# 接口

什么是接口：当一个抽象类，如果抽象类中的所有方法都是抽象的，那么我们就可以把它定义为一个接口，接口是对行为的抽象。类是对属性和行为的抽象。接口中没有属性的。

interface 接口名{

方法定的定义；

….

}

## 接口的特点

1. 接口中的方法的定义不需要Abstract来修饰，默认就是抽象的
2. 接口是不可以实例化的，需要有类来实现接口实现接口的语法：

class 类名 implements 接口名，接口名,…{

//实现每一个接口中的方法

}

一个类实现一个接口就必须要实现这个抽象类中的所有抽象方法。

1. 接口中的方法不能和private，static和final共存
2. 在接口中可以定义“属性“，可以通过接口的实现类的实例来访问，还可以通过接口名来访问（推荐大家 使用），接口中的属性不能修改，我们接口中的属性默认都是final static 的，通常在接口中来定义 属性把它作为常量，常量的名字规范是单词大写，而且多个单词之间用下划线来分隔。比如： FILE\_PATH
3. 接口可以继承接口， 可以多继承

## 案例

范例：使用接口方式计算圆和正方形的面积和周长。

Interface Cal

calLong();计算周长

calArea();计算面积

Class Circle class Rect

Double r double r

calLong(){}计算周长 calLong(){}计算周长

calArea(){}计算面积 calArea(){}计算面积

# 接口在JDK1.8中的特性

在接口中我们之前1.7里面的抽象方法是有强制性必须实现，jdk1.8里面提供了默认的方法，让我们可以选择性的实现，让我们的接口更加灵活。

接口可以定义接口的默认方法：

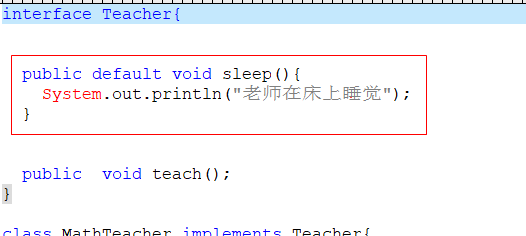
public interface InterDemo {

public default void method(){

//方法体

}

}



# 父子类之间类型转换

## 子类转换成父类。

|  |
| --- |
| class Teacher{  String name = "普通老师";  int age;    public void sleep(){  System.out.println("老师在睡觉");  }  }  class JavaTeacher extends Teacher{    String name = "亮哥老师";    public void sleep(){  System.out.println("java老师在刀尖睡觉");  }    public void makeFriend(){  System.out.println("喜欢交朋友");  }  }  class PhPTeacher extends Teacher{    }  public class ExtendsConverterTest{  public static void main(String [] args){    JavaTeacher jt = new JavaTeacher();  //子类的实例可以自动的转换成父类的类型  Teacher t = jt;  //如果子类里面有对父类的方法的重写 的话，那么通过父类的引用调用重写的方法的时候调用的是子类  t.sleep();  System.out.println(t.name);    //t是Teacher类型，Teacher类中没有makeFriend();,不可以方法子类的特有方法  t.makeFriend();        }  } |

## 父类转换成子类

|  |
| --- |
| class Teacher{  String name = "普通老师";  int age;    public void sleep(){  System.out.println("老师在睡觉");  }  }  class JavaTeacher extends Teacher{    String name = "亮哥老师";    public void sleep(){  System.out.println("java老师在刀尖睡觉");  }    public void makeFriend(){  System.out.println("喜欢交朋友");  }  }  class PhPTeacher extends Teacher{    }  public class ExtendsConverterTest1{  public static void main(String [] args){        Teacher t = new JavaTeacher();  //Teacher t = new PhPTeacher();  //父类如果转换成子类必须要强制转换，但是前提是这个父类的真身是这个子类  JavaTeacher j1 = (JavaTeacher)t;    }  } |

# 多态

多态就是行为具有表现多种功能的能力。

## 继承多态

|  |
| --- |
| abstract class Teacher{  String name;    public abstract void teach();  }  class JavaTeacher extends Teacher{  public void teach(){  System.out.println("java老师在讲java课");  }  }  class PhPTeacher extends Teacher{  public void teach(){  System.out.println("php老师在讲php课");  }  }  class NetTeacher extends Teacher{  public void teach(){  System.out.println("net老师在讲net课");  }  }  class Leader{    // Teacher t = new PhPTeacher()  public void checkTeacher(Teacher t){  //判断t的真身是什么  if(t instanceof JavaTeacher){  System.out.println("java老师开始讲课");  }  if(t instanceof PhPTeacher){  System.out.println("php老师开始讲课");  }  if(t instanceof NetTeacher){  System.out.println("net老师开始讲课");  }      t.teach();  System.out.println("打分");  }    }  public class CheckTeacherTest5{  public static void main(String [] args){  //定义一个领导  Leader l = new Leader();    //创建一个java老师  JavaTeacher jt = new JavaTeacher();  jt.name = "亮哥";  l.checkTeacher(jt);  System.out.println("-------------------------");    //创建一个Php的老师  PhPTeacher pt = new PhPTeacher();  pt.name = "龙哥";  l.checkTeacher(pt);  System.out.println("-------------------------");  NetTeacher nt = new NetTeacher();  nt.name = "豹哥";  l.checkTeacher(nt);  }  } |

