    2. switch
3. 循环结构

java笔记

woldf的java笔记，已经同步到这里

认识java

java简介

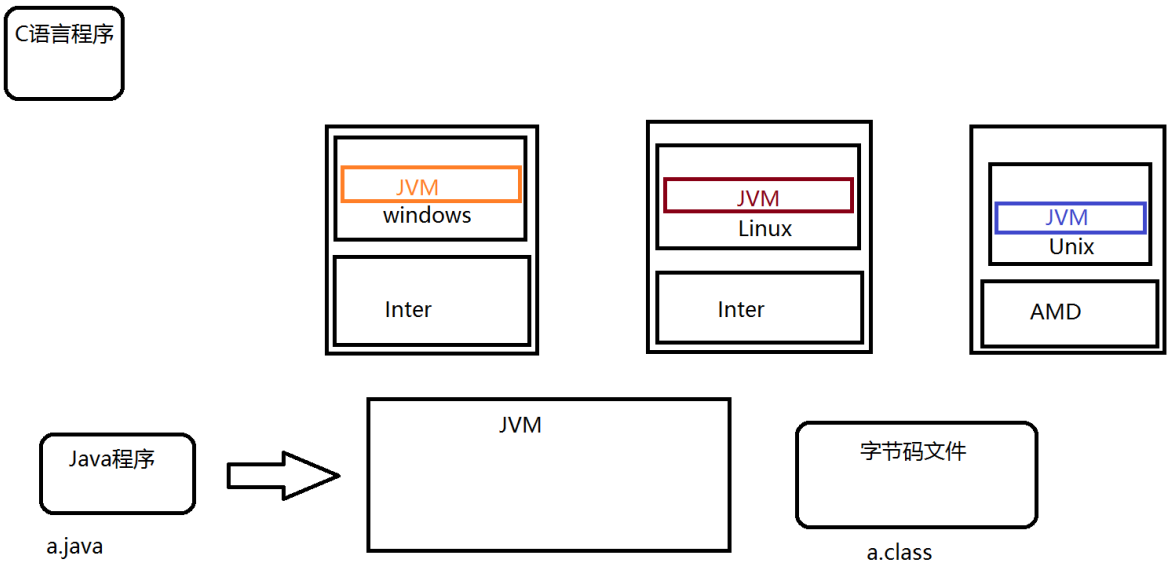
Java是一个非常流行的编程语言，发明于1991年，是Sun公司（现在被Oracle公司收购了）。

Java的分类

- java SE 标准版 主要是java的核心和基础，主要用在桌面应用开发，本学期学习的重点
- java EE 企业版 主要是用来开发网站后台程序（后端），下学期学习的重点 J2ee
- java ME 移动版 已经被淘汰，针对nokia手机

java的特点

- 简单
- 健壮
- 面向对象
- 网络功能
- 安全性
- 跨平台



java的执行过程：

- java源代码文件的扩展名：*.java
- java源代码文件经过JVM（java虚拟机）编译，生成*.class文件

编写java的hello world

- 编译器安装（jdk）
- jdk8（jdk1.8）最流行，最稳定的版本
- 9/10/11/12/15（最新的版本）
- jdk配置：
  - 配置java_home
  - 添加path

第一个java程序

笔记的代码输出结果基于Ubuntu 20.04（wsl2）

openjdk version "1.8.0_282"

- 编写A.java文件

```
public class A{
    public static void main(String[] args){
        System.out.println("hello java!");
```

```
    }
}
```

- 编译A.java文件`javac A.java`
- 执行`java A`

注意1

```
A.java:
public class B{
    public static void main(String[] args){
        System.out.println("hello java!");
    }
}
$javac A.java:
A.java:1: error: class B is public, should be declared in a file named B.java
public class B{
       ^
1 error
```

注意：文件名和公共类的类名必须一样

注意2

```
A.java:
Public class B{
    public static void main(String[] args){
        System.out.println("hello java!");
    }
}
$javac A.java:
A.java:1: error: class, interface, or enum expected
Public class A{
^
1 error
```

注意：

- java区分大小写
- public是关键字，是不能修改的。

注意3

```
A.java:
public class B{
    public static void main(String[] args){
        System.out.println("hello java!")
    }
}
$javac A.java:
A.java:3: error: ';' expected
        System.out.println("hello java!")
                                          ^
1 error
```

