目录

[Day09.    Java](http://code.tarena.com.cn/CGBCode/cgb1903/01-java/%b1%ca%bc%c7/day09.htm#_Toc5984358)

[1     多态](http://code.tarena.com.cn/CGBCode/cgb1903/01-java/%b1%ca%bc%c7/day09.htm#_Toc5984359)

[1.1       类型转换](http://code.tarena.com.cn/CGBCode/cgb1903/01-java/%b1%ca%bc%c7/day09.htm#_Toc5984360)

[1.1.1   向上转型](http://code.tarena.com.cn/CGBCode/cgb1903/01-java/%b1%ca%bc%c7/day09.htm#_Toc5984361)

[1.1.2   向下转型](http://code.tarena.com.cn/CGBCode/cgb1903/01-java/%b1%ca%bc%c7/day09.htm#_Toc5984362)

[1.2   instanceof](http://code.tarena.com.cn/CGBCode/cgb1903/01-java/%b1%ca%bc%c7/day09.htm#_Toc5984363)

[练习1   多态测试](http://code.tarena.com.cn/CGBCode/cgb1903/01-java/%b1%ca%bc%c7/day09.htm#_Toc5984364)

[2     什么是面向对象](http://code.tarena.com.cn/CGBCode/cgb1903/01-java/%b1%ca%bc%c7/day09.htm#_Toc5984365)

[3     抽象类](http://code.tarena.com.cn/CGBCode/cgb1903/01-java/%b1%ca%bc%c7/day09.htm#_Toc5984366)

[3.1       抽象类测试](http://code.tarena.com.cn/CGBCode/cgb1903/01-java/%b1%ca%bc%c7/day09.htm#_Toc5984367)

[练习2   生成一个有序的数字数组](http://code.tarena.com.cn/CGBCode/cgb1903/01-java/%b1%ca%bc%c7/day09.htm#_Toc5984368)

[4     BigDecimal，BigInteger](http://code.tarena.com.cn/CGBCode/cgb1903/01-java/%b1%ca%bc%c7/day09.htm#_Toc5984369)

[练习3  BigDecimal](http://code.tarena.com.cn/CGBCode/cgb1903/01-java/%b1%ca%bc%c7/day09.htm#_Toc5984370)

[5     位运算符](http://code.tarena.com.cn/CGBCode/cgb1903/01-java/%b1%ca%bc%c7/day09.htm#_Toc5984371)

[练习4   位运算](http://code.tarena.com.cn/CGBCode/cgb1903/01-java/%b1%ca%bc%c7/day09.htm#_Toc5984372)

[6     作业](http://code.tarena.com.cn/CGBCode/cgb1903/01-java/%b1%ca%bc%c7/day09.htm#_Toc5984373)

**Day09.  Java**

**1   多态**

一个子类型实例，可以被当做父类型来处理

子类实例

Point3D a =

{

   x,y,z

}

Point a =

{

   x,y,z

}

多态作用：一致的类型

void f(父类型 o1) {

}

所有子类型对象，都可被传递到该方法，被当做一致的父类型来处理

**1.1     类型转换**

**1.1.1 向上转型**

子类型实例，转型成父类型

用父类型的引用变量，来引用子类实例

向上转型后，只能调用父类定义的通用成员，子类特有成员被隐藏

**1.1.2 向下转型**

已经转成父类型的子类实例，转回成子类型

**1.2     instanceof**

运行期类型识别

当多种子类型，都当做父类型来处理，要对某一种子类型进行特殊处理，可以先判断真实类型，再向下转型

对真实类型，及其父类型判断，都返回true

Shape s = new Line();

s instanceof Line      true

s instanceof Shape      true

s instanceof Object     true

**练习1  多态测试**

day0702\_二维点三维点

day0703\_电子宠物

项目：day0901\_图形形状

类：day0901.Test1

             Shape

             Line

             Square

             Circle

Shape 图形形状类

{

   draw(),

   clear()

}

Line 线段类

{

   draw()

   length()

}

Square 方形类

{

   draw()

}

Circle圆形类

{

   draw()

}

Shape

**package** day0901;

**public** **abstract** **class** Shape {

/\*

 \* \*) 作为通用方法

 \* \*) 子类必须实现

 \*/

**public** **abstract** **void** draw();

**public** **void** clear() {

    System.***out***.println("\n\n\n\n\n\n\n\n\n\n\n\n\n\n");

    System.***out***.println("图形已被擦除");

    System.***out***.println("---------------------------");

}

}

Line

**package** day0901;

**public** **class** Line **extends** Shape{

@Override

**public** **void** draw() {

    System.***out***.println("一");

}

//线段自己的方法，与其他类无关

**public** **void** length() {

    System.***out***.println("十几厘米长...");

}

}

Square

**package** day0901;

**public** **class** Square **extends** Shape{

@Override

**public** **void** draw() {

    System.***out***.println("口");

