[Day01 2](#_Toc197810165)

[**Day02** 3](#_Toc197810166)

[Day 03 4](#_Toc197810167)

[Day04 流程控制语句 5](#_Toc197810168)

[Day05 循环高级 7](#_Toc197810169)

[Day06 数组 8](#_Toc197810170)

[Day07 方法 10](#_Toc197810171)

[Day07 面向对象 12](#_Toc197810172)

# Day01

JDK官网 http://www.oracle.com（javac把.java文件编译为.class文件，java运行文件）

配置环境变量（bin）

class（定义一个类）

字面量类型（整数，小数，字符串，字符，布尔型，空类型）

制表符'\t'把字符串补齐8位

数据类型 变量名=数据值；(变量请直接赋值)

二进制（0b）八进制（0）十进制（null）十六进制（0x）

任意进制转十进制（公式：系数\*基数的权次幂 相加）

byte的取值范围（-128~127）

标识符：类，方法，变量的名称；不能以数字开头；由字母，数字，-,$组成；

键盘录入：Scanner (ScannerDomeOne)

(

import java.until.Scanner;

Scanner sc=new Scanner(System.in);

int name=sc.nextIn();

)

IDEA(下载官网：https://www.jetbrains.com/idea/)

IDEA四层结构（project 项目；module 模块；package 包；class 类；）

小数参与运算可能丢失精度

算术运算符：=，-，乘，/，%；

小自动升大，大强制缩小。

byte，short，char升为int；

强制转换类型：目标数据类型 变量名=（目标数据类型）被强制转换变量名；

例：int a=（int）b；（可能精度丢失）

字符串只有拼接操作；

自增自减：先用后加（b=a++），先加后用(b=++a);

加等减等：加后赋值（a+=b），赋值后加（a=+b）；

&与|或！非，^异或；

&&短路与（都真才真），||短路或（有真则真）；

# **Day02**

**三元运算符：**

格式：关系表达式？ 表达式1：表达式2

解释：关系表达式为true运行表达式1，false运行表达式2。

# Day 03

原码，反码，补码（单位：比特byte，8比特=1字节）；

原码:-128（补码：1000 0000）~127

反码：正数的反码不变；

负数进行取反，符号位不变；

补码：负数的反码+1；解决+0和-0的问题，正数和负数的运算，计算机的存储和计算；

Byte：1

Short：2

Int：4

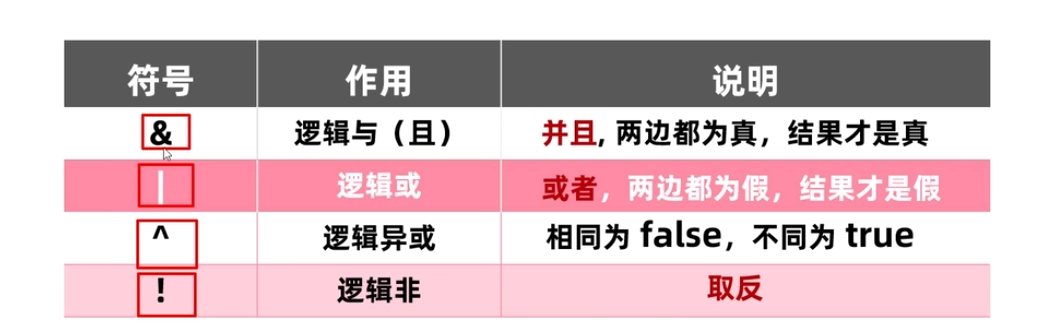
Long：8；

&逻辑与：都为1才为true

|逻辑或：有1就为1；

<<左移：一位是\*2

>>右移：一位/2.



# Day04 流程控制语句

### 顺序结构

先后顺序，依次执行。

### 分支结构

If语句：多条件判断（范围判断）

if（关系表达式1）{

··· 语句体1；

}

Else if（关系表达式2）{

··· 语句体2；

}…

else{

语句体2

}

Switch语句

格式：有限个数据列举出，任选其一

Switch（表达式）{

Case 值1：

语句体1；

Break；

Case 值2：

语句体2；

Break；

。。。

Default：（default可以省略，可以写在任何位置）

语句体n+1；

Break；

}

Case穿透：break不能省略（本次循环结束）

使用场景：如果多个case重复了，可以用case简化。

Eg：switch（表达式）{

Case 值1->语句体；

Case 值1->语句体；

Case 值1->语句体；

Default->语句体；

}

循环结构·

变量的作用范围，只在所属的·大括号里面有效

如果我们把变量定义在循环里面，那么当前的变量只能在本次循环有效

第二次循环开始的时候就会重新定义了变量

##### For循环：

格式：

for（初始化语句；条件判断语句；条件控制语句）{

循环语句

}

执行流程：

1. 执行初始化语句。
2. 执行条件判断语句，看其结果是true还是false

如果是false，循环结束。

如果是true，执行循环语句题。

3．执行条件控制语句

4.回到2继续执行判断语句

##### While循环

格式：

初始化语句；

While（条件判断语句）{

循环体语句；

条件控制语句；

}

For和while对比：

语法上无区别，知道范围用for，不知道范围用while

##### Do。。。while循环

格式

Do{

循环体语句；

条件控制语句；

}while（条件判断语句）；

先执行再判断。

# Day05 循环高级

### 无限循环

已知循环次数

For(;;){

System.out.println(“内容”);

