**JAVA基础**

# 标识符

由字母、数字、下划线，美元符$组成，且数字不能作为开头。

1abc 不是合法的标识符 以数字作为开头

$abc 是 abc\*123 不是，因为包含特殊字符\*

Abc123$ 是

标识符作用：

定义类名 方法名 变量名,这些名字不能是java关键字

必须都是合法的标识符

命名规则：

由字母,数字,下划线\_,美元$组成,但数字不能开头. 驼峰命名法

其还不能是java关键字（在eclipse中，字体颜色发生变化）

类名：必须由有意义的单词（不是拼音）组成，并且单词的首字母都要大写 ，如用果单词过长，可以使用简写

方法名：必须由有意义的单词组成，并且第一个单词的首字母小写，其他单词的首字母都要大写 ，如用果单词过长，可以使用简写

变量名：

普通变量（成员变量，局部变量）必须由有意义的单词组成，并且第一个单词的首字母小写，其他单词的首字母都要大写 ，如用果单词过长，可以使用简写

静态变量：使用static修饰，所以写的时候 sStaticVariable

常变量: final修饰 PI

# 变量（unit2/HelloWorld.java）

变量就好比是一个容器，它是用来装数据的。（杯子是用来装水一样）

变量的命名一定要是合法的标识符，由有意义的单词组成，首字母小写，后面的每个单词的首字母大写。

购物车 总价定义成一个变量， 变量名 totalMoney，shoppingcarMoney….

添加1个商品 5 -》totalMoney = 5

添加1个商品 10 -》totalMoney = 15

添加1个商品 20 –》totalMoney = 35

查看购物车时，就可以看到购物车商品总价

|  |
| --- |
| **public** **static** **void** main(String[] args) {  // 数据类型int，他来表示变量的容器类型，  // 即这个容器可以装载什么样的数据,整型-》整数  // int 数据类型 totalMoney变量名 5变量初始值  // = 代表赋值  // 变量一定是先定义后使用,局部变量必须初始化  **int** totalMoney = 5;  // 在原来的基础上再加  totalMoney = totalMoney + 10;  totalMoney = totalMoney + 20;  System.***out***.println(totalMoney);  } |

## 局部变量

根据作用域划分的，作用域就是他能起作用的地方

|  |
| --- |
| // 方法，方法名为test，通过大括号，闭包表示这个方法内容的范围  **public** **void** test() {  // 局部变量，只在方法内部有效  **int** totalMoney = 5;  System.***out***.println(totalMoney);  } |

成员/全局变量

|  |
| --- |
| **public** **class** HelloWorld {  // 类中，方法外定义全局变量，成员变量  // 类的加载， 内存可以即存即取，或者只要内存中存在，什么时候想取，随时都可以拿到  **static** **int** *totalMoney* = 100;  **public** **static** **void** main(String[] args){  }  } |

# 数据类型

Java是强类型的语言，变量使用前必须先声明类型，什么类型的变量装载什么类型的数据。JavaScript是弱类型的语言

## 基本数据类型（unit2/ DataType.java）

8个

byte short int long float double boolean char

int整型 4个字节 整数

byte 2^8-1=255 -256 1个字节

short 2个字节 短整数

long 8个字节 长整数

float 4个字节 幂次方位 小数

double 8个字节 小数

boolean true false

char 字符 ‘a’ ‘b’ 2个字节

|  |
| --- |
| **public** **static** **void** main(String[] args) {  **byte** bVal = -100;  **int** iVal = 10000000;  **short** sVal = 100;  // 长整型变量要加L  **long** lVal = 10000000000000L;  // float变量，后面必须要加f  **float** fval = 100.0f;  // 字面量如100,100.0,1000  // 在代码中出现100字面量时，默认的数据类型为int类型  // 在代码中出现100.0字面量时，默认的数据类型为double类型  **double** dval = 100.0;  // boolean类型变量的定义  **boolean** isBoo = **true**;  **boolean** isBoo1 = **false**;  **char** cVal = 'c';  } |