注意：

- java每一条语句必须以分号";"结尾
- java中所使用的标点符号都是英文 ,;.()[]{}:

注意4

```
A.java
public class A{
    public static void main(String[] args){
        //这是一条不会被执行的语句
        /*
        这是
        多行
        注释
        */
            System.out.println("hello java!");
    }
}
```

注意：

- 单行注释 //
- 多行注释 /*注释内容*/
- 注释的作用：
    - 解释程序代码
    - 不会被执行

问题：打印下面的图案

##########

##########

##########

##########

```
Test1.java
public class Test1{
    public static void main(String[] args){
            System.out.println("##########");
            System.out.println("##########");
            System.out.println("##########");
            System.out.println("##########");
    }
}
```

```
Test2.java
public class Test2{
    public static void main(String[] args){
            System.out.println("##########\n##########\n##########\n##########");
    }
}
```

java中的标识符与关键字

标识符的作用

给变量、方法、类起名字

标识符的要求

标识符由大小写字母，数字，下划线和美元符号（$）组成，但不能以数字开头

大小写敏感

不能与java语言关键字重名

不能和java类库的类重名

不能由空格、@、#、+、-、/等符号

长度无限制

建议使用有意义的名称

不要使用true或false

```
public class Test3{    public static void main(String[] args){
            int a3 = 19;//合法
            int 3a = 29;//不合法
            int a# = 18;//不合法 不能使用除大小写字母，数字，下划线和美元符号（$）以外的
符号
            int 价格 = 100;// 合法，可以用中文，但不建议
            int class = 1903;//不合法，不能使用关键字作为变量名
            int Class = 1907;//合法
            int $x = 20;//合法
    }
}
```

关键字

有特殊含义、固定不变的单词，都是小写

```
数据类型：byte、short、int、long、char、float、double、boolean
包引入和包声明：import、package
类和接口的声明：class、extends、implement、interface
流程控制：if、else、switch、case、break、default、while、for 、do、continue、return
异常处理：try、catch、finally、throw、throws
修饰符：abstract、final、private、protected、public、static、synchronized
其他：new、instanceof、this、super、void、enum
```

变量和常量

变量：可变

常量：固定不变

变量

变量声明

- 语法： `数据类型 变量名;`
- 举例：`int a;`

变量的定义与初始化

- 初始化：给变量赋值

`变量名称 = 值;`

`a=16;`

- 变量的定义：同时声明一个变量并且给它赋值

`数据类型 变量名称 = 值;`

`int a = 16;`

注意：

```
A.java
public class A{
    public static void main(String[] args){
        int a;
        System.out.println(a);
    }
}
$javac A.java:
A.java:4: error: variable a might not have been initialized
        System.out.println(a);
                           ^
1 error
```

a未赋值，报错

在使用一个变量之前必须先定义

数据类型

数据类型：确定变量所占用的内存空间大小，也可以确定变量所能够做的操作。

基本数据类型：byte boolean char short int long float double

引用数据类型：数组，类，接口

java数据类型分为基础类型（简单类型）和引用类型

数据类型	关键字	在内存中占用 的位数	取值范围	成员默认值
字节型	byte	8	-128~127	(byte)0
短整型	short	16	-32768~32767	(short)0
整型	int	32	-2的31次方~2的次方31-1	0
长整型	long	64	-263~263-1	0L
字符型	char	16	0~65535	‘\u0000’
单精度浮点型	float	32	1位符号,8位指数,23位尾数	0.0F
双精度浮点型	double	64	1位符号,11位指数,52位尾数	0.0D
布尔型	boolean	1	true，false	false

注意：计算机内部使用二进制数据，整型数据采用补码

```
public class test{
    public static void main(String[] args){
        byte  a = 128;
        System.out.println(a);
    }
}
$javac:
test.java:3: error: incompatible types: possible lossy conversion from int to byte
        byte  a = 128;
                  ^
1 error
```