}

未知循环次数

While（true）{

System.out.println(“内容”);

}

Do{

System.out.println(“内容”);

} While（true）;

### 跳转控置语句

Continue:结束本次循环，继续下次循环。

Break：结束整个循环。

Random：随机生成一个数字。

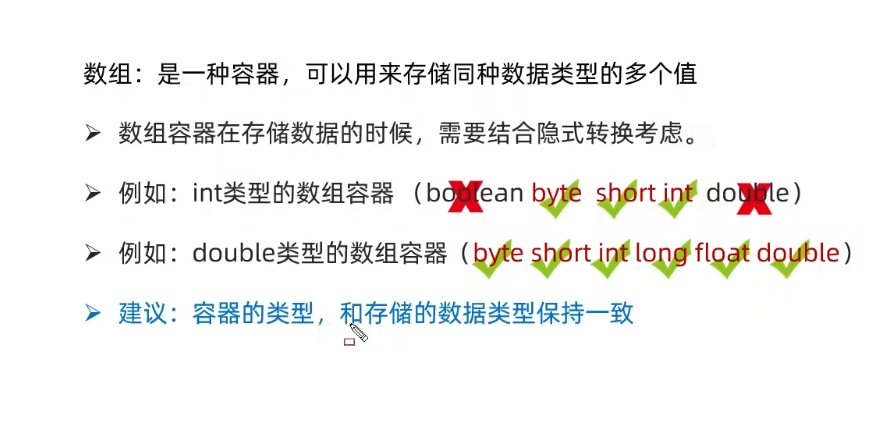
1. 导包
2. 创建对象
3. 生成随机数



从0开始，到-1结束；包头不包尾，包左不包右；

# Day06 数组

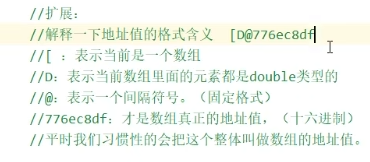
容器，存储同种数组类型的多个值。



例：int [] array

索引：下标。

访问：array在后是查询，在前是赋值。

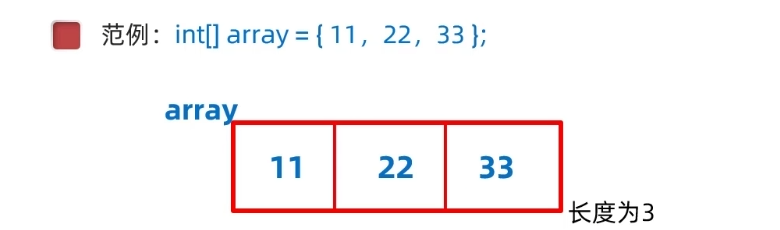


### 初始化

在内存中为数组容器开辟空间，并将数据存入容器的过程。

###### 静态初始化

数据类型[]数组名=new 数据类型[]{ 元素1，元素2，元素3，。。。}

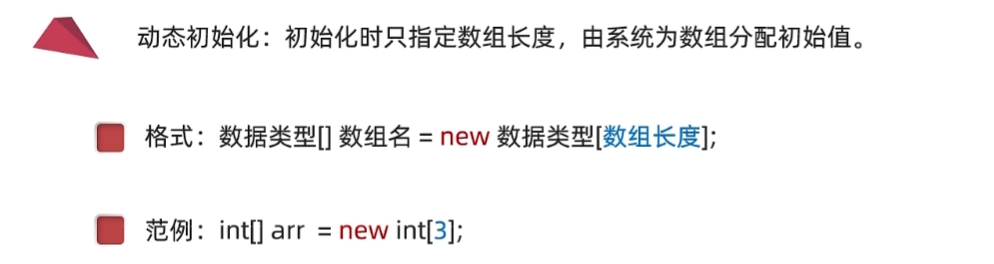


数组的长度属性：数组名.length

快速生成遍历：数组名.fori

###### 动态初始化

数据类型[] 数组名=new 数据类型[数组长度];