}

}

Circle

**package** day0901;

**public** **class** Line **extends** Shape{

@Override

**public** **void** draw() {

    System.***out***.println("一");

}

//线段自己的方法，与其他类无关

**public** **void** length() {

    System.***out***.println("十几厘米长...");

}

}

Test1

**package** day0901;

**import** java.util.Random;

**import** java.util.Scanner;

**public** **class** Test1 {

**public** **static** **void** main(String[] args) {

    /\*

     \* Shape s = null;

     \*

     \* s = Shape实例

     \* s = Line实例

     \* s = Square实例

     \* s = Circle实例

     \*/

    System.***out***.println("按回车执行");

    Shape s = **null**;

**while**(**true**) {

**new** Scanner(System.***in***).nextLine();

**int** r = **new** Random().nextInt(4);

**switch**(r) {

**case** 0:

           System.***out***.println("Shape抽象类，不能创建实例");

**continue**;

**case** 1: s = **new** Line(); **break**;

**case** 2: s = **new** Square(); **break**;

**case** 3: s = **new** Circle(); **break**;

        }

*f*(s);

    }

}

/\*

 \* Shape

 \*   |- Line， length()

 \*   |- Square

 \*   |- Circle

 \*/

**private** **static** **void** f(Shape s) {

    System.***out***.println("---当做Shape父类型处理----------------");

    //获得真实类型的类名,

    //模仿Object中toString()方法的代码

    String n = s.getClass().getName();

    System.***out***.println("真实类型："+n);

    s.draw();

    //当做父类型，只能调用父类定义的通用成员，

    //子类特有成员被隐藏

    //s.length();

    //s引用的实例，真实类型是Line

**if** (s **instanceof** Line) {

        System.***out***.println(

         "向下转回成 Line 类型,再调用长度方法");

        Line line = (Line) s;

        line.length();

    }

**new** Scanner(System.***in***).nextLine();

    s.clear();

}

}

**2   什么是面向对象**

封装、继承、多态

  封装

  类

  对象、实例

  引用

  构造方法

  this

  this.xxx

  this(...)

  方法重载 Overload

  继承

  作用：代码重用、复用

  单继承

  子类对象

  父类对象和子类对象绑定，整体作为一个对象

  调用成员，先找子类，再找父类

  方法重写 Override

  super

  super.xxx()

  super()

  多态

  作用：一致的类型

  向上转型

  向下转型

  instanceof

**3   抽象类**

半成品类，没有写完的类

抽象方法，没有代码，只有方法定义

public abstract void f();

包含抽象方法的类，必须是抽象类

public abstract class A {

}

抽象类不能创建实例

抽象方法的作用：

  作为通用方法，在父类中定义

  要求子类，必须实现这个方法

**3.1     抽象类测试**

day0703\_电子宠物

Pet

day0901\_图形形状

Shape

**练习2  生成一个有序的数字数组**

按回车，随机生成数字，添加到数组，直接生成有序数组

[3,34,56,77,87]

项目：day0902\_随机生成有序数组

类：day0902.Test1

**package** day0902;

**import** java.math.BigDecimal;

**import** java.util.Arrays;

**import** java.util.Random;

**import** java.util.Scanner;

**public** **class** Test1 {

/\*

 \* 二分法查找、折半查找

 \* 在有序数组中查找一个值所在的位置下标，

 \* 如果找不到，返回 -(插入点+1)

 \*

 \* 29

 \*                             mid

 \* [2,3,3,5,6,8,12,13,13,13,17,28,30,32] 14

 \*                                lo

 \*                             hi

 \*  \*) lo和hi定位低位和高位下标

 \*  \*) 中间位置是 mid = (lo+hi)/2;

 \*  \*) mid位置大，hi定位到mid-1

 \*  \*) mid位置小，lo定位到mid+1

 \*  \*) mid位置找到相等的，mid下标就是结果

 \*  \*) 找不到，-(lo+1)，为什么+1，防止lo是0

 \*/

**public** **static** **void** main(String[] args) {

    /\*

     \* Number是抽象父类

     \*

     \* Number[]

     \* []

     \* [45]

     \* [23 45 760.7619368204128]

     \*  Double   Integer

     \*/

    System.***out***.println("按回车执行");

    Number[] a = **new** Number[0];

**while**(**true**) {

**new** Scanner(System.***in***).nextLine();

        Number n = **null**;

        //随机产生整数，或者是浮点数

        //一半机会产生整数 Integer

**if**(Math.*random*() < 0.5) {

           // 682 自动装箱 Integer

           // 再向上转型成父类型 Number

           n = **new** Random().nextInt(1000);

