[Day01 2](#_Toc201329980)

[**Day02** 3](#_Toc201329981)

[Day 03 4](#_Toc201329982)

[Day04 流程控制语句 5](#_Toc201329983)

[Day05 循环高级 7](#_Toc201329984)

[Day06 数组 8](#_Toc201329985)

[Day07 方法 10](#_Toc201329986)

[Day07 面向对象 12](#_Toc201329987)

[Day08 标准的javaBean 14](#_Toc201329988)

[Day09 面向对象综述 16](#_Toc201329989)

[Day10 API&字符串 17](#_Toc201329990)

[Day11 集合 19](#_Toc201329991)

[Day12 学生管理系统 20](#_Toc201329992)

[Day13 面向对象进阶 21](#_Toc201329993)

[Day14 多态 25](#_Toc201329994)

# 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

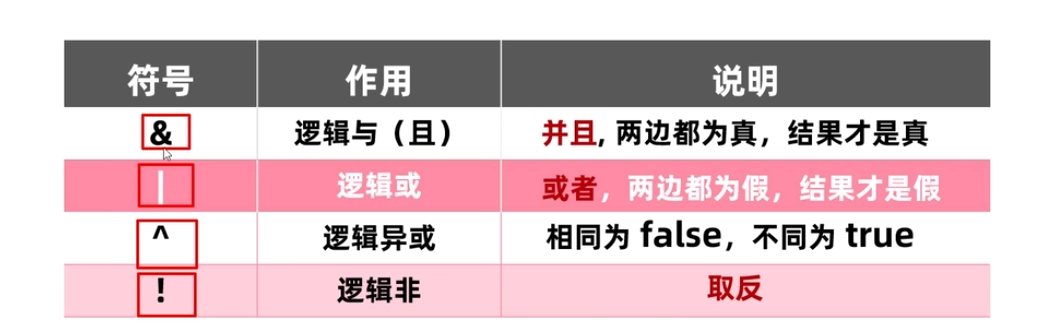
Long：8；

&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。

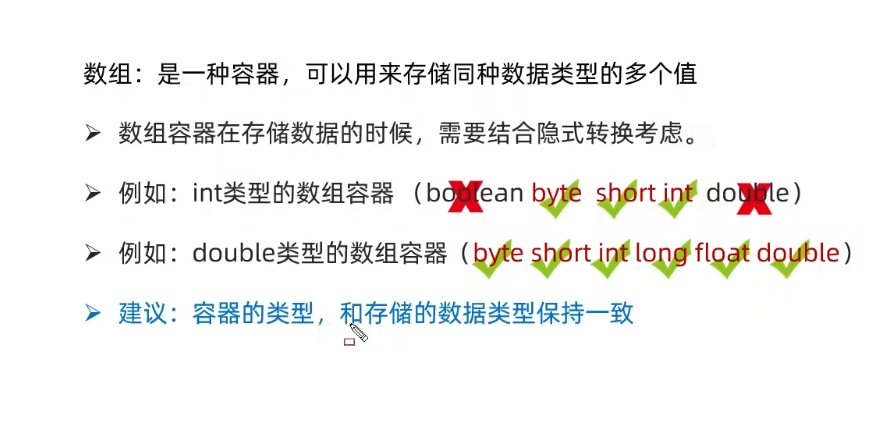
1. 导包
2. 创建对象
3. 生成随机数



从0开始，到-1结束；包头不包尾，包左不包右；

