目录

一、关键字概述及使用.....2
二、标识符及常用命名规则.....2
三、Java 中注释分类格式.....3
四、常量.....3
（一）常量种类.....3
（二）进制及进制的转换.....3
五、变量及数据类型.....4
（一）变量定义.....4
（二）数据类型.....4
（三）float 数据在内存中存储方式.....6
六、运算符.....7
七、控制语句.....8
（一）顺序结构.....8
（二）选择结构.....8
（三）循环结构.....8
（四）跳转控制.....8
八、方法（函数）.....8
九、数组.....8
（一）一维数组.....8
（二）二维数组.....9
附件（ASCII 码）.....11

一、 关键字概述及使用

关键字：被 java 语言赋予特定含义的单词

特点：组成关键字单词的字母全部小写（注：main 不是关键字）

注意： A：goto 和 const 是保留字

B：类似于 Notepad++这样的高级记事本，针对关键字都有特殊的颜色标记

用于定义数据类型的关键字					
class	interface	byte	short	int	void
long	float	double	char	boolean	
用于定义数据类型值的关键字					
true	false	null			
用于定义流程控制的关键字					
if	else	switch	case	default	return
while	do	for	break	continue	
用于定义访问权限修饰符的关键字					
private	protected	public			
用于定义类，函数，变量修饰符的关键字					
abstract	final	static	synchronized		
用于定义类与类之间关系的关键字					
extends	implements				
用于定义建立实例及引用实例，判断实例的关键字					
new	this	super	instanceof		
用于异常处理的关键字					
try	catch	finally	throw	throws	
用于包的关键字					
package	import				
其他修饰符关键字					
native	strictfp	transient	volatile	assert	

二、 标识符及常用命名规则

就是给类,接口,方法,变量等起名字时使用的字符序列

组成规则

- 英文大小写字母
- 数字字符

- $和_

注意事项

- 不能以数字开头
- 不能是 Java 中的关键字
- 区分大小写

常用命名规则：见名知意

- 包：其实就是文件夹，用于把相同的类名进行区分（注：全部小写）
  单级：spianch
  多级：cn.itcast
- 类或者接口
  一个单词：单词的首字母必须大写 eg：Student 等
  多个单词：每个单词的首字母必须大写 eg：HelloWorld 等
- 方法或者变量
  一个单词：单词的首字母小写 eg：main，age 等
  多个单词：从第二个单词开始，每个单词的首字母大写 eg：studentAge 等
- 常量
  一个单词：全部大写 eg：PT 等
  多个单词：每个字母都大写，用_隔开 eg：MAX_AGE 等

三、Java 中注释分类格式

用于解释说明程序的文字

单行注释 格式： //注释文字

多行注释 格式： /* 注释文字 */

文档注释 格式： /** 注释文字 */

四、常量

（一）常量种类

字符串常量 整数常量 小数常量 字符常量 布尔常量 空常量（null）

（二）进制及进制的转换

与 C 语言类似，此不仔细介绍。

五、 变量及数据类型

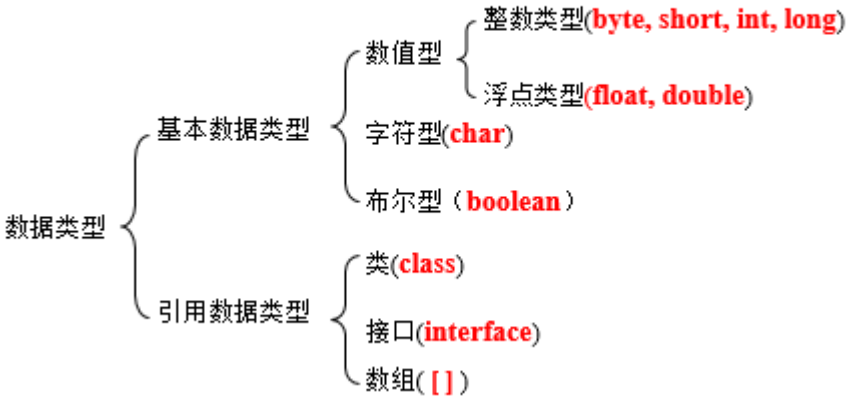
（一）变量定义

在程序执行的过程中，在某个范围内其值可以发生改变的量

数据类型 变量名 = 初始化值;

