# 黑马程序员\_毕向东\_Java基础视频教程

# Java基础学习知识点总结

# 2016年01月06日 day01

## 一、基础知识：软件开发

1、什么是软件？软件：一系列按照特定顺序组织的计算机数据和指令的集合。

2、常见的软件：系统软件：如：DOS，Windows，Linux等。应用软件：如：扫雷，迅雷，QQ等。

3、什么是开发？制作软件。

## 二、基础知识：人机交互方式

4、软件的出现实现了人与计算机之间的更好的交互。

5、交互方式：图形化界面：这种方式简单直观，使用者易于接受，容易上手操作。命令行方式：需要有一个控制台，输入特定的指令，让计算机完成一些特定的操作。较为麻烦，需要记录住一些命令。

## 三、基础知识：常用的DOS命令

6、点击开始在运行栏输入“cmd”进入dos操作界面。

7、课程中常见的命令：

dir：列出当前目录下的文件以及文件夹。

md：创建目录。

rd：删除目录。

cd：进入指定目录。d：进入盘符。

cd..：退回到上一级目录。

cd/：退回到根目录。

del：删除文件。

exit：退出dos命令行。

8、在dos命令行操作文件夹时，如忘记文件夹全称，可以输文件夹前几个名称后面用\*代替。

## 四、基础知识：计算机语言

9、通过DOS命令行的演示，发现原来操作计算机就如同和计算机说话一样。

10、我们告诉它做什么，它就可以做什么。前提是，我们和它说的内容它必须识别才可以。这就是计算机语言。

11、什么是计算机语言？语言：是人与人之间用于沟通的一种方式。例如：中国人与中国人用中文沟通。而中国人要和韩国人交流，就要学习韩语。

12、计算机语言：人与计算机交流的方式。如果人要与计算机交流，那么就要学习计算机语。计算机语言有很多种，如：C，C++，Java等。这里，我们选择其中一种：Java语言。

## 五、基础知识：Java语言介绍

13、Java语言概述。是SUN（Stanford University Network，斯坦福大学网络公司）1995年推出的一门高级编程语言。是一种面向Internet的编程语言。随着Java技术在Web方面的不断成熟，已经成为Web应用程序的首选开发语言。是简单易学，完全面向对象，安全可靠，与平台无关的编程语言。

14、Java语言的三种技术架构。J2EE（Java 2 Platform Enterprise Edition）企业版（现更名：JAVAEE）：是为开发企业环境下的应用程序提供一套解决方案。该技术体系中包含的技术如Servlet Jsp等，主要针对于Web应用程序开发。J2SE（Java 2 Platform Standard Edition）标准版（现更名：JAVASE）：是为开发普通桌面和商务应用程序提供的解决方案。该技术体系是其他两者的基础，可以完成一些桌面应用程序的开发。比如Java版的扫雷。J2ME（Java 2 Platform Micro Edition）小型版（现更名：JAVAME）：是为开发电子消费产品和嵌入式设备提供的解决方案。该技术体系主要应用于小型电子消费类产品，如手机中的应用程序等。

15、Java语言的特点：跨平台性。什么是跨平台性？通过Java语言编写的应用程序在不同的系统平台上都可以运行。原理是什么？只要在需要运行Java应用程序的操作系统上，先安装一个Java虚拟机（JVM Java Virtual Machine）即可。由JVM来负责Java程序在该系统中的运行。

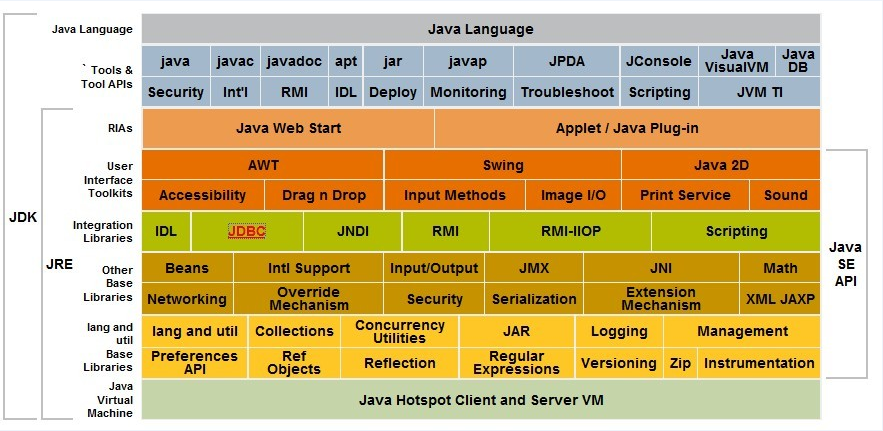
16、Java语言的特点：跨平台性。Java程序：Windows系统：Win版的JVM；Linux系统：lin版的JVM；MAC系统：mac版的JVM。因为有了JVM，所以同一个Java程序在三个不同的操作系统中都可以执行。这样就实现了Java程序的跨平台性。也称为Java具有良好的可移植性。

## 六、基础知识：Java语言的环境搭建

17、明确什么是JRE，JDK；下载JDK；安装JDK；配置环境变量；验证是否成功。

18、什么是JRE，JDK？JRE（Java Runtime Environment Java运行环境）：包括Java虚拟机（JVM Java Virtual Machine）和Java程序所需的核心类库等，如果想要运行一个开发好的Java程序，计算机中只需要安装JRE即可。JDK（Java Development Kit Java开发工具包）：JDK是提供给Java开发人员使用的，其中包含了Java的开发工具，也包括了JRE。所以安装了JDK，就不用再单独安装JRE了。其中的开发工具：编译工具（java.exe）打包工具（jar.exe）等。简单而言：使用JDK开发完成的Java程序，交给JRE去运行。

19、Java6.0 Platform



20、下载JCK（Java Development Kit Java开发工具包）。官方网址：www.oracle.com、java.sun.com。下载完成后，安装在电脑上。然后打开DOS命令行，效验文件是否安装完毕，并利用DOS进行Javac.exe运行测试。为了更方便的使用Java开发工具，需要进行环境变量配置。配置成功后就可以进行Java程序的开发了。

21、环境变量配置。方法一：图形界面操作。我的电脑——属性——高级——环境变量；编辑Path环境变量，在变量值开始处加上Java工具所在目录，后面用分号和其他值分隔开即可；打开DOS命令行，任意目录下敲入javac；如果出现javac的参数信息，配置成功。这种配置方式，一劳永逸。具体流程：右键点击我的电脑打开属性，点击高级系统设置，点击高级，点击环境变量，找到系统变量，新建系统变量（变量名：JAVA\_HOME，变量值：安装JKD的地址。如：D:\ProgramStudio\JDK1.7）点击确定，找到变量名为Path的变量，点击编辑，在变量值栏的最前面（放到前面时，先寻找）加入%JAVA\_HOME%\bin用分号隔开。方法二：DOS界面操作（可用于临时环境变量配置，借助别人的电脑开发Java程序）。具体流程：利用set命令：设置、查看环境变量的值。Set path：查看path的值。配置Java工具包：set path=D:\ProgramStudio\JDK1.7\bin;%path%回车。测试javac。只对当前窗口有效。定义class路径：Set classpath=.;c:myclass。

## 七、基础知识：Java程序开发体验

22、Hello World。将Java代码编写到扩展名为.java的文件中；通过javac命令对该java文件进行编译；通过java命令对生成的class文件进行运行。Java文件——javac.exe编译——class文件——java运行——结果。

23、对于初学者，可以用记事本编写。按部就班，一步一步写代码。打开记事本。Java代码是以类的形式来体现的。代码如下：

class TestFirstOne //Java定义类+类名（首字母大写）

{

public static void main(String[] args) //主函数（保证类的独立运行）

{

System.out.println("hello world"); //输出打印语句

}

}

23、classpath配置。临时配置方式：dos配置。Set Classpath=代码路径。回车。Classpath先在环境变量中找，然后在当前路径（结尾没带分号不找）。Path先在当前路径查找，然后去环境变量中查找。一劳永逸方法和上面JDK环境变量配置方法一样。

24、Hello World组成部门。class是java程序中的关键字（关键字只能是小写）。{}类的区间用大括号定义（类中的内容为了增加阅读性，需要有阶梯感，如使用tab键）。public static void main(String[] args)定义主函数（使用大括号确定函数的内容）。

25、注释。三种方式：//单行注释，符号以后直到回车键的内容；/\*多行注释，符号开始到符号结束以内的内容\*/；/\*\*java注释符号，可被提取出来制作软件开发手册\*/。注释的作用：增加程序的阅读性；调试程序。

26、写代码要养成写注释的习惯。在写代码前注释相关信息。例如：

/\*

需求：定义一个Hello World小程序。

思路：

1，

2，

步骤：

1，通过class关键字定义一个类。将代码都编写到该类中。

2，为了保证该的独立运行。在类中定义个主函数。格式public static void main(String[] args)

3，保存成一个扩展名为java的文件。

4，在dos控制台中通过javac工具对java文件进行编译。

5，在通过java命令对生成的class文件进行执行。

作者：ZC

版本：v1.0

日期：2016.01.06

copyright：ZC

\*/

# 2016年01月07日 day02

## 一、基础知识：Java语言基础组成

1、组成：关键字、标识符、注释、常量和变量、运算符、语句、函数、数组

## 二、基础知识：关键字

2、关键字的定义和特点：定义：被Java语言赋予了特殊含义的单词。特点：关键字中所有字母都为小写。

3、用于定义数据类型的关键字：class、interface、byte、short、int、long、float、double、char、boolean、void。

4、用于定义数据类型值的关键字：true、false、null。

5、用于定义流程控制的关键字：if、else、switch、case、default、while、do、for、break、continue、return。

6、用于定义访问权限修饰的关键字：private、protected、public。

7、用于定义类，函数，变量修饰的关键字：abstract、final、static、synchronized。

8、用于定义类与类之间关系的关键字：extends、implements。

9、用于定义建立实例以及引用实例，判断实例的关键字：new、this、super、instanceof。

10、用于异常处理的关键字：try、catch、finally、throw、throws。

11、用于包的关键字：package、import。

12、其他修饰符关键字：native、strictfp、transient、volatile、assert。

## 三、基础知识：标识符

13、标识符：在程序中自定义的一些名称；由26个英文字母大小写，数字：0-9，符号\_$组成；定义合法标识符规则：数字不可以开头、不可以使用关键字；Java中严格区分大小写；注意：在起名时，为了提高阅读性，要尽量有意义。

14、Java中的名称规范：包名：多单词组成时所有都小写，如：xxxyyyzzz；类名接口名：多单词组成时，所有单词的首字母大写，如：XxxYyyZzz；变量名和函数名：多单词组成时，第一个单词首字母小写，第二个单词开始每个单词首字母大写，如：xxxYyyZzz；常量名：所有字母都大写，多单词时每个单词用下划线连接，如：XXX\_YYY\_ZZZ。

## 四、基础知识：注释

15、注释：对于单行和多行注释，被注释的文字，不会被JVM（Java虚拟机）解释执行；对于文档注释，是Java特有的注释，其中注释内容可以被JDK提供的工具javadoc所解析，生成一套以网页文件形式体现的该程序的说明文档；注释是一个程序员必须要具有的良好编程习惯；初学者编写程序可以养成习惯；先写注释再写代码；将自己的思想通过注释先整理出来，再用代码去体现；因为代码仅仅是思想的一种体现形式而已；用于注解说明解释程序的文字就是注释；提高了代码的阅读性。