# Day06 数组

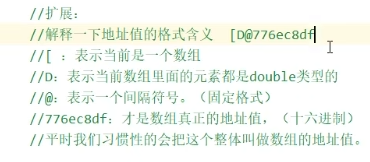
容器，存储同种数组类型的多个值。



例：int [] array

索引：下标。

访问：array在后是查询，在前是赋值。

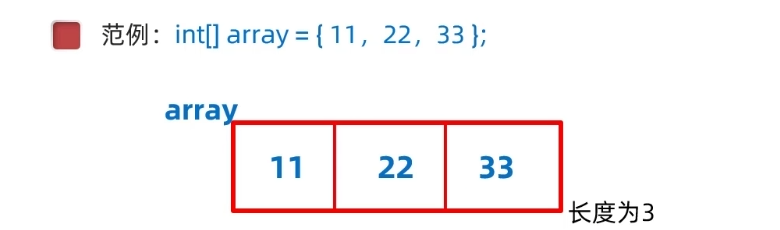


### 初始化

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，。。。}

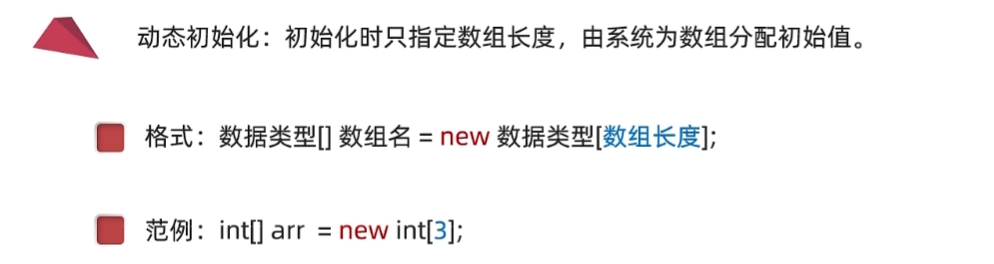


数组的长度属性：数组名.length

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：

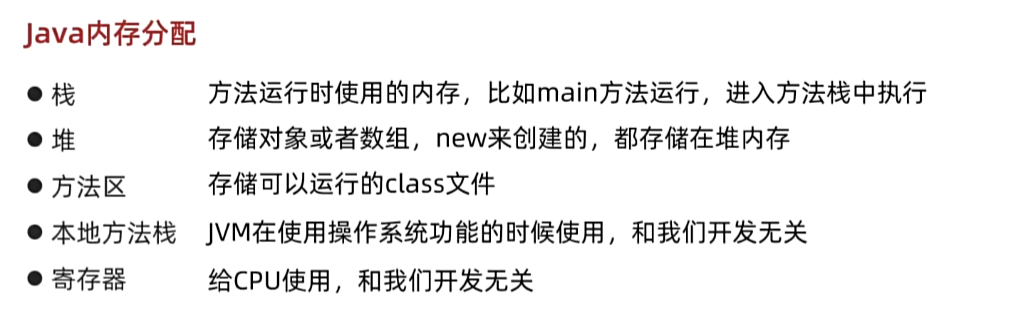
1. 定义数组
2. 修改数组
3. 遍历数组

常见问题

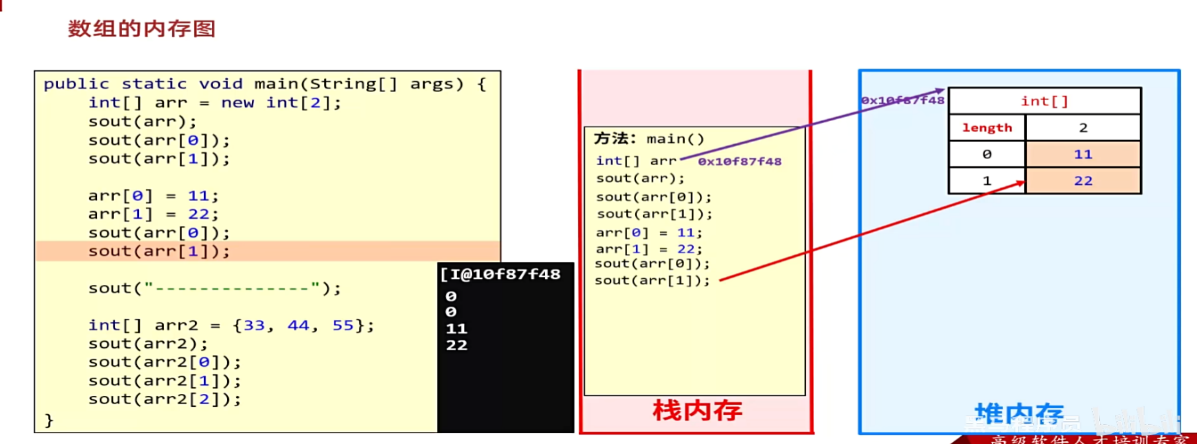
1. 索引越界异常：访问到了不存在的索引。

##### 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={{ },{ }}

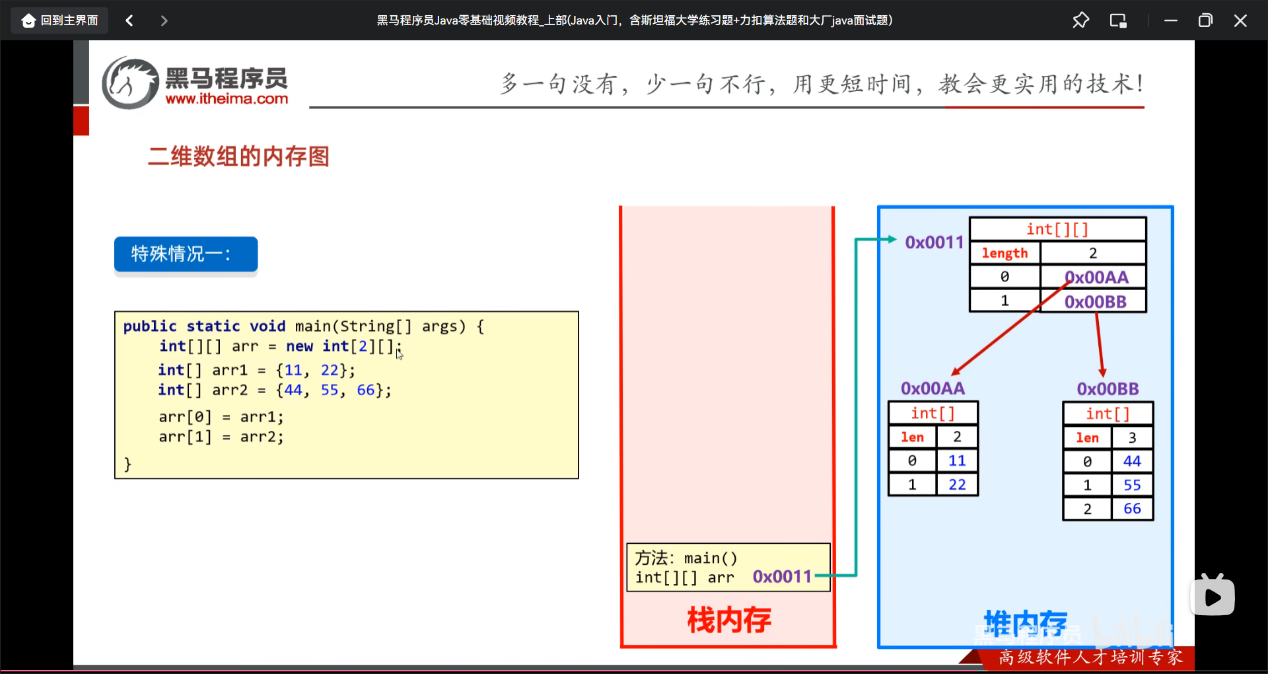
反: Int arr[][]={{ },{ }}

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；

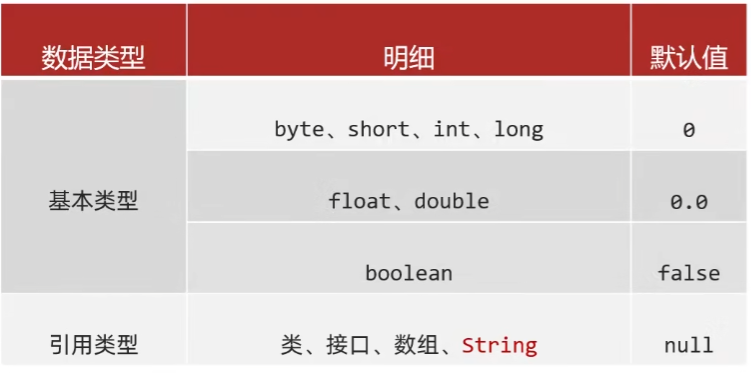