注意：

- 在定义变量或给变量赋值时要注意变量的数据类型和数值大小匹配

```
public class test{
    public static void main(String[] args){
        char  a = '中文';
        System.out.println(a);
    }
}
$javac test.java
test.java:3: error: unclosed character literal
char  a = '中文';
^
test.java:3: error: unclosed character literal
char  a = '中文';
^
test.java:3: error: not a statement
char  a = '中文';
^
3 errors
```

```
public class test{
    public static void main(String[] args){
        char  a = "中";
        System.out.println(a);
    }
}
$javac test.java
test.java:3: error: incompatible types: String cannot be converted to char
        char  a = "中";
                  ^
1 error
```

注意：

- 在java中char类型占用2B空间，所以它能保存中文（一个），而且要注意，char类型要用英文单引号

```
public class test{
    public static void main(String[] args){
        float a = 3.14;
        System.out.println(a);
    }
}
$javac test.java
test.java:3: error: incompatible types: possible lossy conversion from double to
float
        float a = 3.14;
                  ^
1 error
```

注意：

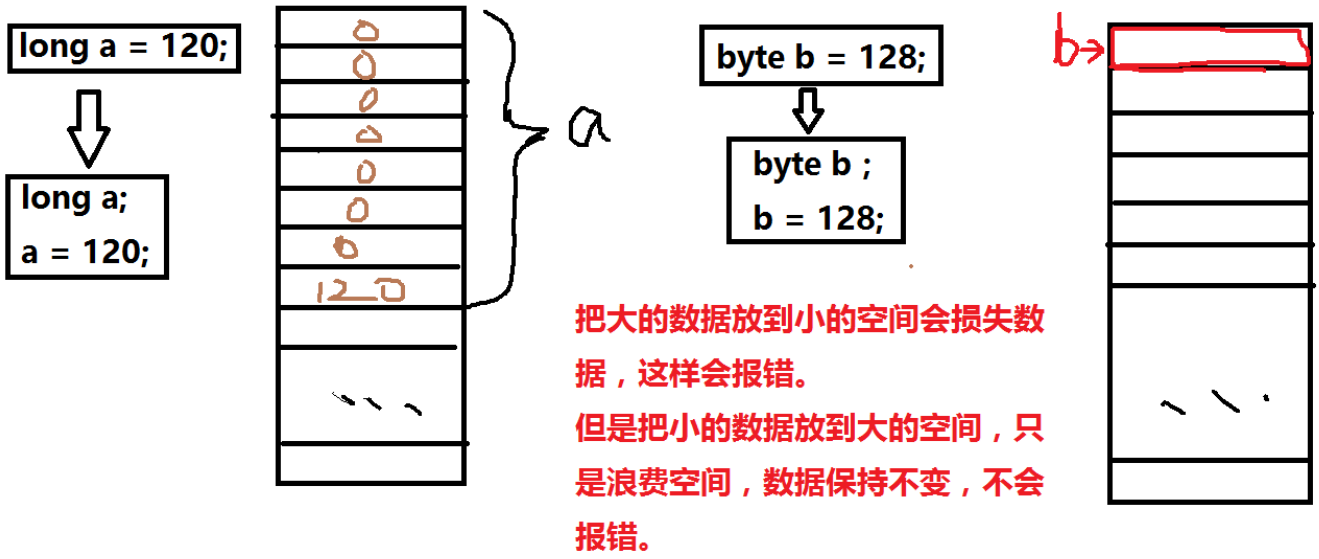
- java语言中，所有的浮点数都是double类型
- 若表示浮点数float则需要在数据后面添加f或F 例如： `float a = 3.14f;`

```
public class test{
    public static void main(String[] args){
        long a = 120;
        System.out.println(a);
    }
}
//不报错
```

注意：

将内存占用小的数据类型赋值给内存大的不会报错，但是会浪费空间；

可以在数据类型后面添加l或L表示long类型：`long a = 120L`




```
public class test{
    public static void main(String[] args){
        boolean a = true;//不会报错
        boolean b = false;//不会报错
        boolean c = 1;//会报错
        System.out.println(a);
    }
}
$javac test.java
test.java:5: error: incompatible types: int cannot be converted to boolean
        boolean c = 1;
                    ^
1 error
```

