2. Java编程基础

2.1. 变量与标识符

2.1.1. 变量分类

- 按所属的数据类型划分:
  - 基本数据类型变量
  - 引用数据类型变量
- 按被声明的位置划分:
  - 局部变量: 方法或语句块内部定义的变量
  - 成员变量: 方法外部、类的内部定义的变量

2.1.2. 标识符

- Java对包、类、方法、参数和变量等要素命名时使用的字符序列

- 命名规则：
  - 由字母、数字、下划线(_)和美元符号($)组成
  - 不能以数字开头
  - 区分大小写
  - 长度无限制
  - 不能是Java中的关键字和保留关键字
- 标识符命名习惯：驼峰命名法、见名知意。以单词或者单词组合命名

2.2. 八种基本数据类型

- byte：字节型，占用1个字节，取值范围-128~127，默认值：0
- short：短整型，占用2个字节，取值范围-32768~32767，默认值：0
- int：整型，占用4个字节，默认值：0
- long：长整型，占用8个字节，默认值：0
- char：字符型，占用2个字节，取值范围0~65535，默认值：'\u0000'
- float：单精度浮点型，默认值：0.0F
- double：双精度浮点型，默认值：0.0D
- boolean：布尔型，取值范围仅仅只有true，false，默认值为：false

2.3. 基本数据类型的声明

2.3.1. 声明变量的格式

- 声明变量：类型 变量名;
- 声明的同时给变量赋值：类型 变量名 = 值;
- 给已声明过的变量赋值：变量名 = 值;

2.4. 进制与转换

2.4.1. 进制

- 十进制：日常使用的进制
- 二进制：计算机中所有数据都是以二进制进行保存的，逢二进一
  - 计算机中信息的存储单位：
    - 位(Bit)：表示一个二进制数码0或1，是计算机存储处理信息的最基本单位。
    - 字节(Byte)：一个字节由8个位组成，它表示作为一个完整处理单位的8个二进制数码
- 八进制：可以用3个二进制位表示一位8进制数
- 十六进制：可以用4个二进制位表示一位16进制数，一般以0x或0X开头

2.4.2. 补码

- 计算机内的二进制数值是以补码的形式进行表示的
- 正数：正数的补码和其原码的形式是相同的
- 负数：将改数的绝对值的二进制形式，按位取反再加一
- 二进制补码数值的最高位（最左位）是符号位：该位为0，表示数值为正数；改位为1，表示数值为负数
- 使用补码的原因：使用补码，可以将符号位和其他位统一处理；同时减法也可以按加法来处理；另外，两个用补码表示的数相加时，如果最高位（符号位）有进位，则该进位被舍弃

2.4.3. 进制转换

- 二进制数转换成十进制数：
  - 按权相加：把二进制数首先写成加权系数展开式，然后按十进制加法规则求和
  - 原码1011.01 = $1\times2^3+0\times2^2+1\times2^1+1\times2^0+0\times2^{-1}+1\times2^{-2}$ = 8+0+2+1+0+0.25 = 11.25
- 十进制转换成二进制：
  - 整数部分：“除2取余，逆序排列”法
  - 小数部分：“乘2取整，顺序排列”法

2.5. 基本数据类型转换

2.5.1. 自动转换

- byte、short、int：这三者在计算时会自动转换成int
- 如果把int值转换为float值，或者long转换为double值，不需要强制转换，但是可能会丢失精度
- byte-->short-->int-->long和char-->int和float-->double

2.5.2. 强制转换

- 把大的容量类型转换为小的容量类型，需要添加强制类型转换
- 有可能会造成精度降低或数据溢出，使用时要小心
- boolean类型不能转换成任何其他数据类型

2.6. 关键字

abstract	assert	boolean	break	byte
case	catch	char	class	const
continue	default	do	double	else
enum	extends	final	finally	float
for	goto	if	implements	import
instanceof	int	interface	long	native

abstract	assert	boolean	break	byte
new	package	private	protected	public
return	strictfp	short	static	super
switch	synchronized	this	throw	throws
transient	try	void	volatile	while

- 保留关键字：goto和const

2.7. 转义字符

\n	换行(LF)，将当前位置移到下一行开头
\r	回车(CR)，将当前位置移到本行开头
\t	水平制表符(HT)，（跳到下一个TAB位置）
\\	代表一个反斜线字符“\”
\'	代表一个单引号（撇号）字符
\"	代表一个双引号字符

2.8. 运算符与优先级

2.8.1. 算数运算符

- 运算顺序从左到右

运算符	描述	示例	结果
+	加法	5+5	10
-	减法	5-3	2
*	乘法	2*3	6
/	除法	10/3	3
%	取余（取模）	10%3	1
++	自增（前，后）		
--	自减（前，后）		