16、Java中的注释格式：单行注释：格式：//注释文字；多行注释：格式：/\*注释文字\*/；文档注释：格式：/\*\*注释文字\*/。

## 五、基础知识：常量

17、常量表示不能改变的数值。

18、Java中常量的分类：整数常量：所有整数；小数常量：所有小数；布尔型常量：较为特有，只有两个数值：true和false；字符常量：将一个数字字母或者符号用单引号（’ ’）标识；字符串常量：将一个或者多个字符用双引号（” ”）标识；null常量：只有一个数值就是null。

19、对于整数：Java有三种表现形式：十进制：0-9，满10进1；八进制：0-7，满8进1，用0表示开头；十六进制：0-9，A-F，满16进1，用0x开头表示。

## 六、基础知识：进制转换

20、进制的转换特点：八位二进制表示一个字节（基本的数据单元）。三个二进制位表一个八进制位。四个二进制位表示一个十六进制位。ASCII码编码表。

21、二进制和十进制之间的转换：十进制转二进制：原理：对十进制数进行除2运算（除二取余法）；二进制转十进制：原理：二进制乘以2的N次的过程。

22、快捷方法：8421法则。

23、负数的二进制表现形式：对应的正数二进制取反加1。负数的最高位都是1。

## 七、基础知识：变量

24、变量的概念：内存中的一个存储区域；该区域有自己的名称（变量名）和类型（数据类型）；该区域的数据可以在同一类型范围内不断变化；理解：变量就如同数学中的未知数；变量：就是将不确定的数据进行存储，也就是需要在内存中开辟一个空间。

25、为什么要定义变量：用来不断的存放同一类型的常量，并可以重复使用。

26、使用变量注意：变量的作用范围（一对{}之间有效）；初始化值。

27、定义变量的格式：数据类型 变量名=初始化值；注：格式是固定的、记住格式，以不变应万变。

28、Java语言是强类型语言，对于每一种数据都定义了明确的具体数据类型，在内存总分配了不同打小的内存空间。数据类型：基本数据类型（数值型（整数类型（byte，short，int，long）；浮点类型（float，double））；字符型（char）；布尔型（boolean））；引用数据类型（类（class）；接口（interface）；数据（[]））。注意：整数默认：int，小数默认：double。

29、数据类型空间：byte 8位二进制（一个8位二进制或1个字节）；short 16位二进制（二个8位二进制或2个字节）；int 32位二进制（四个8位二进制或4个字节）；long 64位二进制（八个8位二进制或8个字节）；float（单精度）32位二进制（四个8位二进制或4个字节）；double（双精度）64位二进制（八个8位二进制或8个字节）；char 取值：0-65535；boolean 取值：true，false。

30、类型转换。不同类型的数据不能进行运算。需要进行数据类型提升，从小往大提升，自动数据类型提升。强制类型转换：格式：变量=（类型名）（需转换的数据）；什么时候使用强制转换：如：小数保留整数部分。

31、表达式的数据类型自动提升：所有的byte型、short型和char的值将被提升到int型；如果一个操作数是long型，计算结果就是long型；如果一个操作数是float型，计算结果就是float型；如果一个操作数是double型，计算结果就是double型。分析：system.out.println（‘a’）与system.out.println（‘a’+1）的区别。

## 八、基础知识：运算符

32、算数运算符：+正号，-负号，+加，-减，\*乘，/除，%取模，++自增（前），++自增（后），--自减（前），--自减（后），+字符串相加（字符串数据和任何数据使用+都是相连接，最终都会变成字符串）。

33、转义字符：通过\ 来转变后面字母或者符号的含义。在Linux系统中换行是一个字符来表示\n，windows系统中，换行由\n\r实现。\n:换行。\b:退格。相当于backspace键。\r:按下回车键。\t:制表符。相当于tab键。

34、赋值运算符：符号：=：赋值，+=：a+=5等价于a=a+5，-=，\*=，/=，%=。

35、比较运算符：==：相等于；！=：不等于；<：小于；>：大于；<=：小于等于；>=：大于等于；instanceof：检查是否是类的对象。注意：比较运算符的结果都是boolean型，也就是要么是true，要么是false；比较运算符“==”不能误写成“=”。

36、逻辑运算符：&：AND（与）两边的boolean表达式结果，有一个为false，那么结果就是false，只要两边都为true，结果为true；|：OR（或）两边的boolean表达式结果，只要有一个为true，结果为true，只有两边都有false，结果为false；^：XOR（异或）两边的boolean表达式结果相同结果为false，两边不同结果为true；！：NOT（非）；&&：AND（短路）；||OR：（短路）。逻辑运算符用于连接boolean类型的表达式。&和&&的特点：&：无论左边是真是假，右边都运算；&&：当左边为假时，右边不运算。|和||的特点：|：两边都做运算；||：但左边为真时，右边不运算。

37、位运算符：<<：左移，如：3<<2=12等价于3\*2^2=12；>>：右移，如：6>>2=1等价于6/2^2=1；>>>：无符号右移，如：3>>>1=1；&：与运算，如：6&3=2；|：或运算，如：6|3=7；^：异或运算，如：6^3=5；~：反码，如：~6=-7；位运算是直接对二进制进行运算。<<：其实就是乘以2的移动的位数次幂；>>：就是除以2的移动的位数次幂。>>：最高位补什么由原有数据的最高位值而定；如果最高位0，右移后，用0补空位；如果最高位1，右移后，用1补空位。>>>：无论最高位是什么，右移后，都用0补。一个数异或同一个数两次，结果还是拿个数。

38、三元运算符：格式：（条件表达式）？表达式1：表达式2；如果条件为true，运算后的结果是表达式1；如果条件为false，运算后的结果是表达式2。如：获取两个数中大数，int x=3,y=4,z;z=(x>y)?x:y//z变量存储的就是两个数的大数。

## 九、基础知识：if语句

39、程序流程控制：顺序结构，判断结构，选择结构，循环结构。

40、判断结构：if语句三种格式：一层判断：if(条件表达式){执行语句;}；两层判断：if(条件表达式){执行语句;}else{执行语句;}；多层判断：if(条件表达式){执行语句;}else if(条件表达式){执行语句;}else{执行语句;}。

41、if else结构间歇格式：变量=（条件表达式）？表达式1：表达式2；三元运算符：好处：可以简化if else代码；弊端：因为是一个运算符，所以运算完毕需要有一个结果。

42、解决DOS命令下，编译无法识别GDK码方法：方法一：使用-encoding参数指明编码方式：javac -encoding UTF-8 XX.java。方法二：保存java文件时，选择编码格式为ANSI。

## 十、基础知识：switch语句

43、选择结构：switch语句：格式：switch(表达式){case 取值1：执行语句；break；case 取值2：执行语句；break；……default：执行语句；//break；}

44、switch（表达式）：表达式只接收四种类型：byte，short，int，char。

45、if语句与switch语句的区别：if除了能判断数值，还能判断具体区间，switch判断区间很累；对于数字是布尔型时，只有if能判断。

46、if语句与switch语句的用法：如果要对具体数字进行判断，数值不多，而且符合byte、short、int、char这四种类型，虽然两个语句都可以使用，建议用switch完成。因为效率稍高。其他情况一般用if。当结果为布尔型或区间时，用if完成。If的使用范围更广。

# 2016年01月08日 day03

## 一、基础知识：while语句

1、循环结构：代表语句：while、do while、for。

2、while语句格式：while（条件表达式）{执行语句；}。

3、do while语句格式：do{执行语句；}while（条件表达式）；。Do while特点是条件无论是否满足，循环体至少被执行一次。

4、while和do while的区别：while：先判断条件，只有条件满足才执行循环体。do while：先执行循环体，再判断条件，条件满足，再继续执行循环体。简单一句话：do while：无论条件是否满足，循环体至少执行一次。

## 二、基础知识：for语句

5、for语句格式：for（初始化表达式；循环条件表达式；循环后的操作表达式）{执行语句；}。注意：a：for里面的连个表达式运行的顺序，初始化表达式只读一次，判断循环条件，为真就执行循环体，然后再执行循环后的操作表达式，接着继续判断循环条件，重复找个过程，直到条件不满足为止。b：while与for可以互换，区别在于for为了循环而定义的变量在for循环结束就是在内存中释放。而while循环使用的变量在循环结束后还可以继续使用。c：最简单无限循环格式：while（true），for（；；），无限循环存在的原因是并不知道循环多少次，而是根据某些条件，来控制循环。

6、for和while的区别：a：变量有自己的作用域。对于一个变量来将，如果这个变量仅仅用于控制循环的次数，用作循环增量时用for语句。循环结束时，该变量在内存被释放。用于优化内存。b：for和while可以进行互换，如果定义循环增量，用for更为合适。

7、什么时候使用循环结构？当要多某些语句执行很多次时，就是用循环结构。

8、循环注意：一定要明确哪些语句需要参与循环，哪些不需要。

9、两种思想：累加思想：原理：通过变量记录住每次变化的结果。通过循环的形式，进行累加动作。计数器思想：原理：通过一个变量记录住数据的状态变化。也通过循环完成。

10、循环嵌套：语句嵌套形式，其实就是语句中含有语句。

11、其他流程控制语句：break（跳出），continue（继续）。Break语句：应用范围：选择结构和循环结构。Continue语句：应用于循环结构。注意：a：这两个语句离开应用范围，存在是没有意义的。b：这两个语句单独存在下面都不可以有语句，因为执行不到。c：continue语句是结束本次循环继续下次循环。d：标号的出现，可以让这两个语句作用于指定的范围。

12、注意：标号只能用于循环语句上，给循环起名字。

13、continue：只能作用于循环结构。继续循环。特点：结束本次循环，继续下一次循环。

14、总结：a：break和continue语句作用的范围。b：break和continue单独存在时，下面可以有任何语句。因为都执行不到。

## 三、基础知识：函数

15、函数的定义：什么是函数？函数就是定义在类中的具有特定功能的一段独立小程序；函数也称为方法。

16、函数的格式：修饰符 返回值类型 函数名（参数类型 形式参数1，参数类型 形式参数2，…）{执行语句；return返回值；}。返回值类型：函数运行后的结果的数据类型。参数形式：是形式参数的数据类型。形式参数：是一个变量，用于存储调用函数时传递给函数的实际参数。实际参数：传递给形式参数的具体数值。Return：用于结束函数。返回值：该值会返回给调用者。

17、函数的特点：定义函数可以将功能代码进行封装；便于对该功能进行复用；函数只有被调用才会被执行；函数的出现提高了代码的复用性；对于函数没有具体返回值的情况，返回值类型用关键字void表示，那么该函数中的return语句如果在最后一行可以省略不写。注意：函数中只能调用函数，不可以在函数内部定义函数；定义函数时，函数的结果应该返回给调用者，交由调用者处理。

18、重点：函数的应用。如何定义一个函数呢？a：既然函数是一个独立的功能，那么该功能的运算结果是什么先明确。因为这是在明确函数的返回值类型。b：再明确在定义该功能的过程中是否需要未知的内容参与运算。因为是在明确函数的参数列表（参数的类型和参数的个数）。

19、函数的重载（overload）：重载的概念：在同一个类中，允许存在一个以上的同名函数，只要它们的参数个数或者参数类型不同即可。重载的特点：与返回值无关，只看参数列表。重载的好处：方便与阅读，优化了程序设计。重载示例：//返回两个整数的和int add(int x,int y){return x+y;}；//返回三个整数的和int add(int x,int y,int z){return x+y+z;}；//返回两个小数的和double add(double x,double y){return x+y;}。