常量

字面常量：

- 12 3.14 true 'h' -129

符号常量：

- final 数据类型 变量名称 = 值;

```
public class test{
    public static void main(String[] args){
        final int a = 123;
        a = 127;
        System.out.println(a);
    }
}
$javac test.java
test.java:4: error: cannot assign a value to final variable a
        a = 127;
        ^
1 error
```

注意：不能给常量重新赋值

```
public class test{
    public static void main(String[] args){
        final int b;
        b = 129;
        System.out.println(b);
    }
}
```

注意：不会报错， 第一次赋值不会报错

```
public class test{
    public static void main(String[] args){
        //final int a = 123;
        //a = 127;
        final int b;
        b = 129;
        b = 130;
        System.out.println(b);
    }
}
$javac test.java
test.java:7: error: variable b might already have been assigned
        b = 130;
```

```
        ^
1 error
```

注意：会报错，不能给常量重新赋值

```
public class test{
    public static void main(String[] args){
        final int b;
        b = 129;
        b = 129;
        System.out.println(b);
    }
}
$javac test.java
test.java:7: error: variable b might already have been assigned
        b = 129;
        ^
1 error
```

注意：会报错，不能给常量重新赋值

表达式与运算符

表达式

- 表达式：就是常量、变量、运算符的组合；

运算符

- 运算符的作用：指明对操作数的运算方式。
- 按操作数的个数分为：
  - 单目运算符(-a)
  - 双目运算符(a+b)
  - 三目运算符(a?c:b)
- 按功能分类
  - 算数运算符 +，—，*，/，%，++，——
  - 关系运算符 >，<，>=，<=，==，!=
  - 逻辑运算符 !，&&，||，&，|
  - 赋值运算符 =，+=，—=，*=，/=等

    - 位运算符
    - 条件运算符 ?:
    - 其他 ·， []，instanceof， () 等
- 运算符的优先级别

算数运算符

`+ - * / % ++ --`

- +表示做算数加法或字符串拼接

```
public class test{
        public static void main(String[] args){
                int a = 2;
                double  b = 3;
                int c = a+b ;
                System.out.println(c);
        }
}
$javac test.java
test.java:5: error: incompatible types: possible lossy conversion from double to
int
                int c = a+b ;
                          ^
1 error
```

注意：执行任何运算首先统一数据类型，需要类型转换

类型转换：

- 自动类型转换：小的数据类型自动转换为大的数据类型。int + duble --->double + double
- 强制类型转换

修改方法：

```
//方法1
public class test{
        public static void main(String[] args){
                int a = 2;
                double  b = 3;
                double c = a+b ;//int 改为 double
                System.out.println(c);
        }
}
//结果: 5.0
```

```
//方法2
public class test{
        public static void main(String[] args){
                int a = 2;
                double  b = 3;
                int c = int(a+b);//强制类型转换
                System.out.println(c);
        }
}
//结果: 5
```

强制类型转换的语法：(目标数据类型)需要转换的值

- *表示算数乘法 要实现y=2x+1表达式为y=2*x + 1
- /表示算数除法,(注意不要写成反斜杠\)

```
public class test{
        public static void main(String[] args){
                int a = 5;
                int b = 2;
                int c = a/b;// 5/2 int/int=int
                System.out.println(c);
        }
}
//结果:2
```

```
public class test{
        public static void main(String[] args){
                int a = 5;
                int b = 2;
                double c = a/b;// int/int=int,然后进行类型转换为double,所以先得到结果
2,再转换为2.0
                System.out.println(c);
        }
}
//结果:2.0
```

```
public class test{
        public static void main(String[] args){
                int a = 5;
                double b = 2;
                double c = a/b;// int/double=double/double 所以得结果2.5
                System.out.println(c);
        }
}
//结果:2.5
```

注意:

- 两个int类型做除法,结果一定是int类型;而且计算结果很有可能是错的(与数学计算结果不一致)
- 在除法中,只要有一个是小数类型,结果一定是小数类型,而且计算结果和数学计算一致

```
public class test{
        public static void main(String[] args){
                double c=5/2.;
                System.out.println(c);
        }
}
//结果:2.5
```

```
public class test{
        public static void main(String[] args){
                double c=5/.2;
                System.out.println(c);
        }
}
//结果:.2
```