## 接口多态

|  |
| --- |
| interface Teacher{      public void teach();  }  class JavaTeacher implements Teacher{  String name;  public void teach(){  System.out.println("java老师在讲java课");  }  }  class PhPTeacher implements Teacher{  String name;  public void teach(){  System.out.println("php老师在讲php课");  }  }  class NetTeacher implements Teacher{  String name;  public void teach(){  System.out.println("net老师在讲net课");  }  }  class Leader{    // Teacher t = new PhPTeacher()  public void checkTeacher(Teacher t){  //判断t的真身是什么  if(t instanceof JavaTeacher){  System.out.println("java老师开始讲课");  }  if(t instanceof PhPTeacher){  System.out.println("php老师开始讲课");  }  if(t instanceof NetTeacher){  System.out.println("net老师开始讲课");  }      t.teach();  System.out.println("打分");  }    }  public class CheckTeacherTest6{  public static void main(String [] args){  //定义一个领导  Leader l = new Leader();    //创建一个java老师  JavaTeacher jt = new JavaTeacher();  jt.name = "亮哥";  l.checkTeacher(jt);  System.out.println("-------------------------");    //创建一个Php的老师  PhPTeacher pt = new PhPTeacher();  pt.name = "龙哥";  l.checkTeacher(pt);  System.out.println("-------------------------");  NetTeacher nt = new NetTeacher();  nt.name = "豹哥";  l.checkTeacher(nt);  }  } |

# API的使用

|  |
| --- |
| public class APITest{  /\*  java.lang这个包下的所有类都是自动引入的。    自然对数：Math.E  圆周率：Math.PI  绝对值：Math.abs(-1.5)  把小数去掉整数加1 Math.ceil(12.1)  把小数位舍弃 Math.floor(12.9)  比较两个数的大小 Math.max(12.9, 27)  比较两个数的大小 Math.min(12.9, 27)  计算某个数的n次幂 Math.pow(3, 3)  四舍五入 Math.round(4.5)  开平方 Math.sqrt(16));    \*/  public static void main(String[] args){  System.out.println(Math.E);  System.out.println(Math.PI);    int result = Math.abs(-1);  System.out.println("-1的绝对值是 ："+result);    double result1 = Math.ceil(1.3);  System.out.println("向上取整的小数 ："+result1);      result1 = Math.floor(1.8);  System.out.println("向下取整的小数 ："+result1);      result1 = Math.max(1.8, 2.9);  System.out.println("求最大值 ："+result1);    result1 = Math.min(1.8, 2.9);  System.out.println("求最小值 ："+result1);      result1 = Math.pow(3, 3);  System.out.println("求次幂 ："+result1);    result1 = Math.sqrt(16);  System.out.println("开平方 ："+result1);    long l = Math.round(1.6);  System.out.println("四舍五入 ："+l);  }  } |

# 包的使用

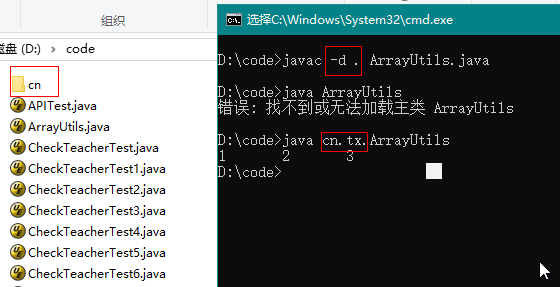
包：就是文件夹

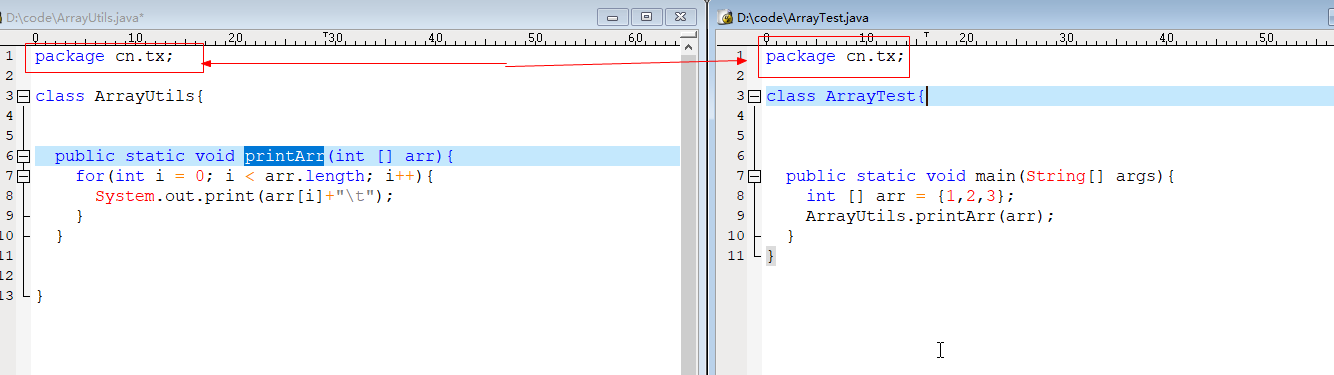
作用：对类做分类管理，可以区分同名不同包的类。

语法: package 包名（不同级别用“.”来分隔）;，如 package com.rl;, 位置放置文件的有效代码第一行（注释不算）

如何编译：

Javac –d . 源文件





相同包下的类可以直接使用

如果两个类在不同的包下，不能直接使用的



我们必须要通过类的引入才能使用其他包下的类。