20、什么时候用重载？当定义的功能相同，但参与运算的未知内容不同。那么，这时就定义一个函数名称以表示其功能，方便阅读，而通过参数列表的不同来区分多个同名函数。

## 四、基础知识：数组

21、数组的定义：概念：同一种数据类型的集合。其实数组就是一个容器。数组的好处：可以自动给数组中的元素从0开始编号，方便操作这些元素。关键字：new：用来在内存中产生一个容器实体。格式一：元素类型[]数组名=new元素类型[元素个数或数组长度]；示例：int [] arr = new int[5]；格式二：元素类型[]数组名=new元素类型[]{元素，元素，…}；示例：int[] arr = new int[]{3,5,1,7}；int[] arr = {3,5,1,7}。

22、内存结构：Java程序在运行时，需要在内存中分配空间。为了提高运行效率，又对空间进行了不同区域的划分，因为每一片区域都有特定的处理数据方式和内存的管理方式。A：栈内存（栈区）：用于存储局部变量，当数据使用完，所占空间会自动释放。B：堆内存（堆区）：数组和对象，通过new建立的实例都存放在堆内存中。每一个实体都有内存地址。实体中的变量都有默认初始化值。实体不在被使用，会在不确定的时间内被垃圾回收器回收。C：方法区，本地方法区，寄存器。

# 2016年01月09日 day04

## 一、基础知识：数组

1、获取数组中的元素。通常会用到遍历。

2、数组中有一个属性可以直接获得到数组元素个数。length。使用方式：数组名称.length。

3、数组排序：选择排序：内循环结束一次，最值出现头角标位置。冒泡排序：第一圈：最值出现在了最后位。

4、java中排序工具：import.java.util.\*；Array.sort(arr)。java中已经定义好的一种排序方式。开发中，对数组排序，要使用该句代码。

5、二维数组：数组中的数组：格式一：int[][] arr=new int[3][2]；定义了名称为arr的二维数组；二维数组中有3个一维数组；每一个一维数组中有2个元素；一维数组的名称分别为arr[0]，arr[1]，arr[2]；给第一个一维数组1脚标位赋值为78写法是：arr[0][1]=78。格式二：int[][] arr=new int[3][]；二维数组中有3个一维数组；每个一维数组都是默认初始值null；可以对这个三个一维数组分别进行初始化：arr[0]=new int[3]；arr[1]=new int[1]；arr[2]=new int[2]。

# 2016年01月10日 day05

## 一、面向对象：概述

1、理解面向对象：面向对象是相对面向过程而言；面向对象和面向过程都是一种思想；面向过程：强调的是功能行为；面向对象：将功能封装进对象，强调了具备了功能的对象；面向对象是基于面向过程的。让我们的角色从执行者变为指挥者。面向过程是面向对象的基础。

2、一切皆对象，境界：万物皆对象。

3、写程序时：先考虑对象。有对象时，直接用。没对象时，自己造一个对象。

4、人开门：名称提炼法。事物：人、门。示例：人{开门（门）{门。开（）；}}；门{开（）{操作门轴等。}}。

5、面向对象的三个特征：封装，继承，多态。

6、以后的开发：其实就是找对象使用。没有对象，就创建一个对象。

7、找对象，建立对象，使用对象，维护对象的关系。

8、类和对象的关系：类就是：对现实生活中事物的描述。对象：就是这类事物，实实在在的个体。映射到java中，描述就是class定义的类。具体对象就是对于java在堆内存中用new建立实体。

9、类与对象示例：图纸：汽车，汽车，汽车。可以理解为：类就是图纸；汽车就是堆内存中的对象。

10、描述事物其实就在是描述事物的属性和行为。

11、属性对应是类中的变量，行为对应的类中的函数（方法）。其实定义类，就是在描述事物，就是在定义属性和行为。属性和行为共同成为类中的成员（成员变量和成员函数（方法））。

12、引用变量：类类型变量。记住：类类型变量指向对象。

13、就需要该对象做使用，在java指挥方式是：对象.对象成员。

14、成员变量和局部变量：作用范围：成员变量作用于整个类中。局部变量作用于函数中，或者语句中。在内存中的位置：成员变量：在堆内存中，因为对象的存在，才在内存中存在。局部变量：存在栈内存中。

15、匿名对象：匿名对象是对象的简化形式；匿名对象两种使用情况：当对对象方法仅进行一次调用的时；匿名对象可以作为实际参数进行传递。

## 二、面向对象：封装

16、封装的定义：是指隐藏对象的属性和实现细节，仅对外提供公共访问方式。

17、封装的好处：将变化隔离；便于使用；提高重用性；提高安全性。

18、封装原则：将不需要对外提供的内容隐藏起来；把属性都隐藏，提供公共方法对其访问。

19、函数是最小的封装体；类是较小的封装体；包是中级封装体；框架是较大的封装体。

20、关键字private：私有，权限修饰符：用于修饰类中的成员（成员变量，成员函数）。在类中隐藏。私有只在本类中有效。

21、注意：私有仅仅是封装的一种表现形式。

22、之所以对外提供访问方式，就是因为可以在访问方式中加入逻辑判断语句。对访问的数据进行操作。提高代码健壮性。

23、关键字private：将成员变量私有化。对外提供对应的set，get方法对其进行访问。提高对数据访问的安全性。

## 三、面向对象：构造函数

24、构造函数的特点：函数名与类名相同；不用定义返回值类型；不可以写return语句。

25、构造函数的作用：给对象进行初始化。注意：默认构造函数的特点；多个构造函数是以重载的形式存在的。

26、构造函数：对象一建立就会调用与之对应的构造函数。构造函数的作用：可以用于给对象进行初始化。

27、构造函数的小细节：当一个类中没有定义构造函数时，那么系统会默认给该类假如一个空参数的构造函数。当在类中自定义了构造函数后，默认的构造函数就没有了。

28、构造函数和一般方法（函数）的区别：A：构造函数和一般函数在写法上有不同。B：在运行上也有不同。构造函数是在对象一建立就运行。就对象初始化。而一般方法是对象调用才执行，是给对象添加对象具备的功能。C：一个对象建立，构造函数只运行一次。而一般方法可以被该对象调用多次。

29、什么时候定义构造函数呢：当分析事物时，该事物存在具备一些特征或者行为，那么将这些内容定义在构造函数中。

30、构造代码块：作用：给对象进行初始化。对象一建立就运行，而且优先于构造函数执行。

31、构造代码块和构造函数区别：构造代码块是给所有对象进行统一初始化，而构造函数是给对应的对象进行初始化。

32、构造代码块中定义的是不同对象共性的初始化内容。

## 四、面向对象：this关键字

33、关键字this：看上去，是用于区分局部变量和成员变量同名情况。

34、关键字this的特点：就代表本类的对象，到底代表哪一个呢？this代表它所在函数所属对象的引用。简单说：哪个对象在调用this所在的函数，this就代表哪个对象。

35、this的应用：当定义类中功能时，该函数内部要用到该函数的对象时，这时用this来表示这个对象。但凡本类功能内部使用到了本类对象，都用this代替。

36、this语句：用于构造函数之间函数互相调用。this语句：只能定义在构造函数的第一行。因为初始化要先执行。

37、this语句只能用于构造函数间，不能用于一般函数。

# 2016年01月11日 day06

## 一、面向对象：static关键字

1、static（静态）关键字：用于修饰成员（成员变量和成员函数）。被修饰后的成员具备以下特点：随着类的加载而加载；优先于对象存在；被所有对象所共享；可以直接被类名调用。使用注意：静态方法只能访问静态成员；静态方法中不可以写this，super关键字；主函数是静态的。

2、用法：是一个修饰符，用于修饰成员（成员变量，成员函数）。当成员被静态修饰后，就多了一个调用方式，除了可以被对象调用外，还可以直接被类名用。类名.成员变量。

3、实例变量和类变量的区别：A，存放位置：类变量随着类的加载而存在于方法区中。实例变量随着对象的建立存在于堆内存中。B，生命周期：类变量生命周期最长，随着类的消失而消失。实例变量随着对象的消失而消失。

4、静态使用注意事项：A，静态方法只能访问静态成员。非静态方法既可以访问静态也可以访问非静态。B，静态方法中不可以定义thsi，super关键字。因为静态优先于对象存在，所有静态方法中不可以出现this。C，主函数是静态的。

5、静态优缺点：优：对对象的共享数据进行单独空间的存储，节省空间。没有必要每个对象中都存储一份。可以直接被类名调用。缺：生命周期过长。访问出现局限性。（静态虽好，只能访问静态。）

6、public static void main(String[] args) 主函数：是一个特殊的函数，作用程序的入口，可以被jvm调用。

7、主函数的定义：public：代表着该函数访问权限是最大的。static：代表主函数随着类的加载就已经存在了。void：代表主函数没有具体的返回值。main：不是关键字，但是是一个特殊的单词，可以被jvm识别。（String[] args）：函数的参数，参数类型是一个数组，该数组中的元素是字符串。字符串类型的数组。

8、主函数是固定格式的：jvm识别。jvm在调用主函数时，传人的是new String[0]；。

9、什么时候使用静态？要从两方面下手：因为静态修饰的内容有成员变量和函数。什么时候定义静态变量（类变量）呢？当对象中出现共享数据时，该数据静态所修饰。对象中的特有数据要定义成非静态存在于堆内存中。

10、什么时候定义静态函数呢？当功能内部没有访问到非静态数据（对象的特有数据），那么该功能可以定义成静态的。

11、静态的应用：A每一个应用程序中都有共性的功能，可以将这些功能进行抽取，独立封装。以便复用。B虽然可以通过建立ArrayTool的对象使用这些工具方法，对数组进行操作。但是：1，对象是用于封装数据的，可以ArrayTool对象并未封装特特有数据。2，操作数组的每一个方法都没有用到ArrayTool对象中的特有数据。C这时就考虑，让程序更严谨，是不需要对象的。可以将ArrayTool中的方法都定义成static的，直接通过类名调用即可。D将方法都静态后，可以方便于使用，但是该类还是可以被其他程序建立对象的。为了更为严谨，强制让该类不能建立对象。可以通过将构成函数私有化完成。

12、接下来，将ArrayTool.class文件发送给其他人，其他人只要将该文件设置到classpath路径下，就可以使用该工具类。但是，很遗憾，该类中到底定义了多少个方法，对方不清楚。因为该类并没有使用说明书。开始制作程序的说明书。java的说明书通过文档注释来完成。

13、注释符号：/\*\*\*/；@author作者；@version版本；@param参数；@return返回值。

14、静态代码块。格式：static{静态代码块中的执行语句。}。

15、静态代码块的特点：随着类的加载而执行，只执行一次,并优先于主函数。用于给类进行初始化的。

16、重点：对象的初始化过程：Person p=new Person(“zhangsan”,20);该句话都做了什么事情？A因为new用到了Person。Class。所有会先找到Person。Class文件并加载到内存中。B执行该类中的static代码块，如果有的话，给Person.class类进行初始化。C在堆内存中开辟空间，分配内存地址。D在堆内存中建立对象的特有属性，并进行默认初始化。E对属性进行显示初始化。F对对象进行构造代码块初始化。G对对象进行对应的构造函数初始化。H将内存地址付给栈内存中的p变量。

## 二、面向对象：单例设计模式

17、设计模式：解决某一类问题最行之有效的方法。java中23中设计模式：单例设计模式：解决一个类在内存只存在一个对象。

18、想要保证对象唯一。1，为了避免其他程序过多建立该类对象。先禁止其他程序建立该类对象。2，还为了让其他程序可以访问到该类对象，只好在本类中，自定义一个对象。3，为了方便其他程序对自定义对象的访问，可以对外提供一些访问方式。