}

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。

## 构造方法

也叫做构造器，构造函数，在创建对象的时候给成员变量赋值的。

创造对象的时候，虚拟机自动调用，给成员变量进行初始化。

修饰符 类名（参数）{

方法体；

}

空参和全参。

构造方法的重载：有参的和无参的，虽然方法名相同，但是参数不同，所以叫重载。

# Day08 标准的javaBean

1. 见名知意
2. 成员变量使用private修饰
3. 提供至少两个构造方法
4. 成员变量要有对应的set和get方法。

Ptg插件。

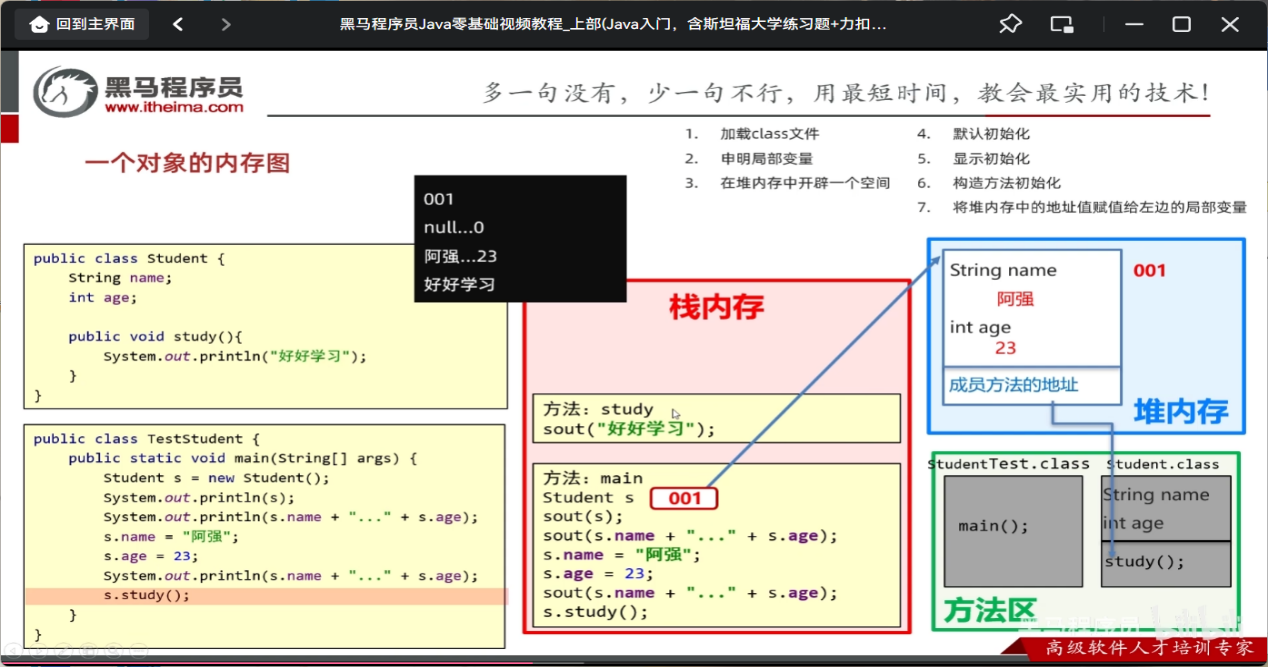
#### JavaBean内存分配

栈

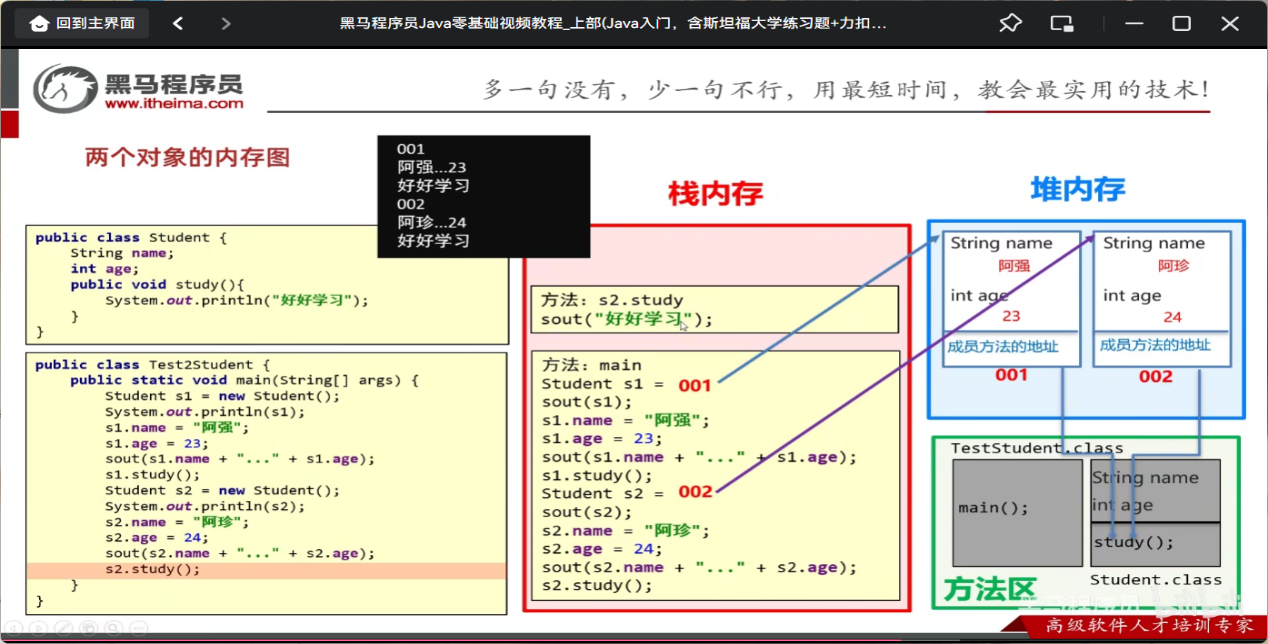
堆

方法区

一个对象的内存图



两个对象的内存图



**基本数据类型**：数据值是存储在自己的空间中的。（赋的真实的值）

**引用数据类型**：数据值是存储在其他的空间中（比如new在堆），自己空间中存储的是地址值。（赋值给其它变量，赋的地址值。）

###### This的内存原理

作用：区分局部变量和成员变量（代表方法调用者的地址值。）P89

**成员变量**：类中方法外。

**局部变量**：方法内，方法申明上。

# Day09 面向对象综述

SOUF(

第一部分参数：要输出的内容%s占位

第二部分参数：填充的数据

)

键盘录入（**不能混用**）P92

第一套：

NextInt();接收整数

Nextdouble();接收小数

