semantic ambiguity 语义含混

compiler(编译器):把源代码source code（一段程序）转换成处理器processor可以识别的语言

C++或C通常用compiler把源代码直接转化成机械码(machine language),但是不同的CPU有不同的machine language，所以同样的一个可执行的文件，不同的平台上就不能执行，那么就需要对不同的平台对source code 重新编译（re-compile）

Java则彻底不同

1. Java compiler:

把Java source code转换成byte code（字节码）

机械码是针对叫做Java Virtual Machine的虚拟机的（fictitious computer）

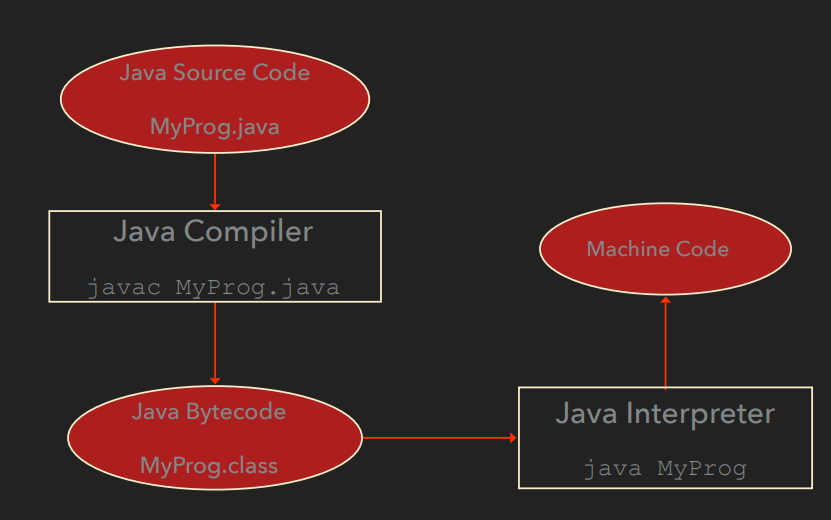
2. Java interpreter（直译器）

执行这个JAVA VIRTUAL MACHINE(JVM)虚拟机

把Java byte code转换成machine language并且执行它

于是java compiler实际上是没有与任何特定实际接触的

只要被编译成byte code，那么java程序就可以被任意平台电脑使用



在java中，我们有两种不同的程序

1. application 应用

autonomous applications自主应用 或独立程序

由当地的操作系统OS(operating system)进行(通过JVM)

可以使用图形graphics 和GUI图形用户界面或简单的IO控制台（console）

必须有个主要方法

2. Applets

由网络浏览器进行（通过JVM）

必须嵌入HTML页面

必须用GUI

一些概念

Algorithm算法：一步步的过程来解决一个问题/展示为自然语言

Pseudocode伪代码：算法被写成更正式的语言/看起来像代码，但实际上没有specific syntax(具体的语法)

