**Android开发先从学习Java开始**

目录

**[☞Java语言的历史 2](#_Toc460572507)**

**[☞Java语言特性 2](#_Toc460572508)**

**[☞\*Java基本语法入门 2](#_Toc460572509)**

**[☞\*Java流程控制 6](#_Toc460572510)**

**[☞\*Java数组 12](#_Toc460572511)**

**[☞Java字符串 14](#_Toc460572512)**

**☞Java语言的历史**

最早由Sun公司在1990年开发，1995年JDK（Java Development Kit）1.0版本正式发布。09年Sun被Oracle收购。由于Java独立于平台，它拥有无可比拟的能力，可以开发出很多大型软件。像Android app的开发也是基于Java语言。

Java开发分为三个方向：桌面开发，web开发，嵌入式系统开发

**☞Java语言特性**

支持面向对象，语法简单，分布性，可移植性，解释性，安全性，健壮性，支持多线程

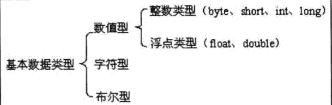
**☞\*Java基本语法入门**

**①标识符**

Java程序中必须使用的，但不能随便使用，有一定的规则。它可以简单理解为一个名字，用来**标识一些内容**或**一些属性**

e.g.int i = 100; i就是标识符，由程序员所起，但有一定规则：标识符是由字母、数字、下划线和美元符号组成，第一个字符不能是数字。非法的标识符如7bus、4word，合法的标识符，合法的标识符如s\_b、\_pp。

**②关键字**

 是指Java语言中已经被赋予特定意义的一些单词，不可以把这些字作为标识符来使用。Java中共有47个关键字。

**③基本数据类型**

|  |  |  |  |
| --- | --- | --- | --- |
| 类型 | 位数 | 最大值~最小值 | 详情&举例 |
| **byte** | 8位 | -128（-2^7）~127（2^7-1） | ·有符号的，以二进制补码表示的**整数**，默认值为0  ·**定义(声明)变量**的写法：  byte a = 100，byte b = -50。 |
| **short** | 16位 | -32768（-2^15）~ 32767（2^15 - 1） | ·有符号的，以二进制补码表示的**整数**，默认值为0  ·**定义(声明)变量**的写法：  short s = 1000，short r = -20000。 |
| **int** | 32位 | -2^31~2^31 - 1 | ·有符号的，以二进制补码表示的**整数**，默认值为0  ·**定义(声明)变量**的写法：  int a = 100000, int b = -200000。 |
| **long** | 64位 | -2^63~2^63 -1 | ·有符号的，以二进制补码表示的**整数**，默认值为0  ·**定义(声明)变量**的写法：  long a = 100000L，Long b = -200000L。 |
| **float** | 32位 | -- | ·符合IEEE 754标准的浮点数(小数)  默认值是0.0f，是**单精度浮点数据类型**  ·**定义(声明)变量**的写法：  float f1 = 234.5f。 |
| **double** | 64位 | -- | ·符合IEEE 754标准的浮点数(小数)  默认值是0.0d，是**双精度浮点数据类型**  ·**定义(声明)变量**的写法：  double d1 = 123.4。 |
| **char** | 16位 | 最小值是 '\u0000'（即为0）  最大值是 '\uffff'（即为65,535） | ·单一的16位Unicode字符  char数据类型可以储存任何字符；  ·**定义(声明)变量**的写法：  char letter =‘A’ |
| **boolean** | 1位 | -- | ·只有两个取值：true和false；  默认值是false；  ·**定义(声明)变量**的写法：  boolean one = true |

**④变量与常量**

在程序执行的过程中，其**值不能改变**的量称为常量，其**值能够被改变**的量称为变量。变量与常量的声明都必须使用**合法的标识符**，并且只有在**声明后**才能使用。

**声明变量**：定义一个变量，就是要告诉编译器这个变量属于哪一种数据类型，这样编译器才知道要配置多少空间，以及能存放什么样的数据。变量都有一个合法的变量名，在声明变量时可以不给予赋值，也可以直接赋给初值。

e.g. int i = 10; //赋初值

char letter = ‘A’; //赋初值

double d; //不赋初值，只声明

**声明常量**：在程序运行过程中一直不会改变的量称为常量。常量在整个程序中只能被赋值一次，且声明时必须赋值。声明常量的标准语法：final 数据类型 常量名称[=值]（常量名称一般采用全大写）

e.g.final double PI = 3.1415926

**⑤运算符**

**a**.**赋值运算符**：即“=”，其功能是将右方操作数所含的值赋值给左方的操作数，语法：

变量类型 变量名 = 所附的值;

e.g.：int a = 3;

