语言基础

001

计算机的发展

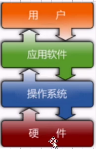


物联网

002

操作系统和平台相关性

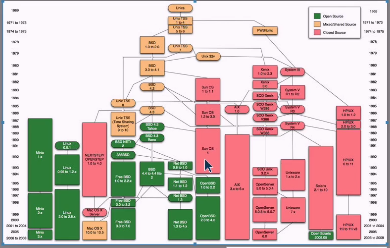
OS



Windows发展



Unix发展



平台相关性：

不同的平台都有指定的指令格式

Linux没有扩展名，不是按照扩展名来区分的。

平台相关性关系到应用软件的迁移

003

Windows键盘的操作和常见命令

004 太极和八卦

005进制概述

十进制：0 1 2 3 4 5 6 7 8 9；

二进制：0 1

八进制：0 1 2 3 4 5 6 7

十六进制：0 1 2 3 4 5 6 7 8 9 A B C D E F

七进制：0 1 2 3 4 5 6 7

电脑存储单位：

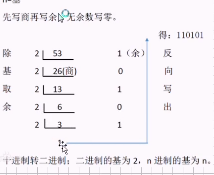
二进制存储：bit byte k MB GB TB PB

一个字节8位即 1byte = 8 bits

1k=1024byte

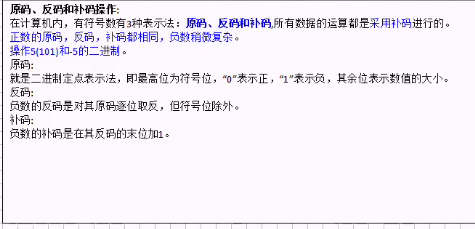
1MB=1024K

进制转换：

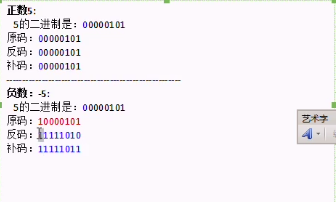


007

原码 补码（所有运算都是补码运算） 反码



正数的原码、反码、补码都相同

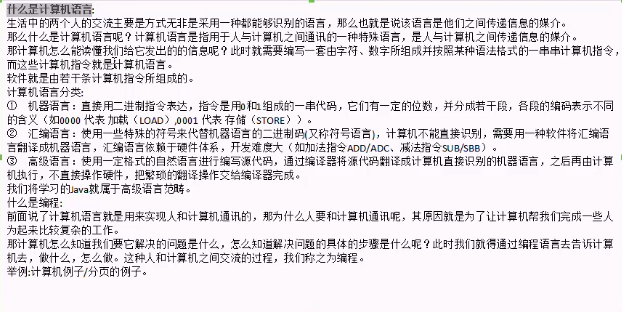


反码是原码的相应位0变1，1变0；符号为不变

补码是在反码的基础上在最后一位上+1

SUN和java

008



机器语言

二进制指令

汇编语言

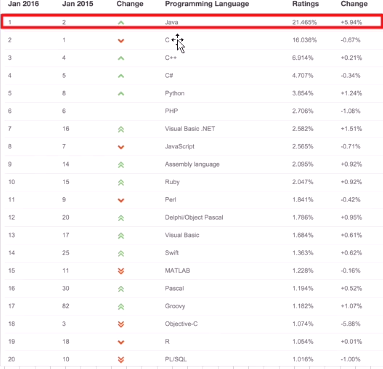
二进制指令的符号

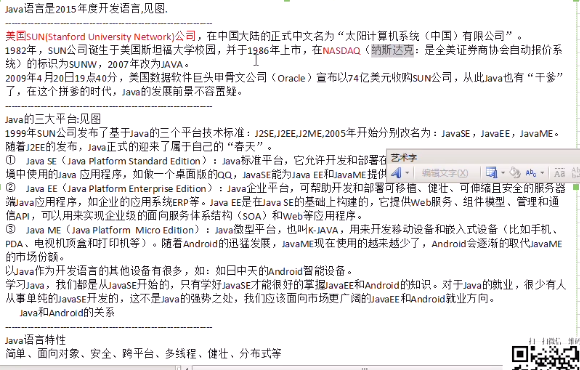
高级语言

自然语言进行编码，然后利用编译器生成计算机直接识别的机器语言

009

2015-2016语言排名



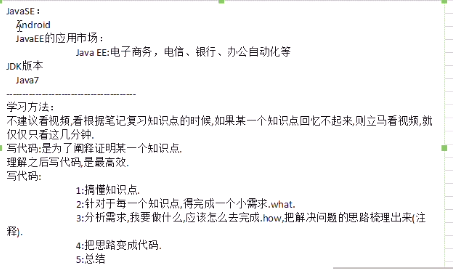
java三大平台：

j2se:开发桌面程序

j2ee：开发BS

j2me：嵌入式语言

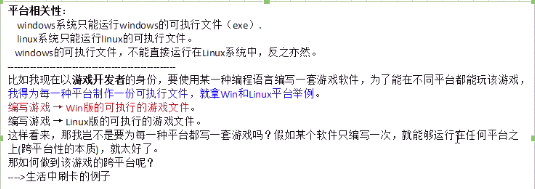
010 javase课程体系和学习方法



Java核心是1.4,增加不同的内容是从而有不同的版本。

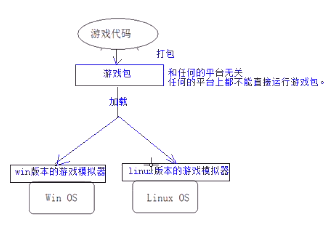
步骤很重要，思维很重要；写代码之前一定要先写思路，开始代码可能与思路出入较大，但是写多了最后思路与代码就基本一致了