        } **else** {//一半机会产生浮点数 Double

           // 917.72345546 自动装箱 Double

           // 再向上转型成父类型 Number

**double** d = Math.*random*() \* 1000;

           //舍入运算，保留1位小数

           d = BigDecimal

                 .*valueOf*(d)

                 .setScale(1, BigDecimal.***ROUND\_HALF\_UP***)

                 .doubleValue();

           n = d;

        }

        //System.out.println(n);

        //二分法查找，在a数组中，查找n的下标位置

        //index是正数，就是插入点

        //index是负数（-(插入点+1)），用-index-1计算插入点

**int** index = *binarySearch*(a, n);

**if**(index<0) {

           index = -index-1;

        }

        /\* 5

         \* [2,3,3,5,6,8, ]

         \*        i

         \* 5

         \* [2,3,3,5,5,6,8]

         \*        i                         \*/

        a = Arrays.*copyOf*(a, a.length+1);

        //index是插入点，把插入点后面的数据，向后挪一位

        System.*arraycopy*(a,index,a,index+1,a.length-index-1);

        a[index] = n; //把数据放入插入点位置

        System.***out***.println(Arrays.*toString*(a));

    }

}

**private** **static** **int** binarySearch(

        Number[] a, Number n) {

**int** lo = 0;

**int** hi = a.length-1;

**int** mid;

    //lo和hi不能交叉，交叉就是找不到

**while**(lo<=hi) {

        //中间位置

        mid = (lo+hi)/2;

        //mid位置的值和 n 比较

**if** (a[mid].doubleValue() > n.doubleValue()) {

           hi = mid-1;

        } **else** **if**(a[mid].doubleValue() < n.doubleValue()) {

           lo = mid+1;

        } **else** {

**return** mid;

        }

    }

**return** -(lo+1);

}

}

**4   BigDecimal，BigInteger**

BigDecimal精确的浮点数运算

BigInteger超大整数运算

把运算数，先都封装成 BigDecimal 对象，再使用 BigDecimal 提供的计算方法来计算

创建 BigDecimal 对象

BigDecimal bd = BigDecimal.valueOf(2);

方法

  add(BigDecimal bd)

  subtract(BigDecimal bd)

  multiply(BigDecimal bd)

  divide(BigDecimal bd)

  divide(BigDecimal bd,保留位数,舍入方式)

  setScale(保留位数,舍入方式)

**练习3  BigDecimal**

项目：day0903\_BigDecimal

类：day0903.Test1

**package** day0903;

**import** java.math.BigDecimal;

**import** java.util.Scanner;

**public** **class** Test1 {

**public** **static** **void** main(String[] args) {

    System.***out***.println("输入两个浮点数");

**double** a = **new** Scanner(System.***in***).nextDouble();

**double** b = **new** Scanner(System.***in***).nextDouble();

    System.***out***.println(a+b);

    System.***out***.println(a-b);

    System.***out***.println(a\*b);

    System.***out***.println(a/b);

    /\*

     \* 2               4.35

     \* 1.9             100

     \*/

    System.***out***.println("------------------------");

    BigDecimal bd1 = BigDecimal.*valueOf*(a);

    BigDecimal bd2 = BigDecimal.*valueOf*(b);

    BigDecimal bd3;//用来保存计算结果

    bd3 = bd1.add(bd2);

    System.***out***.println(bd3.doubleValue());

    bd3 = bd1.subtract(bd2);

    System.***out***.println(bd3.doubleValue());

    bd3 = bd1.multiply(bd2);

    System.***out***.println(bd3.doubleValue());

    bd3 = bd1.divide(

        bd2, 50, BigDecimal.***ROUND\_HALF\_UP***);

    System.***out***.println(bd3.doubleValue());

    System.***out***.println(bd3.toString());

}

}

阶乘

**package** day0306;

**import** java.math.BigInteger;

**import** java.util.Scanner;

**public** **class** Test1 {

**public** **static** **void** main(String[] args) {

    System.***out***.print("输入整数，求阶乘：");

**int** n = **new** Scanner(System.***in***).nextInt();

    //把n变量的值取出来，传递到f()方法

    String r = *f*(n);

    System.***out***.println(r);

}

/\*

 \* r = n

 \*

 \* i

 \* n-1, r=r\*i

 \* 3, r=r\*i

 \* 2, r=r\*i

 \* 1, r=r\*i

 \*/

**private** **static** String f(**int** n) {

    /\*//如果参数n的值是0，直接返回1

    if(n == 0) {

        return 1;

    }

    long r = n;

    //i循环从n-1到1

    for (int i = n-1; i >= 1; i--) {

        r=r\*i;

    }

    //向调用位置返回计算结果

    return r;\*/

**if**(n == 0) {

**return** "1";

    }

    BigInteger r = BigInteger.*valueOf*(n);

**for** (**int** i = n-1; i >= 1; i--) {

        r = r.multiply(BigInteger.*valueOf*(i));

