数字（Number）

@M了个J
李明杰

https://github.com/CoderMJLee
http://cnblogs.com/mjios

码拉松

实力IT教育 www.520it.com

基本类型的缺陷

- 对比引用类型，基本类型存在的一些缺陷
  - 无法表示不存在的值（null 值）
  - 不能利用面向对象的方式去操作基本类型（比如直接用基本类型调用方法）
  - 当方法参数是引用类型时，基本类型无法传递

- 解决方案：可以自己将基本类型包装成引用类型

```
public class IntObject {
    public int value;
    public IntObject(int value) {
        this.value = value;
    }
}
```

```
IntObject[] data = {
    new IntObject(-100), new IntObject(100),
    null, new IntObject(0)
};
for (IntObject intObject : data) {
    if (intObject == null) {
        System.out.println("没有值");
    } else {
        System.out.println(intObject.value);
    }
}
```

包装类（Wrapper Class）

- 其实 Java 中已经内置了基本类型的包装类（都在 java.lang 包中）

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

- 数字类型的包装类（Byte\Short\Integer\Long\Float\Double）最终都继承自 java.lang.Number

自动装箱、拆箱（Autoboxing and Unboxing）

- 自动装箱：Java 编译器会自动将基本类型转换为包装类（调用 valueOf 方法）
- 自动拆箱：Java 编译器会自动将包装类转换为基本类型（调用 xxxValue 方法）

```java
Integer i1 = 10;
// Integer i1 = Integer.valueOf(10);

void add(Integer num) {}
add(20);
// add(Integer.valueOf(20));
```

```java
// 下面这句代码也是正确的
Object num = 10;
```

```java
Integer i1 = 10;
int i2 = i1;
// int i2 = i1.intValue();

System.out.println(i1 == 10);
// System.out.println(i1.intValue() == 10);

Integer[] array = { 11, 22, 33, 44 };
int result = 0;
for (Integer i : array) {
    // i.intValue() % 2 == 0
    if (i % 2 == 0) {
        // result += i.intValue();
        result += i;
    }
}
```

包装类的判等

- 包装类的判等，不要使用 ==、!= 运算符，应该使用 equals 方法

```java
Integer i1 = 88;
Integer i2 = 88;
Integer i3 = 888;
Integer i4 = 888;
// 不推荐
System.out.println(i1 == i2); // true
System.out.println(i3 == i4); // false
// 推荐
System.out.println(i1.equals(i2)); // true
System.out.println(i3.equals(i4)); // true
```

```java
Integer i1 = 88;
Integer i2 = Integer.valueOf(88);
Integer i3 = new Integer(88);
System.out.println(i1 == i2); // true
System.out.println(i1 == i3); // false
```

- IntegerCache 类中缓存了 [-128, 127] 范围的 Integer 对象
- Integer.valueOf 方法会优先去 IntegerCache 缓存中获取 Integer 对象

小码哥教育 SEEMYGO

使用注意

- 【基本类型数组】与【包装类数组】之间是不能自动装箱、拆箱的

```
public static void test1(Integer[] nums) {}
public static void test2(int[] nums) {}

int[] nums1 = { 11, 22 };
test1(nums1); // error
Integer[] nums2 = nums1; // error

Integer[] nums3 = { 11, 22 };
test2(nums3); // error
int[] nums4 = nums3; // error
```

Math

- java.lang.Math 类提供了常见的数学计算功能

```
// 自然常数，自然对数函数的底数
public static final double E = 2.7182818284590452354;
// 圆周率
public static final double PI = 3.14159265358979323846;
```

```
Math.abs(-100) // 求绝对值：100
Math.max(100, 200) // 求最大值：200
Math.min(100, 200) // 求最小值：100
Math.floor(3.9) // 向下取整：3.0
Math.ceil(3.1) // 向上取整：4.0
Math.round(3.5) // 四舍五入：4
Math.pow(4, 2) // 4的2次方：16.0
Math.sqrt(16) // 16的平方根：4.0
```

```
Math.exp(2) // E的2次方
// 求ln8的值，以E为底数、8为真数的对数
Math.log(8)
// 角度转为弧度
double degree = 90;
double radian = Math.toRadians(degree);
// 三角函数
Math.sin(radian)
Math.cos(radian)
Math.tan(radian)
```

```
Math.random() // 生成[0.0, 1.0)范围的随机数
```

Random

- java.util.Random 可以更方便地生成各种随机数

```
// 生成各种随机值
Random r = new Random();
r.nextBoolean()
r.nextInt()
r.nextLong()
r.nextFloat()
```

```
// 生成[0, 99]范围的整数
int num1 = (int) (Math.random() * 100);
int num2 = new Random().nextInt(100);

// 生成[10, 99]范围的整数
int num3 = (int) (Math.random() * 90) + 10;
int num4 = new Random().nextInt(90) + 10;
```

```
// 输出4位的大写字母验证码
Random r = new Random();
for (int i = 0; i < 4; i++) {
char c = (char) (r.nextInt(26) + 'A');
System.out.print(c);
}
System.out.println();
// DWKV
```