011 引出扩平台型和可移植性

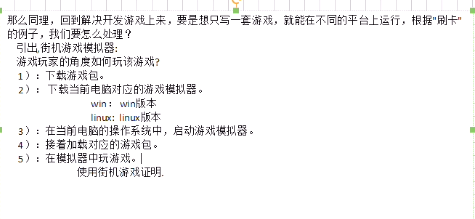




012 实现街机游戏的跨平台性

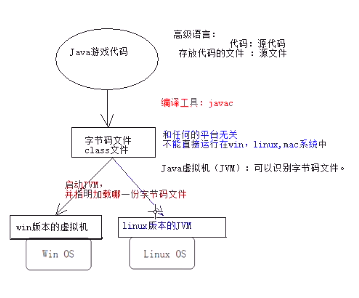


对于开发者来说只需要开发游戏包就可以了

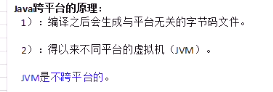


游戏包和平台无关，游戏模拟器和平台无关。

013 java的跨平台性原理



启动jvm的命令：java

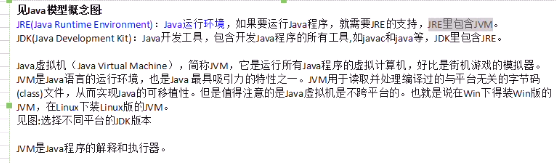


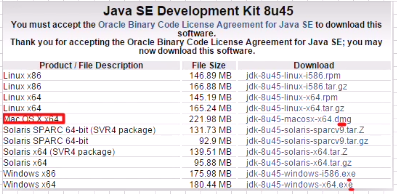
014 jdk-jre-jvm概述

Jdk：java development kit

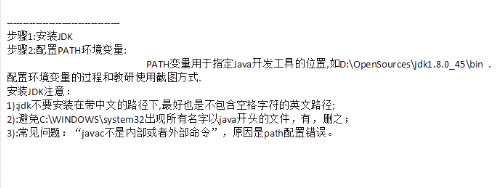
Jre：java runtime environment

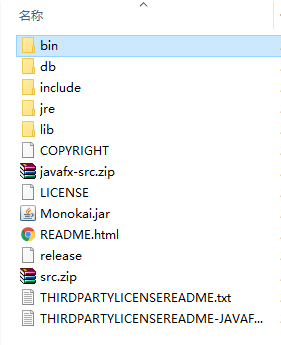
Jvm:java virtual machine

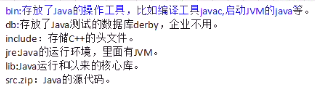




015 搭建java开发和运行环境







016 第一个java程序

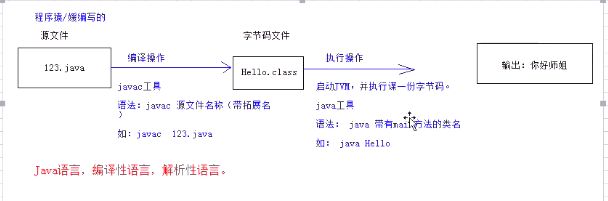
编写源代码

编译：javac 源文件

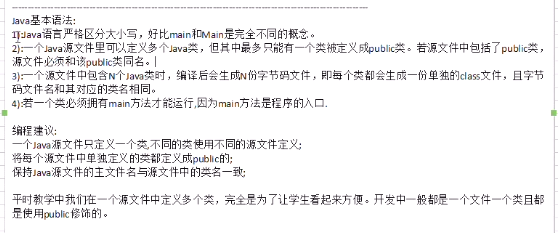
启动jvm加载字节码文件并解释执行：java 源文件名



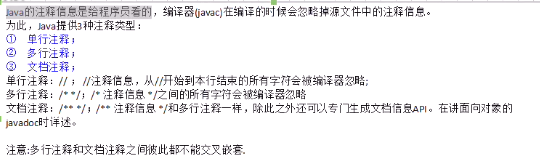
017 java的编译和运行机制



019 java基本语法

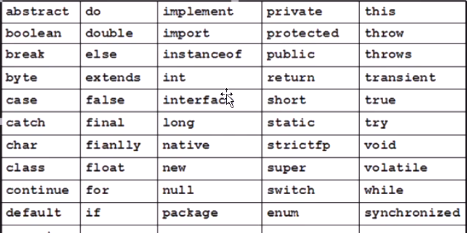


020 java的三大注释

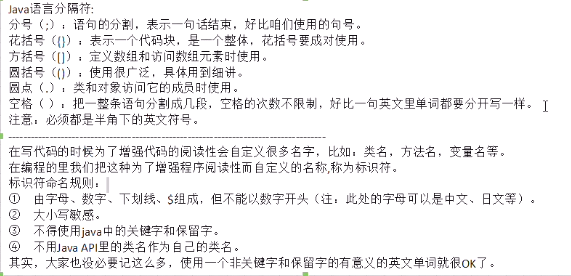


021 java中的关键字和保留字

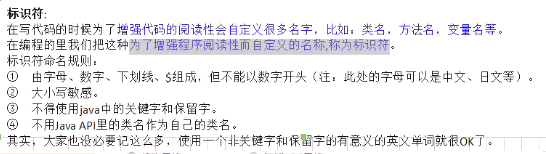




022 java中的语言分隔符



025 java语言的标识符规范



024 Editplus工具

Day 02

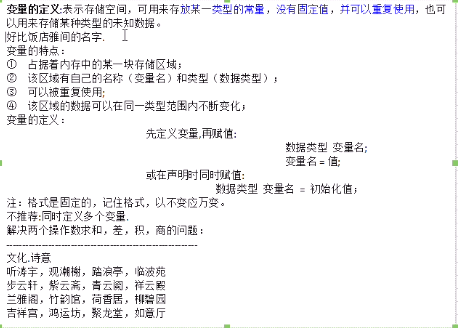
001 字面量和常量



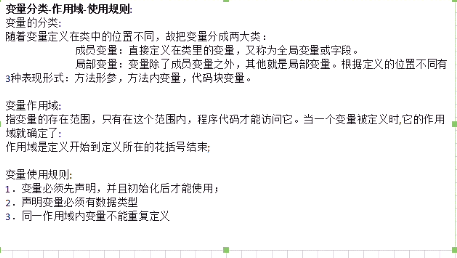
字面量就是直接给出的一个值（可以是整数小数true false等）

002 引出变量

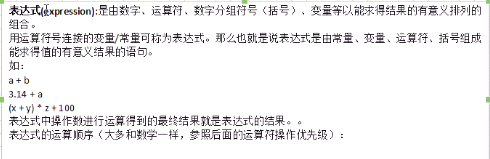
003 变量的定义和基本使用



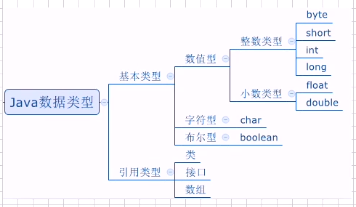
004 变量的分类、作用域和使用规则



005 java表达式



006 数据类型和分类



基本数据类型/原生数据类型

数值型：

整数型：byte short int long

小数型：float double

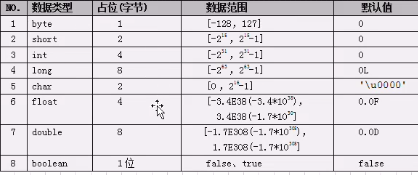
不同整数类型在内存中占用的空间大小不同。

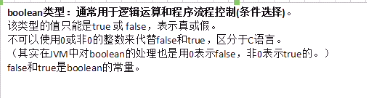
字符型:char

布尔型:Boolean,true和false

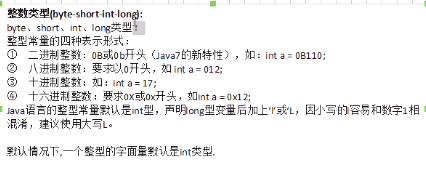
引用数据类型/对象数据类型

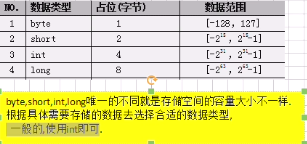
007 布尔类型

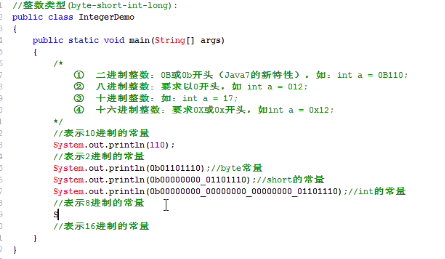




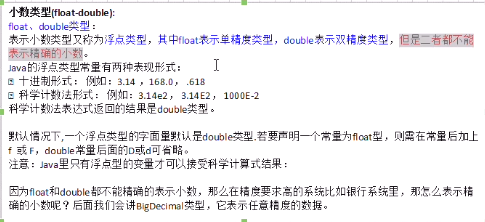
008 整数类型



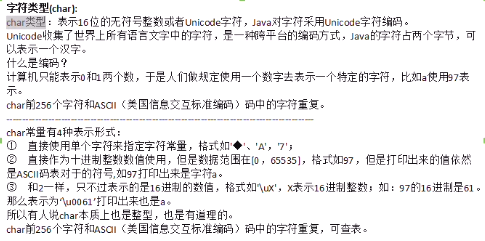




009 小数类型



010 字符类型



从某方面来讲，char就是整数类型，0~2^16-1，最高位不是符号位

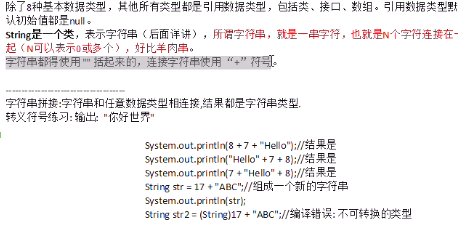
Boolean在jvm中也是由0和1来表示的

char c1=65;