Program 程序：algorithm被写成programming 语言，有具体的语法

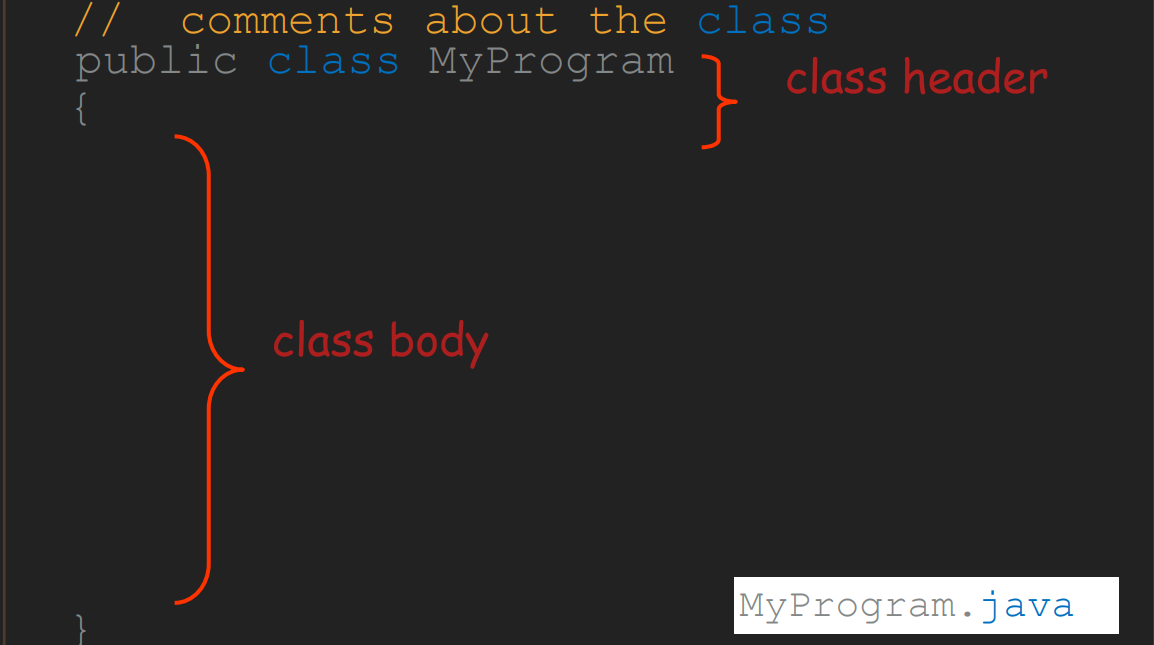
一个java program

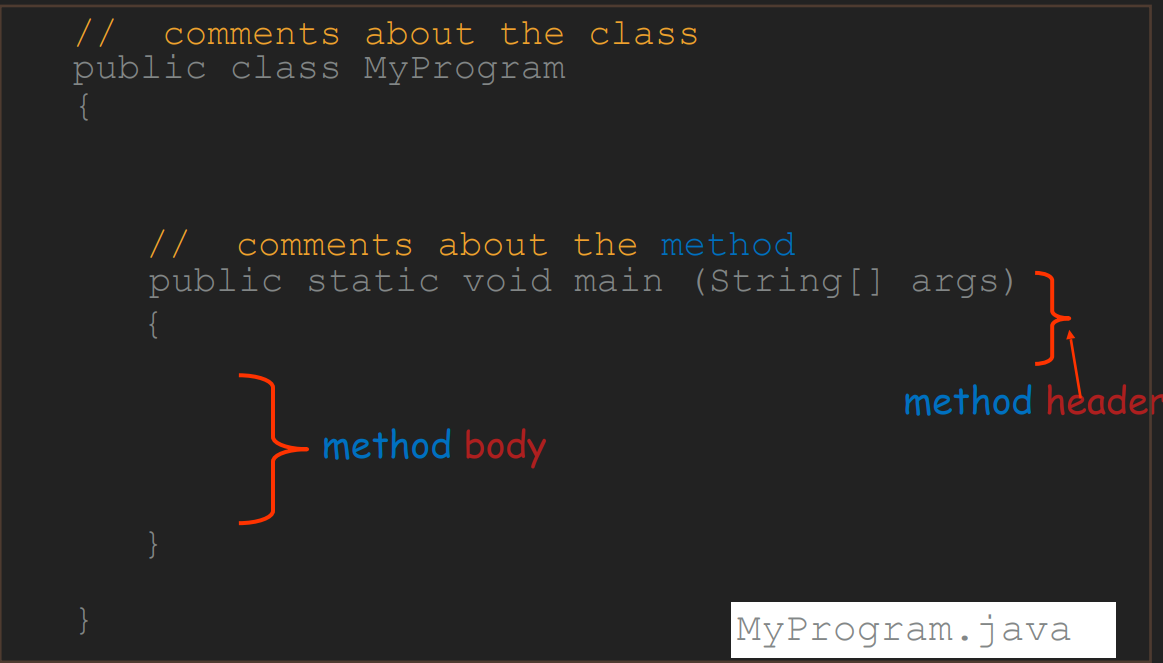
由一个或多个class组成

一个class由一个或多个method组成，method就是action

一个method包含着编程语句/指令programming statements/instructions

总包含着一个叫做main的method





SYNTAX AND SEMANTICS

Syntax rules语法规则

