Java程序设计

计算机科学与技术

贡正仙

zhxgong@suda.edu.cn

第6章 接口

- 导读

  重点：

  接口回调与多态

  难点：

  接口的设计

6.1 接口

为何引入接口？

◆ 为了克服Java**单继承**的缺点，Java使用了接口，一个类可以实现多个接口。

◆ 面向接口编程是使用接口来约束类的行为，并为类和类之间的通讯建立实施标准。

◆ 面向接口编程是对多态特性的一种体现。

6.1 接口的定义

接口的定义分为接口的声明和接口体。

- 接口声明

关键字interface来声明，格式：

interface 接口的名字

- 接口体

接口体中包含常量定义(默认 public static final,可省)和方法定义(默认 public abstract,可省)两部分。

```
[修饰符] interface 接口名 [extends 父接口名列表]{
    [public] [static] [final] 常量;
    [public] [abstract] 方法;
}
```

6.1 接口的定义

```
interface Printable{
      int MAX=100;
      void add();
}
```

```
interface Printable{
      public static final int MAX=100;
      public abstract void add();
}
```

静态常量

System.out.println(Printable.Max)

Java中的接口是一个特殊的抽象类，接口中的所有方法都没有方法体。（JDK8之前）

6.2 接口的实现

一个类通过使用关键字implements声明自己实现一个或多个接口。

实现单接口

```
interface  Fee
{
    void  charge();
}

class Bus implements Fee{
      void charge(){
          //实现charge
       }
}
```

6.2 接口的实现

实现多接口

```
interface  Fee
{
 void  charge();
}
```

```
interface  TemperatureControl
{
    void  controlTemperature();
}
```

```
class Taxi implements Fee,TemperatureControl
{
      void charge(){
          //实现charge
       }
      void  controlTemperature(){
          //实现controlTemperature
       }
…..
}
```