System.out.println(c1); //输出A

011 最常用的引用类型

String

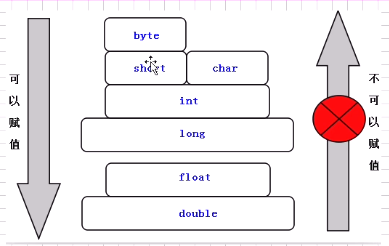


012 数据过大和溢出



数字运算都是二进制运算，即先转换为二进制在运算

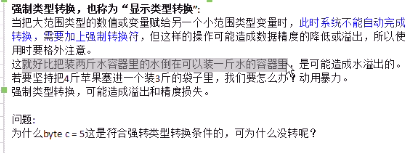
013 基本类型的自动转换



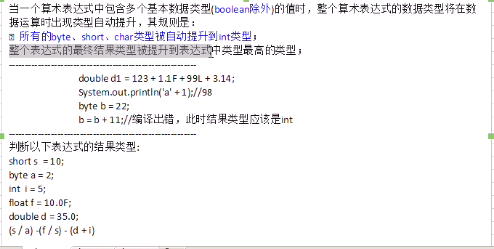


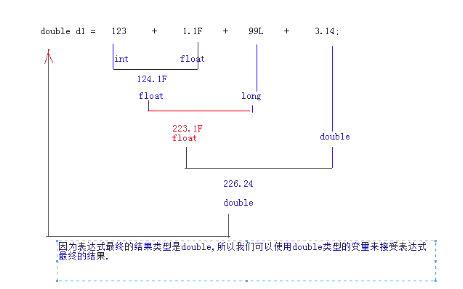


014 基本数据类型的强制转换

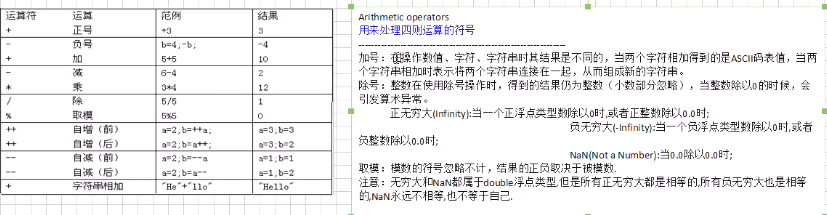


015 表达式类型的自动提升

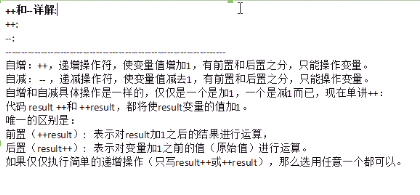




016 算数运算符

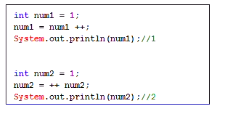


017 ++和--的详解

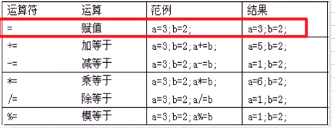


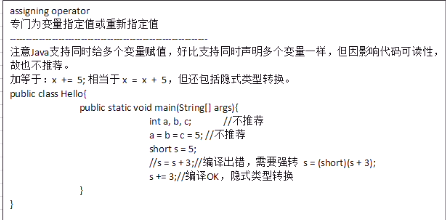
++ -- 都只能操作变量，不能操作常量（字面常量和定义的常量）

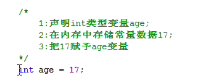
018 自增的面试题



019 赋值运算符

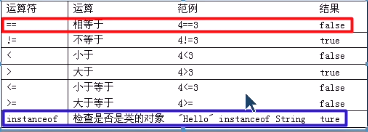




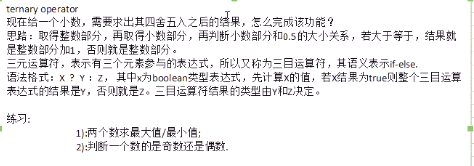


+= -= \*= /= 包含隐式类型转换

020 比较运算符



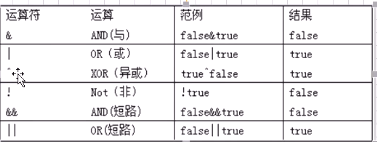
021 三元运算符

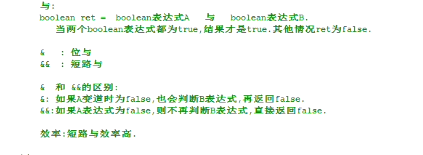


022 分页业务逻辑算法

边界条件判断

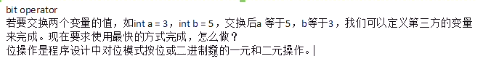
023 逻辑运算符

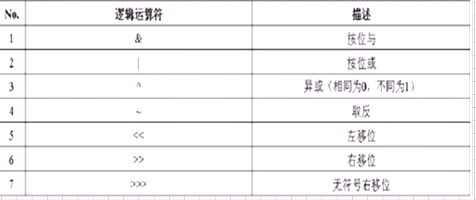


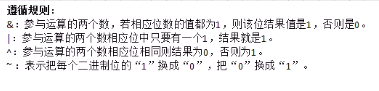


短路运算符效率更高

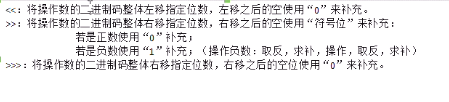
024 为运算符





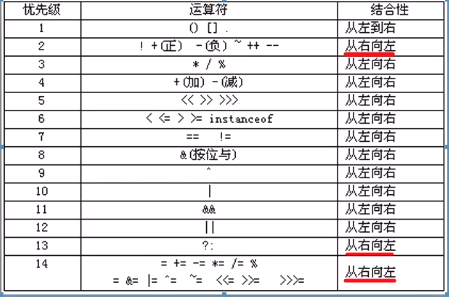


026 移位操作



027 交换两个变量的值

028 运算符操作的优先级



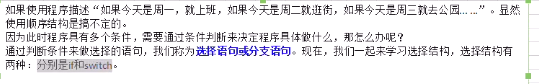
028 小结

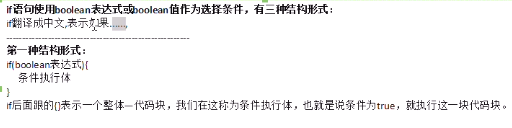
Day03

001 顺序结构

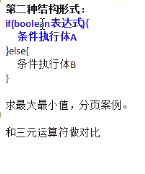