**b**.**算术运算符**：

算术运算符用在数学表达式中，它们的作用和在数学中的作用一样。下表列出了所有的算术运算符。

表格中的实例假设整数变量A的值为10，变量B的值为20：

|  |  |  |
| --- | --- | --- |
| 运算符 | 描述 | 例子 |
| + | 加法 - 相加运算符两侧的值 | A + B等于30 |
| - | 减法 - 左操作数减去右操作数 | A – B等于-10 |
| \* | 乘法 - 相乘操作符两侧的值 | A \* B等于200 |
| / | 除法 - 左操作数除以右操作数 | B / A等于2 |
| ％ | 取模 - 左操作数除右操作数的余数 | B%A等于0 |
| ++ | 自增 - 操作数的值增加1 | B++等于21 |
| -- | 自减 - 操作数的值减少1 | B--等于19 |

注意：自增、自减可以放在操作元前也可以放在操作元后。**放在前面的会将变量的值加1（减1），然后再使该变量参与表达式的运算；放在后面的会先使变量参与表达式的运算，然后再将该变量加1（减1）**

e.g.：int a = 4;

int b = a++; // b为4，a为5

int c = ++a; // c为5，a为5

**c.关系运算符**

|  |  |  |
| --- | --- | --- |
| 运算符 | 描述 | 例子（A=10,B=20） |
| == | 检查如果两个操作数的值是否相等，如果相等则条件为真。 | （A == B）为假(非真)。 |
| != | 检查如果两个操作数的值是否相等，如果值不相等则条件为真。 | （A != B）为真。 |
| > | 检查左操作数的值是否大于右操作数的值，如果是那么条件为真。 | （A > B）非真。 |
| < | 检查左操作数的值是否小于右操作数的值，如果是那么条件为真。 | （A < B）为真。 |
| >= | 检查左操作数的值是否大于或等于右操作数的值，如果是那么条件为真。 | （A> = B）为假。 |
| <= | 检查左操作数的值是否小于或等于右操作数的值，如果是那么条件为真。 | （A <= B）为真。 |

**d.逻辑运算符**

|  |  |  |
| --- | --- | --- |
| 操作符 | 描述 | 例子（A=true, B=false） |
| && | 称为逻辑与运算符。当且仅当两个操作数都为真，条件才为真。 | （A && B）为假。 |
| | | | 称为逻辑或操作符。如果任何两个操作数任何一个为真，条件为真。 | （A | | B）为真。 |
| ！ | 称为逻辑非运算符。用来反转操作数的逻辑状态。如果条件为true，则逻辑非运算符将得到false。 | ！（A && B）为真。 |

**e.三元运算符**

三元运算符是Java中唯一 一个三目运算符，其操作元有3个，第一个是条件表达式，其余的是两个值，**条件表达式成立时运算取第一个值，不成立时取第二个值**。

boolean b = 20 < 45 ? true : false; // 过程是先①判断20<45是否成立②成立则取true，否则取false③将true这个结果赋予变量b。即最终结果是b=true。

**☞Java流程控制**

在对基本数据类型有了一定了解后，那我们该如何运用数据呢？这就需要对数据进行一系列的操作，包括前面讲到的运算符，也包括接下来讲的流程控制。

**1.if语句**

if条件语句是一个重要的编程语句，它用于告诉程序在某个条件成立的情况下执行某段程序，而在另一种情况下执行另外的语句。

最简单的例子如下：

int a = 100;

if (a == 100) {

System.out.print(“a的值是100”);

}

上面这段程序讲的是如果a的值等于100，控制台就会输出“a的值是100”这句话，否则将不会输出。

if语句的语法格式如下：

if(布尔表达式){

语句序列

}

**解析\*：**

**使用if条件语句，判断是否执行紧跟在条件之后的那个语句**。关键字if之后是作为条件的“布尔表达式”，如果该表达式返回的结果为true，则执行其后的语句序列；若为false，则不执行if条件之后的语句。if语句可分为简单的if条件语句、if...else语句和if...else if多分支语句。

**布尔表达式：**是if语句的**必要**参数，表示它最后返回的结果必须是一个布尔值。它可以是一个单纯的布尔变量或常量，或者使用关系或布尔运算符的表达式。

**语句序列：**可以是一条或多条语句或者为空什么都不写，当表达式的值为true时执行这

些语句。语句序列仅有一条语句，则可以省略条件语句中的大括号。如上例可写成

int a = 100;

if (a == 100)

System.out.print(“a的值是100”);