例子：InterfaceMultipleImpliment1.java

6.2 接口的实现

既继承又实现接口

```
class A extends B implements Xable, Yable{


}
```

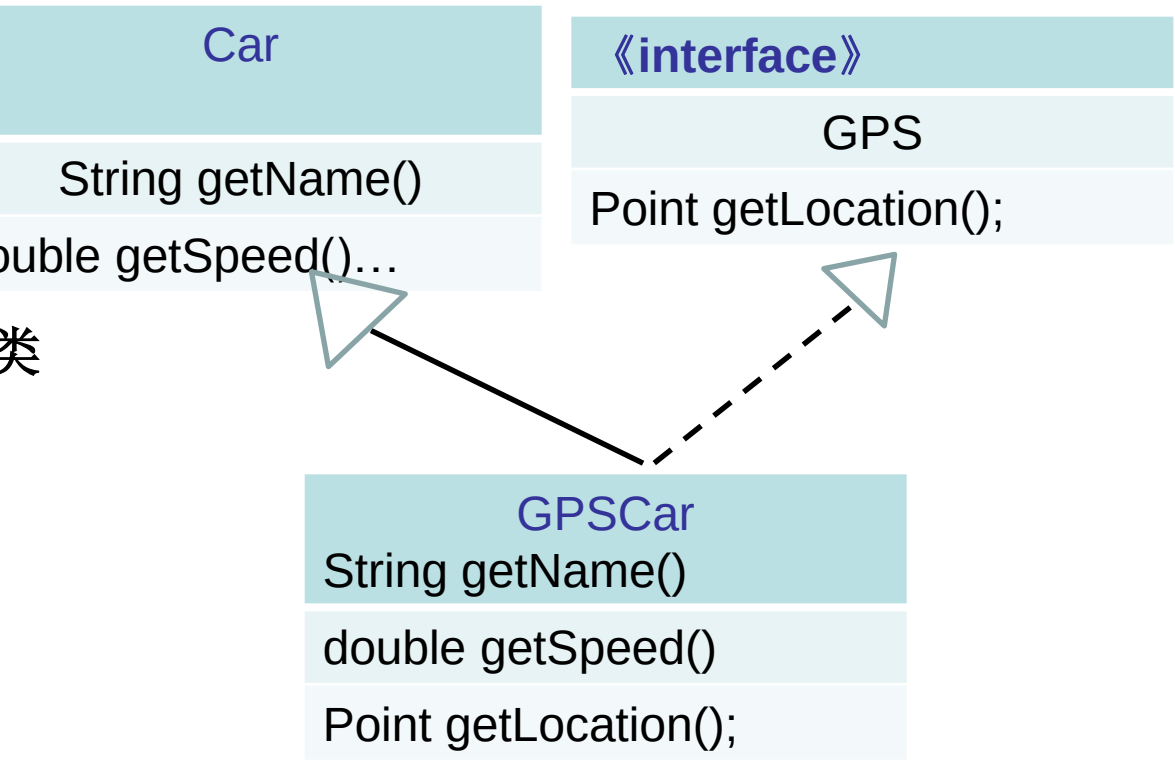
→

6.2 接口的实现

既继承又实现接口

例子

GPS接口
Car类



GPSCar类

见 InterfaceAndExtend.java

6.2 接口的实现

一个经典模式：接口–抽象类–类

接口Monitor-》

实现Monitor的抽象类MonitorAB-》

继承抽象类MonitorAB的LCDMonitor

例子：InterfaceAndExtend2.java

6.2 接口的实现

理解接口：

(1)接口只关心操作，并不关心操作的具体实现 ；

(2)接口可以增加很多类都需要具有的功能，一个类可以实现多个接口，不同的类可以实现相同的接口；

(3)接口的思想在于它可以增加很多类都需要具有的功能，而且实现相同的接口类不一定有继承关系。

```
class Cinema implements Fee,TemperatureControl
{}
class Taxi implements Fee,TemperatureControl
{}
```

6.2 接口的实现

如果一个类实现某个接口，但没实现它声明的所有方法，该类必须是**abstract**类

```
interface  Fee
{
    void  charge();
    void print();
}
abstract class Partial  implements Fee
{
     void charge(){
          //…….
     }
}
```

思考

```
interface  Fee
{
    void  charge();
    void print();
    void payback();
}
```

```
Class A implements Fee{
    void charge(){}
    void print(){}
}
```

void payback(){}

```
Class B implements Fee{
    void charge(){}
    void print(){}
}
```

6.3 接口的继承

接口是可以被继承的。

和类的继承相似：当子类继承父类接口时，子类会获得父类接口中定义的所有抽象方法、常量属性等。

```
interface  Fee
{
    void  charge();
    void print();
}
```

```
interface  FeeAdvance extends Fee
{
    void  payback();
}
```

6.3 接口的继承

和类的继承相似：当子类继承父类接口时，子类会获得父类接口中定义的所有抽象方法、常量属性等。

与类的继承不一样：接口可以实现多继承，也就是说接口可以有多个直接父接口。（见下例）

6.3 接口的继承

```
interface  Fee
{  void  charge();
}
interface  TemperatureControl
{
    void  controlTemperature();
}

interface HunHe extends Fee, TemperatureControl
{
}
```

例子：InterfaceMultipleImpliment2.java

6.4 接口与抽象类

接口与抽象类的**共同点**如下：

（1）接口与抽象类都不能被实例化，能被其他类实现和继承

（2）接口和抽象类中都可以包含抽象方法，实现接口或继承抽象类的普通子类都必须实现这些抽象方法。

• 6.4 接口与抽象类

如果一个类实现某个接口，但没实现它声明的所有方法，该类必须是**abstract**类

```
interface  Fee
{
    void  charge();
    void print();
}
abstract class Partial  implements Fee
{
     void charge(){
          //…….
     }
}
```

6.4 接口与抽象类

接口与抽象类的用法差别如下：

（1）接口中只能包含抽象方法，不能包含普通方法；抽象类中可以包含普通方法。(JDK8之前 P112)

（2）接口中不能定义静态方法；抽象类中可以定义静态方法。(JDK8之前)