#### 默认值（unit2/ DefaultValDt.java）

|  |
| --- |
| **public** **class** DefaultValDt {  **static** **int** *iVal*;  **static** **byte** *bVal*;  **static** **short** *sVal*;  **static** **long** *lVal*;  **static** **float** *fVal*;  **static** **double** *dVal*;  **static** **boolean** *booVal*;  **static** **char** *cVal*;  **public** **static** **void** main(String[] args) {  // + 字符串连接  System.***out***.println("int 类型-》" + *iVal*);  System.***out***.println("byte 类型-》" + *bVal*);  System.***out***.println("short 类型-》" + *sVal*);  System.***out***.println("long 类型-》" + *lVal*);  System.***out***.println("float 类型-》" + *fVal*);  System.***out***.println("double 类型-》" + *dVal*);  System.***out***.println("boolean 类型-》" + *booVal*);  System.***out***.println("char 类型-》" + *cVal*);  }  } |

#### 类型转换（unit2/DataTrans.java）

自动类型转换

表数范围小的自动转换为表数范围大的

强制类型转换

表数范围大的强制转换为表数范围小的

|  |
| --- |
| **public** **class** DataTrans {  **public** **static** **void** main(String[] args) {  // **TODO** Auto-generated method stub  **int** iVal = 10;  **long** lVal = 10000000000000L;  // long表示的数据范围比int大，在讲一个long类型的值赋给int时  // 需要强制转换  iVal = (**int**) lVal;  System.***out***.println(iVal);  // 自动转换  lVal = iVal;  **float** fVal = 10.0f;  **double** dVal = 11.8;  // double表示的数据范围比float大，再讲一个double赋值给  // float类型的值时，需要强制转换，  // 将double类型的值变为float类型的值  fVal = (**float**) dVal;  // 从float转到double自动转换  dVal = fVal;  // 自动转换  dVal = iVal;  // 强制转换,会带来一个问题，就是精度丢失  dVal = 11.8;  iVal = (**int**) dVal;  System.***out***.println(iVal);  }  } |

#### 引用数据类型

String字符串

数组 枚举 接口

|  |
| --- |
| **public** **class** StringDemo {  **public** **static** **void** main(String[] args) {  // 字符串使用双引号括起来，  String str = "上海.............";  str = str + "是魔都！";  System.***out***.println(str);  }  } |
| // + 从左向右运算  String str1 = 10 + 20 + "10";  String str2 = str + 10 + 20;  System.out.println(str1);// 3010  System.out.println(str2);// 上海.............是魔都！1020 |

# 运算符

* 算术运算符：+ , -, \* / ,% ,++ ,--
  + 注意,自加和自减放在变量前后的区别：放在前面先完成自加和自减再把变量拿出来用,在后面先拿出来用,再自加和自减
  + 自加和自减只能用于操作变量,不能用于操作常量
* 赋值运算符：=、+=、\*=、/=、%=;
* 位运算符：&、 |、 ~ 、^、 << 、>>、 >>>。
* 扩展后的赋值运算符
* 比较运算符<、>、>=、<=、!=、==
* 逻辑运算符：&&、 || 、!、 ^
* 三目运算符 条件? true的取值 : false 的取值
* 优先级,,,尽量用小括号,,引起,这样可读性更好