定义我们怎么把符号,reserved words保留字，identifier标识符组合在一起来构建一个有效程序

Semantics

定义一个statement的意味着什么

一个程序必须做到syntactically correct,但没必要做到semantically correct

三种错误

Compile-time (syntax) errors

Compiler会找到语法或其他基本的错误

不会创造出一个可运行的程序

Run-time errors

在程序运行时出现的问题

导致程序反常的结束

Logical (semantic) errors…又叫做bug

算法上出了问题

Compiler识别不出问题

程序可运行，但出来的结果是错的

消除bug的过程叫做debugging

Compiler: javac Hello.java

byte-code program: Hello.class

Interpreter: java Hello

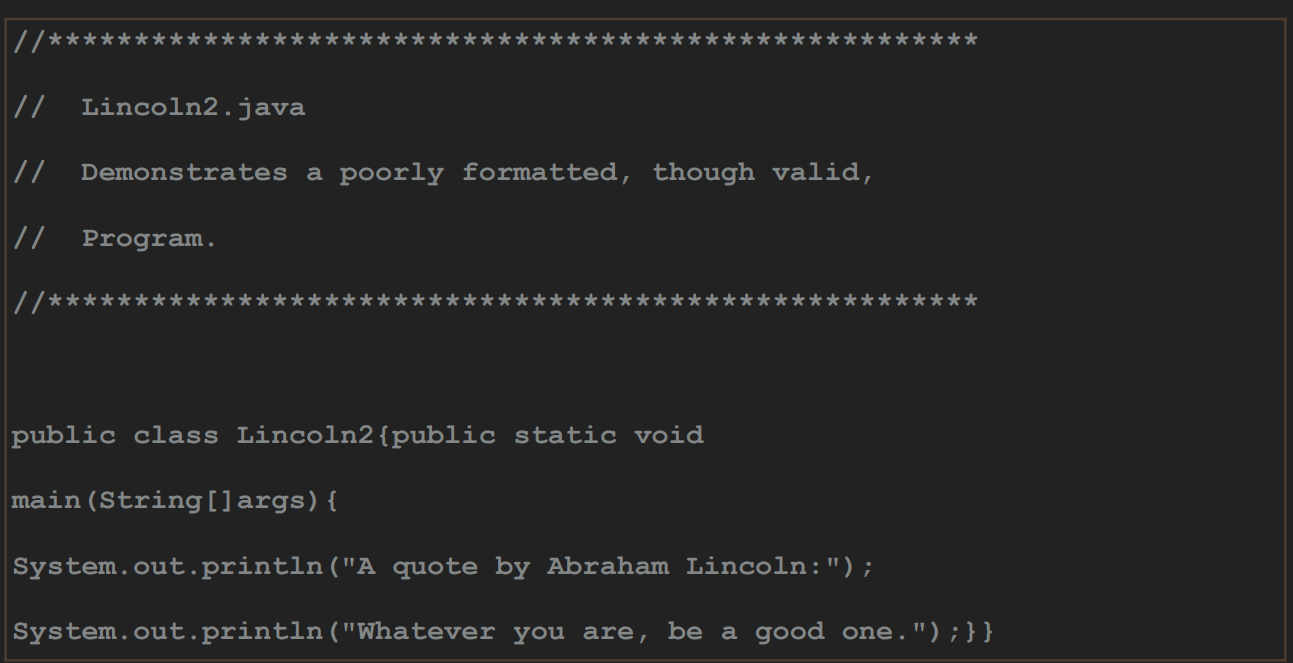
IDE开发环境，比如eclipse

1.Comment //后面的东西，没啥实际意义，但一定要写，包括姓名，目的，大致思路

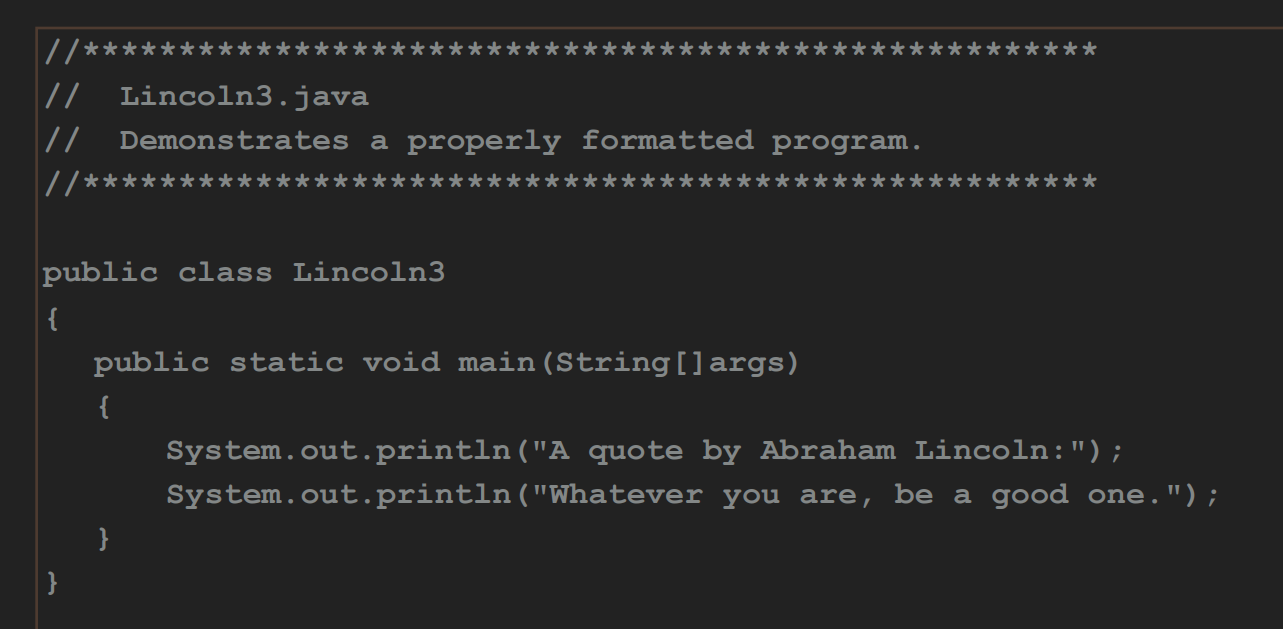
2.Identifiers:码农用来对代码，变量命名所用的文字，可以由字母，数字，下划线\_，$美元符号组成，不能以数字开头，不能用有特殊意义的文字

