Java语言

第一章 Java语言概述

定义：Java语言是由美国sun公司，在1995年推出的高级编程语言。所谓语言，是计算机的语言，人们可以使用编程语言对计算机下达命令，让计算机完成人们需要的功能。2009年甲骨文公司收购Sun公司。

Java语言

Bit—Byte—KB—MB—GB—TB—PB—EB—ZB

文件夹在计算机中以字节大小进行表示。

命令提示符（cmd）

输入cmd，打开的为dos界面，即dos命令。

刚打开的命令窗口即为根路径

启动： win+R 输入cmd回车

切换盘符： 盘符名称+：回车

进入文件夹： cd+文件夹名称 回车

进入多级文件夹： cd 文件夹1\文件夹2\文件夹3 回车

注：可以按Tab键进行文件夹的切换或者补全

返回上一级： cd+.. 回车

直接返回根路径： cd + \

查看当前的内容： dir

清屏： cls

创建目录： md

删除目录： rd

删除文件：del

退出： exit

软件开发：软件，即一系列按照特定顺序组织的计算机数据和指令的集合，有系统软件和应用软件之分。

人机交互方式：图形化界面：这种方式简单直接，使用者易于上手，容易接受。

命令行方式：需要有一个控制台，输入特定的指令，让计算机完成一些操作。较为麻烦，需要记住一定的指令。

Java语言程序开发环境的搭建

第一代语言：

打孔机——纯机器语言。

第二代语言：

汇编。

第三代语言：

1. pascal、Fortran面向过程的语言

C++面向对象/面向过程

Java跨平台的纯面向对象的语言

.NET跨语言的平台（由微软公司新推出，与Java相抗衡）

Java虚拟机——JVM

JVM：Java虚拟机，简称JVM，是运行所有Java程序的假想计算机，是Java程序的运行换环境，是Java最具吸引力的特征之一。我们编写Java程序代码时，都运行在JVM之上。

跨平台：任何软件的运行，都必须要运行在操作系统上，而我们用Java编写的软件可以运行在任何的操作系统上，这个特性称为Java语言的跨平台特性。该特性是由JVN实现的，我们编写的程序运行在JVM上，而JVM运行在操作系统之上。

Java的虚拟机本身不具备跨平台功能，每个操作系统下都有不同版本的虚拟机。跨平台的为Java程序。

JRE：是Java程序的运行环境，包含运行时所需的核心类库（环境信息）。

JDK：是Java程序的开发工具包，包含JRE和开发人员使用的工具。

Java的基本元素

标识符、关键字、注释、分隔符

标识符的作用：用于命名变量、类型、数组、文件、类和方法等。

标识符的语法：由字母、下划线（\_）、美元符号（$）和数值组成的字符串。首字符不能是数字。其中的字母是包括汉字在内的各个国家的文字（Unicode 字符）

Java语言注释的种类：单行注释(//)、多行注释(/\*…… \*/)和

文档注释(/\*\*…… \*/)。

文档注释(/\*\*…… \*/)：文档注释属于多行注释，有其特殊

的用途。Java文档生成器（javadoc命令）可以扫描Java

语言程序，提取文档注释，生成该程序的Java帮助文档。

javadoc命令是用来生成自己[API](https://baike.baidu.com/item/API/10154" \t "https://baike.baidu.com/item/javadoc/_blank)文档的。

使用方式：使用命令行在目标文件所在目录输入javadoc +文件名.java。

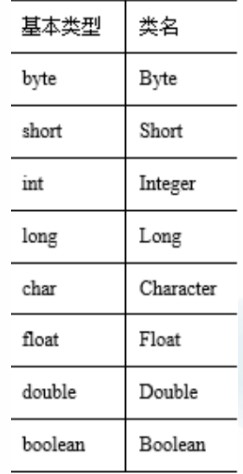
APl文档：目的是提供[应用程序](https://baike.baidu.com/item/%E5%BA%94%E7%94%A8%E7%A8%8B%E5%BA%8F" \t "https://baike.baidu.com/item/API/_blank)与开发人员基于某[软件](https://baike.baidu.com/item/%E8%BD%AF%E4%BB%B6" \t "https://baike.baidu.com/item/API/_blank)或硬件得以访问一组[例程](https://baike.baidu.com/item/%E4%BE%8B%E7%A8%8B" \t "https://baike.baidu.com/item/API/_blank)的能力，而又无需访问原码，或理解内部工作[机制](https://baike.baidu.com/item/%E6%9C%BA%E5%88%B6" \t "https://baike.baidu.com/item/API/_blank)的细节。

整型：byte（1字节）、short（2字节）、int、long.

