Java核心API (包装类,BigDecimal,BigInteger)

# 包装类

因为java中有8个基本类型数据,他们不是以对象的形式存在,他们自然也不是Object的子类.那么在参与面向对象开发时会遇到很多阻碍,为此,java提供了针对8中基本类型数据的包装类使之可以将基本类型转化为以对象形式存在,参与面向对象开发 。Java语言的8种基本类型分别对应了8种"包装类"。每一种包装类都封装了一个对应的基本类型成员变量，同时还提供了针对该数据类型的实用方法，包装类使用前提：JDK1.5+。

|  |  |
| --- | --- |
| 基本类型 | 包装类 |
| int | Integer |
| char | Character |
| byte | Byte |
| short | Short |
| long | Long |
| float | Float |
| double | Double |
| boolean | Boolean |

包装类的目的：用于将基本类型数据当作引用类型看待。

包装类的名字：除了Integer(int)，Character(char)外，其余包装类名字都是基本类型名首字母大写。

## 装箱：基本类型转为对象类型

方式一：Double d=new Double(2.2);//装箱

方式二：Double d=Double.valueOf(2.2);//基本类型对应的包装类都有valueOf方法

## 拆箱：对象类型转为基本类型

double num=d.doubleValue();//拆箱

JDK1.5包装类自动拆装箱（原理）：在编译源程序的时候，编译器会预处理，将未作拆箱和装箱工作的语句自动拆箱和装箱。

Integer i = 10; //自动装箱，基本类型直接赋值给对象，实际已经执行了 Integer i = Integer.valueOf(10);

int num = i; //自动拆箱，对象类型直接赋值给基本类型，实际上执行了 int num = i.intValue();

注意：自动包装类的比较应该使用equal()方法。自动包装箱规范要求boolean,byte,char<=127,介于-128-127之间的short和int被包装到固定的对象中。所以Integer a = 100; Integer b = 100;//a==b一定成立

## 包装类的其他常用方法

包装类的一些常用功能：将字符串转换为其类型，方法是：parseXXX，XXX代表其类型。这里要特别注意！一定要保证待转换的字符串描述的确实是或者兼容要转换的数据类型！否则会抛出异常！

|  |
| --- |
| String numStr="123";  System.out.println(numStr+1);//1231  int num=Integer.parseInt(numStr);  System.out.println(num+1)//124  long longNum=Long.parseLong(numStr);  System.out.println(longNum);//123  double doubleNum=Double.parseDouble(numStr);  System.out.println(doubleNum);//123.0 |

Integer提供了几个方法：将一个整数转换为16进制的形式，并以字符串返回；将一个整数转换为2进制的形式，并以字符串返回。

|  |
| --- |
| String bStr=Integer.toBinaryString(num);  String hStr=Integer.toHexString(num); |

所有包装类都有几个共同的常：获取最大、最小值。

|  |
| --- |
| int max=Integer.MAX\_VALUE;//int最大值  int min=Integer.MIN\_VALUE;//int最小值  System.out.println(Integer.toBinaryString(max));  System.out.println(Integer.toBinaryString(min)); |

# BigDecimal类

表示精度更高的浮点型，在java.math.BigDecimal包下，该类可以进行更高精度的浮点运算。需要注意的是，BigDecimal可以描述比Double还要高的精度，所以在转换为基本类型时，可能会丢失精度！

## BigDecimal的使用

|  |
| --- |
| 创建一个BigDecimal实例，可以使用构造方法  BigDecimal（String numberFormatString），  用字符串描述一个浮点数作为参数传入。  BigDecimal num1=new BigDecimal("3.0");  BigDecimal num2=new BigDecimal("2.9");  //运算结果依然为BigDecimal表示的结果  BigDecimal result=num1.subtract(num2);//num1-num2 System.out.println(result);  float f=result.floatValue();//将输出结果转换为基本类型float  int i=result.intValue();//将输出结果转换为基本类型int |

2）BigDecimal可以作加add、减subtract、乘multiply、除divide等运算：这里需要注意除法，由于除法存在结果为无限不循环小数，所以对于除法而言，我们要制定取舍模式，否则会一直计算下去，直到报错（内存溢出）。

result=num1.divide(num2, 8, BigDecimal.ROUND\_HALF\_UP); // 小数保留8位，舍去方式为四舍五入

|  |
| --- |
| BigDecimal bigDecimal = new BigDecimal("1.0");  BigDecimal b2 = new BigDecimal("2.0");  System.out.println(bigDecimal.doubleValue());  System.out.println(bigDecimal.intValue());  System.out.println(bigDecimal.add(b2));  System.out.println(bigDecimal.subtract(b2));  System.out.println(bigDecimal.multiply(b2));  System.out.println(bigDecimal.divide(b2)); |

# BigInteger类

使用描述更长位数的整数"字符串"，来表示、保存更长位数的整数，在java.math.BigInteger包下。

## BigInteger的使用

|  |
| --- |
| 1) 创建BigInteger  BigInteger num=new BigInteger("1");  /num=new BigInteger(1); // 不可以，没有这样的构造器  //这种方式我们可以将一个整数的基本类型转换为BigInteger的实例  num=BigInteger.valueOf(1);  2）理论上：BigInteger存放的整数位数只受内存容量影响。  3）BigInteger同样支持加add、减subtract、乘multiply、除divide等运算。  BigInteger b1 = BigInteger.valueOf(1);  BigInteger b2 = new BigInteger("2");  System.out.println(b1.add(b2).intValue());  System.out.println(b1.subtract(b2));  System.out.println(b1.multiply(b2));  System.out.println(b1.divide(b2)); |