002 if结构

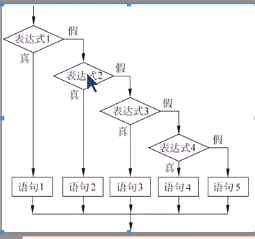




003 if-else结构

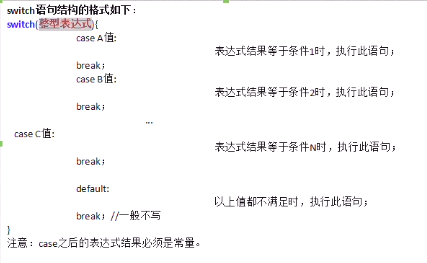


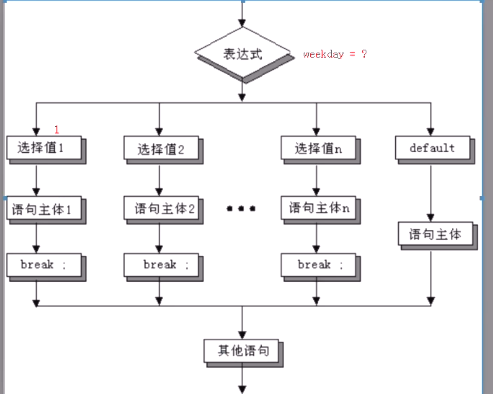
004 if-else if-else语句

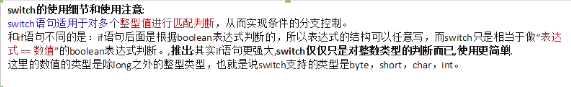


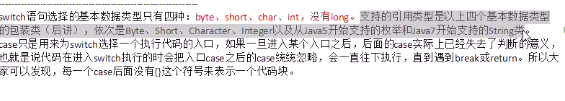
005

006 switch语句的简单使用



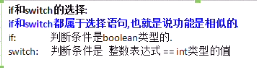






穿透

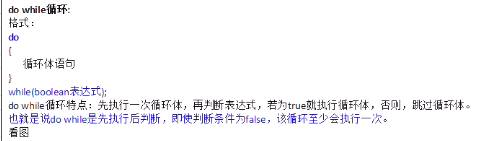
008if和switch的选择



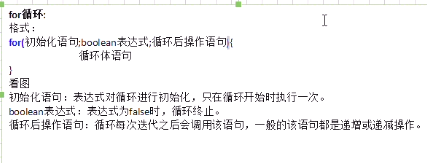
009引出循环语句

010While循环

011do-while循环



012 for循环





014嵌套循环



015嵌套循环性能优化

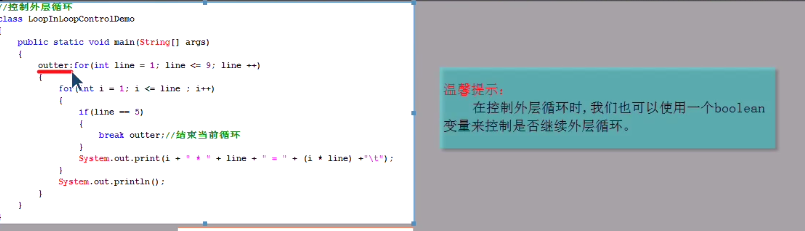
019控制循环语句

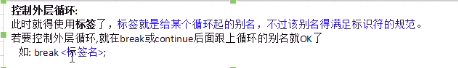
Break

Continue

Return

Goto



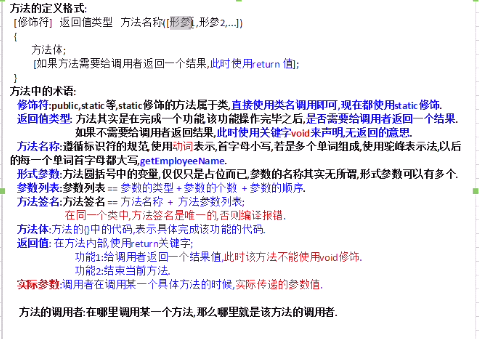


Day 04

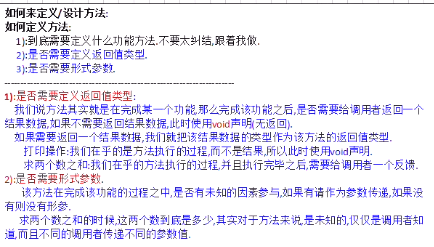
01 引出方法

02方法定义和基本调用操作

03方法中的专业术语

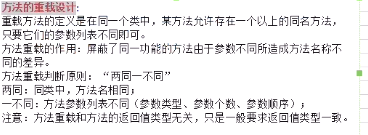


04如何设计方法

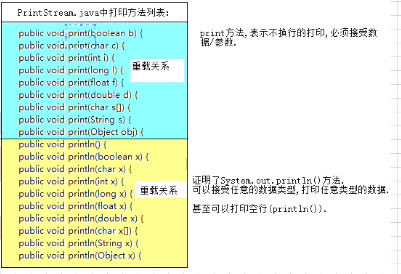


05方法的执行流程

06方法重载设计

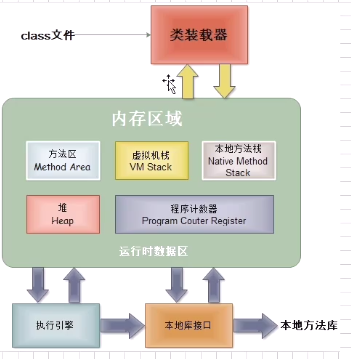


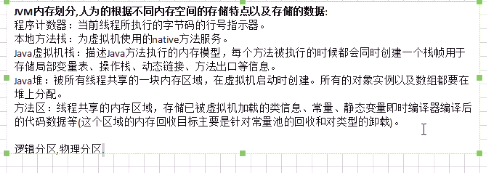
08方法重载在jdk中的使用

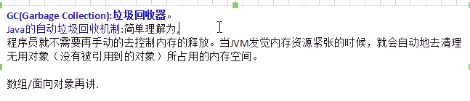


09 方法的递归操作

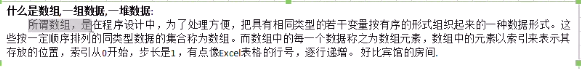
10jvm内存模型



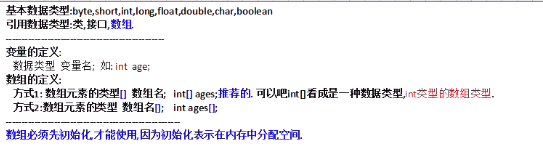




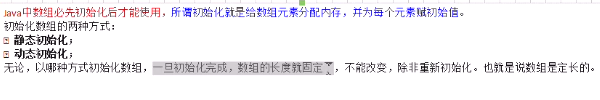
011引出数组和数组定义



类型一致



012数组的静态初始化和内存分析



数组必须先初始化才能使用，即在内存中分配空间。

数组初始化后长度就固定了

数组是定长的，元素个数是固定的。

静态初始化：

Int[] var=new int[]{1,2,3};