    }

**return** r.toString();

}

}

**5   位运算符**

+-\*/% == != > >= < <= && || ! ++ -- ? : = += () instanceof

|  |  |
| --- | --- |
| & | 位与，两位都是1，结果才是1 |
| | | 位或，任意一位是1，结果就是1 |
| ^ | 异或，不同是1  对同一个值异或两次得到原值 |
| ~ | 求反，1变0,0变1 |
| >> | 带符号右移位  符号位是1，左边补1 |
| >>> | 不带符号右移位  左边都补0 |
| << | 左移位 |

**练习4  位运算**

项目：day0904\_位运算

类：day0904.Test1

**package** day0904;

**import** java.util.Scanner;

**public** **class** Test1 {

**public** **static** **void** main(String[] args) {

    /\*

     \* 746345

     \* 00101010 11111001 10111010 10111011

     \*        <<--------24-----------

     \*        ---------24---------->>>

     \* 00000000 00000000 00000000 10111011

     \*/

    System.***out***.print("输入整数：");

**int** n = **new** Scanner(System.***in***).nextInt();

    n = n<<24>>>24;

    System.***out***.println(n);

    /\*

     \* 00101010 11111001 10111010 10111011

     \* 00000000 00000000 00000000 11111111 &

     \* --------------------------------------

     \* 00000000 00000000 00000000 10111011

     \*/

    System.***out***.print("输入整数：");

    n = **new** Scanner(System.***in***).nextInt();

    n = n & 0x000000ff;

    System.***out***.println(n);

}

}

int的四个字节值，拆分成4个byte值

00101010 11111001 10111010 10111011

00101010

11111001

10111010

10111011

Test2

**package** day0904;

**import** java.util.Arrays;

**import** java.util.Scanner;

**public** **class** Test2 {

**public** **static** **void** main(String[] args) {

    System.***out***.print("整数：");

**int** n = **new** Scanner(System.***in***).nextInt();

**byte**[] a = *f1*(n);

    System.***out***.println(Arrays.*toString*(a));

**int** r = *f2*(a);

    System.***out***.println(r);

}

**private** **static** **int** f2(**byte**[] a) {

    /\*

     \* 00101010

     \* 11111001

     \* 10111010

     \* 10111011

     \*

     \* r = 00101010 11111001 10111010 00000000

     \*

     \* 每一个字节值先和0x000000ff求位与，

     \* 再左移24,16,8,0位

     \* 再和r求位或，结果保存到r

     \*/

**int** r = 0;

**for** (**int** i = 0; i < a.length; i++) {

        r = r | ((a[i]&0x000000ff)<<(24-8\*i));

    }

**return** r;

}

**private** **static** **byte**[] f1(**int** n) {

    /\*

     \* 00101010 11111001 10111010 10111011

     \*

     \* n右移24,16,8,0，再强转byte

     \*    i 0  1  2 3

     \*/

**byte**[] a = **new** **byte**[4];

**for** (**int** i = 0; i < a.length; i++) {

        a[i] = (**byte**) (n>>(24-8\*i));

    }

**return** a;

}

}

作业

**package** day0905;

**import** java.util.Scanner;

**public** **class** Test1 {

**public** **static** **void** main(String[] args) {

    System.***out***.println("email:");

    String e = **new** Scanner(System.***in***).nextLine();

    String name = *getName*(e);

    System.***out***.println(name);

    System.***out***.println("对称：");

    String s = **new** Scanner(System.***in***).nextLine();

**if** (*duiChen*(s)) {

        System.***out***.println("对称");

    } **else** {

        System.***out***.println("不对称");

    }

}

**private** **static** String getName(String e) {

    /\*

     \* "abc@def.com"

     \*     |

     \*/

**int** index = e.indexOf("@");

**if** (index == -1) {

**return** "格式错误";

    }

**return** e.substring(0, index);

}

**private** **static** **boolean** duiChen(String s) {

    /\*

     \*        j

     \* abcdedcba

     \*  i

     \*/

**for** (**int** i = 0,j=s.length()-1; i<j; i++,j--) {

        //i和j位置字符不相等

**if**(s.charAt(i) != s.charAt(j)) {

**return** **false**;

        }

    }

**return** **true**;

}

}

**6   作业**

  重写

  day0307\_99乘法表

  day0601\_士兵

  day0602\_今晚吃鸡

  day0604\_递归求阶乘

  day0704\_冒泡排序

  day0901\_图形形状

  day0902\_随机生成有序数组

  有序数组的并归

  两个有序数组，合并成一个有序数组

[1,2,2,4,5]

            j

[2,2,4,5,7,7,8,9]

          k

[1 2 2 2 2 4 4 5 5                  ]

                     i

  j和k的较小值放入i位置，对应下标递增

  j越界，复制k后面的值

  k越界，复制j后面的值