Java程序块的组成

控制流程

运算符

变量

数据类型

Java程序块块

**数据类型：**确定要存储在内存中的数据类型

**变量：**存储数据的基本单元

**运算符：**是一个符号，用于操作一个或多个参数以得出结果程序是按顺序执行的，**控制流语语句**允许改变此顺序

## Java的基本语法

修饰 class 类名 {

程序代码

}

HelloWorld示例:

**public** **class** HelloWorld {

**public** **static** **void** main(String[] args) {

System.***out***.println("Hello World");

}

}

在编写java代码时，需要特别注意几个关键：

1. java中的程序代码可分为结构定义语句和功能执行语句；
2. java语言严格区分大小写
3. 在编写代码时，为了便于阅读，通常会使用一种良好的格式进行排版
4. Java程序中一句连续的字符串不能分开在俩行中书写

注释：

1. 单行注释

int c = 10; //定义一个整型变量

2.多行注释

/\*

int c = 10; int x = 5;

\*/

标示符：标识符可以由任意顺序的大小写字母、数字、下划线（\_）和美元（$）组成，但标识符不能以数字开头

定义标示符号时还应该遵循以下规则：

1. 包名所有字母一律小写
2. 类名和接口名每个单词的首字母都要大写
3. 常量名所有字母都大写，单词之间用下划线连接
4. 变量名和方法名的第一个单词首字母小写，从第二个单词开始每个单词首字母大写
5. 在程序中，应该尽量使用游意义的英文单词来定义标识符，使得程序便于阅读。

## 变量

在程序运行期间，随时可能产生一些临时数据，应用程序会将这些数据保存在内存单元中，每个内存单元都用一个内存单元来标识。这些内存单元被称为变量，定义的标识符就是变量名，内存单元中存储的就是变量值。

int a = 1;

int b = a+a;

int age = 20;

数据类型 变量名 初始值

**变量的命名规则**

**变量要能体现出变量的作用；**

**变量名由1个或多个英文单词组成**

**boolean量一般用is开头**

**变量的作用域**

**Public static voi main(String[] args) {**

**int x = 4;**

**{**

**int y = 9;**

**}**

**}**

**上面的代码中有俩层大括号。其中，外层大括号所标识的代码区域时变量x的作用域，内层大括号所标识的代码区域就是变量y的作用域**

## 修饰符

Java可以使用修饰符来修饰类中方法和属性。主要有两类修饰符：

* 访问控制修饰符 : default, public , protected, private
* 非访问控制修饰符 : final, abstract, strictfp

## 关键字

|  |  |  |
| --- | --- | --- |
| **类别** | **关键字** | **说明** |
| 访问控制 | private | 私有的 |
| protected | 受保护的 |
| public | 公共的 |
| 类、方法和变量修饰符 | abstract | 声明抽象 |
| class | 类 |
| extends | 扩允,继承 |
| final | 最终值,不可改变的 |
| implements | 实现（接口） |
| interface | 接口 |
| native | 本地，原生方法（非Java实现） |
| new | 新,创建 |
| static | 静态 |
| strictfp | 严格,精准 |
| synchronized | 线程,同步 |
| transient | 短暂 |
| volatile | 易失 |
| 程序控制语句 | break | 跳出循环 |
| case | 定义一个值以供switch选择 |
| continue | 继续 |
| default | 默认 |
| do | 运行 |
| else | 否则 |
| for | 循环 |
| if | 如果 |
| instanceof | 实例 |
| return | 返回 |
| switch | 根据值选择执行 |
| while | 循环 |
| 错误处理 | assert | 断言表达式是否为真 |
| catch | 捕捉异常 |
| finally | 有没有异常都执行 |
| throw | 抛出一个异常对象 |
| throws | 声明一个异常可能被抛出 |
| try | 捕获异常 |
| 包相关 | import | 引入 |
| package | 包 |
| 基本类型 | boolean | 布尔型 |
| byte | 字节型 |
| char | 字符型 |
| double | 双精度浮点 |
| float | 单精度浮点 |
| int | 整型 |
| long | 长整型 |
| short | 短整型 |
| null | 空 |
| 变量引用 | super | 父类,超类 |
| this | 本类 |
| void | 无返回值 |
| 保留关键字 | goto | 是关键字，但不能使用 |
| const | 是关键字，但不能使用 |

## 数据类型



|  |  |  |
| --- | --- | --- |
| 数据类型 | 占用空间 | 保存范围 |
| boolean布尔型 | 1字节 | true或false |
| byte字节型 | 1字节 | -128~127 |
| char字符型 | 2字节 | 0~65535 |
| short短整型 | 2字节 | -32768~32767 |
| int整型 | 4字节 | -2147483648~2147483647 |
| Long常整型 | 8字节 | -9223372036854775808~9223372036854775807 |
| float浮点型单精度 | 4字节 | 3.402823e+38~1.401298e-45 |
| double浮点型双精度 | 8字节 | 1.797693e+308~4.90000003-324 |

## 运算符

**1.运算符**

|  |  |
| --- | --- |
| 运算符 | 说明 |
| + | 加法运算符 |
| - | 减法运算符 |
| \* | 乘法运算符 |
| / | 除法运算符 |
| % | 求余数运算符 |
| ++ | 自增运算符 |
| -- | 自减运算符 |

**public** **class** HelloWorld {

**public** **static** **void** main(String[] args) {

//System.out.println("Hello World");

**int** a = 10;

System.***out***.println(a++);

System.***out***.println(++a);

}

}

结果：

10

12

**2.复合赋值运算符**

|  |  |
| --- | --- |
| 表达式 | 等价代码 |
| a+=3 | a=a+3 |
| a-=3 | a=a-3 |
| a\*=3 | a=a\*3 |
| a/=3 | a=a/3 |
| a%=3 | a=a%3 |

**3.逻辑运算符**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **操作符** | **描述** | | **例子** | |
| && | | 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 |

public class Test {

public static void main(String[] args) {

boolean a = true;

boolean b = false;

System.*out*.println("a && b = " + (a&&b));

System.*out*.println("a || b = " + (a||b) );

System.*out*.println("!(a && b) = " + !(a && b));

}

}

**4.条件运算符**

variable x = (expression) ? value if true : value if false

public static void main(String[] args){

int a , b;

a = 10;

// 如果 a 等于 1 成立，则设置 b 为 20，否则为 30

b = (a == 1) ? 20 : 30;

System.out.println( "Value of b is : " + b );

// 如果 a 等于 10 成立，则设置 b 为 20，否则为 30

b = (a == 10) ? 20 : 30;

System.out.println( "Value of b is : " + b );

}

**5. instanceof 运算符**

对象 **instanceof （class/interface）**

String name = "name";

boolean result = name instanceof String; // 由于 name 是 String 类型，所以返回真

System.out.println(result);

**6.数据类型转换**

**低byte、char、short、int、long、float、double高**

**在同一个表达式中，如一个操作数的类型于其他操作数据类型共同存在，则低级别类型的数据自动提升为该机类型的数据**

**强制类型转换有以下几点要求：**

**转换的源数据类型和目标数据类型要兼容**

**浮点类型掐指变量转换为整型时，直接将小数位舍去**

**高级别数据类型强制转换为低级别的数据类型时，数据可能会溢出或者精确度下降**