19、这三步怎么用代码体现呢？1，将构造函数私有化。2，在类中创建一个本类对象。3，提供一个方法可以获取到该对象。

20、对于事物该怎么描述，还怎么描述。当需要将该事物的对象保证在内存中唯一时，就将以上的三部加上即可。

21、单例设计方法一：这个是先初始化对象。称为：饿汉式。Single类一进内存，就已经创建好了对象。开发一般用饿汉式：安全、简单。

22、单例设计方法二：对象被调用时，才初始化，也叫做对象的延时加载。称为：懒汉式。Single类进内存，对象还没有存在，只有调用了getInstance方法时，才建立对象。

23、记住原则：定义单例，建议使用饿汉式。但面试时：考懒汉式的多。特别是如何解决懒汉式的缺点。

# 2016年01月12日 day07

## 一、面向对象：继承

1、继承：1，提高代码的复用性。2，让类与类之间产生了关系。有了这个关系，才有了多态的特性。关键字：extends。

2、注意：千万不要为了获取其他类的功能，简化代码而继承。必须是类与类之间有所属关系才可以继续。所属关系is a。

3、java语言中：java只支持单继承，不支持多继承。原因：因为多继承容易带来安全隐患：当多个父类中定义了相同功能，当功能内容不同时，子类对象不确定要运行哪一个。但是java保留这种机制，并用另一种体现形式来完成表示，多实现。

4、java支持多层继承。也就是一个继承体系。如何使用一个继承体系中的功能呢？想要使用体系，先查阅父类的描述，因为父类中定义的是该体系中共性功能。通过了解共性功能，就可以知道该体系的基本功能。那么这个体系已经可以基本使用了。

5、那么在具体调用时，要创建最子类的对象，为什么呢？一是因为有可能父类不能创建对象，二是创建子类对象可以使用更多的功能，包括基本的也包括特有的。

6、简单一句话：查阅父类功能，创建子类对象使用功能。

7、this代表本类对象的引用，super代编父类对象的引用。

8、子父类出现后，类成员的特点：类中成员：A，变量。B，函数。C，构造函数。

9、子父类中的变量：如果子类中出现非私有的同名变量时，之类要访问本类中的同名变量，用this子类要访问父类中的同名变量，用super。super的使用和this的使用几乎一致。this代表的是本类对象的引用。super代表的是父类对象的引用。

10、子父类中的函数：当子类出现和父类一模一样的函数时，当子类对象调用该函数，会运行子类函数的内容。如同父类的函数被覆盖一样。这种情况是函数的另一个特性：重写（覆盖）。

11、修改以往的源码绝对是灾难。

12、单子类继承父类，沿袭了父类的功能，到子类中，但是子类虽具备该功能，但是功能的内容却和父类不一致，这时，没有必要定义新功能，而是使用覆盖特殊，保留父类的功能定义，并重写功能内容。

13、注意：覆盖：子类覆盖父类，必须保证子类权限大于等于父类权限，才可以覆盖，否则编译失败。静态只能覆盖静态。

14、记住大家：重载：只看同名函数的参数列表。重写：子父类方法要一模一样。

15、子父类中的构造函数：发现在对子类对象进行初始化时，父类的构造函数也会运行。那是因为子类的构造函数默认第一行有一条隐式的语句super（）；super（）；会访问父类中空参数的构造函数。而且子类中所有的构造函数默认第一行都是super（）；。

16、为什么子类一定要访问父类中的构造函数：因为父类中的数据子类可以直接获取，所有之类对象在建立时，需要先查看父类是如何对这些数据进行初始化的。所有之类在对象初始化时，要先访问一下父类中的构造函数。如果要访问父类中指定的构造函数，可以通过手动定义super语句的方式来指定。注意：super语句一定一定在子类构造函数第一行。

17、子类的实例化过程。结论：子类的所有的构造函数默认都会访问父类中空参数的构造函数。因为子类每一个构造函数内的每一行都有一句隐式的super（）；当父类中没有空参数的构造函数时，子类必须手动通过super语句或者this语句形式来指定要访问父类中的构造函数。当然：子类的构造函数第一行也可以收到指定this语句来访问本类中的构造函数。子类中至少会有一个构造函数会访问父类中的构造函数。

18、关键字：final：final可以修饰类，方法（函数），变量；final修饰的类不可以被继承；final修饰的方法（函数）不可以被覆盖；final修饰的变量是一个常量，只能被赋值一次；内部类只能访问被final修饰的局部变量。

## 二、面向对象：抽象类

19、当多个类中出现相同功能，但是功能主体不同，这时也可以进行向上抽取。这时，只抽取功能定义，而不抽取功能主体。

20、抽象：看不懂。

21、抽象类的特点：A，抽象方法一定定义在抽象类中。B，抽象方法和抽象类都必须被abstract关键字修饰。C，抽象类不可以被new创建对象。因为调用抽象方法没意义。D，抽象类中的抽象方法要被使用，必须由子类复写其所有的抽象方法后，建立子类对象调用。

22、如果子类只覆盖了部分抽象方法，那么该子类还是一个抽象类。

23、抽象类和一般类没有太大的不同。该如何描述事物，就如何描述事物，只不过，该事物中出现了一些看不懂的东西。这些不确定的部分，也是该事物的功能，需要明确出现，但是无法定义主体。通过抽象方法来表示。

24、抽象类比一般类多了抽象方法（函数）。就是在类中可以定义抽象方法。

抽象类不可以实例化（即不可以new）。特殊：抽象类中可以不定义抽象方法，这样做仅仅是不让该类建立对象。

25、什么是模板方法呢？在定义功能时，功能的一部分是确定的，但是有一部分是不确定，而确定的部分在使用不确定的部分。那么这时就将不确定的部分暴露出去。由该类的子类去实现。

## 三、面向对象：接口

26、接口：格式：interface{}。接口中的成员修饰符是固定的。成员常量：public static final；成员函数：public abstract。接口的出现将“多继承”通过另一种形式体现出来，即“多实现”。

26、接口：初期理解：可以认为是一个特殊的抽象类。当抽象类中的方法都是抽象的，那么该类可以通过接口的形式来表示。class用于定义类。interface用于定义接口。

27、接口：是不可以创建对象的，因为有抽象方法。需要被子类实现，子类对接口中的抽象方法全部覆盖后，子类才可以实例化。否则子类是一个抽象类。接口可以被类多实现，也是对多继承不支持的转换形式。java支持多实现。用关键字：implements。

28、接口与接口之间支持多继承，类与类之间只支持单继承。

29、接口的特点：接口是对外暴露的规则；接口是程序的功能扩展；接口可以用来多实现；类与接口之间是实现关系，而且类可以继承一个类的同时实现多个接口；接口与接口之间可以有继承关系。

# 2016年01月13日 day08

## 一、面向对象：多态

1、多态的定义：某一类事物的多种存在形态，例：动物中猫，狗。猫这个对象对应的类型是猫类型：猫x=new 猫（）；同时猫也是动物中的一种，也可以把猫称为动物。动物y=new 猫（）；动物是猫和狗具体事物中抽取出来的父类型；父类型引用指向了子类对象。

2、多态的体现：父类的引用指向了自己的子类对象。父类的引用也可以接收自己的子类对象。

3、多态的前提：必须是类与类之间有关系。要么继承，要么实现。通常还有一个前提：存在覆盖。

4、多态的弊端：提高了扩展性，但是只能使用父类的引用访问父类中的成员。

5、多态的好处：多态的出现大大的提高了程序的扩展性。

6、引用数据类型转换：类型提升。向上转型。如果想要调用猫的特有方法时，如何操作？强制将父类的引用，转成子类类型。向下转型。注意：千万不要出现这样的操作，就是将父类对象转成子类类型。我们能转换的是父类引用指向了自己的子类对象时，该引用可以被提升，也可以被强制转换。多态自始至终都是子类对象在做着变量。

7、关键字：instanceof。判断引用数据类型是否一致。

8、在多态中成员函数的特点：在编译时期：参阅引用型变量所属的类中是否有调用的方法。如果有，编译通过，如果没有编译失败。在运行时期：参阅对象所属的类中是否有调用的方法。简单总结就是：成员函数在多态调用时，编译看左边，运行看右边。

9、在多态中，成员变量的特点：无论编译和运行，都参考左边（引用型变量所属的类）。

10、object类：是所有对象的直接或者间接父类，传说中的上帝。该类中定义的肯定是所有对象都具备的功能。

11、Object类中已经提供了对对象是否相同的比较方法。如果自定义类中也有比较相同的功能，没有必要重新定义。只要沿袭父类中的功能，建立自己特有的比较内容即可。这就是覆盖。

# 2016年01月15日 day09

## 一、面向对象：内部类

1、内部类的定义：将一个类定义在另一个类的里面，对里面那个类就称为内部类（内置类，嵌套类）。

2、内部类访问特点：内部类可以直接访问外部类中的成员，包括私有成员。之所以可以直接访问外部类中的成员，是因为内部类中持有了一个外部类的引用，格式：外部类名.this。而外部类要访问内部类中的成员必须要建立内部类的对象。

3、访问格式：当内部类定义在外部类的成员位置上，而且非私有，可以在外部其他类中。

可以直接建立内部类对象。格式：外部类名.内部类名 变量名=外部类对象.内部类对象；Outer.Inner in=new Outer().new Inner();。

4、当内部类在成员位置上，就可以被成员修饰符所修饰。比如，private：将内部类在外部类中进行封装。static：内部类就具备了static的特性。当内部类被static修饰后，只能直接访问外部类中的static成员。出现了访问局限。

5、在外部其他类中，如何直接访问static内部类的非静态成员呢？new Outer.Inner().function();。在外部其他类中，如何直接访问static内部类的静态成员呢？Outer.Inner().function();。

6、注意：当内部类中定义了静态成员，该内部类必须是static的。当外部类中的静态方法访问内部类时，内部类也必须是static的。

7、当描述事物时，事物的内部还有事物，该事物用内部类来描述。因为内部事物在使用外部事物的内容。

8、内部类定义在局部时，1，不可以被成员修饰符修饰；2，可以直接访问外部类中的成员，因为还持有外部类中的引用；但是不可以访问它所在的局部中的变量。只能访问被final修饰的局部变量。

9、匿名内部类：1，匿名内部类其实就是内部类的简写格式。2，定义匿名内部类的前提：内部类必须继承一个类或者实现接口。3，匿名内部类的格式：new 父类或者接口（）{定义子类的内容}。4，其实匿名内部类就是一个匿名子类对象。而且这个对象有点胖。可以理解为带内容的对象。5，匿名内部类中定义的方法最好不要超过3个。

## 二、面向对象：异常

10、异常：异常的体系：Throwable：（Error：通常出现重大问题如：运行的类不存在或者内存溢出等；不编写针对代码对其处理。Exception：在运行时运行出现的一起情况，可以通过try catch finally）。Exception和Error的子类名称都是以父类名作为后缀。

11、异常：就是程序在运行时出现不正常情况。

12、异常由来：问题也是现实生活中一个具体的事物，也可以通过java的类的形式进行描述，并进行封装。其实就是java对不正常情况进行描述后的对象体现。

13、对于问题的划分：两种：一种是严重的问题。一种是非严重的问题。对于Error一般不编写针对性的代码对其进行处理。对于严重的，java通过Error类进行描述。对于非严重的，java通过Exception类进行描述。对于Exception可以使用针对性的处理方式进行处理。