|  |
| --- |
| **public** **class** OperatorDemo {  **public** **static** **void** main(String[] args) {  // + - \* /  **int** x = 10;  **int** y = 20;  **int** z = x + y;  System.***out***.println(z);  // 105是整形，10也是，所以計算結果是整數10,  // 整數10賦值給a  **double** a = 105/10;  // 复合赋值运算符，知道  a /= 10 ;// a = a/10  // 105.0是浮点型，即double类型，10是整形，所以計算結果是整數10.5,  // 浮点数10.5賦值給a  **double** b = 105.0/10;  System.***out***.println(a);  System.***out***.println(b);  // 取余  **int** c = 105 % y;  System.***out***.println(c);  // ++ -- 自增与自减，在前在后意思有所区别  System.***out***.println(c++); // 5 先取值执行语句，在运算加法  // c++ ==> c = c + 1  // ++c ==> c = c + 1  System.***out***.println(++c); // 7 先运算，再取值  System.***out***.println(c--); // 7  System.***out***.println(--c); // 5  // 比较运算符 false true  **int** j = 10;  **int** k = 100;  System.***out***.println(j >= k);// false  System.***out***.println(j <= k);// true  // 基本数据类型==比较他们的数值  System.***out***.println(j == k);// false  System.***out***.println(j != k);// true  // 逻辑运算符 与 或 非 boolean类型的值进行运算  // 与 && （true && true=》true），其他均为false，一个为false便为false  // 或 || （false || false=》false），其他均为true，一个为true便为true  // 非 ！ （！fasle=》true ！true=》false）  // 异或 （fasle ^ fasle =>false, true ^ true=>fasle)其他都是true，  //异或： 若两边相同则为false，不一样时为true  System.***out***.println((j > k) && **true**);// false  System.***out***.println((j < k) && (20 > 10)); // true  System.***out***.println((j > k) || **true**);// true  // j>k=>false !false=>true  System.***out***.println(!(j > k));// true  System.***out***.println("====");  System.***out***.println(!(j > k) ^ (j < k));// false  // 三目运算符 (条件true/false)?(结果1 条件为true时) :(结果2 条件为false时)  // 将j、k中的最大值赋值给l,j>k若为true，则l=j,若为false，则l=k  **int** l = j > k ? j : k;  }  } |

优先级，有小括号先算小括号里面的

## 流程控制语句

#### 分支结构

If语句

If

If else

else 的隐含条件就是对前面条件取反。

If (else if) else

|  |
| --- |
| **public** **class** IfDemo {  **public** **static** **void** main(String[] args) {  **int** score = 80;  **if**(score >= 60) {  System.***out***.println("考试及格");  }  **if**(score >= 90) {  System.***out***.println("考试优秀");  } **else** {  // if那里的条件为false时  System.***out***.println("考试非优秀");  }  // 从上向下执行，只要有一个条件成立，那么其他的便不会再执行  **if**(score >= 90) {  System.***out***.println("考试优秀");  } **else** **if**(score >= 70){  // 隐含条件 score < 90  System.***out***.println("考试良好");  } **else** **if**(score >= 60){  // 隐含条件 score < 70  System.***out***.println("考试及格");  } **else** {  // 隐含条件 score < 60  System.***out***.println("考试不及格");  }  }  } |

#### Switch语句

Switch-case-break

1、使用break语句，防止case穿透  
2、 default可以省略，但不推荐省略  
3、 switch语句中控制表达式的类型只能是byte、short、char、int、String（JDK7新增）和枚举

|  |
| --- |
| **public** **class** SwitchDemo {  **public** **static** **void** main(String[] args) {  **int** x = 6;  // 如果不加break,會出現case語句穿透的问题  // switch条件数据类型要求，byte short int char 枚举  // jdk 1.7及以上可以使用string类型  **switch**(x) {  **case** 1:  System.***out***.println("x = 1");  // 跳出switch语句  **break**;  **case** 2:  System.***out***.println("x = 2");  **break**;  **case** 3:  System.***out***.println("x = 3");  **break**;  **case** 4:  System.***out***.println("x = 4");  **break**;  **default**:  System.***out***.println("x未知");  **break**;  }  System.***out***.println("跳出switch语句后执行");  String str = "a";  // switch根据条件到case语句中寻找匹配项，找到匹配项之后便执行其对应的语句  // 若遇到break，则跳出switch语句，继续想向下执行，若在所有的case中没有找到  // 匹配项，则会执行里面的default内容，default可以排switch语句中的任何位置  **switch**(str) {  **case** "a" :  System.***out***.println("str = a");  **break**;  **case** "b":  System.***out***.println("str = b");  **break**;  **default**:  System.***out***.println("str unknown");  **break**;  }  }  } |