Unicode（统一码、万国码、单一码）是计算机科学领域里的一项业界标准，包括字符集、编码方案等。它为每种语言中的每个字符设定了统一并且唯一的二进制编码，以满足跨语言、跨平台进行文本转换、处理的要求。

包装类：Java语言中的基本数据类型不是面向对象的，因此Java类库中提供了包装类，每个基本数据类型都有一个相对应的类。

除了Integer和Character之外，其他的与类名基本一致。类名首字母大写。



例如：int n = 200;

Integer in = new Integer (n);

其中，in是一个Integer类型的对象，用值20进行初始化，

不同于整型变量n。

字符常量：简单字符常量（‘A’ ‘Z’ ‘@’）和转义字符常量（ ‘\’’ ‘\\’ ‘\n’ ‘\t’）

字符串常量：”Hello word!”

null常量仅有”null”，用于给符合数据类型的变量赋初值等。

Java变量类型可以分为7类：

1.类变量（class variable）：包括类变量和接口变量，定义

类变量时使用修饰词static，接口变量可以不使用修饰

词static。创建类或接口时，变量按照缺省值初始化；卸载

类或接口时，其类变量随之消失。

2.实例变量（instance variable）：不使用修饰词static的类

变量。

3.阵列组件（Array components）：属于未命名的变量，按

缺省值初始化，创建的对象即数组。阵列是若干对象，阵列

中的所有组件具有相同类型。

4.方法形参（Method parameters）：类的方法的形式参数。

5.构造形参（Constructor parameters）：类构造函数的

形式参数。

6.例外形参（exception parameter）：例外处理程序的形

式参数。

7.局部变量（Local variables）：构造函数、方法或程序块

内声明的变量，只在其声明范围内有效。

.final变量

使用修饰词final的变量只能被赋值一次，程序执行过程

中其值不会改变。

例如：

public static final String start = "程序运行开始！！！"

public static final double PI = 3.1415926;

4.变量声明

Java语言的修饰词

**有public、protected、private、abstract、static和final。**

声明变量时，除了指定类型之外，还可以使用修饰词来限定

它的使用范围。

**声明变量时，先指定修饰词（可以复数个），再指定变量类**

**型，最后是变量列表。**

例如：

public static byte ByteSpace = 0x20;

private float length, width, height;

运算符的优先级：

域运算符 ::

初等运算符 () [] ->

单目运算符 ++ --

算术运算符 （先乘除，后加减）

关系运算符 （ > < <=）

逻辑运算符

条件运算符

赋值运算符

逗号运算符

高 低

!——算术运算符——关系运算符——&&和||——赋值运算符

方法：当先后级顺序忘记的时候，可以使用圆括号改变运算顺序。

注意：先乘除后加减，同级运算从左到右，括号里优先。

Java中有很多的特殊运算符：

条件运算符（ ？ ：）

对象运算符（instanceof）

圆括号

方括号（表达式中的方括号[]也是一种语法形式，用来取数组元

素的运算符，数组的下标值写在方括号之中。）

类型转换

Javaz中类型转换分为两种：自动转换和强制转换。

一般而言按照byte→short或char→int→long→float

→double的原则进行类型转换，也可以进行越级转换。

赋值发生时，则必须将赋值运算符右边的表达式的值强制

转换为左边变量的类型。

Java语言可以认为的改变数据类型，称为强制类型转换。强制类型转换表达式：

(<类型名>) (<表达式>)

例如：：计算半径为8.0圆的面积，结果只需要整数部分时，

用(int)进行强制转换：

int s, r=8.0;

s = (int) (3.1415926 \* r \* r);

其中，表达式的圆括号是必要的，如果写成s = (int)

3.1415926 \* r \* r，则意义完全不同。

第二章 Hello Word 入门程序

程序开发的步骤说明

Java程序开发的三个步骤：编写、编译、运行。

Java源程序——编译器——Java字节码文件——JVM运行

Javac.exe:编译器 java.exe 解释器

编译时使用的是文件名.java 文件 而运行则使用的编译自动生成的 文件名.class文件

注意：文件名必须与源代码编写的类的名称一样（有大小写之分）

基础：

// 第一行的第三个单词必须和所在的文件名称完全一样，大小写也要一样。

// piblic class 后面代表定义一个类的名称，类是Java当中所有源代码的基本组织单位。

public clsaa HelloWorld {

// 第二行的内容是万年不变的固定写法，代表main方法。

// 这一行代表程序执行的起点。

public static void main(String[] args) {

// 第三行代表打印输出语句（其实就是屏幕显示）

// 希望显示什么东西，就在小括号当中填写什么。

System.out.println(“HelloWorld!”);