区别：动态：明确元素个数，不明确具体数值。

静态：明确具体数据，直接操作。

主要要求：

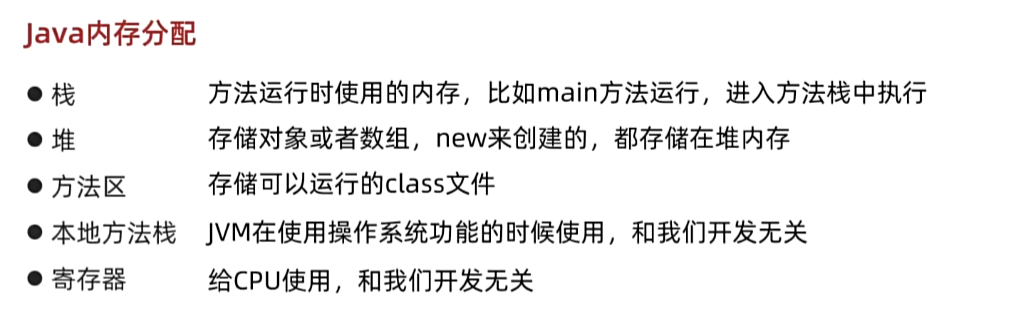
1. 定义数组
2. 修改数组
3. 遍历数组

常见问题

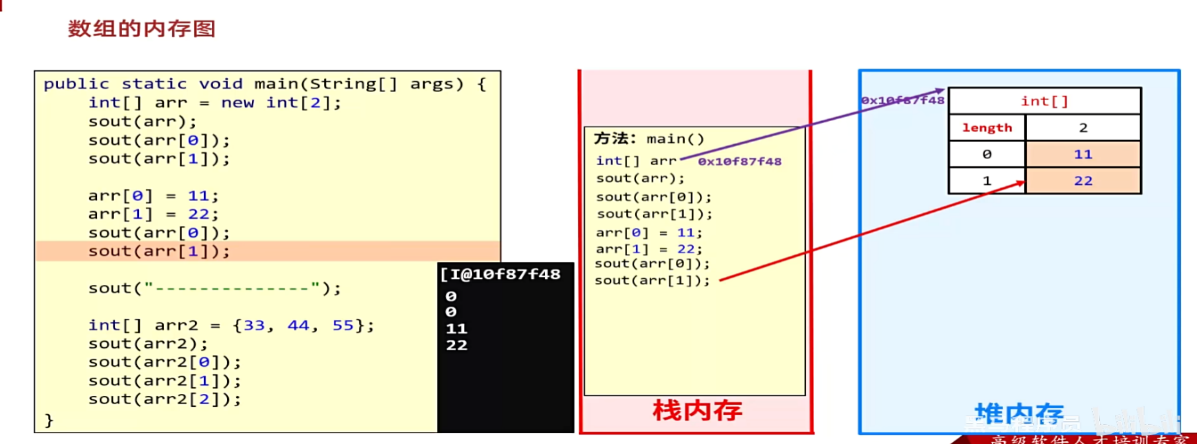
1. 索引越界异常：访问到了不存在的索引。

##### 数组的内存图

1. Java的内存分配：栈，堆，方法区，本地方法栈，寄存器。



内存图：



# Day07 方法

方法（method）是程序中最小的执行单元。

//调用方法。

在mian方法中：

看到方法进入方法

执行完毕回到调用处

##### 传参



#### 形参和实参

形参：指方法定义中的参数。

实参：方法调用中的参数。

**形参和实参必须一一对应，否则会报错。**

#### 带返回值的调用

Public static 返回值类型 方法名（参数）{

**方法体；**

return返回值；

}

1. 直接调用；
2. 赋值调用；
3. 输出调用。

##### 方法的重载

1. 在同一个类中，定义了多个同名的方法，这些同名的方法具有同种的共能。
2. 每个方法具有不同的参数类型或者参数个数，这些同名的方法，就构成了重载关系。
3. 同一个类中，方法名相同，参数不同的方法，与返回值无关。

