doc.canglaoshi.org

**一、java开发**

java是一门高级的编程语言，计算机只认识0和1

编译运行的过程（重点）

编译期：.java（源文件）---编译---生成.class字节码

运行期：JVM负责加载并运行.java字节码文件

名词：

JVM：java的虚拟机，负责加载并运行.class字节码文件

JRE：java的运行环境，除了包含JVM，还包含java运行时所需要的环境工具

JRE=JVM+JAVA系统类库（类库理解为工具）

JDK：java开发工具，除了包含JRE，还包含java编译运行等工具

JDK=JRE+编译/运行所需要的工具

如果电脑仅仅只是想运行Java程序，至少需要给电脑装JRE

如果电脑想要开发java程序，需要给电脑安装JDK

**二、编译器工具**

用编译器工具可以快速的开发一个java程序

Eclipse （手动挡）和 IDEA（自动挡）

需要ctrl+s保存 自动保存

IDEA是JetBarine公司提供的两个版本

免费版和收费版

java语言就是开源的，开源：对用户开放IDEA（免费版），Eclipse（开源的）

java -version 查看java是否安装好

**三、新建项目**

开发步骤：

1、新建一个java工程/项目 --------小区

2、新建一个java的包 --------楼/单元

都在src文件夹创建包（package）

3、新建一个java的类 --------房子

在创建好的package下面新建类（class）

java的注释：

1、单行注释：//写上解释性文本

2、多行注释：/\*\*/

3、文档注释：写日志等 /\*\*写上文档注释 \*/

左右{}：代码需要写在{}里面

main功能------主要负责程序运行，也就意味着我们想要执行的代码要放到main里面

main功能快捷键-----在英文状态下输入main回车即可

代码逐语句执行 第一行---第二行---第三行

sout功能------println ------打印信息并显示在下方窗口中的功能

" "在双引号中间写你要输出的信息，需要注意分号必须英文 ;

1、java是区分大小写的！

2、写代码期间，输入法必须是英文

3、每句话写完必须分号结尾

Java编译方式有三种：

Compile------对类进行编译（不管类中的内容是否修改过，一定会编译）（不用）

Rebuild------对项目进行编译（不管项目是否修改过，一定会全部编译）（不用）

Make------默认的编译方式（只编译修改过的文件）

**四、变量----可以变化的数据**

**变量（存数据的）**

1、变量的声明：在银行开了个账户

int a;//声明一个int（整数）变量 变量的名字为a

int b,c,d;//声明了三个int的变量 变量名为 b c d

int a;在声明一个a会出先同一个问题吗？----不可以，编译错误：已存在名为a的变量

2、变量的初始化：给账号存钱

复制号： =

int e = 100;//声明了一个int的变量名 e 里面存储了100

int f;