小码哥教育 SEEMYGO

UUID

- UUID（Universally Unique Identifier），通用唯一标识符
- UUID 的目的是让分布式系统中的所有元素都能有唯一的标识符，而不需要通过中央控制端来做标识符的指定

- 可以利用 java.util.UUID 类的 randomUUID 方法生成一个 128 bit（32 位 16 进制数）的随机 UUID

```
// 964cb88d-a383-4705-99d2-72cc90a493f5
System.out.println(UUID.randomUUID());
```

数字格式化

- 可以使用 System.out.printf 或者 System.out.format 输出格式化的字符串
- 可以使用 String.format 创建格式化的字符串

转换符	作用
d	十进制整数
f	浮点数
n	换行，跟 \n 效果一样

标记	作用
08	8个字符的宽度，前面用 0 补齐
+	显示符号（正数+，负数–）
,	显示分组字符（本地化）
–	左对齐
.3	保留 3 位小数
10.3	10 个字符的宽度，保留 3 位小数

小码哥教育 SEEMYGO

数字格式化

```
long n = 461012;
System.out.format("%d%n", n);        //  "461012"
System.out.format("%08d%n", n);      //  "00461012"
System.out.format("%+8d%n", n);      //  " +461012"
System.out.format("%,8d%n", n);      // " 461,012"
System.out.format("%+,8d%n%n", n); //  "+461,012"

double pi = Math.PI;
System.out.format("%f%n", pi);         // "3.141593"
System.out.format("%.3f%n", pi);       // "3.142"
System.out.format("%8.3f%n", pi);    // "   3.142"
System.out.format("%08.3f%n", pi);     // "0003.142"
System.out.format("%-8.3f%n", pi);     // "3.142"
```

```
String str = String.format("The PI is %.2f", Math.PI);
// The PI is 3.14
System.out.println(str);
```

小码哥教育 SEEMYGO

DecimalFormat

- 使用 java.text.DecimalFormat 可以更好地控制前 0、后 0、前缀、后缀、分组分隔符、十进制分隔符等

```
void customFormat(String pattern, double value) {
    DecimalFormat fmt = new DecimalFormat(pattern);
    System.out.println(fmt.format(value));
}
```

```
// 123,456.789
customFormat("###,###.###", 123456.789);
// 123456.79
customFormat("###.##", 123456.789);
// 000123.780
customFormat("000000.000", 123.78);
// $12,345.67
customFormat("$###,###.###", 12345.67);
```

字符串转数字

- 使用包装类的 valueOf、parseXX 方法

```
// 12
Integer i1 = Integer.valueOf("12");
// 12
int i2 = Integer.parseInt("12");
// 255（十六进制解析FF）
int i3 = Integer.parseInt("FF", 16);

// 12.34
Float f1 = Float.valueOf("12.34");
// 12.34
float f2 = Float.parseFloat("12.34");
```

数字转字符串

- 使用字符串的 valueOf 方法、包装类的 toString 方法

```
// 12.34
String str1 = String.valueOf(12.34);
// 255
String str2 = Integer.toString(255);
// ff
String str3 = Integer.toString(255, 16);
// 12.34
String str4 = Float.toString(12.34f);
```

小码哥教育 SEEMYGO

高精度计算

- float、double 存储的只是小数的近似值，并非精确值。因此不适合用来进行高精度计算

```
double d1 = 0.7;
double d2 = 0.7;
// 0.48999999999999994
System.out.println(d1 * d2);
```

```
0.7 = 0b0.101100110...
0.7 * 2 = 1.4 取出整数部分1
0.4 * 2 = 0.8 取出整数部分0
0.8 * 2 = 1.6 取出整数部分1
0.6 * 2 = 1.2 取出整数部分1
0.2 * 2 = 0.4 取出整数部分0
0.4 * 2 = 0.8 取出整数部分0
0.8 * 2 = 1.6 取出整数部分1
0.6 * 2 = 1.2 取出整数部分1
0.2 * 2 = 0.4 取出整数部分0
...
```

- 建议使用 java.math.BigDecimal 来进行高精度计算

小码哥教育 SEEMYGO

BigDecimal

- 建议使用字符串初始化 BigDecimal，因为 float、double 存储的是近似值，不是精确值

```
BigDecimal v1 = new BigDecimal("0.7");
BigDecimal v2 = new BigDecimal("0.7");
System.out.println(v1.add(v2)); // 加，1.4
System.out.println(v1.subtract(v2)); // 减，0.0
System.out.println(v1.multiply(v2)); // 乘，0.49
System.out.println(v1.divide(v2)); // 除，1
System.out.println(v1.setScale(3)); // 保留3位小数，0.700
```