3.Indentation:留白，在java中，空格没有具体意义，并会被缩略，通过留白你的代码会容易阅读

错误的indentation

没留白

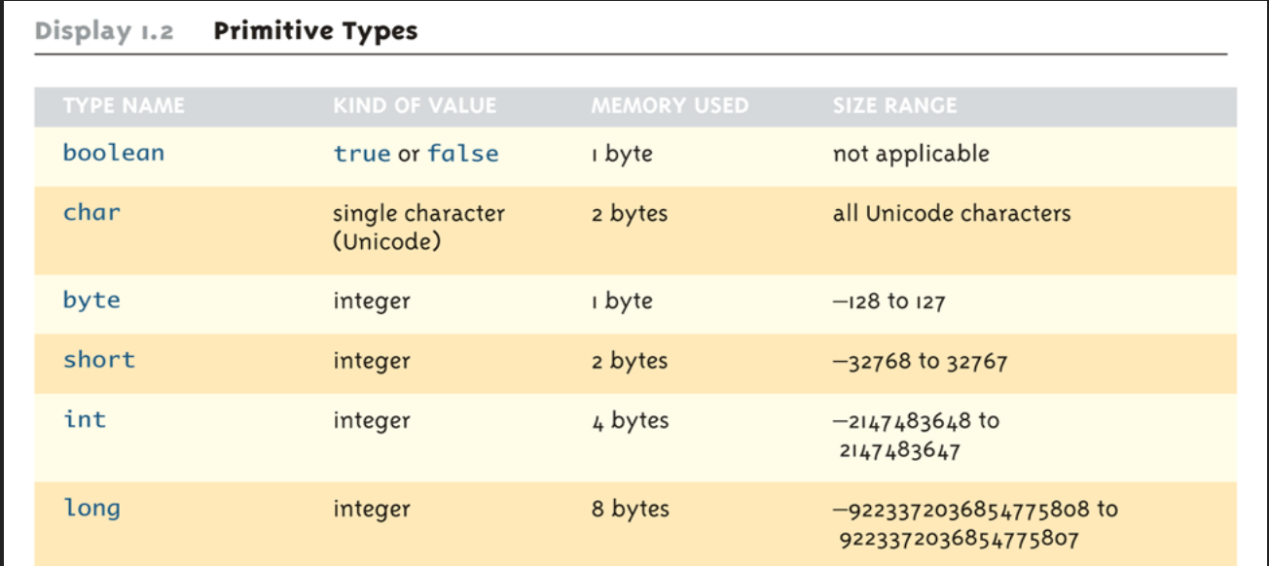
瞎比留白

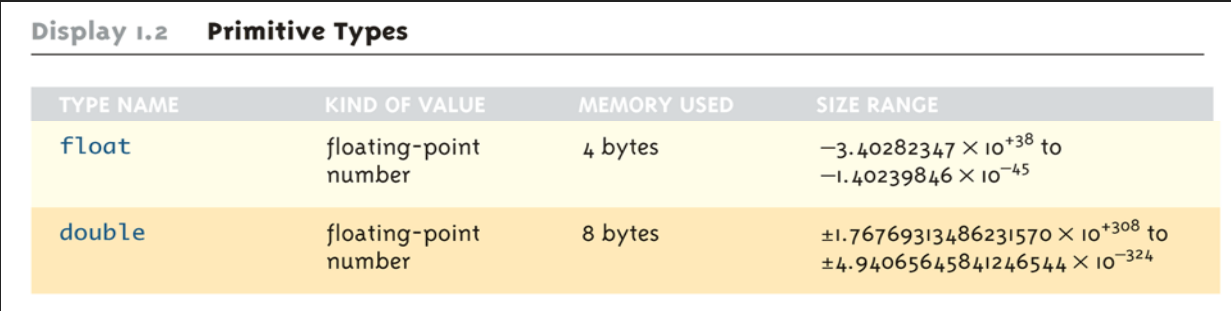
破费

4.primitive types

Java中一共有八种type

Numeric：4种来表示整数(byte, short, int, long)比如-3,-5 两种来表示浮点数（float, double）比如3.5





Characters:表示字符(char)比如a

Boolean values:表示对或错(boolean)

Floating point numbers只是近似值，比如6.666666666666……会给你显示成6。666666666667

一个char存储一个字符可以表示的有大小写26个字母/标点符号.，等/个位数字1234等/特殊符号&这种/控制符号 “\n””\t”这种

Boolean的值只有reserved words true/false

5.variables变量：存储中一个location的名字，用来存储信息（价格，大小等），在被使用前必须写清楚——变量的名字/存储信息的类型/可以在程序的任意地方写清楚这些信息,但是必须要在第一次运转之前写

int total int是存储信息的类型，total是名字

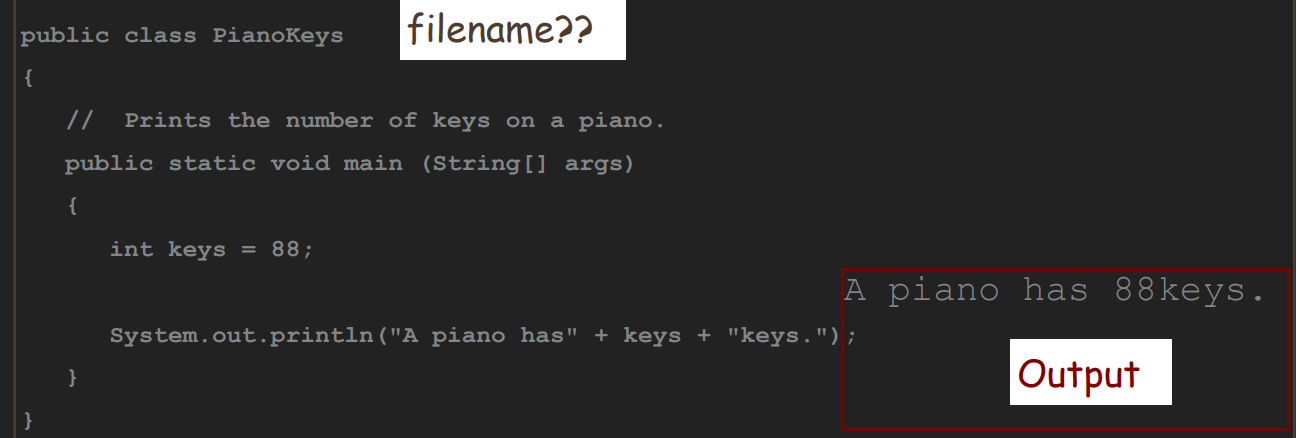
int count, temp, result 多个变量可以一次批量创造

uninitialized :给出了一个变量但没给出具体值

在某些情况下一个uninitialized 变量会给一个默认值—最好别依赖他/清晰地赋值可以让程序更整洁

一个变量可以得到一个初始值通过declaration

int sum = 0



他给key附了一个88的变量

Print 里没打双引号的KEY实际上是88

CONSTANTS 常量

与变量相同，但是只能有一个值 只要这个程序激活

在运转程序时，只要你改变这个constant，那么compiler 就会出问题

用 final modifier（最后改变）final int MIN\_AGE = 18;