（二）数据类型

Java语言是强类型语言，对于每一种数据都定义了明确的具体数据类型，在内存总分配了不同大小的内存空间



类 型	占用存储空间	表数范围
byte	1字节	-128～127
short	2字节	-2^{15}～$2^{15}-1$
int	4字节	-2^{31}～$2^{31}-1$
long	8字节	-2^{63}～$2^{63}-1$

类 型	占用存储空间	表数范围
float	4字节	-3.403E38～3.403E38
double	8字节	-1.798E308～1.798E308

整数默认：int 小数默认：double

- boolean 类型不能转换为其他的数据类型
- 默认转换
  byte,short,char—int—long—float—double
  byte,short,char 相互之间补转换，他们参与运算首先转换为 int 类型
- 强制转换
  目标类型 变量名=(目标类型)(被转换的数据);
- 字符串数据和其他数据做+，结构是字符串类型
  这里的+不是+，而是字符串连接符

Eg1：

面试题：

```
byte b1=3,b2=4,b;
b=b1+b2;
b=3+4;
```

哪句是编译失败的呢？为什么呢？

```
12 class DataTypeDemo6 {
13     public static void main(String[] args) {
14         //定义了三个byte类型的变量，b1，b2，b3
15         //b1的值是3，b2的值是4，b没有值
16         byte b1 = 3,b2 = 4,b;
17 
18         //b = b1 + b2; //这个是类型提升，所有有问题
19 
20         b = 3 + 4; //常量，先把结果计算出来，然后看是否在byte的范围内，如果在就不报错。
21     }
22 }
```

Eg2：

面试题：

```
byte b = 130;
```

有没有问题?如果我想让赋值正确，可以怎么做?结果是多少呢?

```
 6 class DataTypeDemo7 {
 7     public static void main(String[] args) {
 8         //因为byte的范围是：-128到127。
 9         //而130不在此范围内，所以报错。
10         //byte b = 130;
11 
12         //我们可以使用强制类型转换
13         byte b = (byte) 130;
14 
15         //结果是多少呢?
16         System.out.println(b);
17     }
18 }
19 /*
20     分析过程：
21         我们要想知道结果是什么，就应该知道是如何进行计算的。
22         而我们又知道计算机中数据的运算都是补码进行的。
23         而要得到补码，首先要计算出数据的二进制。
24 
25         A:获取130这个数据的二进制。
26             00000000 00000000 00000000 10000010
27             这是130的原码，也是反码，还是补码。
28         B:做截取操作，截成byte类型的了。
29             10000010
30             这个结果是补码。
31         C:已知补码求原码。
32                     符号位      数值位
33             补码：   1          0000010
34 
35             反码：   1          0000001
36 
37             原码：   1          1111110
38 */
```

结果为原码：-126

Eg3:

面试题：

```
6  class DataTypeDemo9 {
7      public static void main(String[] args) {
8          System.out.println("hello"+'a'+1); //helloa1
9          System.out.println('a'+1+"hello"); //98hello
10
11         System.out.println("5+5="+5+5); //5+5=55
12         System.out.println(5+5+"=5+5"); //10=5+5
13
14     }
15 }
```

结果是：

```
E:\JavaSE\day02\code\代码\05_变量和数据类型>javac DataTypeDemo9.java

E:\JavaSE\day02\code\代码\05_变量和数据类型>java DataTypeDemo9
helloa1
98hello
5+5=55
10=5+5
```