Next();接受字符串

遇到空格，制表符，回车就停止接收。这些符号后面的数据就不会接收了。

第二套体系

Nextline();接收字符串

可以接收空格，制表符，遇到回车才停止接收数据。

令数组元素为空，则为删除该元素。

# Day10 API&字符串

API:应用程序编程接口。

Java API:jdk中提供的各种功能的java类。

##### String:是一个常量，创建后不可变，string值不能被改变可共享。

创建string对象：

1. 直接赋值
2. New

###### Java的内存模型：

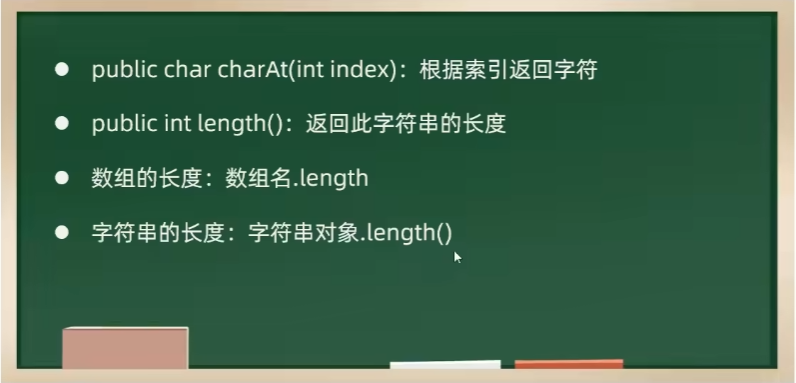
当双引号**直接赋值**时，系统会检查该字符串在传尺是否存在。

不存在—创建新的

存在—复用。

equals方法，比较字符串内容是否相等。

**String遍历：**

****

**//char类型的变量在参与计算的时候自动类型提升为int，查询ascii码表。**

###### Substring（包头不包尾）

1.获取一个手机号  
String phoneNumber="18337765145";  
2.截取手机号的前三位  
String start=phoneNumber.substring(0,3);//第0索引到第3索引  
3.截取手机号的后四位  
String end=phoneNumber.substring(7);  
4.拼接  
String result=start+" \*\*\*\* "+end;  
5.打印  
System.*out*.println(result);

