCom();
```



## 6.2.1 接口变量与回调机制

com=object;



- 接口回调非常类似于上转型对象调用子类的重写方法。
- 接口不能调用类中其它非接口方法。

## ■■■ 6.2.1 接口变量与回调机制举例

```
interface ShowMessage {  
    void 显示商标(String s) ;  
}  
  
class TV implements ShowMessage {  
    public void 显示商标(String s) {  
        System.out.println ("电视" +s);  
    }  
}  
  
class PC implements ShowMessage {  
    public void 显示商标 (String s) {  
        System.out.println ("电脑" +s);  
    }  
}
```

```
public class Example6_3 {  
    public static void main (String args[]) {  
        ShowMessage sm;  
        sm=new TV();  
        sm.显示商标 ("长城牌电视机");  
        sm=new PC();  
        sm.显示商标 ("ThinkPad X280笔记本");  
    }  
}
```

Eclipse演示



## ■■■ 6.2.2 接口的多态性

- 由接口产生的多态就是指：不同的类在实现同一个接口时可能具有不同的实现方式，那么接口变量在回调接口方法时具有多种形态。



## 6.2.2 接口的多态性

```
1 package shu.ces.java.chap6;
2
3 interface ComputerAverage {
4     public double average(double a,double b);
5 }
6
7 class A implements ComputerAverage {
8     public double average(double a,double b) {
9         double aver=0;
10        aver=(a+b)/2;
11        return aver;
12    }
13 }
14
15 class B implements ComputerAverage {
16     public double average(double a,double b) {
17         double aver=0;
18         aver=Math.sqrt(a*b);
19         return aver;
20     }
21 }
```

## 6.2.2 接口的多态性

```
23 public class Example6_4 {  
24     public static void main(String args[]) {  
25         ComputerAverage computer;  
26         double a=11.23, b=22.78;  
27  
28         computer = new A();  
29         double result = computer.average(a,b);  
30         System.out.printf("%5.2f和%5.2f的算术平均值:%5.2f\n",a,b,result);  
31  
32         computer = new B();  
33         result= computer.average(a,b);  
34         System.out.printf("%5.2f和%5.2f的几何平均值:%5.2f",a,b,result);  
35     }  
36 }  
37 }
```

Eclipse演示

## 6.2.2 接口的多态性

```
interface CompurerAverage {  
    public double average(double ... x);  
}  
  
class Gymnastics implements CompurerAverage {  
    public double average(double ... x) {  
        int count=x.length;  
        double aver=0,temp=0;  
        for(int i=0;i<count;i++) {  
            for(int j=i;j<count;j++) {  
                if(x[j]<x[i]) {  
                    temp=x[j];  
                    x[j]=x[i];  
                    x[i]=temp;  
                }  
            }  
        }  
        for(int i=1;i<count-1;i++) {  
            aver=aver+x[i];  
        }  
        if(count>2)  
            aver=aver/(count-2);  
        else  
            aver=0;  
        return aver;  
    }  
}
```

## 6.2.2 接口的多态性

```
class School implements CompurerAverage {  
    public double average(double ... x) {  
        int count=x.length;  
        double aver=0;  
        for(double param:x) {  
            aver=aver+param;  
        }  
        aver=aver/count;  
        return aver;  
    }  
}  
  