注意:2.0和2.是一样的;0.2和.2是一样的

- %表示取余

```
public class test{
        public static void main(String[] args){
                int a=13%3;
                System.out.println(a);
        }
}
//结果:1
```

```
public class test{
        public static void main(String[] args){
                int a=-13%3;
                System.out.println(a);
        }
}
//结果:-1
```

```
public class test{
        public static void main(String[] args){
                int a=-13%-3;
                System.out.println(a);
        }
}
//结果:-1
```

```
public class test{
        public static void main(String[] args){
                double a=-13%3.2;
                System.out.println(a);
        }
}
//结果:-0.1999999999999993
```

- ++自增一

```
public class test{
        public static void main(String[] args){
                int a = 7;
                a++;
                System.out.println(a);
        }
}
//结果:8
```

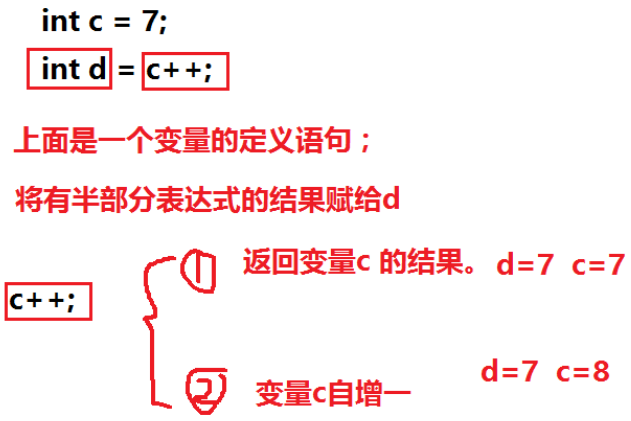
```
public class test{
        public static void main(String[] args){
                int a = 7;
                ++a;
                System.out.println(a);
        }
}
//结果:8
```

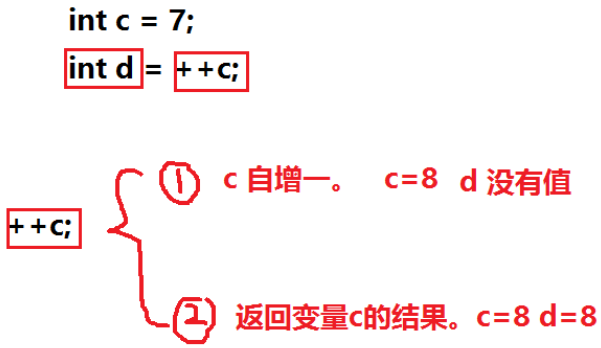
注意:单独使用时,++a与a++无任何区别

```
public class test{
        public static void main(String[] args){
                int c = 7;
                int d = a++;
                System.out.println(c);//8
                System.out.println(d);//7
        }
```

```
}
//结果:8;7
```

```
public class test{
        public static void main(String[] args){
                int c = 7;
                int d = ++a;
                System.out.println(c);//8
                System.out.println(d);//8
        }
}
//结果:8;8
```





- --自减一(同上)

关系运算符

`>, <, >=, <=, ==, !=`

结果一定是boolean,值为true或false

```
public class test{
        public static void main(String[] args){
                int a = 2;
                int b = 3;
                int c = a > b;
                System.out.println(c);
                //System.out.println(b);
        }
}
$javac test.java
test.java:5: error: incompatible types: boolean cannot be converted to int
                int c = a > b;
                        ^
1 error
```

- 注意:关系运算符的结果是boolean类型,在Java中,boolean类型的值泵赋给int类型的变量

```
public class test{
        public static void main(String[] args){
                int a = 2;
                int b = 3;
                System.out.println(a>b);
        }
}
结果:false
```

```
public class test{
        public static void main(String[] args){
                int a = 3;
                int b = 3;
                boolean c = a >= b;
                System.out.println(c);
        }
}
结果:true
```

```
public class test{
        public static void main(String[] args){
                int a = 3;
                int b = 3;
                boolean c = a == b;
                System.out.println(c);
        }
}
结果:true
```

逻辑运算符

`!, &&, ||, &, |`

- and(与)&&
- or(或)||
- not(非)!

```
public class test{
        public static void main(String[] args){
                int a = 3;
                int b = 5;
```

```
                System.out.println(a && b);
        }
}
test.java:6: error: bad operand types for binary operator '&&'
                System.out.println(a && b);
                                   ^
  first type:  int
  second type: int
1 error
```