避免无意中试图改变一个值/促进整个程序中使用的值的更新

6.output

System.out.print 显示括号（parenthesis）里的内容

System.out.println 显示括号里的内容+自动换行

int x = 1, y = 2; System.out.println("x+y="+x+y); 会显示12因为x+y=x然后 y显示的是2

System.out.println("x+y="+(x+y));x+y=x+y

在JAVA中，加号又能表示加号，又能表示连接符，只有两边都是数值类型时，认作加号

ESCAPE SEQUENCES 转义序列

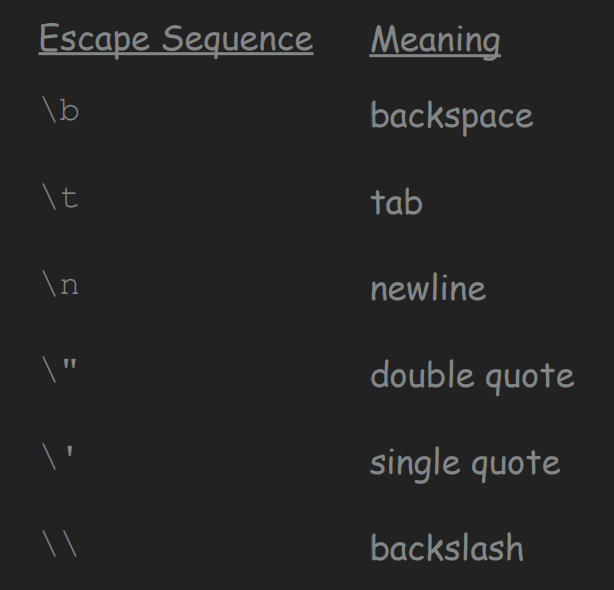
要print出一个双引号就必须用到ESCAPE SEQUENCES

Sequence是一系列的字符来表达的一个特殊字符

用一个反斜杠开始

认作一个单一字符

System.out.println ("I said \"Hi\" to her.");



B将当前位置移到前一列

T 跳到下一个tab位置

N新换一行

“双引号，‘单引号

\r\n \r是回到开头，\n是新的一行

[\\单斜杠](file:///\\单斜杠)

System.out.println ("I said \"Hi\" to her."); 就直接把\’’看成双引号就行了，

7.ASSIGNMENT赋值

用来改变变量的值

赋值的符号是=

Syntax语法 Variable = Expression 左边是变量 右边是变化

结果被存储在左边的变量中

8- ARITHMETIC EXPRESSIONS 算术表达 operator符号 operand数

运算符号Arithmetic operators:

Addition + Subtraction - Multiplication \* Division / Remainder %

除法规则，如果两个数都是整数，那么答案也是整数10 / 8 equals 1 8 / 12 equals 0

有一个小数 10.0 / 8 equals 1.25 8 / 12.0 equals 0.6667

10%8=2

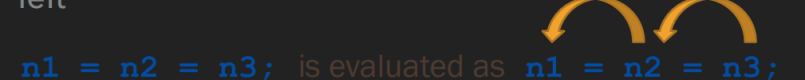
OPERATOR PRECEDENCE 符号优先级

1：括号中的 2：一元运算符 3：乘，除，余 4：加减 string concatenation字符串连接 5:赋值运算符

运算符是一元、二元还是三元，要看它要求有几个数参加运算，比如"++"运算符，只需要一个数据单元参加运算，如++a，它就是一元运算符；而"+"、"-"等运算符，如a+b，要求有两个数据单元参加运算，它就是二元运算符

对于一元运算符，同优先级从右往左看+-+rate is evaluated as +(-(+rate))