（三）float 数据在内存中存储方式

float 类型数字在计算机中用 4 个字节存储。遵循 IEEE-754 格式标准：

一个浮点数有 2 部分组成：底数 m 和指数 e

底数部分 使用二进制数来表示此浮点数的实际值

指数部分 占用 8bit 的二进制数，可表示数值范围为 0-255

但是指数可正可负，所以，IEEE 规定，此处算出的次方必须减去 127 才是真正的指数。

所以，float 类型的指数可从-126 到 128

底数部分实际是占用 24bit 的一个值，但是最高位始终为 1，所以，最高位省去不存储，在存储中占 23bit

科学计数法：

格式：

SEEE EEEE EMMM MMMM MMMM MMMM MMMM MMMM

S 表示浮点数正负

E 指数加上 127 后的值得二进制数据

M 底数

举例：

17.625 在内存中的存储

首先要把 17.625 换算成二进制：10001.101

整数部分，除以 2，直到商为 0，余数反转。

小数部分，乘以 2，直到乘位 0，进位顺序取。

在将 10001.101 右移，直到小数点前只剩 1 位：

1.0001101 * 2^4 因为右移动了四位

这个时候，我们的底数和指数就出来了

底数：因为小数点前必为 1，所以 IEEE 规定只记录小数点后的就好。所以，此处的底数为：0001101

指数 :实际为 4 ,必须加上 127(转出的时候 ,减去 127) ,所以为 131。也就是 10000011

符号部分是整数，所以是 0

综上所述，17.625 在内存中的存储格式是：

01000001 10001101 00000000 00000000

换算回去：自己做。

六、运算符

算术运算符 赋值运算符 比较运算符 逻辑运算符 位运算符 三目运算符

Eg1：

面试题：

```
Short s = 1; s = s+1;
Short s = 1; s += 1;
```

上面两个代码有问题吗，如果有，哪里有问题？

```
15 class OperatorTest {
16     public static void main(String[] args) {
17         //short s = 1;
18         //s = s + 1;
19         //System.out.println(s);
20 
21         short s = 1;
22         s += 1; //好像是 s = s + 1;
23         System.out.println(s);
24     }
25 }
```

结果：第一个会报错，提示可能有损失精度，而第二个正常运行

- 为什么第二个没有问题？

由于扩展的赋值运算符其实隐含了一个强制性转换，

即如 s +=1; 不是等价于 s = s+1;

而是等价于 s = (s 的数据类型)(s+1)；

七、控制语句

（一）顺序结构

（二）选择结构

if 流程　if-else 流程　else if 流程　if 语句嵌套

switch-case 语句

（三）循环结构

while 语句　do-while 语句　循环嵌套　for 循环

（四）跳转控制

break 语句　continue 语句　return 语句

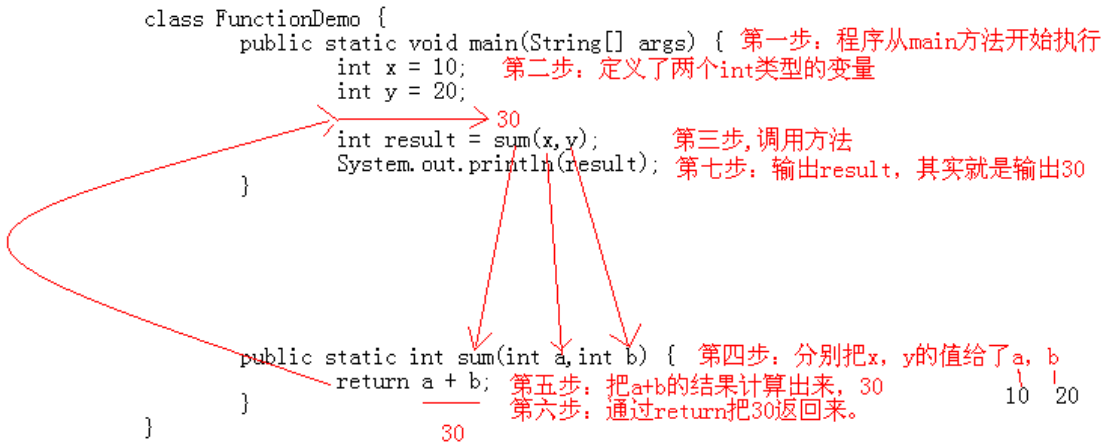
八、方法（函数）

方法就是完成特定功能的代码块。在很多语言里面都有函数的定义，函数在 Java 中被称为方法。

格式：

```
修饰符 返回值类型 方法名(参数类型 参数名 1，参数类型 参数名 2…)
{
    函数体;
    return 返回值;
}
```

执行过程：



九、数组

（一）一维数组

与 C 语言用法一致。

静态初始化　格式：数据类型[] 数组名 = new 数据类型[]{元素 1,元素 2,…};

```
int[] arr = new int[]{1,2,3};
```

