

# 抽象类和抽象方法

## 一：知识梳理

### 1.基本概念：

在面向对象的概念中，所有的对象都是通过类来描述的，但并不是说所有的类都是用来描述对象的，当一个类中没有包含足够的信息以描绘一个具体的对象时，这样的类就是抽象类。

例：形状就是一个抽象的概念。不同的子类计算形状面积的方法是不一样的。可提供抽象方法来被不同的子类所实现。

```
//抽象类Shape
abstract public class Shape {
    abstract double area(); //抽象方法
}
```

从例子中可看出，抽象了是用关键字 **abstract** 修饰的。抽象类中有一种特殊方法，即用 **abstract** 关键字来修饰的方法，称为“抽象方法”。

### 2.抽象类和抽象方法的声明格式：

```
abstract class <类名>{
    成员变量;
    方法（）{方法体}; //一般方法
    abstract 方法（）; //抽象方法
}
```

### 3.抽象类和抽象方法的特点：

(1) 抽象方法不允许直接实例化，换句话说抽象类不能创建对象，它只能作为其他类

的父类。 但可以通过向上转型，指向实例化。

(2) 抽象方法只有声明，不能有实现，也就是仅有方法头，而没有方法体和操作实现。

如：`abstract double area();`

4. 定义抽象类的意义在于：

(1) 为其子类提供一个公共的类型（父类引用指向子类对象）；

(2) 封装子类中的重复内容（成员变量和方法）；

(3) 将父类设计成抽象类后，既可借由父子继承关系限制子类的设计随意性，在一定程度上避免了无意义父类的实例化。

## 二．重点注意

- 含有抽象方法的类，只能被定义成抽象类。

如下面，不定义为抽象类时会报错：

```
public class Shape {  
    abstract double area(); // 抽象方法  
}
```

正确的代码为：

```
// 抽象类 Shape  
abstract public class Shape {  
    abstract double area(); // 抽象方法  
}
```

- 抽象类不一定包含抽象方法。

例子：

```

abstract public class Shape {
    public void girth(){
        System.out.println("图形周长为..."); //一般方法
    }
}

```

- 在抽象类中的成员方法可以包括一般方法和抽象方法

```

//抽象类Shape
abstract public class Shape {
    public void girth(){
        System.out.println("图形周长为..."); //一般方法
    }
    abstract double area();//抽象方法
}

```

- 抽象类不能被实例化,即使抽象类里不包含抽象方法,这个抽象类也不能创建实例。

抽象类的构造方法主要是用于被其子类调用。

例子：

Shape 抽象类中不含抽象方法：

```

abstract public class Shape {
    public void girth(){
        System.out.println("图形周长为..."); //一般方法
    }
}

```

测试类中实例化 Shape,编译器会报错：

```

Shape s=new Shape();

```

- 一个类继承抽象类后，必须实现其所有抽象方法，否则也是抽象类，不同的子类对

父类的抽象方法可以有不同的实现。

如父类为：

```
//抽象类父类Shape
abstract public class Shape {
    abstract double area();//抽象方法
}
```

则其子类 Circle 有两种做法：

方案一：重写抽象方法 area(),使方法得以实现

```
//子类圆Circle类
public class Circle extends Shape {
    //属性：圆的半径r
    public double r;
    public Circle (double r){
        this.r=r;
    }
    //重写父类中area()方法
    public double area(){
        return(double)(3.14*r*r);
    }
}
```

方案二：子类 Circle 类也定义为抽象类


```
//子类圆Circle类
public abstract class Circle extends Shape {
    //属性：圆的半径r
    public double r;
    public Circle (double r){
        this.r=r;
    }
}
```


- 即使父类是具体的，但其子类也可以是抽象的。如 Object 是具体的，但可以创建抽象子类。

- **abstract 方法不能用 static 和 private 修饰；对于类，不能同时用 final 和 abstract 修饰，因为 final 关键字使得类不可继承，而 abstract 修饰的类如果不可以继承将没有任何意义。两者放在一起，会起冲突**

如以下用法都会引起编译器报错：

```
 static abstract double area(); //抽象方法
```

```
 private abstract double area(); //抽象方法
```

```
 abstract final class Shape {  
    abstract double area(); //抽象方法  
}
```

### 三：一个完整准确的抽象类例子

声明一个抽象类 Shape,有抽象成员方法 area()。Shape 派生出两个子类圆 Circle 类和矩形 Rectangle 类。Shape 里声明了抽象方法 area(),该方法分别在两个子类里得到实现。

代码如下：

```
package com.imooc.shape;  
//抽象类父类Shape  
abstract public class Shape {  
    abstract double area(); //抽象方法  
}
```

---

```
//子类圆Circle类
public class Circle extends Shape {
    //属性：圆的半径r
    public double r;
    Circle (){

    }
    //创建带参构造函数（参数为r）
    public Circle (double r){
        this.r=r;
    }
    public double getR() {
        return r;
    }
    public void setR(double r) {
        this.r = r;
    }
    //重写父类中area()方法
    public double area(){
        return(double)(3.14*r*r);
    }
}
```

慕课网  
imooc.com

---

```
//子类矩形类Rectangle
public class Rectangle extends Shape {
    //属性：矩形的长length、宽wide
    public double length;
    public double wide;

    Rectangle(){

    }
    //创建带参构造方法（以length和wide为参数）
    public Rectangle(double length,double wide){
        this.length=length;
        this.wide=wide;
    }

    public double getLength() {
        return length;
    }
    public void setLength(double length) {
        this.length = length;
    }
    public double getWide() {
        return wide;
    }
    public void setWide(double wide) {
        this.wide = wide;
    }
    //重写父类的area（）方法
    public double area(){
        return length*wide;
    }
}
```

```
import com.imooc.shape.Circle;
//测试类
public class Test {

    public static void main(String[] args) {
        //创建类的实例，将圆的半径设为3.5，矩形的长宽分别是6和5
        Circle c=new Circle(3.5);
        Rectangle re=new Rectangle(6,5);
        //调用area（）方法，输出结果
        System.out.println("圆的面积为 "+c.area());
        System.out.println("矩形的面积为 "+re.area());

    }

}
```