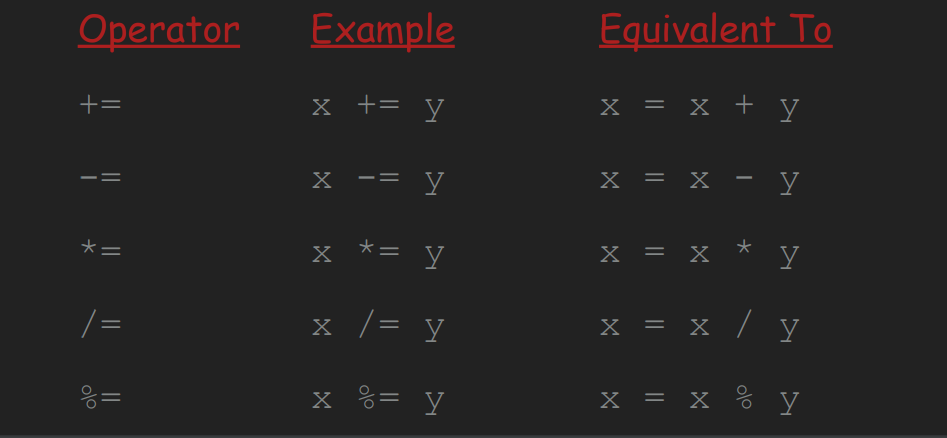
二元运算符同优先级从左往右看base + rate + hours is evaluated as (base + rate) + hours

一系列赋值从右往左看

=（赋值号）优先级小于计算符号，先把右边算完，再赋值

answer = sum / 4 + MAX \* lowest

9.其他赋值符assignment



Increment count++ 就是count = count+1

Decrement count--就是count = count-1

++ 自增(前) b=++a; b=4;a=4; 先让a的值加1，再把a的值赋值给b   
++ 自增(后) b=a++; b=3;a=4; 先把a的值赋值给b，再让a的值加1   
- - 自减(前) b=- -a; b=2;a=2; 先让a的值-1,再把a的值赋值给b   
- - 自减(后) b=a- -; b=3;a=2; 先把a的值赋给b,再让a减1

放在前面就是先自增，在赋值，放在后面是先赋值老的值，再自增

10 - ASSIGNMENT COMPATIBILITY 赋值兼容

通常来说，一种类型的值不能存储到另一个类型的值

int intVariable = 2.99; //Illegal 不行，因为int是Integer整数，2.99不是

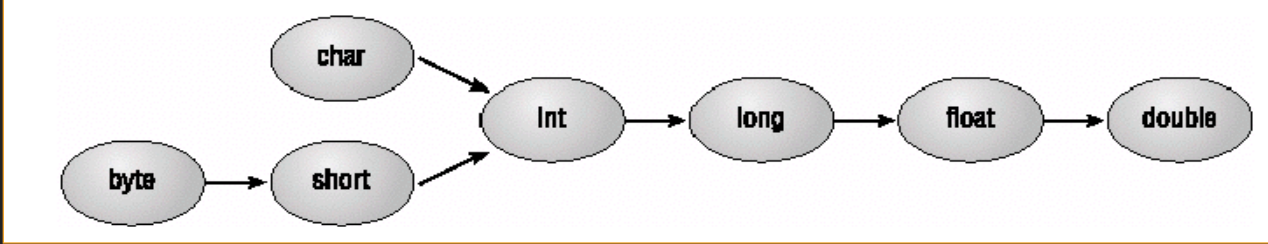
double doubleVariable = 2 可以 整数可以被存在DOUBLE中

2 / 4 答案是0 2/4.0 答案是0.5

在JAVA中，有三种方式来形成类型转换type conversation

1. ARITHMETIC PROMOTION:自动发生只要Operand是不同的type

低等Operand从图中从左往右升级成一个type ，最终转成Int或double



2.ASSIGNMENT CONVERSIONS赋值转换

通常出现在当一种形式的expression被分配到另一种形式的变量

var = expression;

Widening conversion 扩大转换

如果这个变量比现有表达形式有更宽广形式，那么现有表达形式就自动被拓宽

整数和浮点数可兼容， BOOLEAN无法兼容

Narrowing conversion 收缩转换

如果VARIABLE的type比现有表达形式更小 就会出现compilation error 编译错误，因为数据流失了

Casting:码农可以明确指定一种转换方式格式是 (desired\_type) expression\_to\_convert

即（需要的格式）要转换的数(int)3.7

11.Strings

一个变量可以是primitive type原始类型：int, float, boolean, … 也可以是对对象的引用a reference to an object ex: String, Array, …