简化格式：数据类型[] 数组名 = {元素 1,元素 2,…};

```
int[] arr = {1,2,3};
```

动态初始化　格式：数据类型[] 数组名 = new 数据类型[元素个数]；

```
int[] arr = new int[3];
```

注意事项：不要同时动态和静态进行。

```
int[] arr = new int[3]{1,2,3};
/*错误，错因在于系统给予 3 个空间，而用户又让系统去判断给予的几个空间，从而系统不能判断。*/
```

Eg1：

```
/*
        定义一个数组，输出该数组的名称和数组元素值
        赋值后，再输出该数组的名称和数组元素值
*/
class ArrayDemo2
{
        public static void main(String[] args)
        {
        int[] arr = new int[2];
        //输出数组名称
        System.out.println(arr);
        //输出数组元素值
        System.out.println(arr[0]);
        System.out.println(arr[1]);
        System.out.println("------");

        //给数组元素赋值
        arr[0] = 100;
        arr[1] = 200;

        //输出数组名称
        System.out.println(arr);
        //输出数组元素值
        System.out.println(arr[0]);
        System.out.println(arr[1]);
        }
}
```

运行结果：

```
GHB-PC:Desktop Spinach$ javac ArrayDemo2.java
GHB-PC:Desktop Spinach$ java ArrayDemo2
[I@7852e922
0
0
------
[I@7852e922
100
200
```

一维数组的首地址

（二）二维数组

与 C 语言用法一致。

格式 1：数据类型[][] 数组名 = new 数据类型[m][n]；

```
int[][] arr = new int[3][2];
```

注意事项：A.以下格式也可表示二维数组

```
数据类型 数组名[][] = new 数据类型[m][n]；
数据类型[] 数组名[] = new 数据类型[m][n]；
```

B.注意下面定义的区别

```
int x,y;          等价于  int x;int y;
int[] x,y[];      等价于  int[] x;int[] y[];
 /*即 x 为一维数组，而 y 为二维数组 */
```

格式 2：数据类型[][] 数组名 = new 数据类型[m][];

```
int[][] arr = new int[3][];//定义二维数组，但每一维数组不知其大小
arr[0] = new int[3];       //第一个一维数组有 3 个元素
arr[1] = new int[2];       //第二个一维数组有 2 个元素
arr[2] = new int[1];       //第三个一维数组有 1 个元素
```

格式 3：数据类型[][] 数组名 = new 数据类型[][]{{元素 1,…},{ 元素 1,…},…};

```
int[][] arr = new int[][]{{1,2,3},{4,5},{6}};
```

简化格式：数据类型[][] 数组名 = {{元素 1,…},{元素 1,…},…};

```
int[][] arr = {{1,2,3},{4,5},{6}};
```

于 2016/7/18 编著完成

附件（ASCII 码）

常用字符与 ASCII 码对照表

为了便于查询，以下列出 **ASCII 码表**：第 128～255 号为扩展字符（不常用）

ASCII 码	键盘	ASCII 码	键盘	ASCII 码	键盘	ASCII 码	键盘
27	ESC	32	SPACE	33	!	34	"
35	#	36	$	37	%	38	&
39	'	40	(	41	)	42	*
43	+	44	'	45	-	46	.
47	/	48	0	49	1	50	2
51	3	52	4	53	5	54	6
55	7	56	8	57	9	58	:
59	;	60	<	61	=	62	>
63	?	64	@	65	A	66	B
67	C	68	D	69	E	70	F
71	G	72	H	73	I	74	J
75	K	76	L	77	M	78	N
79	O	80	P	81	Q	82	R
83	S	84	T	85	U	86	V
87	W	88	X	89	Y	90	Z
91	[	92	\	93	]	94	^
95	_	96	`	97	a	98	b
99	c	100	d	101	e	102	f
103	g	104	h	105	i	106	j
107	k	108	l	109	m	110	n
111	o	112	p	113	q	114	r
115	s	116	t	117	u	118	v
119	w	120	x	121	y	122	z
123	{	124	|	125	}	126	~