}

}

注意事项：

1. 字符串不是基本类型，而是引用类型。
2. 浮点型可能只是一个近似值，并非明确的值。
3. 数据范围与字节数不一定相关，例如float数据范围比long更加广泛。
4. 浮点数当中默认类型为double，如果一定要使用float类型，需要加一个后缀条件F。
5. 整数的默认为int类型，如果要使用long，需要加一个后缀L。例：System.out.println(100L）。 可以小写。

变量：程序运行期间，内容可以改变的量。

创建一个变量并且使用的格式：

数据类型 变量名称； // 创建了一个变量

变量名称 = 数据值； // 赋值，将右边的数据值，赋值交给左边的变量。

一步到位的格式：

数据类型 变量名称 = 数据值；

public class Demo02Variable {

public static void main(String[] args) {

// 创建一个变量

// 格式：数据类型 变量名称；

Int num1;

// 向变量当中存入一个数据

// 格式：变量名称 = 数据值；

num1 = 10；

// 当打印输出变量名称的时候，显示出来的是变量的内容；

System.out.println(num1); // 10，括号中不用加双引号。

boolean var1 = true;

System.out.println(var1) ; // true;

}

}

强制类型转换(从大到小）

1. 特点：代码需要进行特殊的格式处理，不能自动完成。
2. 格式：范围小的类型 范围小的变量名 = （范围小的类型） 原本范围大的数据;

例： // double - - > int ，强制转化

Int num3 = (int) 3.5;

System.out.println(num3);

输入：3.5 输出：3 //这不是四舍五入，而是精度损失。

注意事项：

1. 强制转换类型一般不推荐使用，因为肯能会造成精度损失（浮点型）、数据溢出（整型）。
2. byte/short/char这三种类型都可以进行数学运算，例如加法“+”；
3. byte/short/char这三种类型在运算的时候，都会首先提升为int类型，然后计算。
4. Boolean类型不能发生数据类型转换。

char zifu1 = ‘A’; //这是一个字符型变量，里面是大写字母A；

System.out.println(ziful1+1); // 66,也就是大写字母 A被当成了65进行处理。

//计算机的底层会用一个数字（二进制）来代表字符A，就是65。

//一旦char类型进行了数字运算，那么字符就会按照一定的规则翻译成一个数字。

byte num4 = 40;

byte num5 = 50;

// byte + byte - -> int + int - - >int

int result1 = num4 + num5 ; // 此处必须为int型，如果为byte类型，会发生数据溢出。

System.out.println(result) (num4 + num5)；// 90；

short num6 = 60 ;

//byte + short - - > int + int - - > int

//int强制转换为short，注意必须保证逻辑上真实大小本来就没有超过short范围，否则会发生数据溢出。

short result2 = (short) (num4 + num6); //100.

数字和字符的对照关系表（编码表）：

ASCII码表：American Standard Code for Information Interchange,美国信息交换标准代码。

Unicode码表：万国码，也是数字和符号的对照关系，开头0-127部分和ASCII完全一样，但是从128开始包含更多的字符。（中文、日语、韩语，甚至表情符号 Emoji(并不是表情符号，而是一种字符）

48 - ‘0’ 65 - ‘A’ 97 - ‘a’

关于ASCII码表

可以通过代码直接查出他的数字。

第一种：

char zifu1 = ‘1’;

// 只要char 类型的数据进行运算，系统会将其自动转换为整数。

System.out.println(zifu1 + 0); // 49;

第二种：

char zifu2 = ‘c’;

//左侧是int类型，右边是char类型；

//char - - > int ,确实是从小到大。

//发生了自动转换。

int num = zifu2;

System.out.println(num); // 99;

char zifu4 = ‘中’; //正确写法；

System.out.println(zifu4 + 0); // 20013;

四则运算中”+“的常见三种用法：

1. 对于数值来说，那就是加法。
2. 对于字符char类型来说，在计算之前，char会被提升为int，然后在计算。
3. 对于字符串String（首字母大写，并不是关键字）来说，加号代表字符串链接操作。
4. 任何数据类型和字符串进行链接的时候，结果都会变成字符串。

例：String str1 = “hello”;

System.out.println(str1); // hello

System.out.println(“hello”+”word”); // helloword 没有加号存在。

String str2 = “java”;

// String + int - - > String

System.out.println(str2 + 20); // java20

//优先级问题

// String + int + int

System.out.println(str2 + 20 + 30); // java2030

//String + (int + int )

System.out.println(str2 + (20 + 30)); //java50

自增自减

单独使用时，i++和++i的结果一样。

混合使用时（如和输出语句连用），i++ 为先用后加，++i 为先加后用。

例如： int num = 5;

num++ // ++num;

//单独使用

System.out.println(num); // 6

//混合使用

System.out.printlin(num++); // 5

System.out.printlin(++num); // 6

比较运算符。是两个数据之间进行比较的运算，运算结果都是布尔值true和false.