（3）接口中只能定义静态常量属性，不能定义普通属性；抽象类里可以定义静态常量属性，也可以定义普通属性。

（4）接口不能包含构造器；抽象类可以包含构造器，抽象类里的构造器为了让其子类调用并完成初始化操作。

（5）接口中不能包含初始化块，但抽象类可以包含初始化块。

（6）一个类最多只能有一个直接父类，包括抽象类；但是一个类可以实现多个接口。

6.5 接口回调

使用接口进行程序设计的核心思想是使用**接口回调**，即接口变量存放实现该接口的类的对象的引用，从而接口变量就可以回调类实现的接口方法。

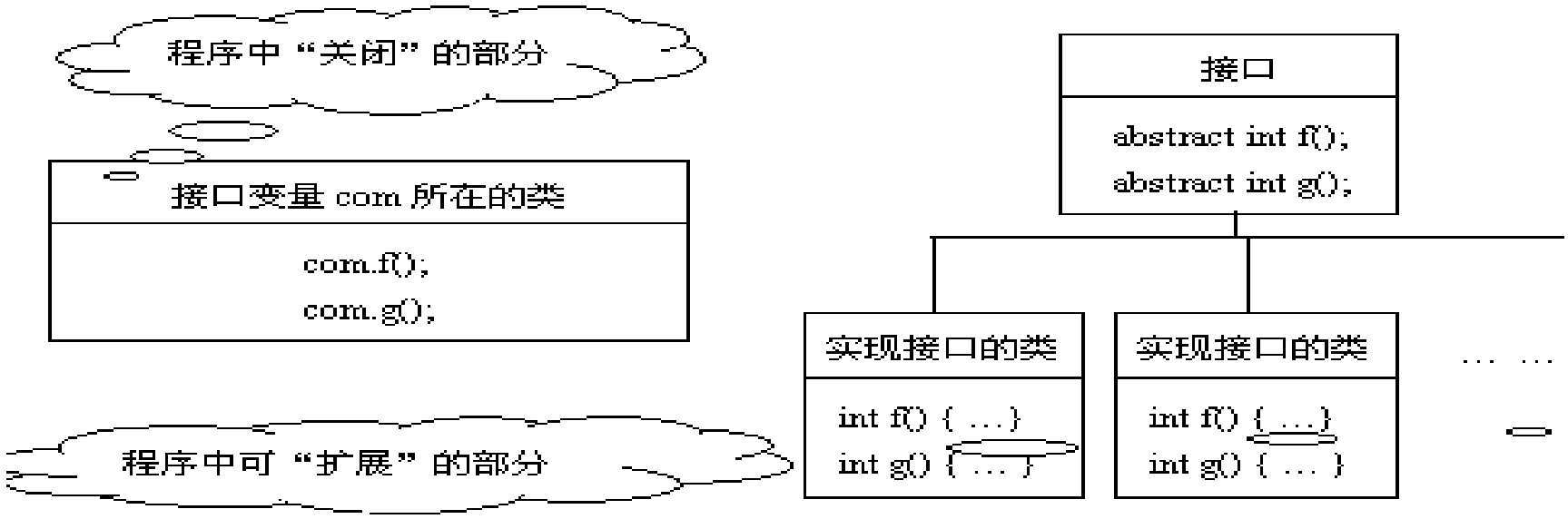


图 5.15　接口与多态的使用

6.5 接口回调

```
interface  ShowMessage {
   void display(String s);
}
```

```
class TV implements ShowMessage {
   public void display (String s) {
     }
}
class PC implements ShowMessage {
   public void display (String s) {
   }
}
```

```
public class Example5_16 {
   public static void main(String args[]) {
      ShowMessage sm;
      sm=new TV();                    //接口变量中存放对象的引用，向上转型混淆
      sm.display(“长城牌电视机”);      //接口回调。
      sm=new PC();                    //接口变量中存放对象的引用
      sm.display("联想奔月5008PC机"); //接口回调
   }
}
```

6.5 接口回调

面向接口编程是对多态特性的一种体现。当接口变量调用被类重写的接口方法时，就是通知相应的对象调用这个方法。

回顾覆盖和向上转型的一个例子

extendClass/UpConverter2.java

改写：

InterfaceAndExtend2.java

```
public static void run(LCDMonitor monitor) {
        monitor.displayText();
        monitor.displayGraphics(
}
public static void run(CRTM
        monitor.displayText();
        monitor.displayGraphics(
}
public static void run(Plasr
        monitor.displayText();
        monitor.displayGraphics();
}
```