- 表达式：由变量、常量运算符组成的式子
- ++：如果是前缀，先对此变量加1，再执行其他的操作；如果是后缀，先执行其他的操作，再对此变量加1

2.8.2. 键盘输入语句

- `Scanner input = new Scanner(System.in)`
- 从键盘接收输入一个整数：int x = input.nextInt();

2.8.3. 赋值运算符

- 将一个值赋给一个变量，运算顺序从右到左

运算符	描述	示例	结果
=	赋值	a = 10	a = 10
+=	加等于	a = 1, a += 3	a = 4
-=	减等于	a = 1, a -=3	a = -2
*=	乘等于	a = 1, a *= 3	a = 3
/=	除等于	a = 10, a /= 3	a = 3
%=	模等于	a = 10, a %= 3	a = 1

2.8.4. 关系运算符

- 比较两边的操作，结果总是boolean类型的

运算符	描述	示例	结果
==	相等于	a = 1, a == 10	false
!=	不等于	a = 1, a != 3	true
<	小于	a = 1, a < 3	true
>	大于	a = 1, a > 3	false
<=	小于等于	a = 10, a <= 3	false
>=	大于等于	a = 10, a >= 3	true

2.8.5. 逻辑运算符

- 对boolean型结果的表达式进行运算，运算结果总是boolean类型

运算符	描述	示例	结果
&	与	false & true	false
|	或	false | true	true
^	异或	true ^ false	true
!	非	! true	false
&&	短路与	false && true	false
||	短路或	false || true	true

- 在计算机中，非0即真，0为假

2.8.6. 位运算符

- 对两个操作数中的每一个二进制位都进行运算
- 功能：

- 按位取反：~
- 按位与：&
- 按位或：|
- 按位异或：^

2.8.7. 位移运算符

- 左移："a<<b;"将二进制形式的a逐位左移b位，最低位空出的b位补0
- 带符号右移："a>>b;"将二进制形式的a逐位右移b位，最高位空出的b位补原来的符号位
- 无符号右移："a>>>b;"将二进制形式的a逐位右移b位，最高位空出的b位补0

2.8.8. 三目运算符：X ? Y : Z

- X为boolean类型表达式，先计算X的值，若为true，整个三目运算符的结果为表达式Y的值，否则整个运算结果为表达式Z的值

2.9. 分支语句

2.9.1. if...else...语句

```
if (表达式) {
    执行语句;
} else {
    执行语句;
}
```

- 注意：表达式是一个逻辑表达式

```
if (表达式1) {
    执行语句1;
} else if (表达式2) {
    执行语句2;
} else if (表达式3) {
    执行语句3;
} else {
    执行语句4;
}
```

- 注意：if...else...可以进行嵌套，if可以单独存在，但是else可以单独存在

2.9.2. switch分支语句

```
switch (表达式) {
    case 取值1:
        语句块1;
        break;
    case 取值2:
        语句块2;
        break;
```

```
    case 取值n:
        语句块n;
        break;
    default:
        语句块n+1;
        break;
}
```

- switch语句与if语句嵌套：

```
switch (表达式) {
    case 取值n:
        语句块n;
        if (表达式) {
            ...
        }
        break;
    default:
        语句块n+1;
        break;
}
```

  - 注意：
    - 表达式的返回值必须是下述几种类型之一：int，byte，char，short，确定的结果，jdk1.7后支持String
    - case子句中的取值必须是常量，且所有case子句中的取值应是不同的
    - default子句是可选的
    - break语句用来在执行完一个case分支后使程序跳出switch语句块；如果case后面没有写break则直接往下面执行
    - case后面的执行体可以写{}也可以不写{}

2.9.3. 分支语句比较

- if和switch都是选择分支语句：
  - 语法不同
  - 语句后表达式值的类型不同
    - if：语句表达式值得类型是boolean
    - switch：分支表达式的值的类型是byte，short，char，int，String和枚举
  - 适用场景
    - if：变量的值在某个区间之内
    - switch：变量的值是某个定值

2.10. 循环语句

2.10.1 while循环

```
while (条件表达式) {
    // 语句块;
}
```

- 符合条件循环继续执行，否则循环退出
- 特点：先判断，再执行

2.10.2. do...while...循环

```
do {
    // 循环操作;
} while (循环条件);
```

- 先执行一遍循环操作，符合条件循环继续执行，否则循环退出
- 特点：先执行，再判断

2.10.3. for循环

```
for (初始化参数(表达式1); 判断条件(表达式2); 更新循环变量(表达式3)) {
    循环体;
}
```

- 关键字：continue表示跳过当次循环，继续下次循环
- for循环的六种写法：
    - 标准写法
    - 表达式1省略，但在外部声明
    - 表达式2省略，死循环
    - 表达式3省略，但是要声明在循环内部
    - 表达式1,3省略，但是要声明在循环在外部和内部
    - 三个表达式都省略，死循环