14、无论Error或者Exception都具体有一些共性内容。比如：不正常情况的信息，引发原因等。

15、异常的处理：java提供了特有的语句进行处理。Try{需要被检测的代码；}catch{异常类 变量}{处理异常的代码：（处理方式）}finally{一定会执行的语句；}。

15、throws Exception//在功能上通过throws的关键字声明了该功能有可能会出现问题。在函数上声明异常。便于提高安全性，让调用者进行处理。不处理编译失败。

16、对多异常的处理。1，声明异常时，建议声明更为具体的异常。这样处理得可以更具体。2，对声明几个异常，就对应有几个catch块。不要定义多余的catch块。如果多个catch块中的异常出现继承关系，父类异常catch块放在最下面。

17、建议在进行catch处理时，catch中一定要定义具体处理方式。不要简单定义一句 e.printStackTrace()，也不要简单的就书写一条输出语句。

18、自定义异常：因为项目中会出现特有的问题，而这些问题并未被java所描述并封装对象。所有对于这些特有的问题可以按照java的对问题封装的思想。将特有的问题，进行自定义的异常封装。

19、当在函数内部出现了throw抛出了异常对象，那么就必须要给对应的处理动作。要么在内部try catch处理。要么在函数上声明让调用者处理。

20、如何定义异常信息呢？因为父类中已经把异常信息的操作都完成了。所有子类只要在构造时，将异常信息传递给父类通过super语句。那么就可以直接通过getMessage方法获取自定义的异常信息。

21、自定义异常：必须是自定义类继承Exception。继承Exception原因：异常体系有一个特点：因为异常类和异常对象都被抛出。他们都具备可抛性。这个可抛性是Throwable这个体系中独有特点。

22、只有这个体系中的类和对象才可以被throws和throw操作。

23、throws和throw的区别：throws使用在函数上。throw使用在函数内。throws后面跟的异常类。可以跟多个。用逗号隔开。throw后跟的是异常对象。

24、Exception中有一个特殊的子类异常RuntimeException运行时异常。如果在函数内容抛出该异常，函数上可以不用声明，编译一样通过。如果在函数上声明了该异常。调用者可以不用进行处理，编译一样通过。

25、之所以不用在函数声明，是因为不需要让调用者处理。当该异常发生，希望程序停止。因为在运行时，出现了无法继续运算的情况，希望停止程序。对代码进行修正。

26、自定义异常时：如果该异常的发生，无法在继续进行运算，就让自定义异常继承RuntimeException。

27、对于异常分两种：1，编译时被检测的异常。2，编译时不被检测的异常（运行时异常，RuntimeException以及其子类）。

# 2016年01月16日 day10

## 一、面向对象：异常

1、finally中存放的是一定会被执行的代码。

2、finally代码块：定义一定执行的代码。通常用于关闭资料。

3、记住一点：catch是用于处理异常。如果没有catch就代表没有被处理过，如果该异常时检测时的异常，就必须声明出去。

4、异常在子父类覆盖中的体现：A，子类在覆盖父类时，如果父类的方法抛出异常，那么子类的覆盖方法，只能抛出父类的异常或者该异常的子类。B，如果父类方法抛出多个异常，那么子类在覆盖该方法时，只能抛出父类异常的子集。C，如果父类或者接口的方法中没有异常抛出，那么子类在覆盖方法时，也不可以抛出异常。如果子类方法发生了异常。就必须要进行try处理。绝对不能抛。

5、异常：是什么？是对问题的描述，将问题进行对象的封装。

6、异常体系：Throwable：|--Error；|--Exception（|--RuntimeException）。

7、异常体系的特点：异常体系中的所有类以及建立的对象都具备可抛性。也就是说可以被throw和throws关键字所操作。只有异常体系具备这个特点。

8、throw和throws的用法：throw定义在函数内，用于抛出异常对象。throws定义在函数上，用于抛出异常类，可以抛出多个用逗号隔开。

9、当函数内容有throw抛出异常对象，并未进行try处理。必须要在函数上声明，都在编译失败。注意：RuntimeException除外。也就是说，函数内如果抛出的RuntimeException异常，函数上可以不用声明。

10、如果函数声明了异常，调用者需要进行处理。处理方法可以throws和try。

11、异常有两种：编译时被检测异常：该异常在编译时，如果没有处理（没有抛也没有try），编译失败。该异常被标识，代表着可以被处理。运行时异常（编译时不检测）：在编译时，不需要处理，编译器不检查。该异常的发生，建立不处理，让程序停止。需要对代码进行修正。

12、异常处理语句：try（需要被检测的代码；） catch（处理异常的代码；） finally（一定会执行的代码；）。有三种结合格式：try catch；try finally；try catch finally。注意：A，finally中定义的通常是关闭资源代码。因为资源必须释放。B，finally只有一种情况不会执行。当执行到System.exit(0);finally不会执行。

13、自定义异常：定义类继续Exception或者RuntimeException。A，为了让该自定义类具备可抛性。B，让该类具备操作异常的共性方法。当要定义自定义异常的信息时，可以使用父类已经定义好的功能。异常异常信息传递给父类的构成函数。class MyException extends Exception{MyException(String message){super(message);}}。

14、自定义异常：按照java的面向对象思想，将程序中出现的特有问题进行封装。

15、异常的好处：A，将问题进行封装。B，将正常流程代码和问题处理代码相分离，方便于阅读。

16、异常的处理原则：A，处理方式有两种：try或者throws。B，调用到抛出异常的功能时，抛出几个，就处理几个。一个try对应多个catch。C，多个catch，父类的catch放到最下面。D，catch内需要定义针对性的处理方式。不要简单的定义printStackTrace,输出语句。也不要不写。当捕获到的异常，本功能处理不了时，可以继续在catch中抛出。

17、异常的注意事项：在子父类覆盖时：A，子类抛出的异常必须是父类异常的子类或者子集。B，如果父类或者接口没有异常抛出时，子类覆盖出现异常，只能try不能抛。

## 二、面向对象：包

18、包（package）：对类文件进行分类管理。给类提供多层命名空间。写在程序文件的第一行。类名的全称的格式是：包名.类名。包也是一种封装形式。

19、关键字protected：/\*保护\*/权限。权限修饰符。可用于修饰包里的父类。

20、总结：包与包之间进行访问，被访问的包中的类以及类中的成员，需要public修饰。不同包中的子类还可以访问直接父类中被protected权限修饰的成员。

21、包与包之间可以使用的权限只有两种，public protected。

22、 public protected default private

同一个类中 可以 可以 可以 可以

同一个包中 可以 可以 可以 不可以

子类 可以 可以 不可以 不可以

不同包中 可以 不可以 不可以 不可以。

23、为了简化类名的书写，使用一个关键字import。import导入的是包中的类。建议，不要写通配符\*，需要用到包中的哪个类，就导入哪个类。

24、建议定义包名不要重复，可以使用url来完成定义，url是唯一的。

25、jar包：java的压缩包：方便项目的携带。方便于使用，只要在classpath设置jar路径即可。数据库驱动，SSH框架等都以jar包体现的。

# 2016年01月17日 day11

## 一、多线程：概述

1、进程：是一个正在执行中的程序。每一个进程执行都有一个执行顺序。该顺序是一个执行路径，或者叫一个控制单元。

2、线程：就是进程中的一个独立的控制单元。线程在控制着进程的执行。一个进程中至少有一个线程。

3、Java VM 启动的时候会有一个进程java.exe。该进程中至少有一个线程负责java程序的执行。而且这个线程运行的代码存在于main方法中。该线程称之为主线程。扩展：其实更细节说明jvm，jvm启动不止一个线程，还有负责垃圾回收机制的线程。

4、如何在自定义的代码中，自定义一个线程呢？通过对API的查找，java已经提供了对线程这类事物的描述。就是Thread类。

5、创建线程的第一种方式：继承Thread类。步骤：1，定义类继承Thread。2，复写Thread类中的run方法。目的：将自定义代码存储在run方法中，让线程运行。3，调用线程的start方法，该方法有两个作用：启动线程；调用run方法。

6、发现运行结果每一次都不同：因为多个线程都获取cpu的执行权。cpu执行到谁，谁就运行。明确一点，在某一时刻，只能有一个程序在运行。（多核除外）cpu在做着快速的切换，以达到看上去是同时运行的效果。我们可以形象把多线程的运行形容为在互相抢夺cpu的执行权。这就是多线程的一个特性：随机性。谁抢到谁执行，至于执行多长，cpu说了算。

7、为什么要覆盖run方法呢？Thread类用于描述线程。该类就定义了一个功能，用于存储线程要运行的代码，该存储功能就是run方法。也就是说Thread类中的run方法，用于存储线程要运行的代码。

8、原来线程都有自己默认的名称。Thread-编号。该编号从0开始。static Thread currentThread()：获取当前线程对象。getName()：获取线程名称。设置线程名称：setName或者构造函数。

9、创建线程的第二种方式：实现Runnable接口。

步骤：1，定义类实现Runnable接口。2，覆盖Runnable接口中的run方法。目的：将线程要运行的代码存放在run方法中。3，通过Thread类建立线程对象。4，将Runnable接口的子类对象作为实际参数传递给Thread类的构造函数。为什么要将Runnable接口的子类对象传递给Thread的构造函数？因为，自定义的run方法所属的对象时Runnable接口的子类对象。所有要让线程去指定指定对象的run方法。就必须明确该run方法所属对象。5，调用Thread类的start方法开启线程并调用Runnable接口子类的run方法。

10、实现方式和基础方法有什么区别呢？实现方式好处：避免了单继承的局限性。在定义线程时，建议使用实现方式。两种方式区别：继承Thread：线程代码存放在Thread子类run方法中。实现Runnable：线程代码存放在接口的子类的run方法中。

## 二、多线程：同步

11、多线程的运行出现了安全问题：问题的原因：当多条语句在操作同一个线程共享数据时，一个线程对多条语句只执行了一部分，汗没有执行完，另一个线程参与进来执行。导致共享数据的错误。解决办法：对多条操作共享数据的语句，只能让一个线程都执行完。在执行过程过，其他线程不可以参与执行。java对于多线程的安全问题提供了专业的解决方式。就是同步代码块。synchronized(对象){需要被同步的代码；}。

12、对象如同锁。持有锁的线程可以在同步中执行。没有锁的线程即使获取了cpu的执行权，也进不去，因为没有获取锁。

13、同步的前提：1，必须要有两个或者两个以上的线程。2，必须是多个线程使用同一个锁。

必须保证同步中只能有一个线程在运行。

14、好处：解决了多线程的安全问题。弊端：多个线程需要判断锁，较为消耗资源。

15、多线程安全问题：如何找到问题：1，明确哪些代码是多线程运行代码。2，明确共享数据。3，明确多线程运行代码中哪些语句是操作共享数据的。

16、同步函数：将关键字synchronized放到函数名称前。即把synchronized当作修饰符使用。

17、同步函数用的是哪一个锁呢？函数需要被对象调用。那么函数都有一个所属对象引用。就是this。所有同步函数使用的锁是this。

18、如果同步函数被静态修饰后，使用的锁是什么呢？通过验证，发现不再是this。因为静态方法中也不可以定义this。静态进内存时，内存中没有本类对象，但是一定有该类对应的字节码文件对象。类名.class 该对象的类型是Class。静态的同步方法，使用的锁是该方法所在类的字节码文件对象。类名.class。

19、死锁：同步中嵌套同步。而锁却不同。

# 2016年01月18日 day12

## 一、多线程：线程间通信

1、wait()，notify()，notifyALL()，用来操作线程为什么定义在了Object类中？这些方法存在于同步中。使用这些方法时必须要标识所属的同步的锁。锁可以是任意对象，所以任意对象调用的方法一定定义Object类中。