**Replace（敏感词替换）**

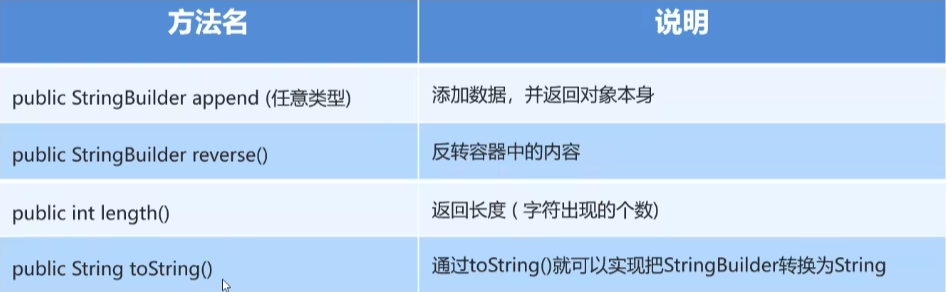
可以通过for循环反复查找敏感词库内的敏感词，进行替换。

##### StringBuilder P105(用于字符串拼接，反转。重要的是业务)

容器，创建之后里面的内容可变。（可提高字符串的操作效率）（打印时是属性值不是地址值）

构造方法

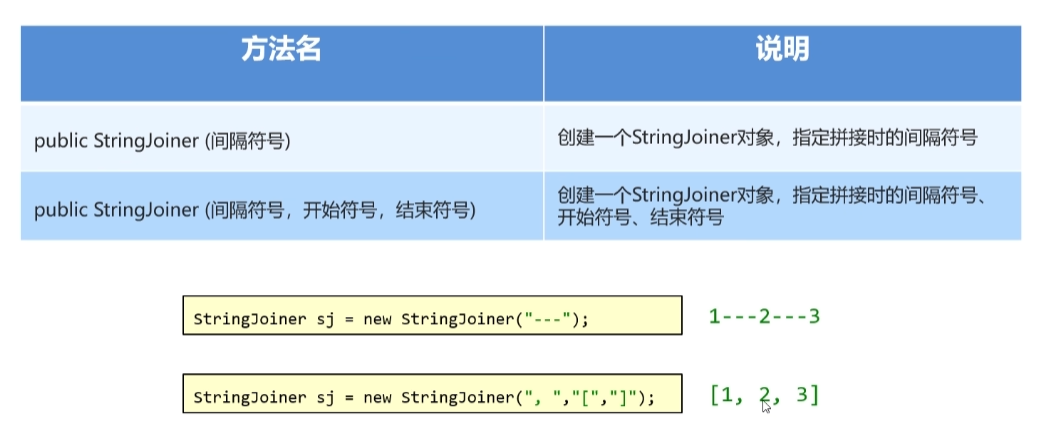




添加方法：.append

反转方法：.reverse

StringJoiner的构造方法





New string（字符数组）；tocharArray把字符串变成字符数组。

# Day11 集合

集合**长度**可变，可以存储**引用数据类型**。

#### ArrayList<E>

泛型：限定集合中存储的数据类型。

打印的不是地址值，而是存储在集合中的数据。

成员方法：



# Day12 学生管理系统

System.exit(0)//虚拟机停止运行。

# Day13 面向对象进阶

### 01 Stractic-静态变量

Stactic表示静态变量，是Java中的一个修饰符，可以修饰成员方法，成员变量。

**被stactic修饰的成员变量，叫静态变量。**

**特点**：被该类的所有对象共享。

不属于对象，属于类。

**调用方式**：类名调用（推荐），对象名调用。

静态变量是绣着类的加载而加载的，优先于对象先出现的。

**被stactic修饰的成员方法，叫成员方法。**

**特点**：多用在测试类和工具类中

Javabean中很少用。