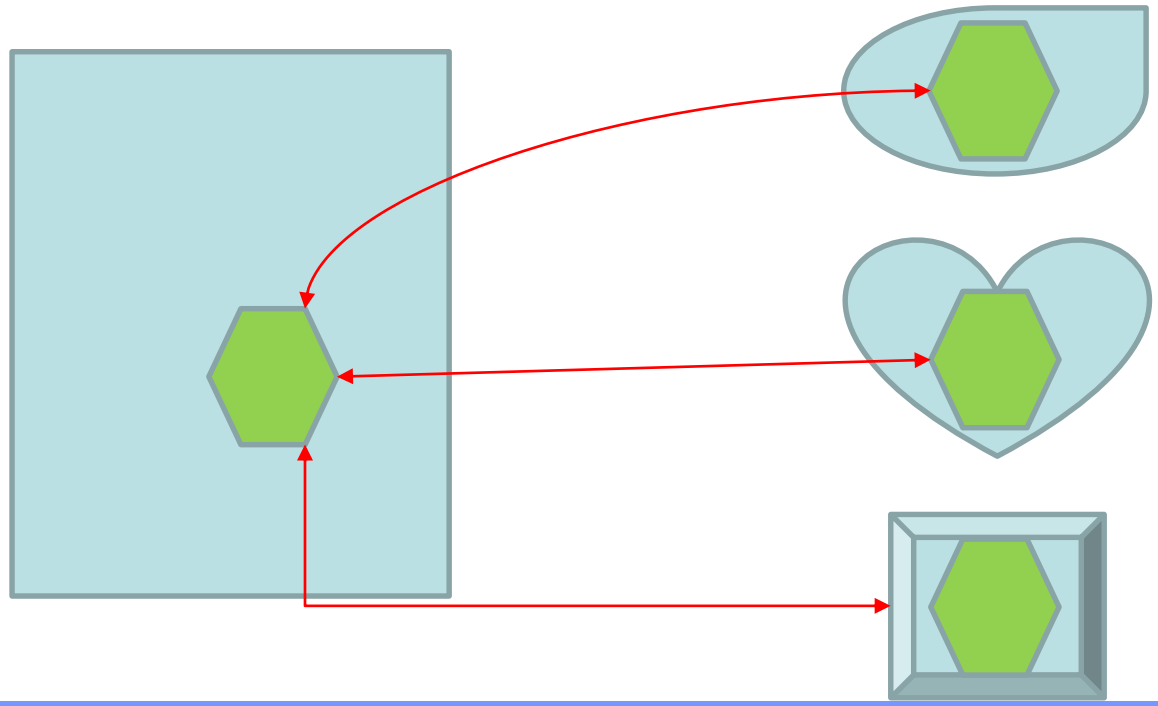
```
public static void run(Monitor monitor) {
   //多态性
   monitor.displayText();
   monitor.displayGraphics();
  if (monitor instanceof LCDMonitor){
  LCDMonitor tempobj= (LCDMonitor)monitor;
  tempobj.getColor();
   }
}
```

```
Monitor m=new LCDMonitor();
        run(m);
        m=new CRTMonitor();
        run(m);
        m=new PlasmaMonitor();
        run(m);
```

```
interface Monitor{
void displayText();
void displayGraphics();
}
        public static void run(Monitor monitor) { //父类实例作为参数
            //多态性
             monitor.displayText();
             monitor.displayGraphics();

        }
LCDMonitor lc=new LCDMonitor();
CRTMonitor crt=new CRTMonitor();
run(lc);
run(crt);
```