Return和break关键字的区别：

Return跟循环没有什么关系，跟方法有关系，表示1结束方法，2返回结果

如果方法执行到return，整个方法全部结束，后面循环也随之结束。

Break关键字和方法没有什么关系，结束循环或者switch的。

###### 二维数组：

一维数组中的一维数组。（分组管理时使用。）

静态初始化：

格式：数据类型[][] 数组名=new 数据类型[][]{{元素1，元素2},{元素1，元素2}}

简化：数据类型[][] 数组名={{元素1，元素2},{元素1，元素2}}

例：int[][] arr={{ },{ }}

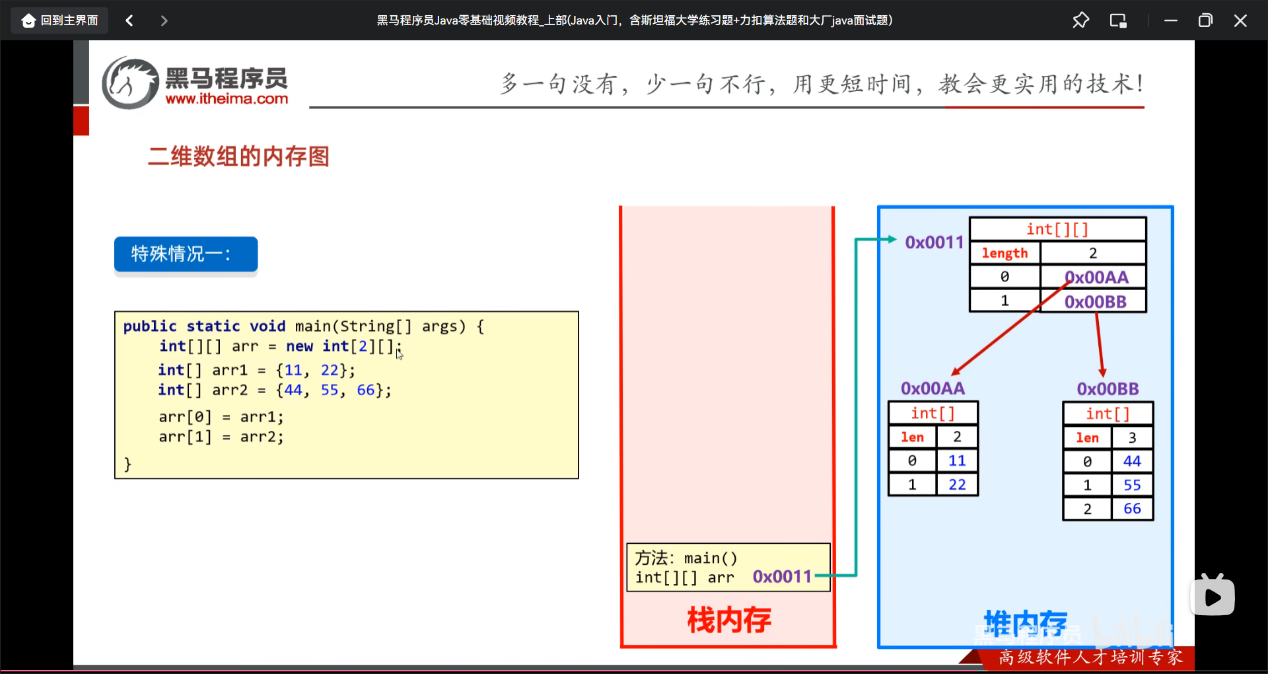
反: Int arr[][]={{ },{ }}

未指定具体位置就输出地址值（索引从0开始）。

动态初始化：

格式：数据类型[][] 数组名=new 数据类型[m][n]

未赋值的是0；



# Day07 面向对象

面向：拿，找。

对象：能干活的东西。

面向对象编程：拿东西过来编程。

获取已有对象并使用。

## 类和对象

类：是共同特征的描述（设计图）：对象：是真实存在的具体实例。

在java中，必须先设计类，才能获取对象

Public class 类名{

成员变量；

成员方法；

构造器；

代码块；

内部类；

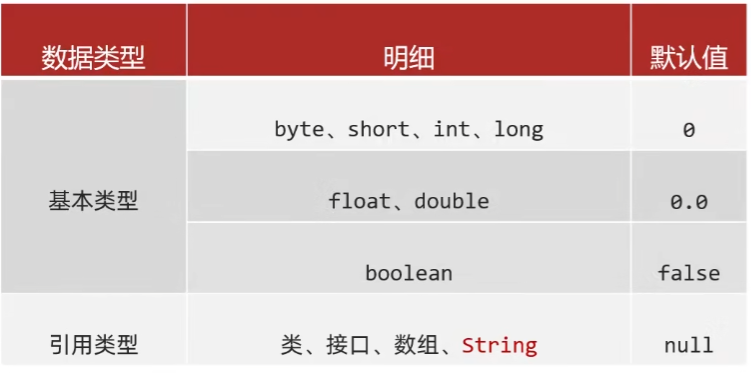
}

类名 对象名=new 类名（）；

JavaBean类：用来描述一类事物的类。

一个文件定义一个类。

默认初始化值：



封装：如何正确设计对象的属性和方法。（对象代表什么，就得封装对应的数据，并提供数据对应的行为。）

继承

多态

Set方法：给私有成员变量赋值（可以添加this用来调用成员变量）。

Get方法：对外提供成员变量。

就近原则：区别成员变量（this.变量名）和局部变量。