**调用方式**：类名调用（推荐），对象名调用。

JavaBean类：用来描述一类事物的类。

测试类：用来检查其他类是否书写正确，带有main方法的类，是程序的入口。

**工具类**：不是用来描述一类事物的，二十帮我们做一些事情的类。

1. 类名见名知意。
2. 私有化结构方法。
3. 方法定义为静态方法。

静态方法**只能**访问静态变量和静态方法。

非静态方法**可以**访问静态变量或者静态方法，**也可以**访问非静态的成员变量和非静态的成员方法。

静态方法中是没有this关键字。

#### 多线程

Mian方法

Public：被jvm调用，访问权限足够大

Static： 被jvm调用，不用创建对象，直接访问

因为mian方法是静态的，所以测试类中其他方法也是静态的

Void： 被jvm调用，不需要给jvm返回值

Mian： 一个通用的名称，虽然不是关键字，但是被jvm直接识别

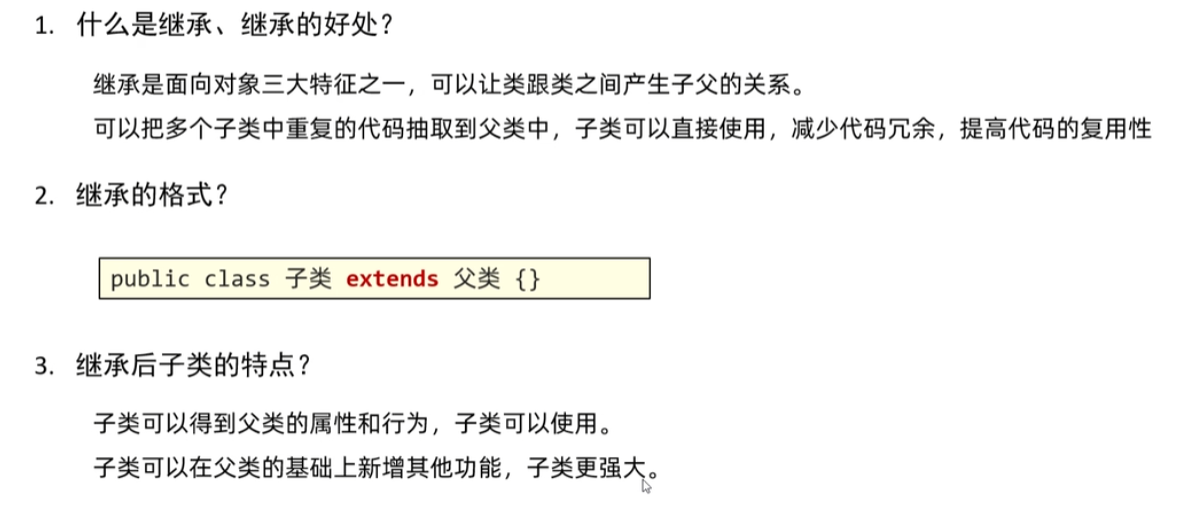
String[] arge：以前用于接收键盘按录入数据，现在没用

继承

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

什么时候用继承？

**当类与类之间，存在相同（共性）的内容**，**并满足子类是父类的一种**，就可以考虑使用继承来优化代码。

特点：

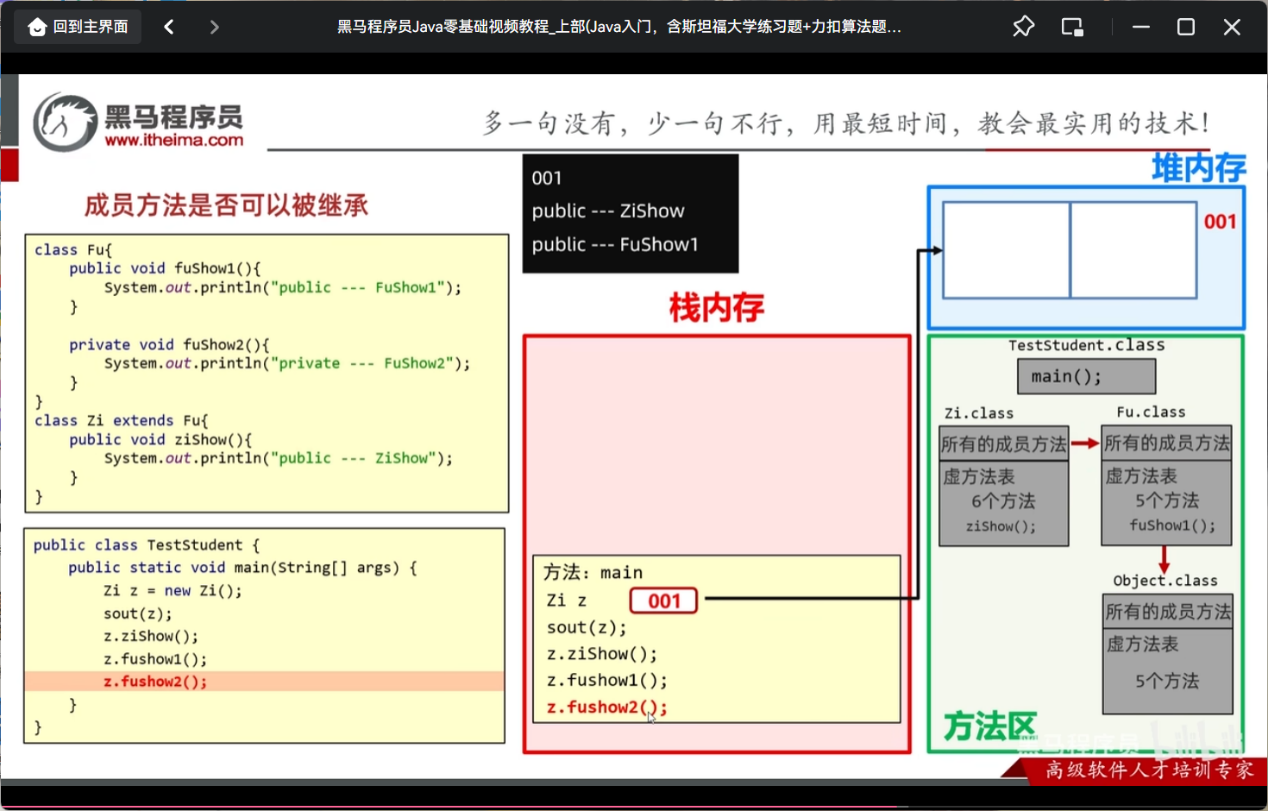
Java只支持**单继承**，不支持**多继承**，但支持**多层继承**。

默认继承或间接继承object类。

私有变量只能在本类中使用。