或者 int[] var={1,2,3}; 这种写法必须初始化

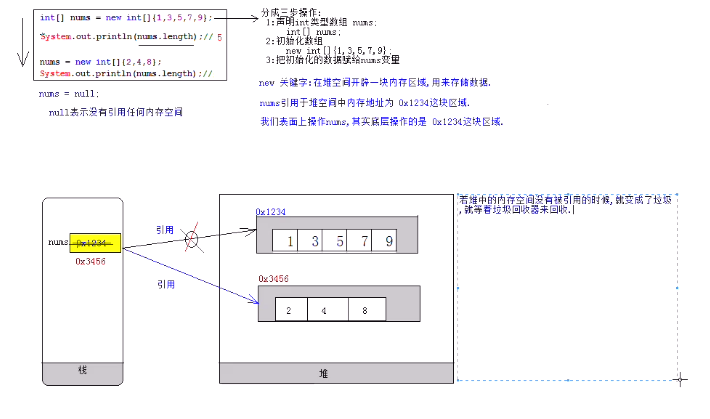
动态初始化：

Int[] var=new int[3];

Var[0]=1;

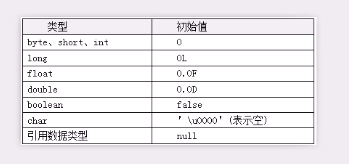
Var[1]=2;

Var[2]=3;