# 循环语句

#### While

#### Do-while

|  |
| --- |
| **public** **class** WhileDemo {  **public** **static** **void** main(String[] args) {  // **TODO** Auto-generated method stub  // 用来循环计数并运算  **int** i = 1;  // 存储计算的和  **int** total = 0;  // 1-100之间所有数的和  // i = 1;i<=100; total = 0=>total=1; ++i=>i=2  // i = 2;i<=100; total = 1=>total=3; ++i=>i=3  // i = 3;i<=100; total = 3=>total=6; ++i=>i=4  // .... ++i=>i=100  // i=100;i<=100; total =..=>total=5050; ++i=>i=101  // i=101; i <=100=>false跳出while循环，继续while之后的  // 语句开始执行  // 先判断条件成立不成立，若成立在执行循环体，否则就不执行  **while**(i <= 100) {  total = total + i;  // i = i + 1;  ++i;  System.***out***.println("===========");  }  System.***out***.println(total);  // do while  // 不管while循环条件成立还是不成立，先执行循环体（循环执行的内容），  // 在执行判断  **do** {  total = total + i;  ++i;  }**while**(i <= 100);  System.***out***.println(total);  // 写循环时，一定要避免死循环，给循环一个出口条件  }  } |

#### For

|  |
| --- |
| **public** **class** ForDemo {  **public** **static** **void** main(String[] args) {  **int** total = 0;  // 求1-100所有整数的和  // int i = 1声明一个变量i，初始值为1  // i <= 100 为真，则执行循环体total+=i =》total=1; i++=>i = 2  // i <= 100为真，则执行循环体total+=i =》total=3; i++=>i = 3  // ... i++=>i = 100  // i <= 100为真，则执行循环体total+=i =》total=..;i++=>i = 101  // i <= 100为假，不在执行循环，继续向下执行  // i的作用域从for循环大括号开始，到大括号结束，如果想让i作用域突破  // for循环，则需要将i写在for循环的外面。  // int i = 1;  **for**(**int** i = 1; i <= 100; i++){  total += i;// total = total + i  }  System.***out***.println(total);  // 嵌套的for循环  **for**(**int** i = 1; i <= 10; i++){  **for**(**int** j = 0; j < 10; j ++) {  System.***out***.println("i = " + i + ", j = " + j);  }  }  }  } |

#### 99乘法表

|  |
| --- |
| //外层循环，负责显示多少行，9行  **for**(**int** i = 1; i <= 9; i ++) {  // 内层循环，负责显示每一列数据，因为每一行多出一列，所以这里<=i  **for**(**int** j = 1; j <= i; j++) {  // 如果所得到的积为2位数，则保留，为一位数则在其后加一个空格  String str = "" + ((i\*j)>9?i\*j:i\*j + " ");  // 输出列值  System.***out***.print(i + "\*" + j + "=" + str + " ");  }  System.***out***.println();  } |

## break

Break跳出循环，继续向下还行

Total=1+2+3…+9

|  |
| --- |
| **for**(**int** i = 1; i <= 100; i++){  **if**(i==10){  // 跳出当前for循环  **break**;  }  total += i;// total = total + i  } |

## Continue

Continue跳过本次循环，继续下一次循环

Total=1+2+3+…+9+11+12+….

|  |
| --- |
| **for**(**int** i = 1; i <= 100; i++){  **if**(i==10){  // 跳出当前for循环  **continue**;  }  total += i;// total = total + i  } |

## Scanner与随机数

|  |
| --- |
| **public** **class** ReturnDemo {  **public** **static** **void** main(String[] args) {  // Math java.lang  // 0-99的随机数  **int** j = (**int**) (Math.*random*() \* 100);// [0.0,1.0)->[0.0,100）  System.***out***.println(j);  // 快捷键导包，ctrl + shift + o  Scanner scanner = **new** Scanner(System.***in***);  // nextInt是一个方法名,方法是阻塞型的方法，  // 方法执行到这里，必须等待用户从键盘输入数据  **int** i = scanner.nextInt();  **if** (i > j) {  System.***out***.println("偏大");  } **else** **if**(i == j){  System.***out***.println("猜对了");  } **else** {  System.***out***.println("偏小");  }  }  } |

## 作业：

1、求1-100所有整数的的和（多思考）

2、输出100以内的素数

3、输出9x9的乘法表

4、猜数游戏：随机生成一个数，然后让用户从键盘读入这个数，提示用户猜大了还是猜小了，直到猜中

5、编写一个Java应用程序。用户从键盘输入一个1—9999之间的数，程序将判断这个数是几位数，并判断这个数是否是回文数。回文数是指将该数含有的数字逆序排列后得到的数和原数相同，例如12121、3223都是回文数

GZT-英文名-第几天作业（只要打包src目录下的东西）