- 注意:逻辑运算符的操作数必须是boolean类型,结果也是boolean

```
public class test{
        public static void main(String[] args){
                int a = 23;
                System.out.println(18 <= a <=35);
        }
}
test.java:4: error: bad operand types for binary operator '<='
                System.out.println(18 <= a <=35);
                                          ^
  first type:  boolean
  second type: int
1 error
```

```
public class test{
        public static void main(String[] args){
                int a = 23;
                System.out.println(18 <= a && a<=35);
        }
}
```

注意:逻辑运算符使用场合关系表达式 逻辑运算符 关系表达式

```
public class test{
        public static void main(String[] args){
                int a = 23;
                int b = 39;
                boolean c = a >b || a >50 && b <30;
                System.out.println(c);
        }
}
false
```

注意:&&的优先级高于||

```
public class test{
        public static void main(String[] args){
                int a = 23;
                boolean c = a> 10 || a++ <25;
                System.out.println(c);//true
                System.out.println(a);//23
        }
}
```

注意:短路效应

- true||其他为了节省时间,java不会执行||右侧的其他部分
- false && 其他为了节省时间,java不会执行&&右侧的其他部分

```
public class test{
        public static void main(String[] args){
                int a = 23;
                boolean c = a< 10|| a++ <25;
                System.out.println(c);//true
                System.out.println(a);//24
        }
}
```

赋值运算符

=, +=, -=, *=, /=

```
int a = 3;
a = a + 2;
a += 2;  //   a += 2  ---->    a = a + 2;
a *= 2;  //   a *= 2 ---->     a = a * 2;
a -= 2; //    a -= 2 ---->     a = a - 2;
a /= 2; //    a /= 2 ---->     a = a / 2;
```

```
public class test{
        public static void main(String[] args){
        int a = 3;
                int b = 2;
                int c = 4;
                a *= b+c;
                System.out.println(a);//18
        }
}
a = a * (b+c)
```

赋值运算符优先级最低

=和==的区别:

- =是赋值运算符,把右边的值给左边的变量
- ==是关系运算符,是比较左右两边的是否相等

===在js中存在

位运算符

| & ~ ^

```
public class test{
        public static void main(String[] args){
                int a = 2;
                int b = 3;
                int c = a | b;
                System.out.println(c);//3
        }
}
```

- ||表示逻辑或运算符,连接两个boolean类型的值
- |表示按位或运算符,连接两个数值型数据,首先把数值转换为二进制数据,然后按位进行或运算
- &&表示逻辑与运算符,连接两个boolean类型的值
- &表示按位与运算符,连接两个数值型数据,首先把数值转换为二进制数据,然后按位进行与运算
- ^异或运算符,`true ^ true = false false ^ true = true false ^ false

加密/数据安全/计算机网络常用按位比较

在不借助中间变量的情况下,交换a和b

条件运算符

?:三目运算符条件?操作数1:操作数2

先判断条件,

true,返回操作数1

false,返回操作数2

```
int a = 2;
int b = 3:
int c = a>b?a:b;
/*if (a>b){
    c = a;
}else{
    c = b;
}*/
```

流程控制

三种基本控制结构:顺序,选择,循环

顺序结构

- 1

选择结构

- if语句
- switch语句

if

```
if (条件表达式){
    要执行的语句
}


if(条件表达式){
    语句块1
}else{
    语句块2
}


if(条件表达式1){
    语句块1
}else if(条件表达式2){
    语句块2
}else if (条件表达式3){
……
```

```
}else{
    语句块n
}
```

```
public class test{
        public static void main(String[] args){
                int age = 19;
                if(age >= 18){
            // 下面的两行是一个语句块，是一个整体，会不会执行要看小括号里面的条件
            // 若条件成立，则执行下面的两行，否则不执行，直接跳出选择结果
                        System.out.println("已经成人了！");
                        System.out.println("你要为自己的行为负责哦");
                }
                System.out.println("程序结束"); // 不管上面的条件表达式的结果如何，这一行一定会执行
        }
}
```

待续

switch

待续

循环结构

- 待续

待续