13 数组的动态初始化和内存分析

不同数据类型的初始值：





14 获取设置遍历数组的元素

15操作数组的常见异常

NullPointerException：空指针异常，数组未初始化就使用

ArrayIndexOutOfBoundsException：数组索引越界异常

16获取数组中最大和最小元素

定义方法求max和min

17格式化打印数组

定义一个方法遍历数组打印元素

18逆序排列数组元素

19元素出现索引

20小结



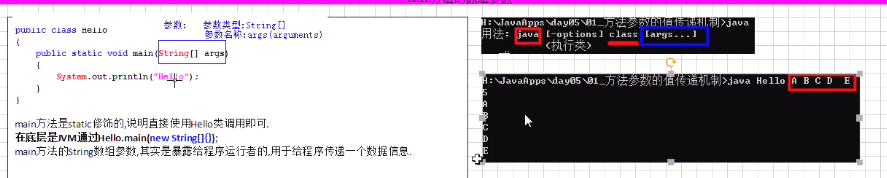
栈：变量名存在栈中；基本变量的值直接存在于栈中。

每一个方法有一个栈帧

堆：使用new关键字在堆中分配空间

Day 05

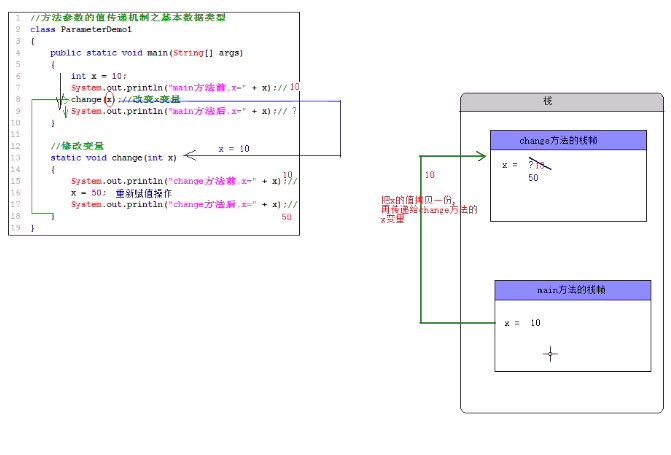
01main方法的数组参数



02参数的值传递机制

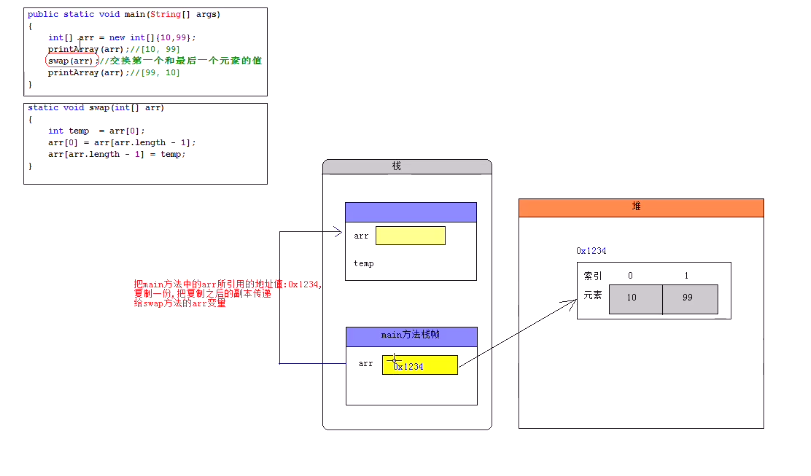
栈：每一个方法都存在一个栈帧，方法的执行和退出为压栈和出栈；

变量名和基本变量的值存储在栈中。



值传递时是先拷贝在传递

03 参数的值传递机制-引用数据类型

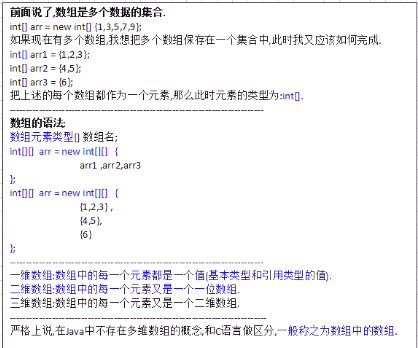


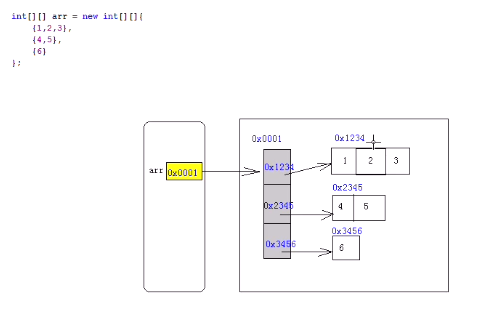
引用传递是先拷贝地址，然后将地址传递，这时两个变量操作的是同一个对空间

堆是共享的

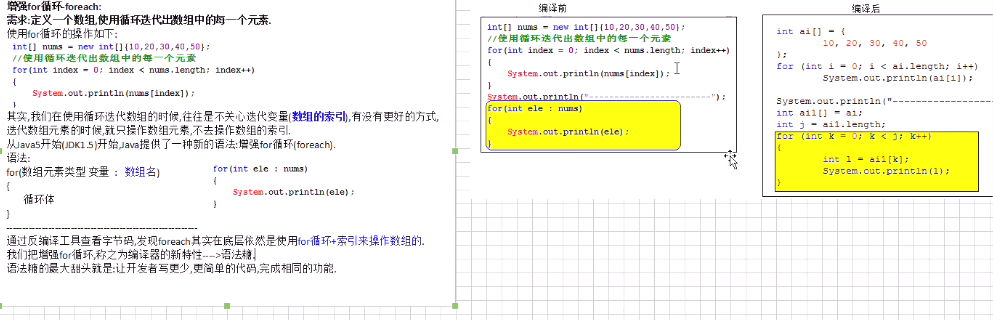
04多维数组

05多维数组的操作





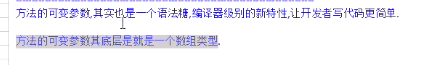
06增强for循环



底层仍然是由for来循环的，只是又重新赋值给了一个新的变量。

因此foreach不能数组的值

07 java5对于数组的支持-方法的可变参数

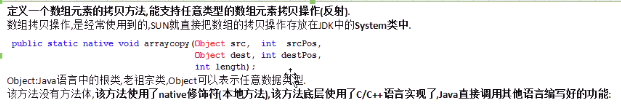


底层实现仍然是数组类型，如果定义两个同名的方法，一个参数为数组，一个为…，编译的时候会报错，定义了两个相同的方法。

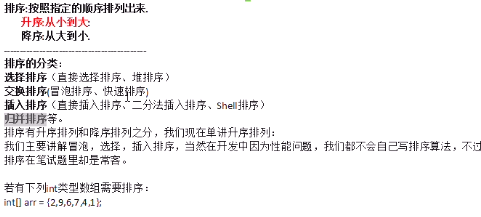
可变参数必须作为方法参数的最后一个。

08实现数组元素的拷贝

09system中的copy方法

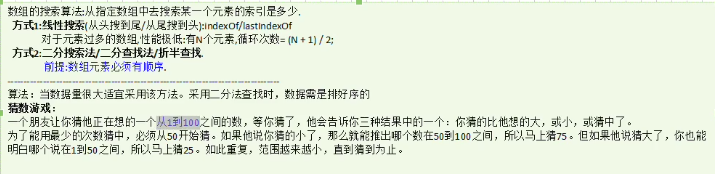


10冒泡排序



11选择排序

12 二分查找



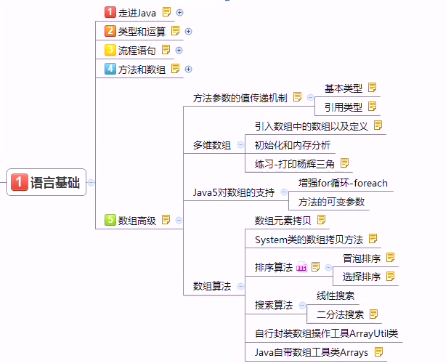
13自行封装操作数组的工具类

定义数组工具类-ArrayUtil

14java内置工具类

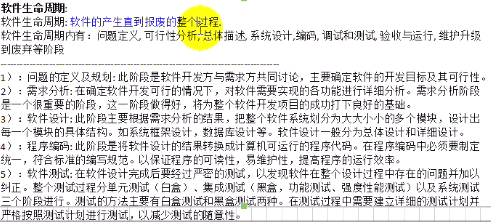
Java默认到lang包里面寻找外部类的属性或者方法，所以lang包不用显示的引入，而其他的包需要显示的引入或者在使用时使用带报名的全名引用。