2、wait()和sleep()有什么区别？wait()：释放资源，释放锁。sleep()：释放资源，不释放锁。

3、wait：notify：notifyALL：都被使用在同步中，因为要对持有监视器（锁）的线程操作。所有要使用在同步中，因为只有同步才具有锁。

4、为什么这些操作线程的方法要定义在Object类中呢？因为这些方法在操作同步中线程时，都必须要标识他们所操作线程只有的锁。只有同一个锁上的被等待线程，可以被同一个锁上notify唤醒。不可以对不同锁中的线程进行唤醒。也就是说，等待和唤醒必须是同一个锁。而锁可以是任意对象，所有可以被任意对象调用的方法定义在Object类中。

5、JDK1.5 中提供个多线程升级解决方案。将同步synchronized替换成现实Lock操作。将Object中的wait，notify，notifyAll，替换了condition对象。该对象可以Lock锁，进行获取。

## 二、多线程：其他操作

6、停止线程：A：定义循环结束标记：因为线程运行代码一般都是循环，只要控制了循环即可。B：使用interrupt（中断）方法：该方法时结束线程的冻结状态，使线程回到运行状态中来。注意：stop方法已经过时不再使用。

7、如何停止线程？只有一种，run方法结束。开启多线程运行，运行代码通常是循环结构。只要控制住循环，就可以让run方法结束，也就是线程结束。特殊情况：当线程处于了冻结状态。就不会读取到标记。那么线程就不会结束。

8、当没有指定的方式让冻结的线程恢复到运行状态时，这时需要对冻结进行清除。强制让线程恢复到运行状态中来。这样就可以操作标记让线程结束。Thread类中提供该方法：interrupt方法。

9、join方法：当A线程执行到了B线程的.join()方法时，A就会等待。等B线程都执行完，A才会执行。join可以用来临时加入线程执行。

# 2016年01月19日 day13

## 一、String类：概述

1、String类：字符串是一个特殊的对象。字符串一旦初始化就不可以被改变。String str=”abc”；String str1=new String(“abc”)；。Str和str1的区别：str在内存中有一个对象。Str1在内存中有两个对象。

## 二、\*\*\*重点：String类：常见操作

常见的操作有哪些？

"abcd"

1，获取。

1.1 字符串中包含的字符数，也就是字符串的长度。

int length()：获取长度。

1.2 根据位置获取位置上某个字符。

char charAt(int index)：获取位置上某个字符。

1.3 根据字符获取该字符在字符串中位置。

int indexOf(int ch)：获取该字符在字符串中位置。返回的是ch在字符串中第一次出现的位置。

int indexOf(int ch,int fromIndex)：从fromIndex指定位置开始，获取ch在字符串中出现的位置。

int indexOf(String str)：返回的是str在字符串中第一次出现的位置。

int indexOf(String str,int fromIndex)：从fromIndex指定位置开始，获取str在字符串中出现的位置。

int lastIndexOf(int ch)：反向索引。

2，判断。

2.1 字符串中是否包含某一个子串。

boolean contains(str);

特殊之处：indexOf(str)：可以索引str第一次出现位置，如果返回-1,表示该str不在字符串中存在。

所以，也可以用于对指定判断是否包含。

if(str.indexOf("aa")!=-1)

而且该方法既可以判断又可以获取出现的位置。

2.2 字符串是否有内容。

boolean isEmpty(str)：原理就是判断长度是否为0。

2.3 字符串是否是以指定内容开头。

boolean startsWith(str);

2.4 字符串是否是以指定内容结尾。

boolean endsWith(str);

2.5 判断字符串的内容是否相同。复写了Object类中的equals方法。

boolean equals(str);

2.6 判断内容是否相同，并忽略大小写。

boolean equalsIgnoreCass();

3，转换。

3.1 将字符数组转成字符串。

构造函数：String(char[])

String(char[],offset,count)：将字符数组中的一部分转成字符串。

静态方法：static String copyValueOf(char[]);

static String copyValueOf(char[] data,int offset,int count);

static String valueOf(char[]);

3.2 将字符串转成字符数组。//重点\*\*。

char[] toCharArray();

3.3 将字节数组转成字符串。

String(byte[])

String(byte[],offset,count)：将字节数组中的一部分转成字符串。

3.4 将字符串转成字节数组。

byte[] getBytes();

3.5 将基本数据类型转成字符串。

static String valueOf(int)

static String valueOf(double)

//3+"";//String.valueOf(3);

特殊：字符串和字节数组在转换过程中，是可以指定编码表的。

4，替换。

String replace(oldchar,newchar);

5，切割。

String[] split(regex);

6，子串。获取字符串中的一部分。

String substring(begin);

String substring(begin,end);

7，转换，去除空格，比较。

7.1 将字符串转成大写或者小写。

String toUpperCase();

String toLowerCase();

7.2 将字符串两端的多个空格去除。

String trim();

7.3 对两个字符串进行自然顺序的比较。

int compareTo(string);

## 三、String类：StringBuffer

8、StringBuffer类：字符串的注组成原理就是通过该类实现的。StringBuffer可以对字符串内容进行增删。StringBuffer是一个容器。很多方法与String相同。StringBuffer是可变长度的。

9、StringBuffer是字符串缓冲区。是一个容器。

10、缓冲区的特点：1，而且长度是可变化的。2，可以直接操作多个数据类型。3，最终会通过toString方法变成字符串。

11、StringBuffer常见操作：

C create U update R read D delete

1，存储。

StringBuffer append()：将指定数据作为参数添加到已有数据的结尾处。

StringBuffer insert(index,数据)：可以将数据插入到指定index位置。

2，删除。

StringBuffer delete(start,end)：删除缓冲区中的数据，包含start，不包含end。

StringBuffer deleteCharAt(index)：删除指定index位置的字符。

3，获取。

char charAt(int index);

int indexOf(String str);

int lastIndexOf(String str);

int length();

String substring(int start,int end);

4，修改。

StringBuffer replace(start,end,string);

void setCharAt(int index,char ch);

5，反转。

StringBuffer reverse();

6，将缓冲区中指定数据存储到指定字符数组中。

void getChars(int srcBegin, int srcEnd, char[] dst, int dstBegin)。

12、JDK1.5版本之后出现了StringBuilder。

StringBuffer是线程同步。单线程使用StringBuffer效率较低。多线程使用StringBuffer保证了安全性。

StringBuilder是线程不同步。单线程使用StringBuilder提高了效率。多线程使用StringBuilder可能存在安全问题，但可以通过自定义锁来解决安全问题。

以后开发，建议使用StringBuilder。

13、升级三个因素：1，提高效率。2，简化书写。3，提高安全性。

## 四、基本数据类型对象包装类

14、基本数据类型对象包装类的最常见作用：就是用于基本数据类型和字符串类型之间做转换。

15、数据类型转换：

A：基本数据类型转成字符串。

基本数据类型+""

基本数据类型.toString(基本数据类型值);

如：Integer.toString(34);//将34整数变成"34"。

B：字符串转成基本数据类型。

静态：基本数据类型包装类.parseInt(字符串);

如：xxx a=Xxx.parseXxx(String);

int a=Integer.parseInt("123");

double d=Double.parseDouble("12.23");

boolean b=Boolean.parseBoolean("true");

非静态：

Integer i=new Integer("123");

int num=i.intValue();

C：十进制转成其他进制。

toBinaryString();

toHexString();

toOctalString();

D：其他进制转成十进制。

parseInt(string,radix);。

# 2016年01月22日 day14

## 一、集合框架：概述

1、集合类：为什么会出现集合类？面向对象语言对事物的体现都是以对象的形式，所有为了方便对多个对象的操作，就对对象进行存储，集合就是存储对象最常用的一种方式。

2、数组和集合类同是容器，有何不同？数组虽然也可以存储对象，但长度是固定的；集合长度是可变的，数组中可以存储基本数据类型，集合只能存储对象。

3、集合类的特点：集合只用于存储对象，集合长度是可变的，集合可以存储不同类型的对象。

4、为什么会出现这么多的容器呢？因为每一个容器对数据的存储方式有所不同。这个存储方式称之为：数据结构。

5、add方法的参数类型是Object。以便于接收任意类型对象。集合中存储的都是对象的引用（地址）。

6、什么是迭代器呢？其实就是集合的取出元素的方式。

## 二、集合框架：List类

7、List:

特有方法。凡是可以操作角标的方法都是该体系特有的方法。

增

add(index,element);

addAll(index,Collection);

删

remove(index);

改

set(index,element);

查

get(index);

subList(from,to);

listIterator();

8、List集合特有的迭代器。ListIterator是Iterator的子接口。在迭代时，不可以通过集合对象的方法操作集合中的元素。因为会发生ConcurrentModificationException异常。所有，在迭代器时，只能用迭代器的方法操作元素，可以Iterator方法时优先的，只能对元素进行判断，取出，删除的操作，如果想要其他的操作如添加，修改等，就需要使用其子接口，ListIterator。该接口只能通过List集合的ListIterator方法获取。

9、Collection

|--List:元素是有序的，元素可以重复。因为该集合体系有索引。

|--ArrayList:底层的数据结构使用的是数组结构。特点：查询速度很快。但是增删稍慢。线程不同步。

|--LinkedList:底层使用的是链表数据结构。特点：增删速度很快，查询速度稍慢。

|--Vector:底层是数据数据结构。线程同步。被ArrayList替代了。

|--Set:元素是无序的，元素不可以重复。

10、枚举就是Vector特有的取出方式。发现枚举和迭代器很像。其实枚举和迭代是一样的。因为枚举的名称以及方法的名称都过长。所有被迭代器取代了。枚举郁郁而终了。

11、LinkedList：特有方法：

addFirst();

addLast();

getFirst();

getLast();

获取元素，但是不删除元素。如果集合中没有元素，会出现NoSuchElementException。

removeFirst();

removeLast();

获取元素，但是元素被删除。如果集合中没有元素，会出现NoSuchElementException。

在JDK1.6出现了替代方法。

offerFirst();

offerLast();

peekFirst();

peekLast();

获取元素，但是不删除元素。如果集合中没有元素，会返回null。

pollFirst();

pollLast();

获取元素，但是元素被删除。如果集合中没有元素，会返回null。

## 三、集合框架：Set类

12、HaseSet是如何保证元素唯一性的呢？是通过元素的两个方法，hashCode和equals来完成。如果元素的hashCode值相同，才会判断equals是否为true。如果元素的hashCode值不同，不会调用equals。

13、注意：对于判断元素是否存在，以及删除操作，依赖的方法时元素的hashcode和equals方法。

# 2016年1月23日 day15

## 一、集合框架：Set类

1、Set：无序，不可以重复元素。

|--HashSet：数据结构是哈希表。线程是非同步的。

保证元素唯一性的原理：判断元素的hashCode值是否相同。

如果相同，还会继续判断元素的equals方法，是否为true。

|--TreeSet：可以对Set集合中的元素进行排序。

底层数据结构是二叉树。

保证元素唯一性的依据：

compareTo方法和return 0。

2、TreeSet排序的第一种方式：让元素自身具备比较性。元素需要实现comparable接口，覆盖comparaeTo方法。这种方式也称为元素的自然顺序，或者默认顺序。