逻辑运算符（ && || ！）

1. 逻辑运算符只能用于boolean。
2. 与、或需要左右各有一个boolean值，但是取反只有唯一的一个boolean值即可。
3. 与、或两种运算符，如果有多个条件，可以连续写。

两种条件：条件A && 条件B

多个条件：条件A && 条件B && 条件C

注意：与“&&”，或“||”。具有短路效果：如果根据左边已经可以判断得到最终的结果，那么右边的代码将不再执行。

即： System.out.printlin(3>4 && 7<8); // false 因为左边的3>4已经是false，已经可以确定结果。右边的代码不用在进行执行。

除法：除法靠零取整。

取余数：先取绝对值，然后把第一个数的符号加在余数上。5%-2 = 1 ； -5%-2 = -1；

三元运算符：（条件判断 ？ 表达式A ： 表达式 B )

注意事项： 、

1.必须同时保证表达式A 与表达式B都符合左侧的数据类型。

int result = 3 > 4 ? 2.5 :10; // 错误，205为浮点型，而前边定义的为int 型。

2.三元运算符的结果必须被使用。

System.out.println(a>b ？a: b); //正确写法；

a > b ? a : b; // 错误写法；没有被利用。

1. 方法入门

概述：就是一个功能抽取出来，把代码单独定义在一个大括号内，形成一个单独的功能。我们需要这个功能的时候，就可以去调用。这样即可实现代码的复用性。

定义一个方法的格式：

public static void 方法名称(){

方法体

}

注意：方法定义好之后，不会执行的，如果想要执行，一定要进行方法的【调用】。

调用格式：

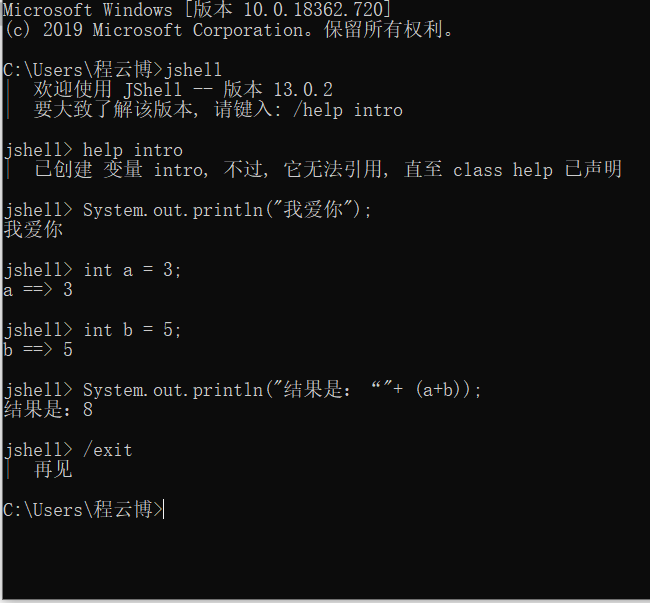
方法名称();

1. JShell脚本工具

JShell脚本工具时JDK9的新特性

当我们编写的代码非常少的时候，而又不愿意编写类，main的方法，也不愿意编译和运行，这个时候就可以使用这个Shell工具。

启动Shell工具，在DOS命令直接输入JShell命令。



想打印什么直接输入既可以，不用进行什么类的定义和编译。

注意：退出时必须输入 ”/exit“，必须有反斜杠存在。

特洛伊木马(Trojan Horse)：是指寄宿在计算机里的一种非授权的远程控制程序。由于特洛伊木马程序能够在计算机管理员未发觉的情况下开放系统权限、泄漏用户信息、甚至窃取整个计算机管理使用权限，使得它成为了黑客们最为常用的工具之一。

对于一个byte/short/char 三中类型来说，如果右侧赋值的数值没有超过范围，那么Javac编译器将会自动隐含的为我们补上一个(byte) (short)(char)。

1. 如果没有超过范围，编译器补上强转。
2. 如果右侧超过了左侧的范围，那么直接编译报错。

分支结构(switch)：

1. 多个case后面的数值不可以重复。
2. switch后边小括号当中只能是下列数据类型：

基本数据类型：byte/short/char/int

引用数据类型：String字符串、enum枚举。

1. 开发工具 Intellij IDEA

IDE：集成开发环境。集成开发环境，是一门专门用来提高Java开发效率的软件。

切换鼠标：按 ins 键

描述：IDEA是一个专门针对Java的集成开发工具(IDE)，由Java语言编写。所以，需要JRE运行环境并配置好环境变量，它可以极大的提升我们的开发效率，可以自动编程，检查错误。在公司中，使用的就是IEDA进行开发。