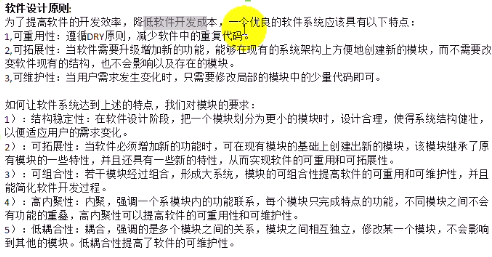
15小结



Day 06

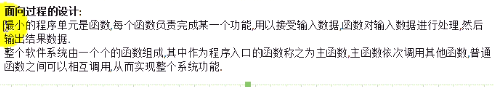
01 软件生命周期和软件设计原则



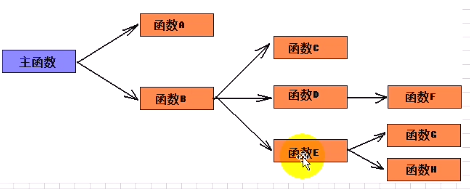


02软件设计之面向过程

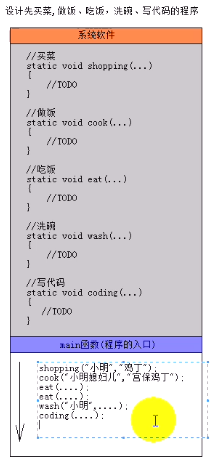


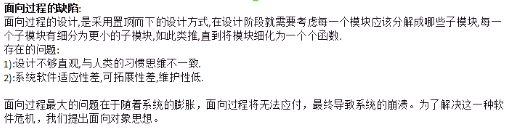


强调功能和行为



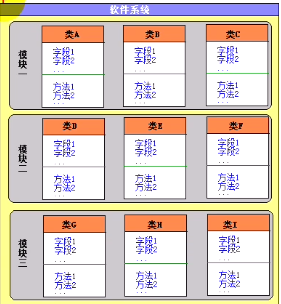
面向过程的程序设计：



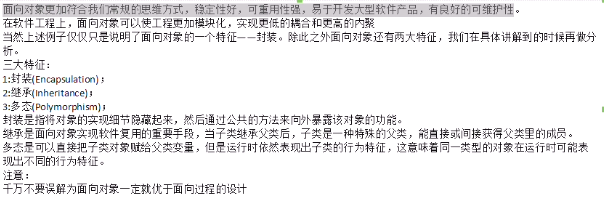


02面向对象

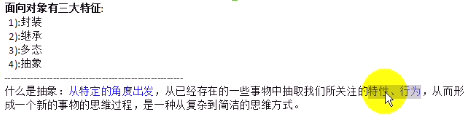




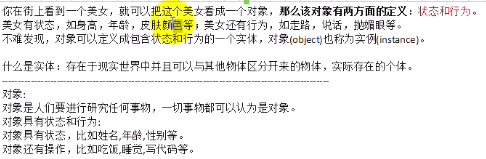
面向对象最小的程序单元是类



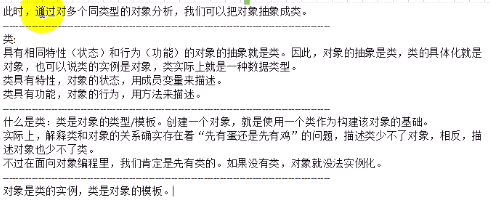
04理解什么是面向对象



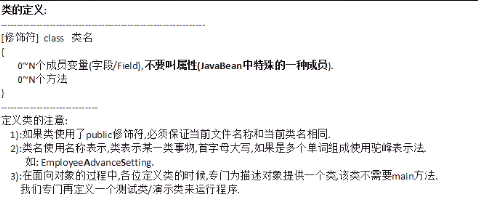
05 对象和类的关系



对象也叫实例



06 类的定义

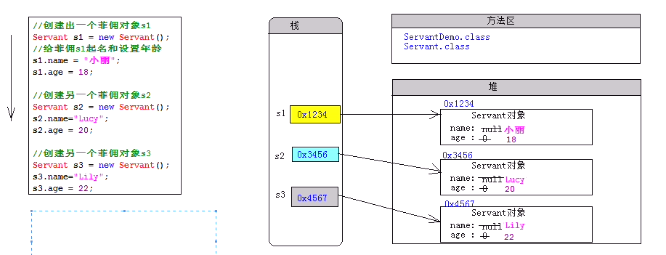


一个java文件只能由一个public修饰的类，且与文件名一致

07对象的创建和调用方法以及操作字段（成员变量）

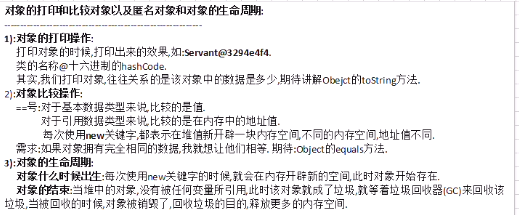
New关键词就会在内存中开辟一块空间

08内存分析-对象的实例化过程



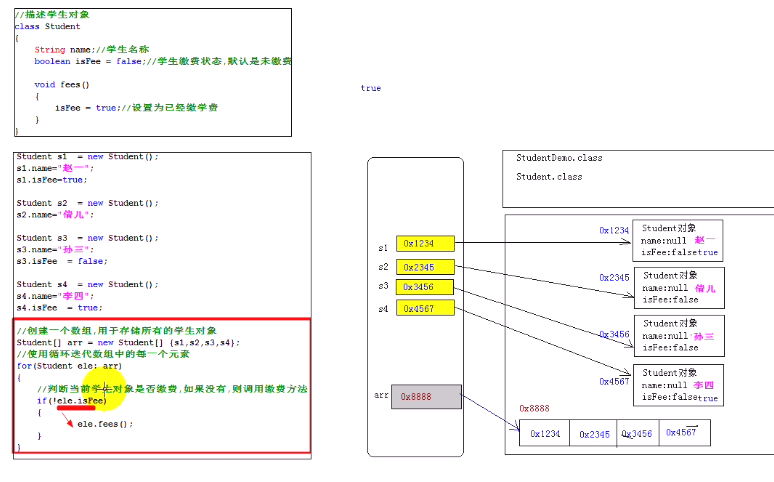


09对象的打印-比较和对象的声明周期-匿名对象

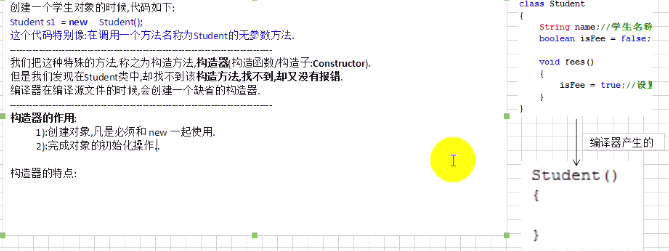




10



11构造器概述和缺省构造器



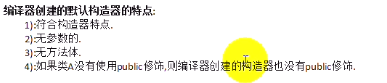
构造器的特点：

1) 构造器的名称和当前类的名称相同

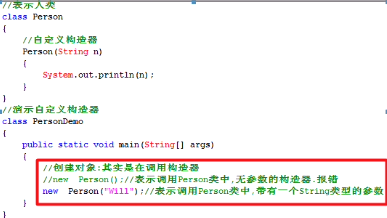
2) 禁止定义返回类型，千万不要使用void作为返回类型

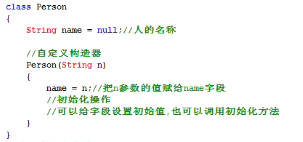
3) 在构造器中不需要使用return语句

其实构造器是有返回值的，返回的是当前创建的引用，即创建时有变量接受，String s1=new String()