f = 200;//初始化为f赋值数据200

3、变量的使用：使用数据，对变量的使用

int g = 5;//声明一个int 变量名为g 并初始化数据为 5

int h = g + 10;//声明变量h ，里面存的数据是 g的数据+10的运算结果

sout("g");//双引号所见即所得

sout(g);//打印变量g中的数据

sout(G);//和g有关系吗，会报错吗（编译错误，变量在使用之前需要声明）

4、变量的命名规范

变量的命名只能包含数字、字母、\_和$（不能以数字开头）

变量严格区别大小写！！！

变量的命名不能以关键字来进行命名

变量名需要见名知意

Java变量名支持中文，不建议使用，建议使用英文

变量的命名使用小驼峰命名法 单词首字母小写，遇到第二个首字母大写

类名的命名规范大驼峰命名法 单子首字母大写，第二个首字母大写

**五、数据类型**

数据类型的作用：决定了当前这个变量可以存储的类型和存储的范围

数据源类型分为两种：基本数据类型 和 引用数据类型（后续讲）

八种基本数据类型：

整数：byte，short，int，long

小数：float，double

字符：char

布尔：boolean（True或者false）

常用的基本数据类型：

整数类型：int，long

小数类型：double

字符类型：char

布尔类型：boolean

不常用：byte（之后会用到），short/float（保留C语言中的类型，工作不同）

------------------------------------------

GB（吉字节）

1GB=1024MB

MB（兆字节）

1MB =1024KB

KB（千字节）

1KB = 1024b

b（字节）

1b=8bit（位）

-------------------------------------------

**整数类型：**

int：在内存中占用4个字节，存储数据 -21亿多~~21亿多（-2147483648----2147483647）

1.整数的直接两默认为int类型，直接量是不可以超出当前类型可存储的范围，否则会报错

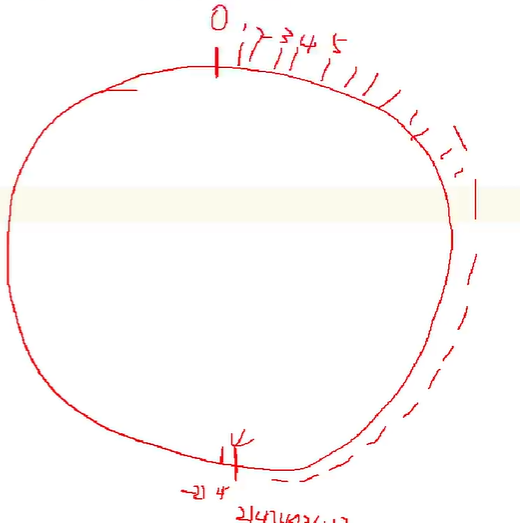
2.整数运算时，结果不会保留小数位（截断删除，不会四舍五入） 5/2

3.整数运算时，若运算结果超出可容纳的范围，则会溢出（溢出不报错），最好不要溢出，数据不准确

（存一个int最大值+1，查看结果）最大值+1 -----溢出为最小值

最大值+2 最小值+1（规律）

一个圆 从1到最小



---------------------------------------------------

long类型：在程序中占8字节，存储范围-900万万亿~~~900万万亿

1.长整数类型的直接两，需要在后面加上大写的L，小写的 l 也可以，避免与1区分不了

long a = 100000000000;报错整数默认为int类型

long a = 100000000000L;不会报错，转为long数据类型

long b = 1000000000（10亿）\* 2 \* 10L;

long c = 1000000000 \* 3 \* 10L;会报错三十亿已经超过int类型

long d = 1000000000L \* 3 \* 10;

long d = 1000000000L \* 3L \* 10L;

2.建议在处理长整数运算时，要在第一个数据后方加上L即可，这样的话整体的会员算结果全部都是长整数，避免报错

int a = 5;

int b = 123445； //a与b占用的空间大小是否是一样的

//声明int占用的都是4个字节，变量占用多大内存，取决于当前变量的数据类型，跟存储的数据无关

**小数类型**

folat（单倍精度）

double（双倍精度），占用8个字节，可存储的范围，天文数字级别，比long大无数倍

1.浮点型直接量默认为double类型，可以在后面加D，默认就是，很少加

double a = 3.14;

double a = 3.14D;

double c = 3.0;

double d = 2.9;

c -d =?不确定（底层原理可能会存在误差），除了银行项目，其余不太在意

double e = 6.0;

double f =4.9;

e - f = ?浮点数，会上下浮动，误差很小

double g = 6.0;

double h =1.9;

g -h = ? 1.9 可能存在误差，也可能不存在误差

2.小数直接量可以在后面加D

3.double类型的数据在进行运算时，可能存在误差，但影响可以忽略不记！！，java有四舍五入的功能，金融行业用BigDecimal

double r = Math.round(4.5);//（Math数学方面的功能）

**字符类型**

char：占用了两个字节，可以存储字符。但是一个char变量只能存一个

字符是指：键盘上的字母，数字，符号.....每个国家的字符都不同

Unicode：同一的字符码（万国码），Unicode中划分了很多个子集，java中char用的就是最常用的子集（ACSLL）码表

1.一个字符对应一个码：表现看是字符，本质上是码（0~65535）之间

2.字符类型的直接量，要用单引号来进行存储，且单引号内只能放单个字符

char temp = 'a'//声明一个char类型的变量，里面存储了a

sout((int)temp)转换为码

char t2 = 97;

sout(t2);//a

3.java提供了特殊符号，如果需要存储，需要转义，转义符 \

//存单引号 '

char T3 = '\''

存\

char T4 = '\\'

**布尔类型;**

boolean类型：占用1个字节，要么存ture，要么false（真成立/假不成立）

boolean a = ture;

boolean b = false; // 不能存其他的

**类型转换**

JAVA分为两种类型转换

自动类型转换（隐式转换）：将小的类型装载到（赋值到）大类型的过程

从小到大：byte--short--int--long--float--double

int a = 100;

long b = a;//隐式转换，从小到大

强制类型转换（显示转换）：将大的类型装载到（赋值到）小类型的过程

long a = 100;

int b = a;//会报错

long字节比较大，如果一定需要转换，需要强转，

long a = 10000000000000L;

int b = (int)a;

