# 1数据类型

## 1.1基本数据类型

byte short int long char float double boolean

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 基本类型 | 默认值 | 存储 | 封装类 | 数据范围 |
| byte | 0 | 1字节 | Byte | -128~127 |
| short | 0 | 2字节 | Shrot | -32768~32767 |
| Int | 0 | 4字节 | Integer | -21亿~21亿 |
| long | 0L(0l) | 8字节 | Long | -2^63~2^63-1 |
| char | ‘\u0000’ | 2字节 | Character | 0~65535 |
| float | 0.0F(0.0f) | 4字节 | Float | 32位EEE754单精度 |
| double | 0.0(0.0d()D) | 8字节 | Double | 64位EEE754双精度 |
| boolean | false | 1位 | Boolean | true/false |

## 1.2引用数据类型

接口 类 数组 String等

## 1.3数据类型转换

通常在进行整形数据处理时，会存在数据溢出，即数据过大或太小产生的错误数据的情况。这时候便需要进行数据类型转换。

描述小数默认类型为double类型，需要定义为float类型的时候需要在数字后加上f或者F如：1.2f。

1. 范围⼩的数据类型可以⾃动变为数据范围⼤的数据类型（在数学计算时）

2. 数据范围⼤的数据类型只有强制转换才能转为数据类型⼩的数据类型。

然后是字符类型，字符类型可以和int型的数据相互转换，但需要注意char类型中的‘0’~‘9’和数字0~9所代表的范围并不同。

最后是布尔类型，在java中的布尔类型与其他语言不同，在其他语言中用0代表false!0表示true，但是在java中并没有这种表达方式，中能用true和false表示。

String作为一种引用类型，其实是一个类，在后面的总结中会详细介绍。

# 2运算符

## 2.1基本运算符

+ - \* / %

这里需要注意的是前置还是后置的问题，后置先运算，后进行自增自减。前置相反。

### 2.2三目运算符

（运算符，结果为boolean类型）？结果1 : 结果2;

例如: 1 > 2 ? x- y: y - x;

等价于

If(1>2){

x-y;

}else{

y-x;

}

### 2.3关系运算符

> < = <= >= == :返回类型均为boolean类型（注意在比较引用类型的时候不可以用==，而应该使用.equals方法，这个在后面会详细介绍）

### 2.4逻辑运算符

(&和&&为与) (|和||为或) (!为非)

其中|| 和 &&代表短路或和短路与，在进行逻辑判断的时候都运用短路与和或。

### 2.5位运算符

& | ~ ^分别代表按位与，按位或，按位取反，以及按位异或

&:同为1才为1，其余为0

|：有一个1就为1，反之为0

~：1变为0，0变为1

^: 相同为0，不同为1

# 3 Java中的逻辑控制

## 3.1 分支结构

If(布尔表达式){

true：表达式1;

}else{

false：表达式2;

}

可以一个if对应多个else代表不同分支，也支持嵌套。

switch(数字|枚举|字符|字符串){

case 内容1 : {

内容满⾜时执⾏语句;

[break;]

} case 内容2 : {

内容满⾜时执⾏语句;

[break;]

} ... default:{

内容都不满⾜时执⾏语句;

[break;]

}

}

需要注意，break通常不能省略。

## 3.2 循环结构

**while循环**

white（循环条件）{

循环体；

更新循环变量；

}

**do...while()循环**

do{

循环体；

更新循环变量；

}while(循环条件)

**for()循环**

for（循环变量初始值;循环终止条件;循环标量更新）{

循环体;

}

## 3.3 循环控制

break 直接结束循环循环。

continue 跳出本次循环，进行下一次循环条件判断。

# 4 方法的定义和使用

## 4.1 方法的定义

方法是一段可以被重复调用代码，格式如下：

访问权限修饰符 (static) (final)返回值类型 方法名（参数列表）{

方法体代码;

return 返回值;

}

## 4.2 方法的重载

在同一个类中方法名相同，参数列表不同（顺序或个数）的方法，需要注意，返回值类型不做要求，也就是说，不能存在只有返回值类型不同的两个方法。

## 4.3 方法的递归

递归就是方法自己调用自己，需要有一个递归的结束条件，而且每次递归的参数需要做一定的调整，保证能越来越趋近于结束条件。所有的分治思想，都可以采用递归的方式去实现。

## 4.4 方法的调用

一般的方法都是需要通过实例化对象来调用的，但是加了static修饰的方法，不依赖于对象，可以直接通过类名调用。具体的会在面向对象的时候详细介绍。

# 5 面向对象