public class Example6_4 {  
    public static void main(String args[]) {  
        CompurerAverage computer;  
        computer=new Gymnastics();  
        double result= computer.average(9.87,9.76,9.99,9.12,9.67,9.73);  
        System.out.printf("体操选手最后得分%5.3f\n",result);  
        computer=new School();  
        result=computer.average(65,89,76,56,88,90,98,46);  
        System.out.println("班级考试平均分数:"+result);  
    }  
}
```

Eclipse演示

## ■■■ 6.2.3 接口与abstract类的比较

- (1) 接口中只可以有常量，不能有变量；而abstract类中既可有常量也可以有变量。
- (2) 接口中全部为抽象方法，不能有非抽象方法； abstract类中既可以有抽象方法，也可以有非抽象方法。



## 6.2.3 接口与abstract类的比较

- (1) 如果某个问题需使用继承才能更好的解决，例如，子类除了需要重写父类的**abstract**方法，还需要从父类继承一些变量或者继承一些重要的非抽象方法，考虑用**abstract**类。
- (2) 如果某个问题不需要继承，只需若干个重要的**abstract**方法的实现细节，则考虑接口。



Eclipse演示





## 第6章 接口与实现

- 1. 接口
  - 2. 接口回调
  - 3. 面向接口编程
- 

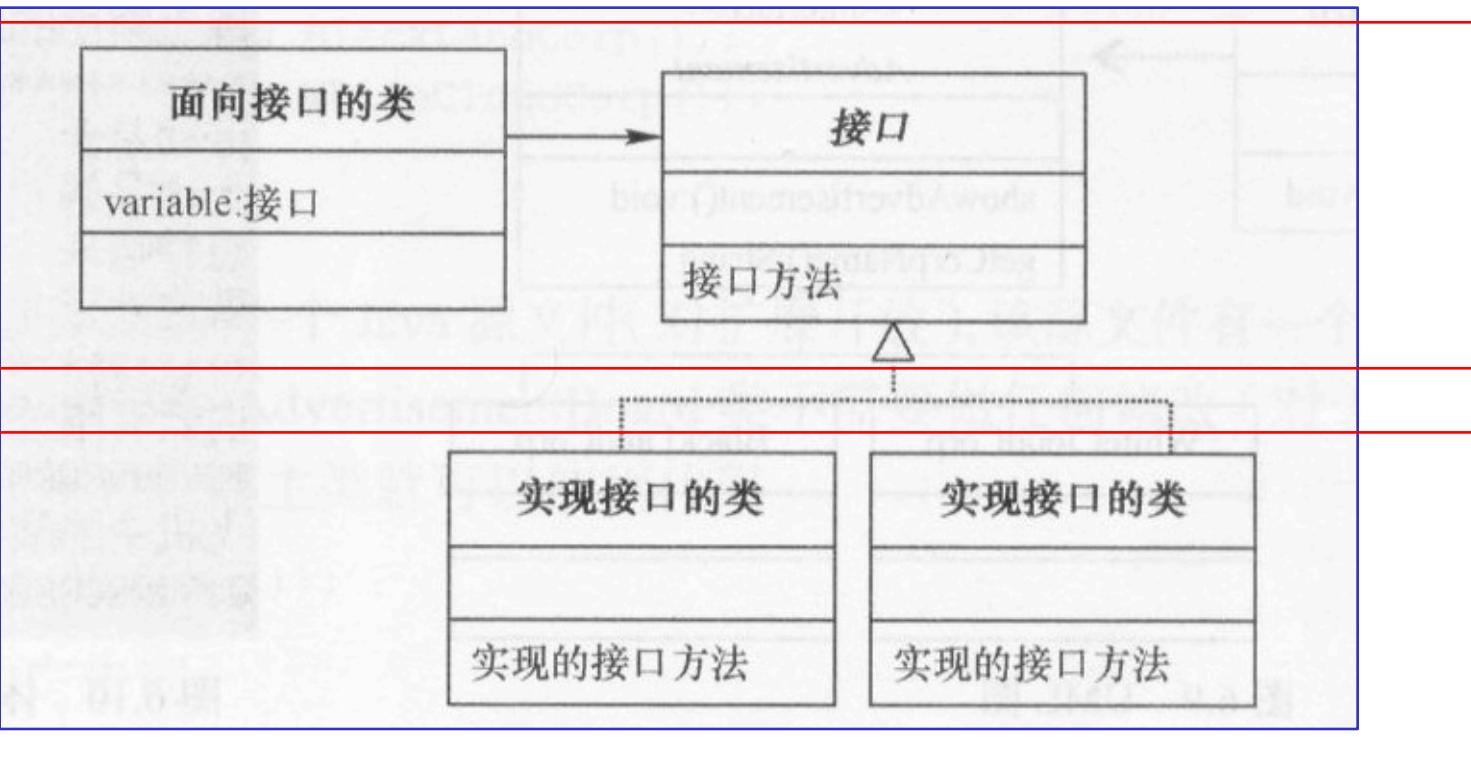
## ■■■ 6.3 面向接口编程

- **面向接口编程思想：** 使用接口回调，即接口变量存放实现该接口的类所创建的对象的引用，从而接口变量可以回调类实现的接口方法。



## 6.3 面向接口编程

- 面向接口编程也可体现程序设计“开-闭”原则，即对扩展开放，对修改关闭。



## ■■■ 6.3 面向接口编程

设计一个广告牌，希望所设计的广告牌可展示许多公司的广告词

### 思考：

#### ① 问题的分析

- 如果我们设计的创建广告牌类中用某个具体公司类，每当用户有新的需求，就会导致修改类的某部分代码。
- 因此，就应当将这部分代码从该类中分割出去，使它和类中其他稳定的代码之间是松耦合关系。
- 将每种可能的变化对应地交给实现接口的类去负责完成。



## 6.3 面向接口编程

### ② 设计: 广告接口

设计接口Advertisement，

有2个方法: `showAdvertisement()` 和 `getCorpName()`

实现接口的类必须重写上述方法

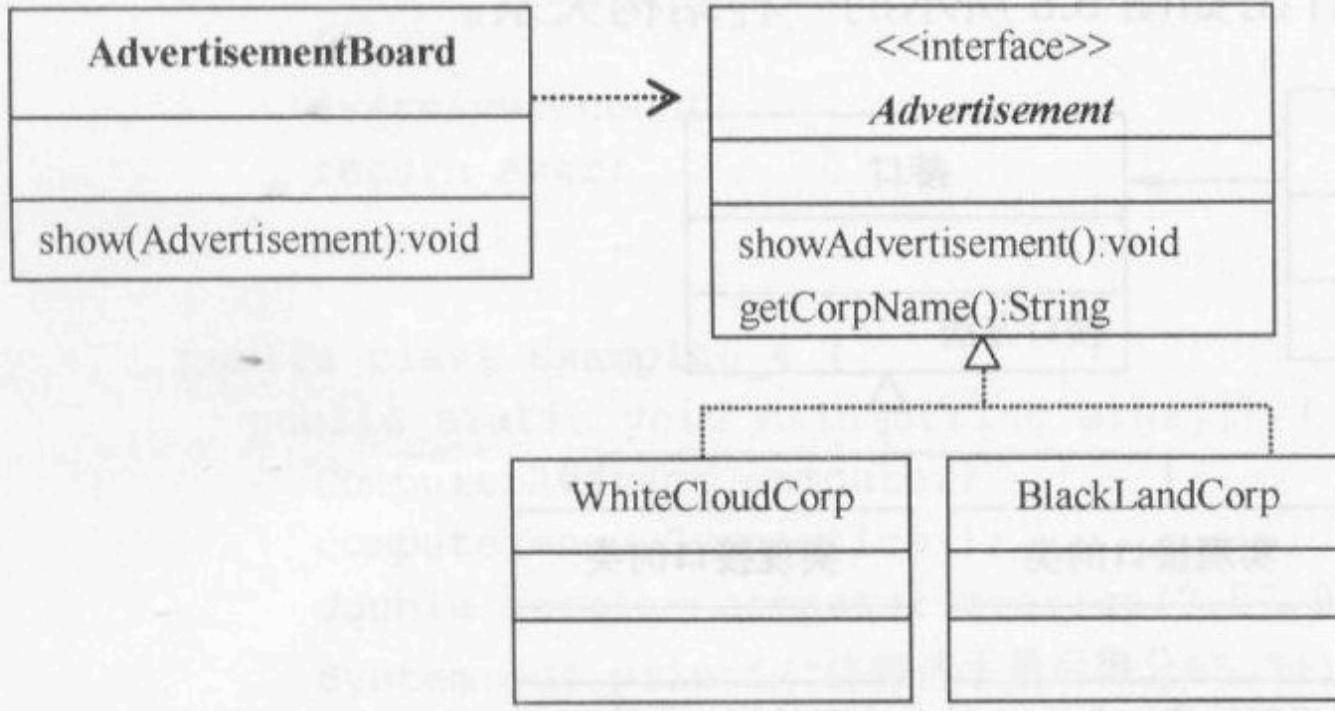
### ③ 设计: 广告牌类

设计AdvertisementBoard类 (广告牌)

该类有一个`show(Advertisement adver)`方法，该方法参数adver是Advertisement接口类型

参数adver可存放任何实现该接口的类所创建对象的引用，并回调类重写的接口方法`showAdvertisement()`来显示公司的广告词，回调类重写的接口方法`getCorpName()`来显示公司的名称。

## 6.3 面向接口编程



## ■■■ 6.3 面向接口编程

**Advertisement.java**