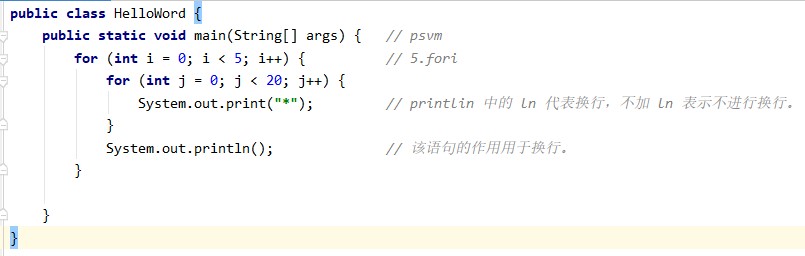
psvm：main方法 sout；输出语句。

IDEA中常用的快捷键：



习题：打印一个5\*20的图表。

正常书写代码：



/\* 复习一下之前学的方法基础入门：

定义格式：

public static void 方法名称() {

方法体

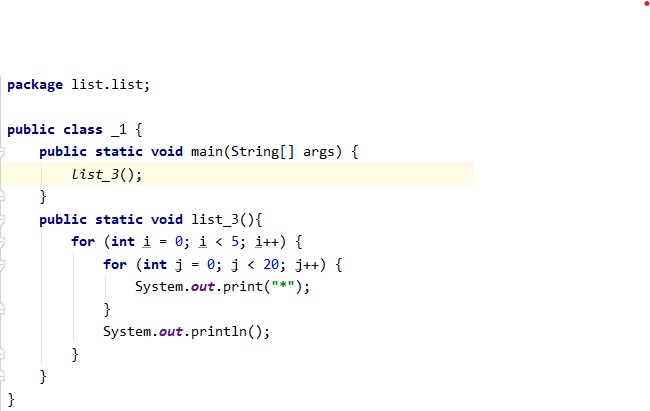
}

调用格式：

方法名称();

\*/

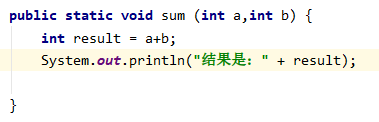
使用方法的调用的代码：



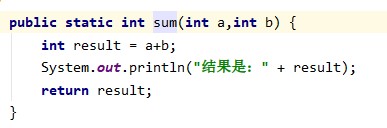
如果调用的方法没有参数，直接调用即可。 即直接 方法名 ();

如果有参数，必须给定数值，如相加 方法名（a:10,b:20); 要给定数值。

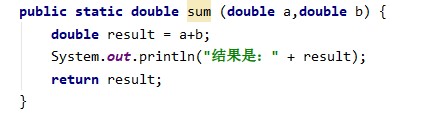
// 如果方法定义的为void 类型，则不需要返回参数，即不用使用return 语句。



// 如果方法定义的为 int或者其他类型，则有参数返回，即必须使用return 语句。

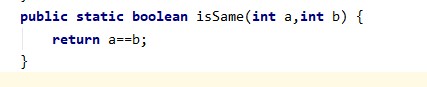


// 如果进行两个小数之间的相加，则把result 定义为 double 类型。返回的即为double 类型。

file:/D:/project-3/project-3.iml

Eg:判断两个数字是否相等。只需要三行代码，即可解决这个问题。

//这个为功能代码，返回值为true或者flase;



//这个为调用语句：

3

// 方法的重载（overloded）：多个方法的名称一样，但是参数不同。

// 参数不同但是功能相同的功能函数，都定义为sum函数。

**public static int** sum (**int** a,**int** b) {  
  
 System.***out***.println(**"有两个参数大的方法执行！"**);  
 **return** a+b;  
}  
**public static int** sum (**int** a,**int** b,**int** c ) {  
  
 System.***out***.println(**"有三个参数大的方法执行！"**);  
 **return** a+b+c;  
}  
**public static int** sum (**int** a,**int** b, **int** c ,**int** d) {  
  
 System.***out***.println(**"有四个参数大的方法执行！"**);  
 **return** a+b+c+d;  
}

// mian 主函数调用。

*// 直接调用sum函数即可，他会自己匹配参数个数。*System.***out***.println(*sum*(20,30)); *// 50;*System.***out***.println(*sum*(20,30,50)); *// 100;*System.***out***.println(*sum*(20,30,10,50)); *// 110;*