**2.switch多分支语句**

在编程中一个常见的问题就是检测一个变量是否符合某个条件，如果不匹配，再用另一个值来检测它，依次类推。**当然，这种问题使用if条件语句也可以完成**。

如：

*If (grade == “A”) {*

*System.out.print(“真棒”);*

*}*

*If (grade == “B”) {*

*System.out.print(“做的不错”);*

*}*

**但这样写的话**，**程序就会显得比较笨重**，程序员需要测试不同的值来给出输出语句。在Java语言中，可以使用switch语句将动作组织起来，就能以一个比较简单明了的方式来实现“多选一”的选择。语法格式如下：

*switch(****表达式****){*

*case* ***常量值1****:*

*语句块1*

*break;*

*........*

*case* ***常量值n****:*

*语句块n*

*break;*

*default:*

*}*

switch语句中**表达式**的值必须是**整形或字符型**，**常量值1~常量值n**必须也是**整形或字符型**。**解析\*：**

1）switch语句首先计算表达式的值，如果表达式的值和某个case后面的变量值相同，则执行该case语句后的若干个语句直到遇到break语句为止。

2）此时如果该case语句中没有break语句，将继续执行后面case中的若干个语句，直到遇到break语句为止。若没有一个变量的值与表达式的值相同，则执行default后面的语句。

3）default语句为可选的，如果它不存在，而且switch语句中表达式的值不与任何case的常量值相同，switch则不做任何处理。

**3.循环语句**

**①while循环语句**

while循环语句也称条件判断语句，它的循环方式为**利用一个条件来控制是否要继续反复执行**这个语句，语法格式如下：

*while(条件表达式){*

*执行语句*

*}*

当条件表达式的返回值为真时，则执行“{}”中的语句，当执行完“{}”中的语句后，重新判断条件表达式的返回值，直到表达式返回的结果为假时，退出循环。如：

*int x =1;*

*int sum = 0;*

*while(x <= 10){*

*sum = sum + x;*

*x++;*

*}*

*System.out.print(“1到10的和是：sum = ” + sum);*

上述程序就是**使用了while循环计算了1到10的和**。

**②do...while循环语句**

do...while循环语句与while循环语句的**区别**：它们之间的区别是while循环语句为先判断条件是否成立再执行循环体，而do...while循环语句则**先执行一次循环**后，再判断条件是否成立。也就是说do...while循环语句中大括号中的程序段至少要被执行一次。

**语法格式如下：(可以尝试一下将上面1~10的和用do…while循环写出来)**

*do{*

*执行语句*

*}while(条件表达式);*

**③for循环语句**

for循环语句是Java程序设计中**最有用的循环语句之一**。一个for循环可以**用来重复执行某条语句，直到某个条件得到满足**。

基本for语句语法格式如下：

*for(表达式1;表达式2;表达式3){*

*语句序列*

*}*

**表达式1：初始化表达式，负责完成变量的初始化。**

**表达式2：循环条件表达式，值为boolean型的表达式，指定为循环条件。**

**表达式3：循环后操作表达式，负责修整变量，改变循环条件。**

**解析\*：**在执行for循环时，首先执行表达式1，完成某一变量的初始化工作；下一步判断表达式2的值，若表达式2的值为true，则进入循环体；在执行完成循环体后紧接着计算表达式3，这部分通常是增加或减少循环控制变量的一个表达式。这样一轮循环结束后，第二轮循环则从计算表达式2开始。

*int sum = 0;*

*for(int i = 2; i <= 100; i+= 2){*

*sum = sum + i;*

*}*

*System.out.print(“2到100之间的所有偶数之和为：” + sum);*

**以上程序利用for循环计算了2到100之间所有偶数之和。**

for循环还有另一种语法格式，我们把它称作**foreach语句，它是for语句的特殊简化版本，**foreach语句不能完全取代for语句，然而任何foreach语句都可以改写成for语句版本。**语法格式**如下：

*for(元素变量x : 遍历对象obj){*

*引用了x的Java语句；*

*}*

foreach语句中的元素变量x，**不必对其进行初始化**。下面通过简单的例子来介绍foreach语句是怎样**遍历一维数组**的。（数组内容后面有，打个预告）

*int arr[ ] = {7,10,1};*

*for(int x : arr){*

*System.out.print(x + “和”)；*

*}*

**(输出结果为：7和10和1)**

**4.跳转语句**

**①break语句**

break语句用于强行退出循环，也就是忽略循环体中任何其他语句和循环条件的限制。break语句可以用于switch、for、while、do...while语句中。

**②continue语句**

continue语句只能应用在for、while、do...while循环语句中，用于让程序跳过本次循环中continue语句后面的语句，并进入下一次循环。

我们举一个例子便于更好理解break语句和continue语句的区别。

*while(x++ < 10){*

*if(x == 3){*

***break;***

*}*

*System.out.print(x + “\t”);*

*}*

上述程序**输出结果是1 2**, **之后便退出了整个while循环**。

但是如果使用continue:

*while(x++ < 10){*

*if(x == 3){*

***continue;***

*}*

*System.out.print(x + “\t”);*

*}*

则**输出结果是1 2 4 5 6 7 8 9 10**，可见上述程序结果**仅仅是不输出3，因为它只是结束了本次循环，进入下一次循环**。

**☞\*Java数组**

先来瞅瞅大概的形式（暂时先了解到二维吧）

|  |  |  |
| --- | --- | --- |
| 维度 | 变量 | 实质 |
| 一维 | int数组a[]={5,2,0} | 数组a存储了3个int类型数据:5,2,0 |
| 二维 | int数组a[2][3]={1,2,3,4,5,6} | 数组a存储6个int数据: |

**要点\*：**

**1）**数组是具有相同类型数据的一组数据的集合。当需要使用的变量很多，而且数据类型相同时，逐个声明就显得非常麻烦，这时可以声明一个数组，然后对数组进行操作。

**2）**数组作为对象允许使用new关键字进行内存分配，在使用数组之前，必须首先定义数组变量所属的类型，即声明数组。

**3）**声明一维数组有两种形式，**语法格式**分别如下：

数组元素类型 数组名字[ ];

数组元素类型[ ] 数组名字;

**数组元素类型**：决定了数组的数据类型，它可以是Java中任意的数据类型，包括基本数据类型和非基本数据类型。

**数组名字**：为一个合法的标识符。

**符号”[ ]”**: 指明该变量是一个数组类型变量，单个”[ ]”表示要创建的数组是一维数组。

**注意\*：**声明数组后，还不能访问它的任何元素，因为声明数组仅仅是给出了数组名字和元素的数据类型，要想真正使用数组还要为其分配内存空间，且分配内存空间时必须指明数组的长度。**分配内存空间的语法格式**如下：

数组名字 = new 数组元素类型[数组元素个数];

**数组名字**：已经声明的数组变量的名称。

**new**：对数组分配空间的关键字。

**数组元素的个数**：指定数组中变量的个数。即数组的长度。

如：

***int arr[ ] = new int[5]; //声明了一个int型的数组并为它分配了空间***

如图为一维数组的内存模式，**注意，在内存空间中数组的下标是从0开始的。**

**4）**数组可以与基本数据类型一样进行初始化操作，数组的初始化可分别初始化数组中的每个元素。数组的初始化有两种方式，如：

int arr[] = new int[]{1,2,3,5,25}; int arr2[] = {34,23,12,6};

**5）\*下图为一维数组的内存模式（以arr2数组为例子）:**

arr2[0] →34 1.内存空间中数组的下标是从0开始

arr2[1] →23 可以推出 2.每个下标对应着一个数据

arr2[2] →12 3.只要知道下标便可输出数据

arr2[3] →6 4.格式如：arr2[下标]便表示对应的数据

除了一维数组以外，还可以拓展到多维数组。如：

int a[][] = new int[2][4];（你是否能够推出二维数组每个数据的下标是如何表示的呢？）

**☞Java字符串**

String类即字符串类型，并不是Java的基本数据类型，但可以像基本数据类型一样使用，用双引号括起来进行声明。

**①创建**

String()：String s = new String(); // s表示一个空字符序列。

String(char a[])：用一个字符数组a创建String对象。

如：

char a[] = {‘g’,’o’, ‘o’, ‘d’};

String s = new String(a); // 这两句等价于String s = new String(“good”);

String(char a[], int offset, int length)：提取字符数组a中的一部分创建一个字符串对象。参数offset开始表示截取字符串的位置，length表示截取字符串的长度。

char a[] = {‘s’,’t’,’u’,’d’,’e’,’n’,’t’};

String s = new String(a, 2, 5); //这两句等价于String s = new String(“udent”);

**②字符串连接：**连接多个字符串时，在每两个连接的字符串之间用“+”相连，连接之后生成一个新的字符串

e.g.String s1 = new String(“hello”);

String s2 = new String(“world”);

String s = s1 + “ ” + s2; //s的结果是hello world

**③常用方法**

获取字符串长度：length()

String s = “We are students”;

s.length(); // 长度为15，因为空格也被计算在内

获取索引（即获取某个字符在字符串中首次出现的位置）：indexOf()

String s = “We are students”;

s.indexOf(“s”); // 返回7，因为String对象的索引是从0开始的，即与数组下标对应。

获取指定索引位置的字符：charAt(int index)

String s = “Hello world”;

char c = s.charAt(6); //字符为w

字符串替换：str.replace(String target, String replacement);

target是str中想要被替换的字符串变量名

replacement是新字符串变量名，用来替换所有出现的target，返回替换后的字符串。如：String str = “That’s an interesting instrument”;

String target = “in”;

String replacement = “out”;

System.out.print(str.replace(target, replacement));

//结果是That’s an outteresting outstrument

判断字符串是否相等：str.equal(String otherstr)

如果两个字符串仍具有相同的字符和长度，则返回true，否则返回false。