**Abstr 抽象类强制子类重写**





虚方法表：非private 非stactic 非final

## 内存分析工具p126

## 查看底层代码：Ctrl+f12

继承中成员变量访问特点：就近原则

先在局部位置找，本类成员位置找，父类成员位置找，逐级往上。

Name，this.name，super.name

调用成员变量时，本类的成员变量this可以省略

方法的重写

当父类的方法不能满足子类现在的需求时，需要进行方法重写。

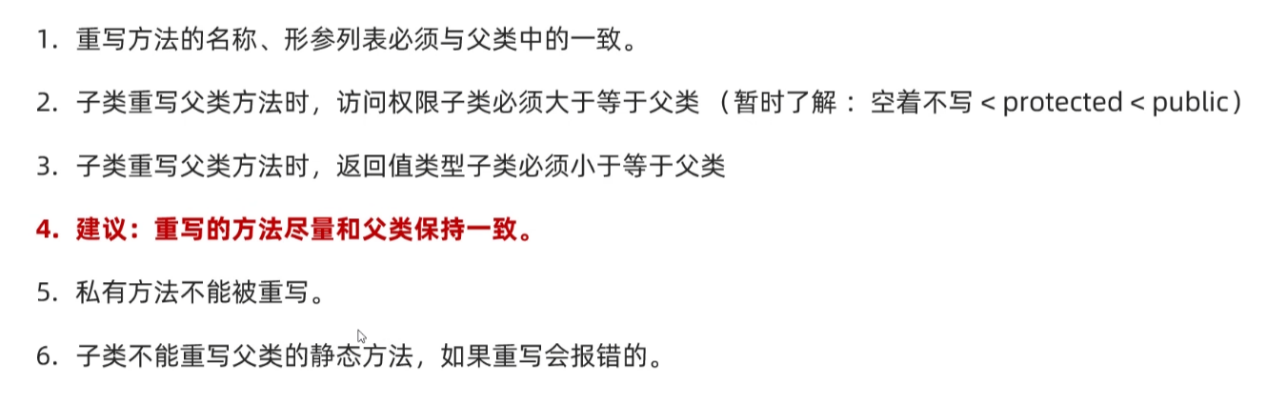
书写格式：

在继承体系中，子类出现了和父类中一摸一样的方法声明，我们就称子类这个方法为重写。

**@Override**重写注解

1. **@Override**是放在重写后的方法上，校验子类重写是语法是否正确。
2. 加上注解后如果有红色波浪线，表示语法错误。
3. 建议重写方法都加**@Override**注解，代码安全，优雅。

本质是覆盖了**虚方法表**中的成员方法

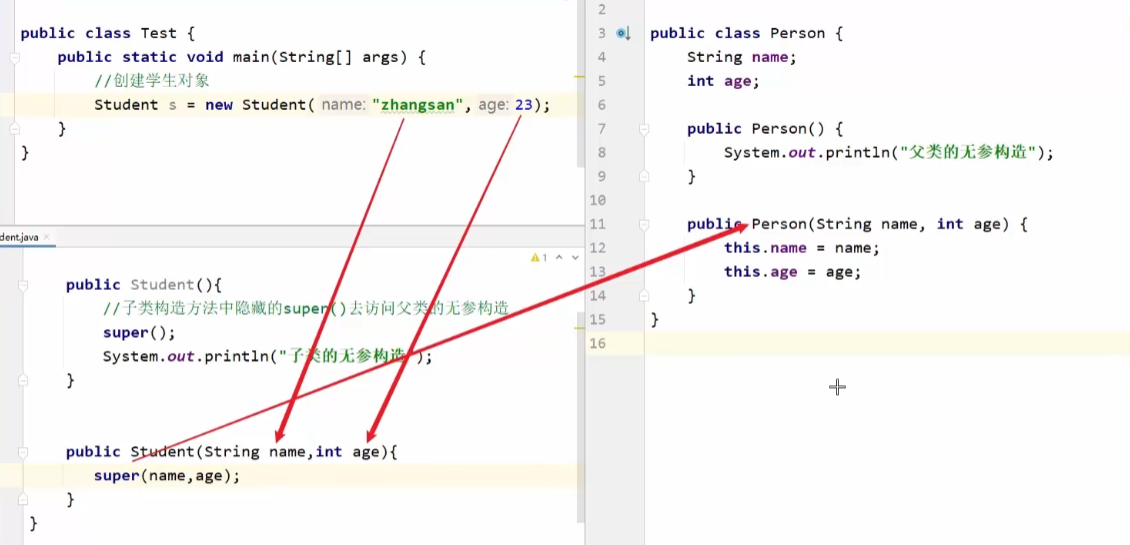


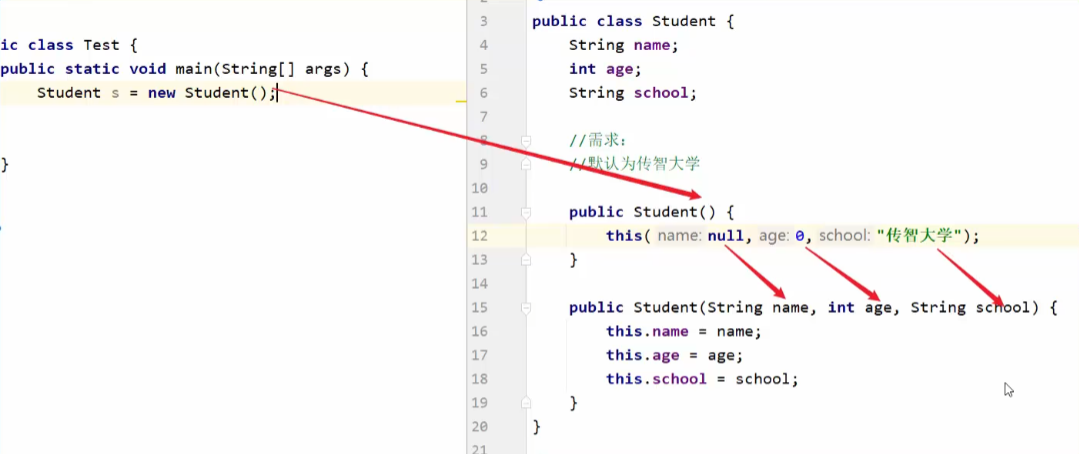
子类构造方法中隐藏的super（）去调用父类的空参构造。

当this（局部变量）被调用时才有值。

继承中构造方法的访问特点是什么？

* 子类不能继承父类的构造方法，但是可以通过super调用。
* 子类构造方法的第一行，有一个默认的super();
* 默认先访问父类中无参的构造方法，在执行自己。
* 如果想要方法问父类有参构造，必须手动书写。