类型转换： （short) int ; 类型说明：40L 声明40为long型，而不是int类型。

在调用输出语句的时候，println方法其实进行了多种数据类型的重载。所以println 的小括号中可以输出多种类型的语句。

1. 数组

数组的初始化：在内存当中创建一个数组，并且向其中赋予一些默认值。

两种常见的初始化方式：

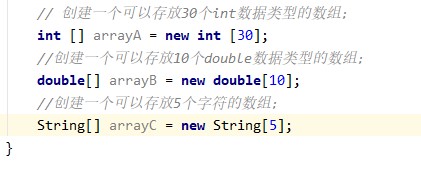
1. 动态初始化（指定长度）：直接指定数组的元素个数。
2. 静态初始化（指定内容）：不指定元素个数，而是指定元素是什么。

动态初始化数组的格式：

**数据类型 [ ] 数组名称 = new 数据类型[数组长度]；**

解释含义：

1. 左侧数据类型：也就是数组当中保存的数据，全都是统一的什么类型。
2. 左侧的中括号：代表我是一个数组。
3. 左侧数组名称：给数组取一个名字。
4. 右侧的new：代表创建数组的动作。
5. 右侧数据类型：必须和左边的数据类型保持一致。
6. 右侧中括号的长度：也就是数组当中，到底可以保存多少个数据，是一个int数字。



静态初始化标准格式：

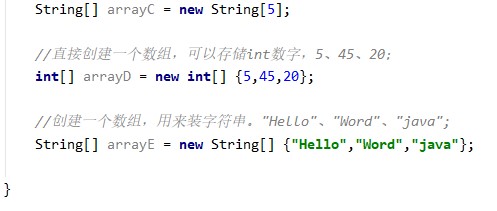
**数据类型[ ] 数组名称 = new 数据类型[ ] { 元素1, 元素2, ....};**

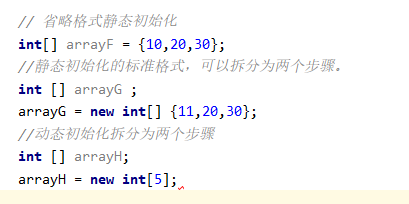
省略格式：

数据类型[ ] 数据名称 = {元素1，元素2，...};

注意事项：

1. 静态初始化没有指定长度但是仍然会自己推算得到长度。
2. 静态初始化标准格式可以拆分成为两个步骤。
3. 动态初始化也可以拆分称为两个步骤。
4. 静态初始化一旦使用省略模式，就不可以在进行拆分。





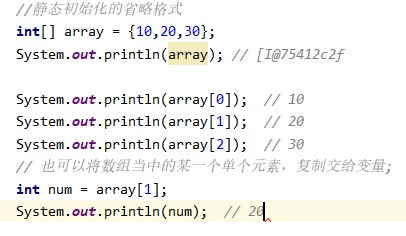
直接打印数组名称，得到的是数组对应的：内存地址哈希值。

内存地址以十六进制表示。

访问数组元素的格式：数组名称[索引值]

索引值：就是一个int数字，代表数组当中元素的编号。

【注意】：索引值从0开始，一直到“数组的长度-1“为之。



使用动态初始化数组的时候，其中的元素将会自动拥有一个默认值。规制如下：

如果是整数类型，那么默认值为0；

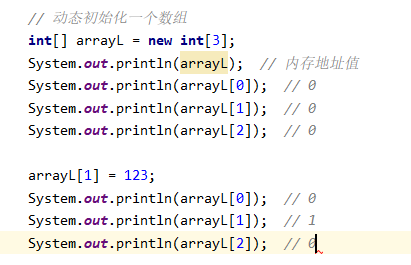
如果是浮点类型，那么默认值为0.0；

如果是字符类型，那么默认值为’\u0000’;