3、TreeSet排序的第二种方式：当元素自身不具备比较性时，或者具备的比较性不是所需要的。这时就需要让集合自身具备比较性。在集合初始化时，就有了比较方式。

4、当元素自身不具备比较性，或者具备的比较性不是所需要的。这时需要让容器自身具备比较性。定义了比较器，将比较器对象作为参数传递给TreeSet集合的构造函数。

5、当两种排序都存在时，以比较器为主。定义一个类，实现Comparator接口，覆盖compare方法。

## 二、集合框架：泛型

6、泛型：JDK1.5版本以后出现的新特性。用于解决安全问题，是一个类型安全机制。

7、泛型的好处：好处：1，将运行时期出现问题ClassCastException，转移到了编译时期。方便于程序员解决问题。让运行时期问题减少。更加安全。2，避免了强制转换麻烦。

8、泛型格式：通过<>来定义要操作的引用数据类型。在使用java提供的对象时，什么时候写泛型呢？通常在集合框架中很常见。只要见到<>就要定义泛型。其实<>就是用来接收类型的。当使用集合时，将集合中要存储的数据类型作为参数传递到<>即可。

9、什么时候定义泛型类？当类中要操作的引用数据类型不确定的时候，早期定义Object来完成扩展。现在定义泛型来完成扩展。

10、泛型类定义的泛型，在整个类中有效。如果被方法使用，那么泛型类的对象明确要操作的具体类型后，所有要操作的类型就已经固定了。为了让不同方法可以操作不同类型，而且类型还不确定，那么可以将泛型定义在方法上。

11、特殊之处：静态方法不可以访问类上定义的泛型。如果静态方法操作的引用数据类型不确定，可以将泛型定义在方法上。

12、? 通配符。也理解为占位符。

泛型的限定：

？ extends E：可以接收E类型或者E的子类型。上限。

？ super E：可以接收E类型或者E的父类型。下限。

# 2016年01月24日 day16

## 一、集合：Map概述

1、Map集合：该集合存储键值对。一对一对往里存，而且要保证键的唯一性。

2、Map框架：

1，添加。

put(K key,V value)

putAll(Map<? extends K,? extends V> m)

2，删除。

clear()

remove(Object key)

3，判断。

containsValue(Object value)

containsKey(Object key)

isEmpty()

4，获取。

get(Object key)

size()

values()

entrySet()

keySet()

3、Map

|--HashTable：底层是哈希表数据结构，不可以存入null键null值，该集合是线程同步的。JDK1.0。效率低。

|--HashMap：底层是哈希表数据结构，允许使用null键和null值，该集合是不同步的。JDK1.2。效率高。

|--TreeMap：底层是二叉树数据结构，线程不同步，可以用于给map集合中的键进行排序。

4、和Set很像。其实大家，Set底层就是使用了Map集合。

5、map集合的两种取出方式：1，keySet：将map中所有的键存入到Set集合。因为Set具备迭代器。所以可以迭代方式取出所有的键，在根据get方法，获取每一个键对应的值。Map集合的取出原理：将map集合转成set集合。在通过迭代器取出。2，Set<Map.Entry<k,v>> entrySet：将map集合中的映射关系存入到了set集合中，而这个关系的数据类型就是：Map.Entry。

6、Map.Entey 其实Entry也是一个接口，它是Map接口中的一个内部接口。

7、什么时候使用map集合呢？当数据之间存在映射关系时，就要先想到map集合。

8、map扩展知识：map集合被使用是因为具备映射关系。

# 2016年01月26日 day17

## 一、集合：Collections

1、fill方法可以将list集合中所有元素替换成指定元素。

## 二、集合：Arrays

2、Arrays：用于操作数组的工具类。里面都是静态方法。

3、asList：将数组变成list集合。

4、把数组变成list集合有什么好处？可以使用集合的思想和方法来操作数组中的元素。注意：将数组变成集合，不可以使用集合的增删方法。因为数据的长度是固定的。如果你增删，那么会发生UnsupportedOperationException。

5、数组变集合：如果数组中的元素都是对象。那么变成集合时，数组中的元素就直接转成集合中的元素。如果数组中的元素都是基本数据类型，那么会将该数组作为集合中的元素存在。

6、集合变数组：指定类型的数组到底要定义多长呢？当指定类型的数组长度小于了集合的size，那么该方法内部会创建一个新的数组，长度为集合的size。当指定类型的数组长度大于了集合的size，就不会新创建数组，而是使用传递进来的数组。所以创建一个刚刚好的数组最优。

7、为什么要讲集合变数组？为了限定对元素的操作。不需要进行增删了。

8、高级for循环：格式：for（数据类型 变量名：被遍历的集合（Collection）或者数组）｛｝。

9、高级for循环、迭代器、ListIterator的区别：

高级for循环对集合进行遍历。只能获取集合元素，但是不能对集合进行操作。

迭代器除了遍历，还可以进行remove集合中元素的动作。

如果使用ListIterator，还可以在遍历过程中对集合进行增删改查的操作。

10、传统for和高级for有什么区别呢？高级for有一个局限性：必须有被遍历的目标。建议在遍历数组的时候，还是希望使用传统for。因为传统for可以定义角标。

11、JDK1.5版本出现的新特性：可变参数：其实就是上一种数组参数的简写形式。不用每一次都手动的建立数组对象。只要将要操作的元素作为参数传递即可。隐式将这些参数封装成了数组。

12、方法的可变参数。在使用时注意：可变参数一定要定义在参数列表最后面。

13、import static：静态导入：例如：import static java.lang.System.\*;//导入了System类中所有静态成员。注意：当类名重名时，需要制定具体的包名。当方法重名时，指定具备所属的对象或者类。

# 2016年01月28日 day18

## 一、其他对象：System

1、System:类中的方法和属性都是静态的。out:标准输出，默认是控制台。int:标准输入，默认是键盘。描述系统一些信息。

2、获取系统属性信息：Properties getProperties();。因为Properties是Hashtable的子类，也就是Map集合的一个子类对象。那么可以通过map的方法取出该集合中的元素。该集合中存储都是字符串。没有泛型定义。

3、Runtime对象：该类并没有提供构造函数。说明不可以new对象。那么会直接想到该类中的方法都是静态的。发现该类中还有非静态方法。说明该类肯定会提供方法获取本类对象。而且该方法是静态的，并返回值类型是本类类型。由这个特点可以看出该类使用了单例设计模式完成。该方式的static Runtime getRuntime();。

4、Math类：ceil方法：返回大于指定数据的最小整数。floor方法：返回小于指定数据的最大整数。round方法：四舍五入。pow方法：幂运算。random：返回0-1之间的随机数。

## 二、IO流：概述

5、IO(Input Outpu)流：IO流用来处理设备之间的数据传输。Java对数据的操作是通过流的方式。Java用于操作流的对象都在IO包中。流按操作数据分为两种：字节流与字符流。流按流向分为：输入流，输出流。

6、IO流常用基类：字节流的抽象基类：InputStream，OutputStream。字节流的抽象基类：Reader，Writer。注：由这四个类派生出来的子类名称都是以其父类名作为子类名的后缀（如：InputStream的子类FilelnputStream，如：Reader的子类FileReader。）。

7、字符流和字节流：字节流两个基类：InputStream OutputStream。字符流两个基类：Reader Writer。先学习字符流的特点：既然IO流是用于操作数据的，那么数据的最常见体现形式是：文件。

# 2016年01月30日 day19

## 一、IO流：Buffered缓冲区

1、字符流的缓冲区：缓冲区的出现提高了对数据的读写效率。对应类：BufferedWriter；BufferedReader。缓冲区要结合流才可以使用。在流的基础上对流的功能进行了增强。

2、缓冲区的出现是为了提高流的操作效率而出现的。所以在创建缓冲区之前，必须要先有流对象。该缓冲区中提供了一个跨平台的换行符。newLine();。

3、字符读取流缓冲区：该缓冲区提供了一次读一行的方法readLine()，方便于对文本数据的获取。当返回null时，表示读到文件末尾。

4、readLine方法返回的时候只返回回车符之前的数据内容。并不返回回车符。

5、装饰设计模式：当想要对已有的对象进行功能增强时，可以定义类，将已有对象传入，基于已有的功能，并提供加强功能。那么自定义的该类称为装饰类。

6、装饰类通常会通过构造方法接收被装饰的对象。并基于被装饰的对象的功能，提供更强的功能。

7、装饰模式比继承要灵活，避免了继承体系臃肿。而且降低了类与类之间的关系。装饰类因为是增强已有对象，具备的功能和已有的是相同的，只不过提供了更强功能。所以装饰类和被装饰类通常是都属于一个体系中的。

## 二、IO流：字节流

8、字符流：

FileReader

FileWriter

BufferedReader

BufferedWriter

字节流：

InputStream

OutputStream

BufferedInputStream

BufferedOutputStream

9、流操作的基本规律：

最痛苦的就是流对象有很多，不知道该用哪一个。

通过两个明确来完成。

1，明确源和目的。

源：输入流。InputStream Reader。

目的：输出流。OutputStream Writer。

2，明确操作的数据是否是纯文本。

是：字符流。

不是：字节流。

3，当体系明确后，再明确要使用哪个具体的对象。

通过设备来进行区分：

源设备：内存，硬盘，键盘。

目的设备：内存，硬盘，控制台。

# 2016年01月31日 day20

## 一、IO流：File类

1、File类：用来将文件或者文件夹封装成对象。方便对文件与文件夹进行操作。File对象可以作为参数传递给流的构造函数。了解File类中的常用方法。

2、File类常见方法：

1，创建。

boolean createNewFile()；在指定位置创建文件，如果该文件已经存在，则不创建，返回false。

和输出流不一样，输出流对象一建立创建对象。而且文件已经存在，会覆盖。

boolean mkdir();创建一级文件夹。

boolean mkdirs();创建多级文件夹。

2，删除。

boolean delete();删除失败返回false。

void deleteOnExit();在程序退出时删除指定文件。

3，判断。

boolean exists();文件是否存在。

boolean isFile();判断是否是文件。

boolean isDirectory();判断是否是文件夹。

boolean isHidden();判断是否是隐藏文件。

boolean isAbsolute();判断是否是绝对路径。

4，获取信息。

String getName();获取文件名称。

String getPath();获取文件相对路径。

String getParent();该方法返回的是绝对路径中的父目录。如果获取的是相对路径，返回null。

如果相对路径中有上一层目录那么该目录就是返回结果。

getAbsolute();获取文件绝对路径。

lastModified();返回文件最后一次被修改的时间。

length();。

3、因为目录中还有目录，只要使用同一个列出目录功能的函数完成即可。在列出过程中出现的还是目录的话，还可以再次调用本功能。也就是函数自身调用自身。这种表现形式，或者编程手法，称为递归。

递归要注意：1，限定条件。2，要注意递归的次数。尽量避免内存溢出。

4、练习：将一个指定目录下的java文件的绝对路径，存储到一个文本文件中。建立一个java文件列表文件。思路：1，对指定的目录进行递归。2，获取递归过程所有的java文件的路径。3，将这些路径存储到集合中。4，将集合中的数据写入到一个文件中。

## 二、IO流：Properties类

5、Properties是hashtable的子类。也就是说它具备map集合的特点。而且它里面存储的键值对都是字符串。是集合中和IO技术相结合的集合容器，该对象的特点：可以用于键值对形式的配置文件。