- 面向接口编程是使用接口来约束类的行为，并为类和类之间的通讯建立实施标准。



6.1 接口

- 使用面向接口编程增加了程序的**可维护性**和**可扩展性**

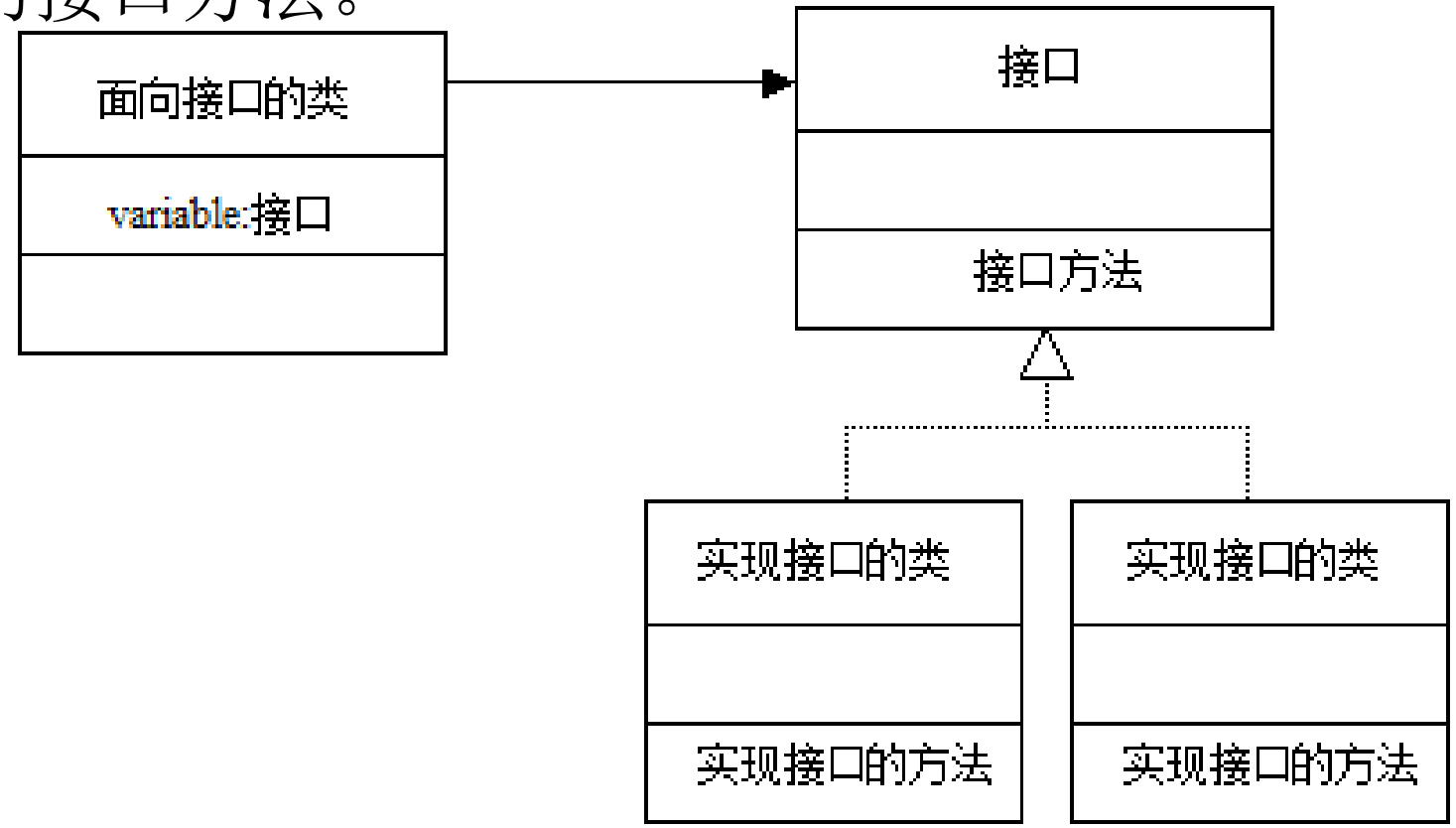
➢ 可维护性

当子类的功能修改时，只要接口不发生改变，系统其他代码就不需要改动。

➢ 可扩充性

当增加一个子类时，测试类和其他代码都不需要改动，如果子类增加其他功能，只需要子类实现其他接口即可。

使用接口可以实现程序设计的“开-闭原则”，即对扩展开放，对修改关闭。当多个类实现接口，接口变量variable所在的类不需要做任何修改，就可以回调类重写的接口方法。



退 出

- 实战Arrays.sort(T[])

实现Comparable的类

➢T是Integer

见InterfaceIntegerComparable.java

public final class Integer extends Number implements Comparable<Integer>

Interface Comparable<T>

int	compareTo(T o) Compares this object with the specified object for order.

- 实战　Arrays.sort(T[])

实现Comparable的类

➢T是自定义类型

```
class Employee {
private int id;
    private String name;
    private int age;
    …
}
```

- 实战

Arrays.sort(T[])

另一方法更灵活的**Comparator**

(集合类)

本章小结

在本章中，我们主要介绍了有关接口和多态性方面的内容。

重点理解：接口回调