如果是布尔类型，那么默认值为false；

如果是引用类型，那么默认值为null；

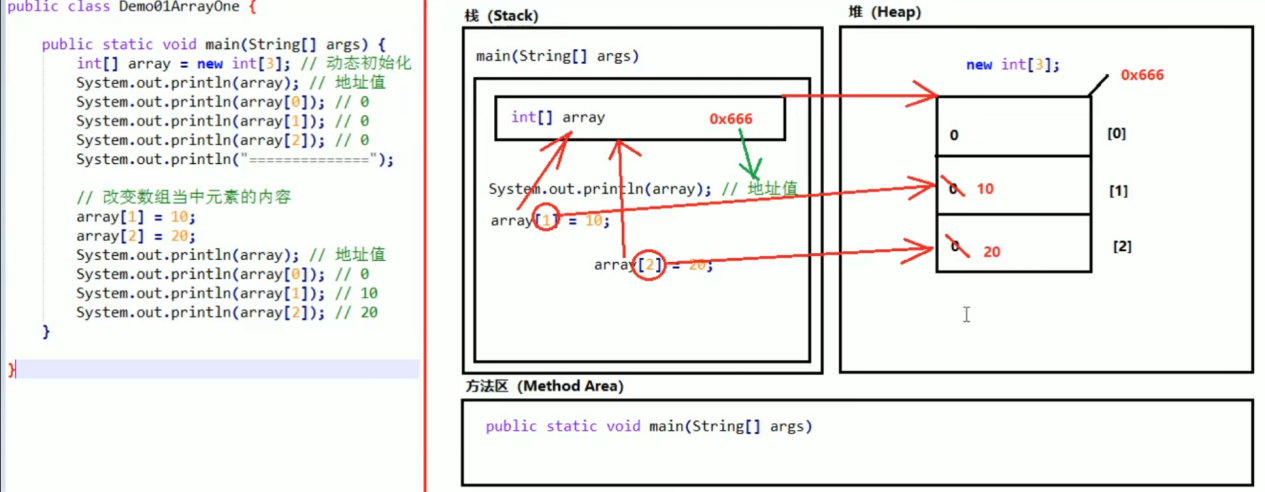
注意事项：静态初始化其实也有默认值的过程，只不过系统自动马上将默认值替换成为了大括号当中的默认值。



java的内存需要划分成为5个部分：



数组中的元素在计算机内存中的传递：



数组中常见的两个问题：

1 数组索引越界异常

数组的索引编号从0开始，一直到“数组的长度-1”为止。

如果访问数组元素的时候，索引编号并不存在，那么将会发生数组索引越界异常

ArraryIndexOutOfBoundsException

原因：索引编号写错了。

解决：修改成为存在的正确索引编号。

2 空数组

所有的引用类型变量，都可以赋值为一个null值。但是代表其中什么都没有。

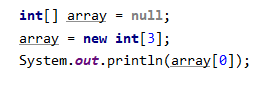
数组必须进行new初始化才能使用其中的元素。

如果只是赋值一个null，没有进行new创建，那么将会发生：

空指针异常 NullPointerException

原因：忘了new

解决：补上nwe

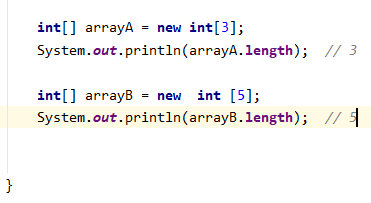


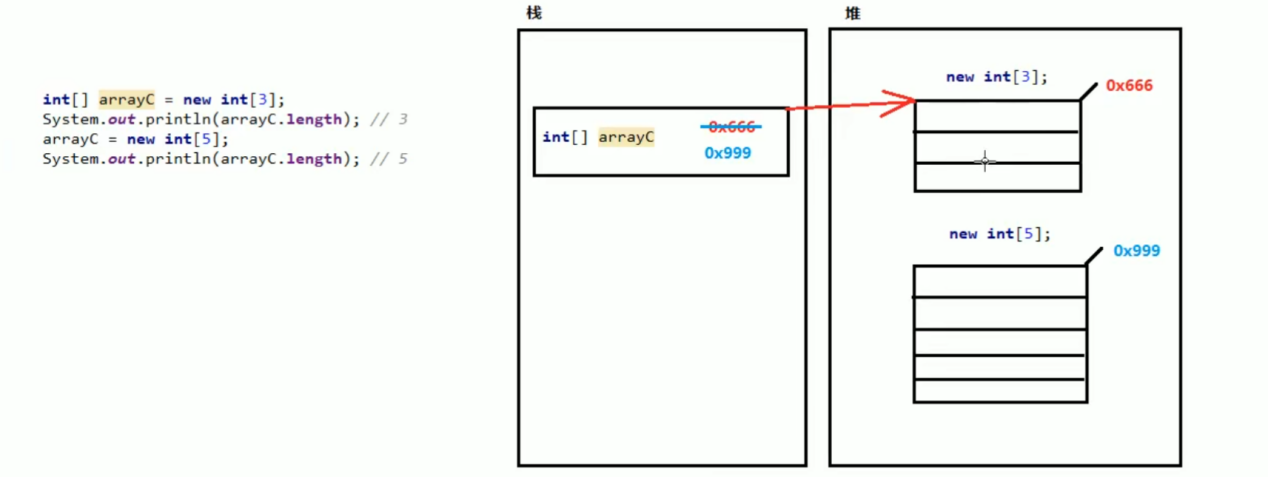
如何获取数组的长度，格式：

**数组名称 .length**

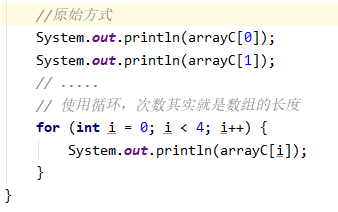
这将会得到一个int数字，代表数组的长度。

数组一旦创建，程序运行期间，长度不可改变。





遍历数组：就是对数组中的每一个元素进行逐一、挨个处理。默认的处理方式就是打印输出。



eg： 数组元素的反转：

本来样子：[1,2,3,4]