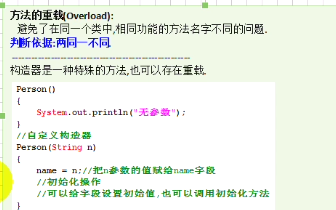


12 自定义构造器和构造器重载

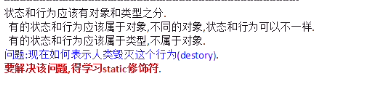
、

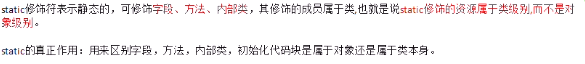


构造器重载：



13static修饰符和特点





Static修饰符的特点：

1)Static 修饰符的成员/字段/方法随着类的加载而加载

当jvm把字节码加载进jvm的时候，static修饰的成员已经在内存中存在

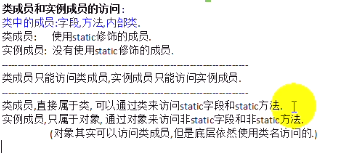
2)由于对象而存在

对象是我们手动通过new关键字创建出来的

3) static修饰的成员被该类型的所有对象所共享

4)直接使用类名访问static成员

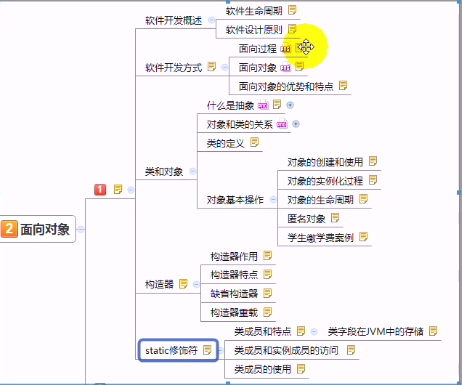
15static成员在jvm中的存储



在static方法中只能调用static成员

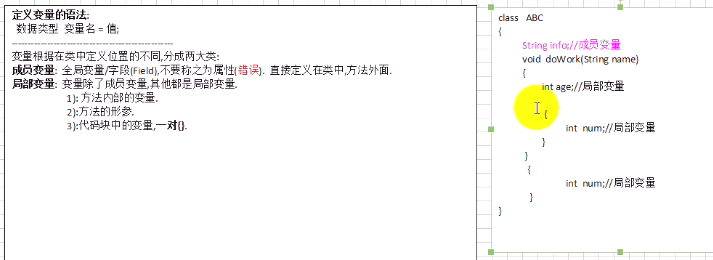
非static可以访问静态成员

15 小结

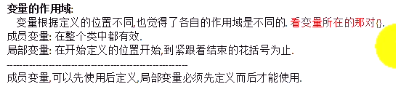


Day 07

01 变量的分类-作用域-初始值

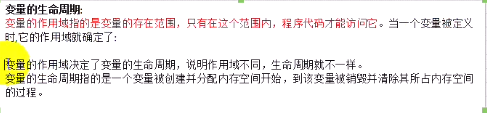




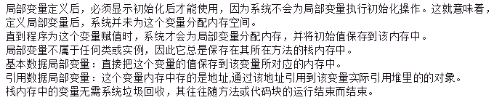


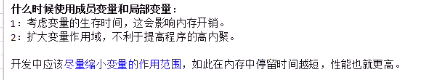
因为调用成员变量是已经定义了对象，而对象存在则成员变量已经存在。

02变量的生命周期和选用







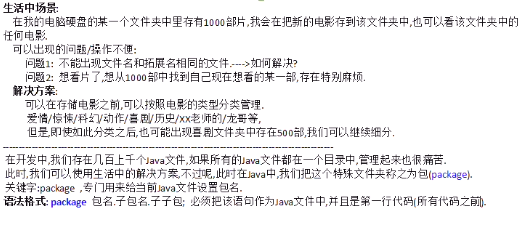


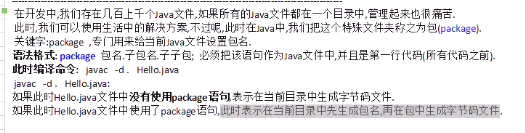
不要随意使用static修饰，非常消耗内存

也不要使用成员变量，因为存在线程的不安全问题（单例模式下）。

能使用局部变量就使用局部变量，因为局部变量是使用时单独开辟空间，用完即销毁。

03package关键字





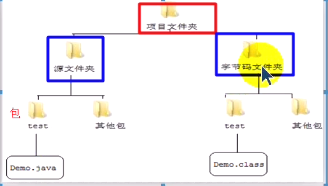
Javac –d . 文件名

运行时执行java pacekage名称.类名

04 package最佳实践



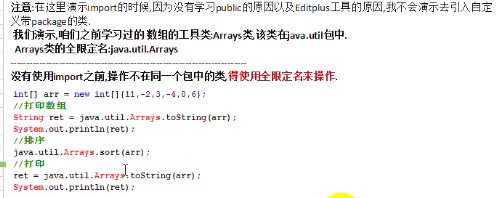
在开发中都是先建立包在定义类



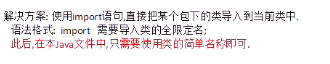
包的建立和字节码的建立时一一对应的。

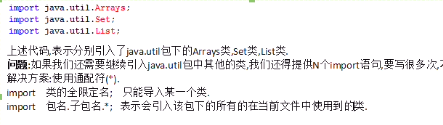


05 import语句



Java.lang包自动被编译器引入

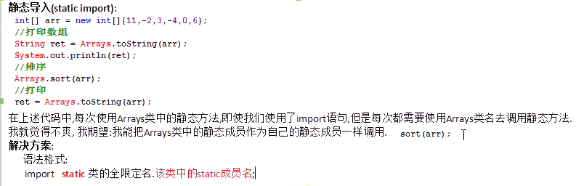




使用通配符\*也是使用哪个类引入哪个类，而不是全部引入，否则会占用很大的内存。

通配符也是语法糖，在编译的字节码文件也是转换为多个import语句

06 static import静态导入

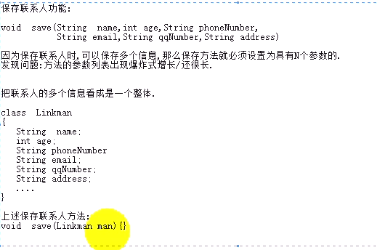


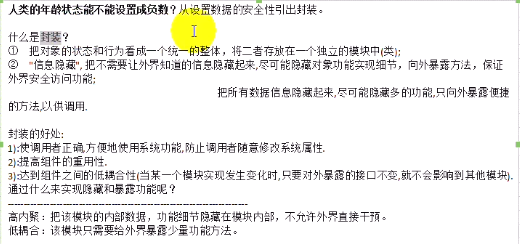
静态导入要导入要使用的成员，也可以使用通配符\*。

成员名可以是成员变量也可以是静态方法

通过反编译工具，其实所谓的静态导入也是一个语法糖，在实际开发中我们不使用静态导入，因为我们分不清方法或字段是当前类的还是导入的。

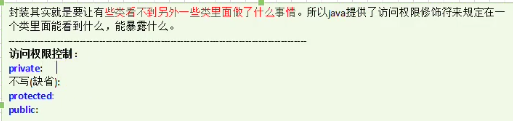
07封装的思想





封装就是使用访问修饰符来修饰成员变量和方法

08权限访问修饰符

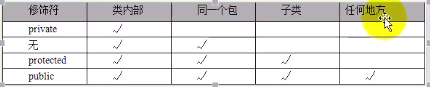


Private 表示私有的，表示类访问权限，只能在本类中访问，离开本类之后就不能直接访问。

不写（缺省） 表示包私有，表示包访问权限，只能在当前包内才能访问。

Protected 表示子类访问权限，同包中的可以访问，即使不同包但是有继承关系也可以访问。

Public表示全局的，可以公共访问权限



一般的字段都使用private修饰，表达隐藏，为了安全性

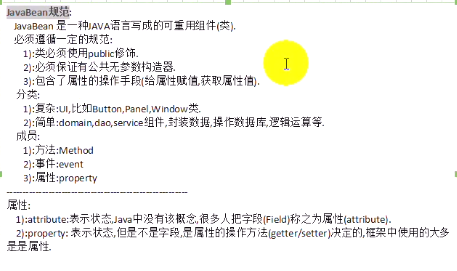
拥有实现细节的方法，一般使用private修饰，不希望外界看到方法的实现细节

一般的方法我们使用public修饰，供外界使用。

一般我们不用缺省，即使使用也仅仅是暴露给同包的其他类

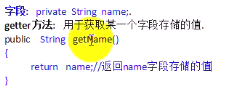
Protected一般在继承关系中父类需要把一个方法暴露给子类

09java bean规范简述

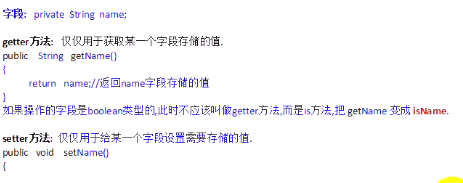


即使手动体统了带参数的构造器，也得提供无参数的构造器

我们讲完封装之后，应该吧对象中的信息隐藏起来（把类中的字段全部使用private西施起来，其他类不能直接访问），为了能让外界访问到本类中的私有变量，我们专门提供getter以及setter方法



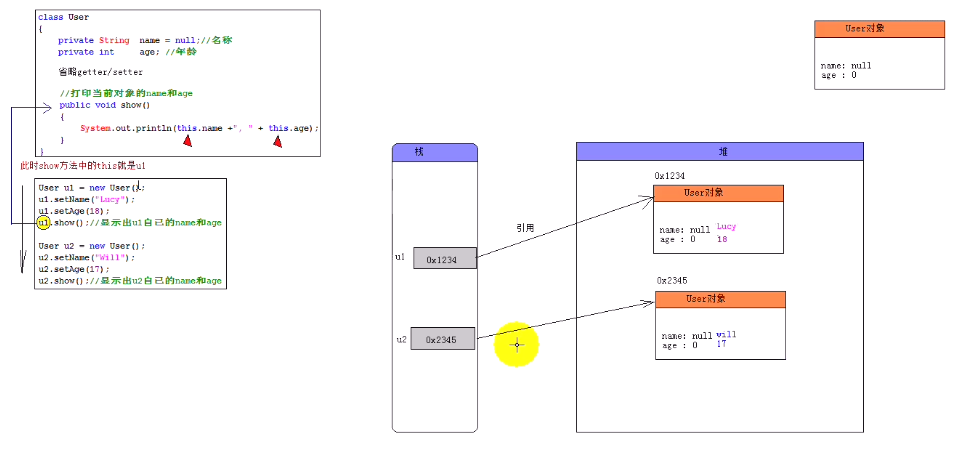
如果操作的字段是布尔类型的，此时不应该叫做getter方法，而是is方法

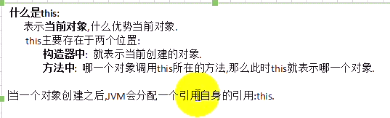


10 引出this关键字

程序有就近原则的思想

11理解this关键字





012this关键字的使用