```
public interface Advertisement { //接口  
    public void showAdvertisement();  
    public String getCorpName();  
}
```

**AdvertisementBoard.java**

```
public class AdvertisementBoard { //负责创建广告牌  
    public void show(Advertisement adver) {  
        System.out.println(adver.getCorpName()+"的广告词如下:");  
        adver.showAdvertisement(); //接口回调  
    }  
}
```

## 6.3 面向接口编程

## WhiteCloudCorp.java

```
public class WhiteCloudCorp implements Advertisement { //PhilipsCorp 实现Advertisement  
接口  
  
    public void showAdvertisement() {  
        System.out.println("@@@@@@@@@@@@@@@@");  
        System.out.printf("飞机中的战斗机，哎 yes!\n");  
        System.out.println("@@@@@@@@@@@@");  
    }  
    public String getCorpName() {  
        return "白云有限公司" ;  
    }  
}
```

## 5.3 面向接口编程

### **BlackLandCorp.java**

```
public class BlackLandCorp implements Advertisement {  
    public void showAdvertisement() {  
        System.out.println("*****");  
        System.out.printf("劳动是爹\n土地是妈\n想种啥来\n就往外接\n");  
        System.out.println("*****");  
    }  
    public String getCorpName() {  
        return "黑土集团";  
    }  
}
```

### **Example6\_5.java**

```
public class Example6_5 {  
    public static void main(String args[]) {  
        AdvertisementBoard board = new AdvertisementBoard();  
        board.show(new BlackLandCorp());  
        board.show(new WhiteCloudCorp());  
    }  
}
```

## 5.3 面向接口编程

**BlackLandCorp.java**

```
public class BlackLandCorp implements Advertisement {  
    public void showAdvertisement() {  
        System.out.println("黑土集团的广告词如下：  
劳动是爹  
土地是妈");  
    }  
}
```

**Example**

```
public class Example6_6 {  
    public static void main(String args[]) {  
        AdvertisementBoard board = new AdvertisementBoard();  
        board.show(new BlackLandCorp());  
        board.show(new WhiteCloudCorp());  
    }  
}
```

Eclipse演示

# 小结

1. 接口的定义与使用
2. 接口回调
3. 面向接口编程