之后样子：[4,3,2,1]

不能使用新数组，就用原来定义的那一个数组

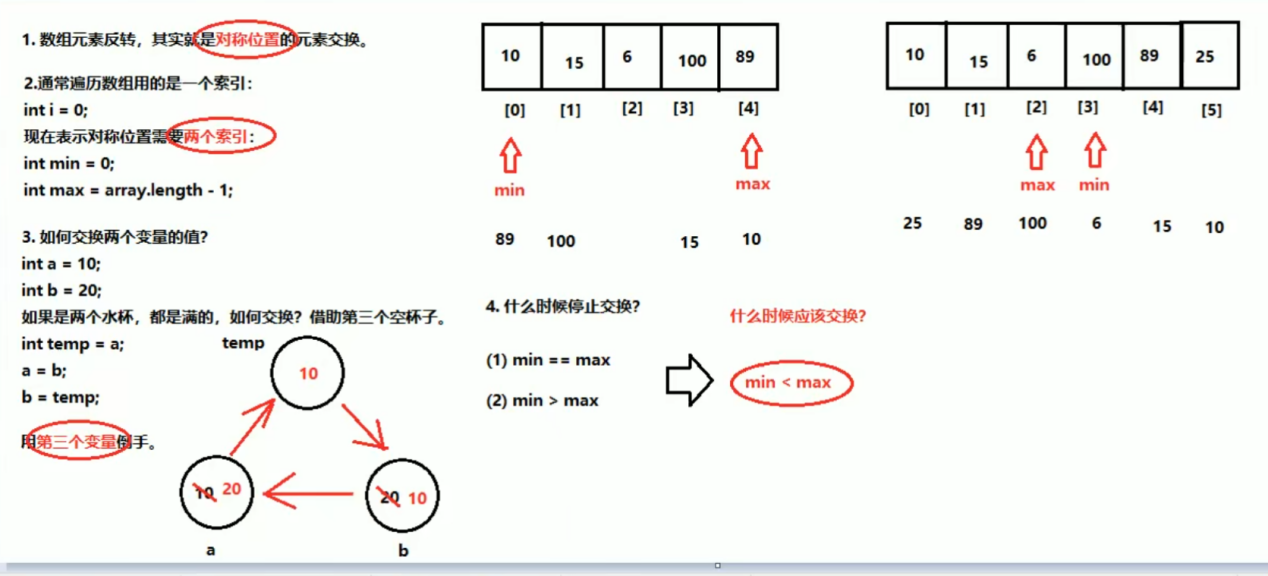
初始化语句：int min = 0 , max = array.length - 1

条件判断：min < max

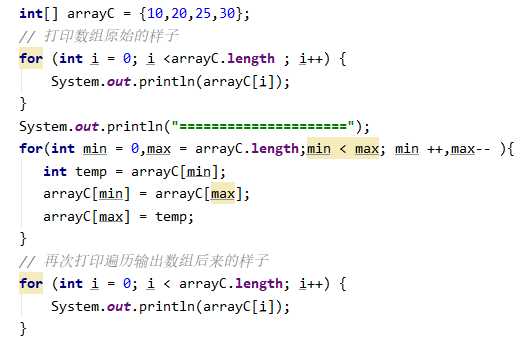
步进表达式：min++ , max--;

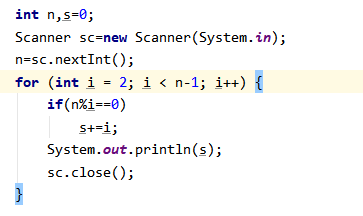
循环体：用第三个变量倒手 temp

方法设计图：



代码：





其中有输入语句：Scanner sc = new Scanner (System.in);

需要赋值的变量= sc.next数据类型； n=sc.nextInt

最后还需要释放这个语句：sc.close();

// 打印一个星矩阵

**for** (**int** i = 0; i < 4; i++) {  
 **for** (**int** i1 = 0; i1 < 5; i1++) {  
 System.***out***.print(**"\*"**);  
 }  
 System.***out***.println();  
}



// 打印右对齐的直角三角形

**for** (**int** i = 0; i < 4; i++) {  
 **for** (**int** i1 = 0; i1 < 3 - i; i1++) {  
 System.***out***.print(**" "**);  
 }  
 **for** (**int** i1 = 0; i1 < i + 1; i1++) {  
 System.***out***.print(**"\*"**);  
  
 }  
 System.***out***.println();  
}

打印结果：