如果long存的值超过int类型，会报错，强转有风险，做的时候要心知肚明（放在红线上有提示）

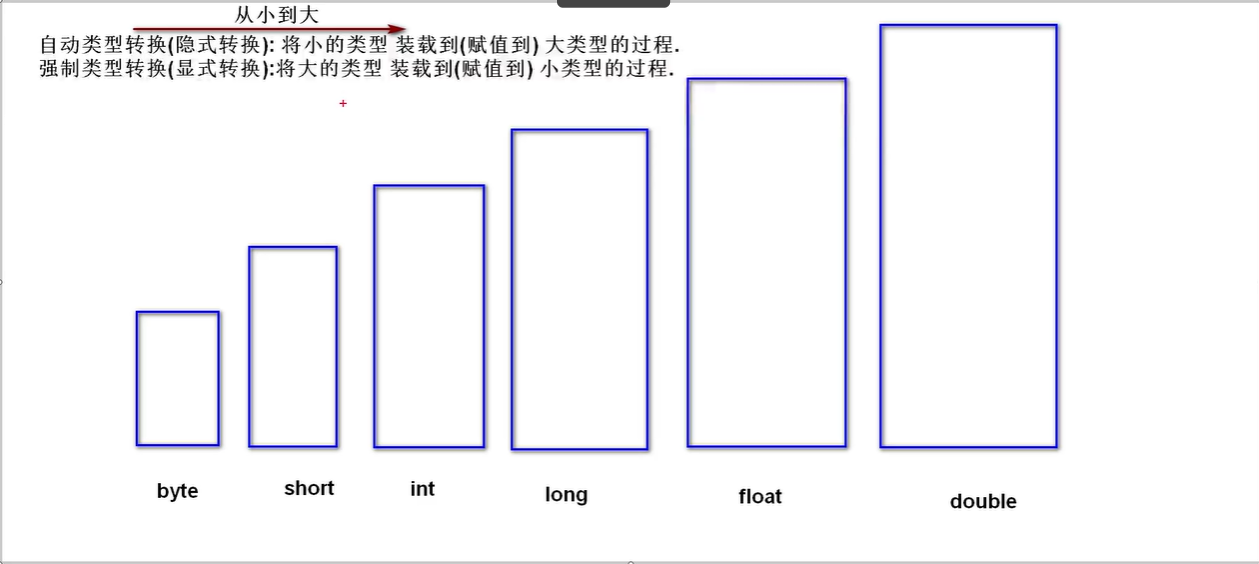
溢出的话会出现不知道的数据

long f = 2147463647;

int g = （int）f;

double h = 3.1415926;

int i = （int）h;//i这个数据是多少？ 3



规则：

1.小类型与大类型进行运算时，系统会自动将类型和结果转换为大的类型

int a = 10;

long b = 20L;

int c = a+b;//报错

int c = （int）（a+b）;

long c = a+b;

2.整数直接量是可以直接赋值给byte，short，char类型的！！但不能超过当前接受数据的类型范围

byte -128~~127 char 0~~65535

3.byte，short，char类型，如果进行运算时，系统会自动将其结果转换为int类型

byte a = 10;

byte b = 10;

byte e = c+d;//会报错，因为是大转小

char f = '我';

char h = '你';

int g = f + h;

**运算符**

**1.数学运算符**

符号： + （加） - （减） \* （乘） / （除） %（取余）

**2.自增自减运算符**

自增(++)：在变量的数据自身基础上增加1

自减(--)：在变量的数据自身基础上减少1

作用域变量

（1）单独运算时（只有当前变量被运算），符号在前在后都一样

//单独运算

int a = 5;

a++;

++a;

（2）参与运算时（赋值也算运算），符号在前在后不一样

符号在前，就先执行符号对应的自增或自减，然后再参与其他运算

符号在后，先参与其他运算，在执行符号对应的自增或自减

int a = 5;

int b = 5 ;

int c = a++;//5

int d = ++b;//6

3.关系运算符

大于（>） 小于（<） 大于等于（>=） 小于等于（<=） 等于（==） 不等于（!=）

关系运算符的判定结果都是boolean类型，要么是（ture）成立，要么是（flase）不成立

int a = 50; int b = 40;

boolean result = a > b;//ture

a的数据大于b的数据吗？-----------

int c = 50; int d = 50;

boolean result1 = a >= d;//或者的关系 true

int e = 100; int f = 99;

f++;//e==f吗

4.逻辑运算符

逻辑运算符的判定结果都是boolean类型，要么是（true）成立，要么是（flase）不成立

（1）逻辑与（并且）：符号 &&（见false即false）

（2）逻辑非（结果取反）：符号 ！（见false即true，见true即flase）（优化代码使用）

true : ！true（非真即假）//false

int a = 5;int b = 10;int c = 5;

boolean result = !(a>b && b>c);//true

int d = 5;int e = 6;

sout(d == e && ++d > 5);//false 从左到右判断

//判断年份是否是闰年的案例

//判断闰年的条件

//条件一：年份能被4整除并且夜年份不能被100整除

//条件二：年份能被400整除

//扫描器，扫描当前界面--控制台（显示台） 用户输入的内容

//Scanner s = new Scanner(System.in);创建一个扫描器

int year = s.nextInt();//获取用户在控制台中输入的int类型的数据

int year = 2020;//年份

boolean r = (year % 4 == 0 && year % 100 ！= 0)||(year % 400 == 0)

（3）逻辑或：符号 ||（见true即true）

短路逻辑

短路与

相亲成功的两个条件

条件一：必须是个女的 男的，不会向下走

并且

条件二：勤俭持家

现象：在与的关系中，如果前面的条件不成立，不会向下执行代码，直接返回false

5.赋值运算符

6.三元（三目）运算符

7.字符串拼接符