# Day14 多态

##### 什么是多态？

同类型的对象，表现出的不同形态。

##### 多态的表现形式

父类类型 对象名称=子类对象；

##### 多态的前提

* 有继承/实现关系；
* 有父类引用指向子类对象
* 有方法重写

##### 多态的好处

使用父类型作为参考，可以接收所有子类对象，体现多态的扩展性与便利性。

##### 多态的特点

调用成员方法：编译看左边，运行看右边

调用成员变量：编译看左边，运行也看左边

##### 多态的优势

在多态形势下，右边对象可以实现解耦合，便于扩展和维护。

Person p=new Student();

p.work()；//业务逻辑发生改变时后续代码无需修改

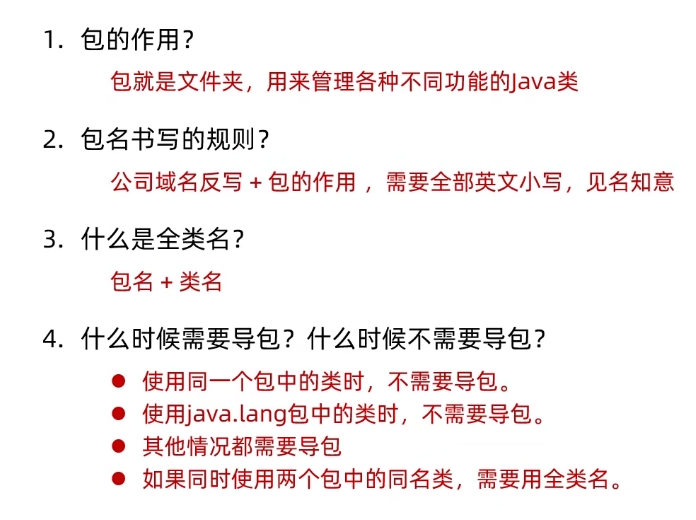
定义方法的时候，使用父类型作为参数，可以接收所有子类对象，体现多态的扩展性与便利

多态不能使用子类特有方法，除非强制类型转换。

强制类型转换：a instanceof Dog d(true); Dog d=(dog) a;

包，final，权限修饰符，代码块

导包关键词：import。



##### Final：

方法：表示该方法是最终方法，不能被重写。

类：表示该类是最终类，不能被继承。

变量：叫做常量，只能被赋值一次。（全部大写，单词之间用下划线隔开）

常量：在实际开发中，常量一般作为系统的配置信息，方便维护，提高可读性。（静态方法只能使用静态常量）

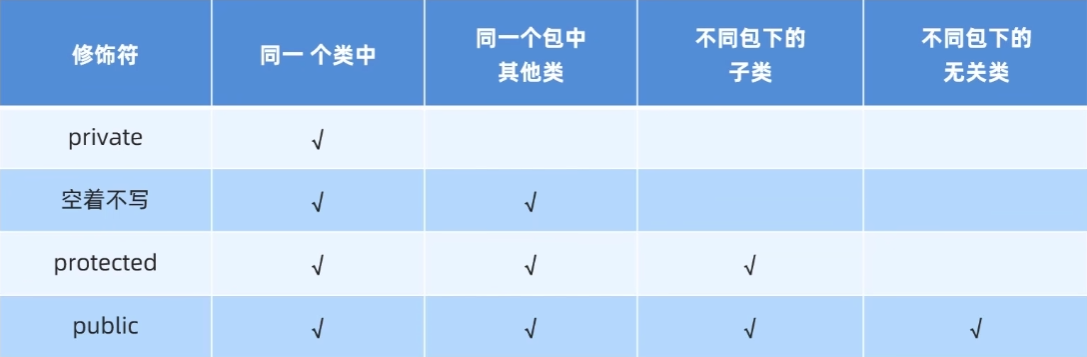
**细节：**

Final修饰的变量是基本类型：那么变量存储的**数据值**不能发生改变。

Final修饰的变量是引用类型：那么变量存储的**地址值**不能发生改变，对象内部的属性值可以发生改变。

##### 权限修饰符

Eg：private<空着不写<protected<public



如果方法中的代码是抽取其他方法中共性代码，这个方法一般也私有。

##### 代码块

**局部代码块：**节省空间，用完就释放。

**构造代码块：**

1. 写在成员位置的代码块
2. 作用：可以把多个构造方法中重复的代码抽取出来
3. 执行时机：我们在创建本类对象的时候会先执行构造代码块在执行构造方法。

**静态代码块：**

格式：static{}

特点：需要通过static关键字修饰，随着类的加载而加载，并且自动触发，只执行一次。

使用场景：在类加载的时候，做一些**数据初始化**的时候使用。

# Day15 抽象类

面向对象的三大特征

封装：对象代表什么，就得封装对应的数据，并提供数据（get和set方法，继承提高代码复用性，多态）

抽象方法：将共性的行为（方法）抽取到父类之后。由于每一个子类执行的内容是不一样，所以，在父类中不能确定具体的方法体。该方法就可以定义为抽象方法。

抽象类，如果一个类中存在抽象方法，那么该类就必须声明为抽象类。

抽象方法定义格式：

Public **abstract** 返回值类型 方法名（参数列表）；

抽象类的定义格式：

Public **abstract** class 类名{}

注意事项！！

* 抽象类不能实例化
* 抽象类中不一定有抽象方法，有抽象方法的类一定是抽象类
* 可以有构造方法
* 抽象类的子类

要么重写抽象类中所有的抽象方法

要么也是抽象类