6、那么在加载数据时，需要数据有固定格式：键=值。

## 三、IO流：其他类

7、打印流：PrintWriter与PrintStream：可以直接操作输入流和文件。

打印流：

该流提供了打印方法，可以将各种数据类型的数据都原样打印。

字节打印流：PrintStream

构造函数可以接收的参数类型：

1，file对象。File。

2，字符串路径。String。

3，字节输出流。OutputStream。

字符打印流：PrintWriter

构造函数可以接收的参数类型：

1，file对象。File。

2，字符串路径。String。

3，字节输出流。OutputStream。

4，字符数出来：Writer。

8、序列流：SequenceInputStream：对多个流进行合并。

9、操作对象：ObjectInputStream与ObjectOutputStream：被操作的对象需要实现Serializable（标记接口）。

# 2016年02月01日 day21

## 一、IO流：其他类

1、RandomAccessFile：随机访问文件，自身具备读写的方法。通过skipBytes(int x),seek(int x)来达到随机访问。

2、管道流：PipedInputStream和PipedOutputStream：输入输出可以直接进行连接，通过结合线程使用。

3、RandomAccessFile：该类不算是IO体系中子类。而是直接继承自Object。但是它是IO包中成员。因为它具备读和写功能。内部封装了一个数组，而且通过指针对数据的元素进行操作。可以通过getFilePointer获取指针位置。同时可以通过seek改变指针的位置。其实完成读写的原理就是内部封装了字节输入流和输出流。通过构造函数可以看出，该类只能操作文件。而且操作文件还有模式：只读r，读写rw等。

4、如果模式为只读r。不会创建文件。会去读取一个已存在文件，如果该文件不存在，则会出现异常。如果模式为读写rw。操作的文件不存在，会自动创建。如果存在则不会覆盖。

5、IO包中的其他类：

操作基本数据类型：DataInputStream与DataOutputStream。

操作字节数组：ByteArrayInputStream与ByteArrayOutputStream。

操作字符数组：CharArrayReader与CharArrayWrite。

操作字符串：StringReader与StringWriter。

6、用于操作字节数组的流对象。

ByteArrayInputStream ：在构造的时候，需要接收数据源，而且数据源是一个字节数组。

ByteArrayOutputStream ：在构造的时候，不用定义数据目的，因为该对象中已经内部封装了可变长度的字节数组。这就是数据目的地。因为这两个流对象都操作的数组，并没有使用系统资源。所以，不用进行close关闭。

在流操作规律讲解时：

源设备：

键盘 System.in，硬盘 FileStream，内存 ArrayStream。

目的设备：

控制台 System.out，硬盘 FileStream，内存 ArrayStream。

用流的读写思想来操作数组。

## 二、IO流：字符编码

7、字符编码：字符流的出现为了方便操作字符。更重要是的加入了编码转换。通过子类转换流来完成：InputStreamReader；OutputStreamWriter。在两个对象进行构造的时候可以加入字符集。

8、编码表的由来：计算机只能识别二进制数据，早期由来是电信号。为了方便应用计算机，让它可以识别各个国家的文字。就将各个国家的文字用数字来表示，并一一对应，形成一张表。这就是编码表。

9、常见的编码表：

ASCII：美国标准信息交换码。用一个字节的7位可以表示。

ISO8859-1：拉丁码表。欧洲码表。用一个字节的8位表示。

GBK2312：中国的中文编码表。用两个字节的16位来表示。

GBK：中国的中文编码表升级，融合了更多的中文文字符号。

Unicode：国际标准码，融合了多种文字。所有文字都用两个字节来表示，Java语言使用的就是Unicode。

UTF-8：最多用三个字节来表示一个字符。

……

10、编码：字符串变成字节数组。

解码：字节数组变成字符串。

String-->byte[]; str.getBytes(charsetName);

byte[]-->String; new String(byte[],charsetName);。

# 2016年02月02日 day22

## 一、GUI：概述

1、GUI：图形用户界面。

GUI：Graphical User Interface（图形用户接口）。用图形的方式，来显示计算机操作的界面，这样更方便跟直观。

CLI：Command line User Inteface（命令行用户接口）。就是常见的Dos命令行操作。需要记忆一些常用的命令，操作不直观。

举例：比如：创建文件夹，或者删除文件夹等。

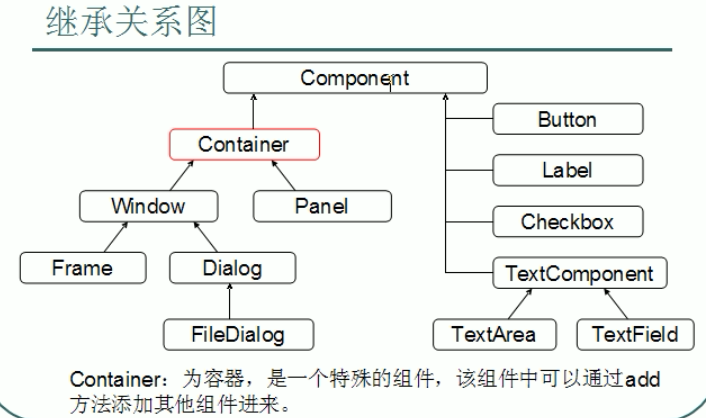
Java为GUI提供的对象都存在java.Awt和javax.Swing两个包中。

2、Awt与Swing：

java.Awt：Abstract Window ToolKit（抽象窗口工具包），需调用本地系统方法实现功能。属重量级控件。

javax.Swing：在Awt的基础上，建立的一套图形界面系统，其中提供了更多的组件，而且完全由Java实现。增强了移植性，属轻量级控件。

3、继承关系图：



4、布局管理器：

容器中的组件的排放方式，就是布局。

常见的布局管理器：

FlowLayout（流式布局管理器）

从左到右的顺序排列。

Panel默认的布局管理器。

BorderLayout（边界布局管理器）

东，南，西，北，中。

Frame默认的布局管理器。

GridLayout（网格布局管理器）

规则的矩阵。

CardLayout（卡片布局管理器）

选项卡。

GridBagLayout（网格包布局管理器）

非规则的矩阵。

5、创建图形化界面：

1，创建frame窗体。

2，对窗体进行基本设置。

比如：大小、位置、布局。

3，定义组件。

4，将组件通过窗体的add方法添加到窗体中。

5，让窗体显示，通过setVisible(true)来完成。

6、建立一个简单的窗体：

Container常用子类：Window Panel（面板，不能单独存在。）

Window常用子类：Frame Dialog。

简单的窗体创建过程：

Frame f = new Frame(“my window”);

f.setLayout(new FlowLayout());

f.setSize(500,400);//设置窗体大小。

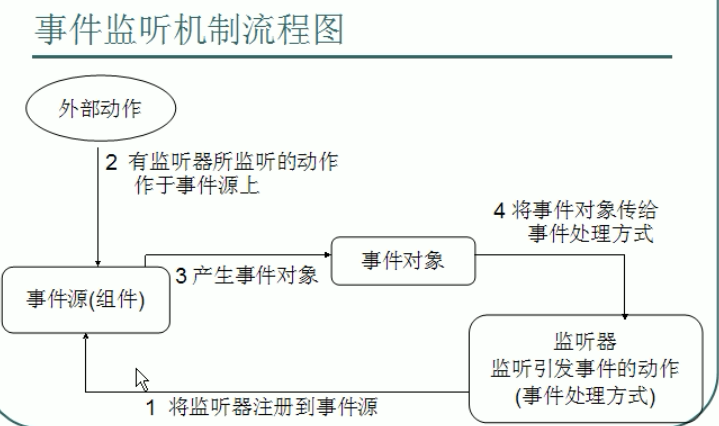
f.setLocation(300,200);//设置窗体出现在屏幕的位置。

f.setVisible(true);。

## 二、GUI：事件监听机制

7、事件监听机制组成：事件源（组件）；事件（Event）；监听器（Listener）；事件处理（引发事件后处理方式）。

8、事件监听机制流程图：



9、事件监听机制的特点：

1，事件源。

2，事件。

3，监听器。

4，事件处理。

事件源：就是awt或者swing包中的那些图形界面组件。

事件：每一个事件源都有自己特有的对应事件和共性事件。

监听器：将可以触发某一个事件的动作（不止一个动作）都已经封装到了监听器中。

以上三者，在java中都已经定义好了。直接获取其对象来用就可以了。

我们要做的事情是，就是对产生的动作进行处理。

# 2016年02月03日 day23

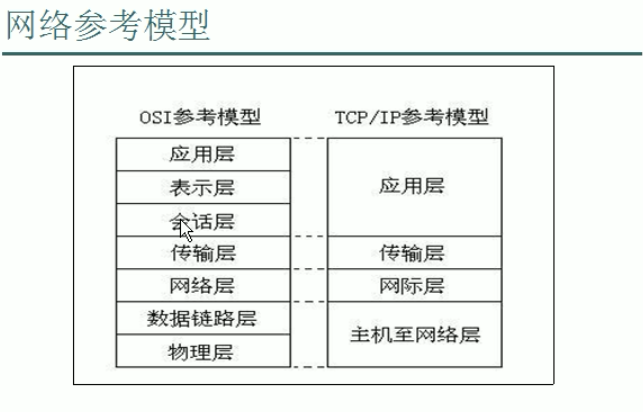
## 一、网络编程：概述

1、网络编程：

网络模型：OSI参考模型。TCP/IP参考模型。

网络通讯要素：IP地址。端口号。传输协议。

2、网络参考模型：



3、网络通讯要素：

IP地址：网络中设备的标识。不易记忆，可用主机名。本地回环地址：127.0.0.1主机名：localhost。

端口号：用于标识进程的逻辑地址，不同进程的标识。有效端口：0~65535，其中0~1024系统使用或保留端口。

传输协议：通讯的规则。常见协议：TCP，UDP。

4、UDP和TCP：

UDP：

将数据及源和目的封装成数据包中，不需要建立连接。

每个数据报的大小在限制在64k内。

因无连接，是不可靠协议。

不需要建立连接，速度快。

特点：面向无连接。数据会被封包。不可靠。速度快。

比如：聊天，视频会议，桌面共享。

TCP：

建立连接，形成传输数据的通道。

在连接中进行大数据量传输。

通过三次握手完成连接，是可靠协议。

必须建立连接，效率会稍低。

特点：面向连接。可大数据量传输。比较可靠。稍微有点慢。

比如：下载。

5、Socket：

Socket就是为网络服务提供的一种机制。

通信的两端都有Socket。

网络通信其实就是Socket间的通信。

数据在两个Socket间通过IO传输。

## 二、网络编程：UDP

6、UDP传输：

DatagramSocket与DatagramPacket。

建立发送端，接收端。

建立数据包。

调用Socket的发送接收方法。

关闭Socket。

发送端与接收端是两个独立的运行程序。

## 三、网络编程：TCP

7、TCP传输：

Socket和ServerSocket。

建立客户端和服务器端。

建立连接后，通过Socket中的IO流进行数据的传输。

关闭socket。

同样，客户端与服务器端是两个独立的应用程序。

# 2016年02月04日 day24

## 一、网络编程：浏览器客户端

1、视频演示讲解。

# 2016年02月6日 day25

## 一、正则表达式

1、正则表达式：符合一定规则的表达式。作用：用于专门操作字符串。

2、正则表达式：符合一定规则的表达式。

作用：用于专门操作字符串。

特点：用一些特定的符号来表示一些代码操作。这样就简化了书写。

所以学习正则表达式，就是在学习一些特殊符号的使用。

好处：可以简化对字符串的复杂操作。

弊端：符号定义越多，正则越长，阅读性越差。

3、具体操作功能：

1，匹配：String matchess();方法。用规则匹配整个字符串，只要有一处不符合规则，就匹配结束，返回false。

2，切割：String split();

3，替换：String replaceAll();

4，获取：将字符串中符合规则的子串取出。

4、获取：操作步骤：

1，将正则表达式封装成对象。

2，让正则对象和要操作的字符串相关联。

3，关联后，获取正则匹配引擎。

4，通过引擎对符合规则的子串进行操作，比如取出。