一个字符串A character string 是一个由string class描述的object，由双引号划定界限

System.out.print("hello"); // string of characters

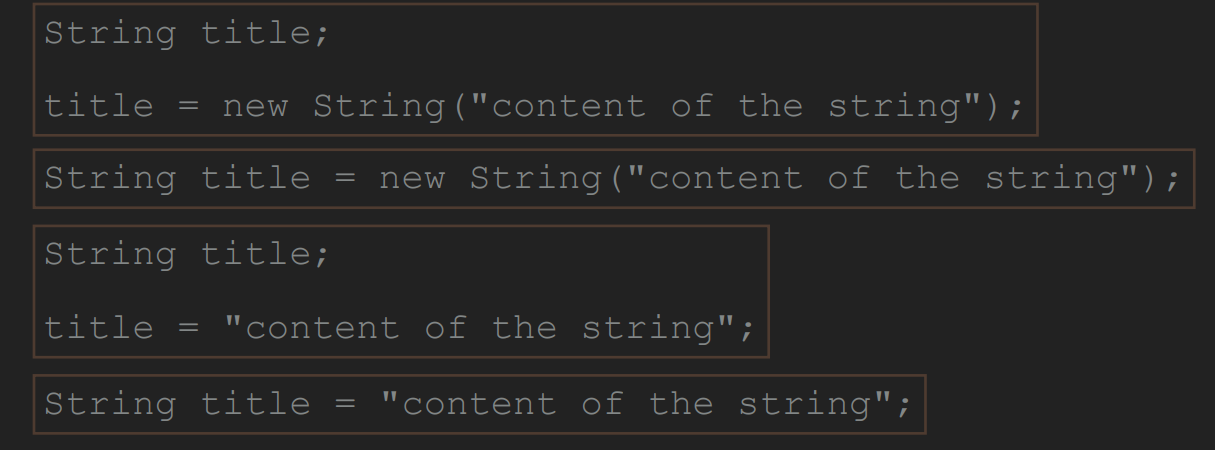
System.out.print('a'); // single character

声明一个string：

1.声明对一个string object的引用 String title;

2.声明object自身 title = new String("content of the string");这个叫String constructor

3.因为string太常见了，我们不用一个新的符号来创建string object

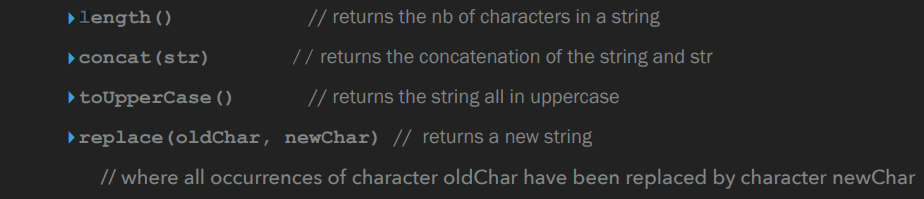
这些格式都行

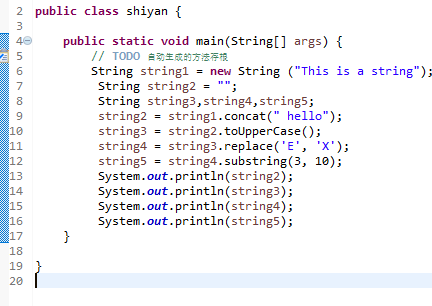
一旦string创建，那么无法改变它的内容或名字

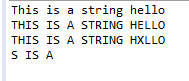
String类提供了：

1：+符号，来连接字符串 比如String solution = "The answer is " + "yes";

2.许多方法来查询string

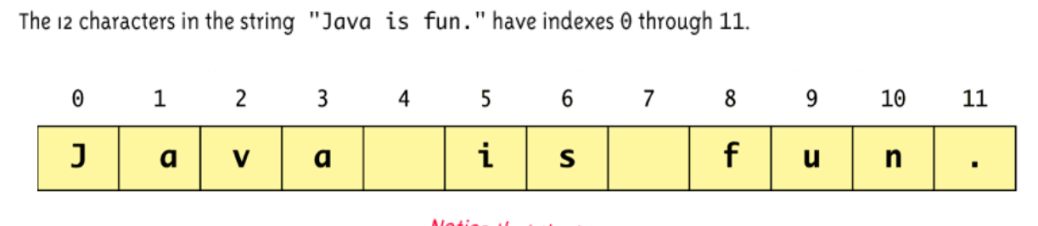






Length求长度，concat连接新词，to uppercase大写，replace把字母换掉 substring截取

String index（串索引）是从0开始索引的

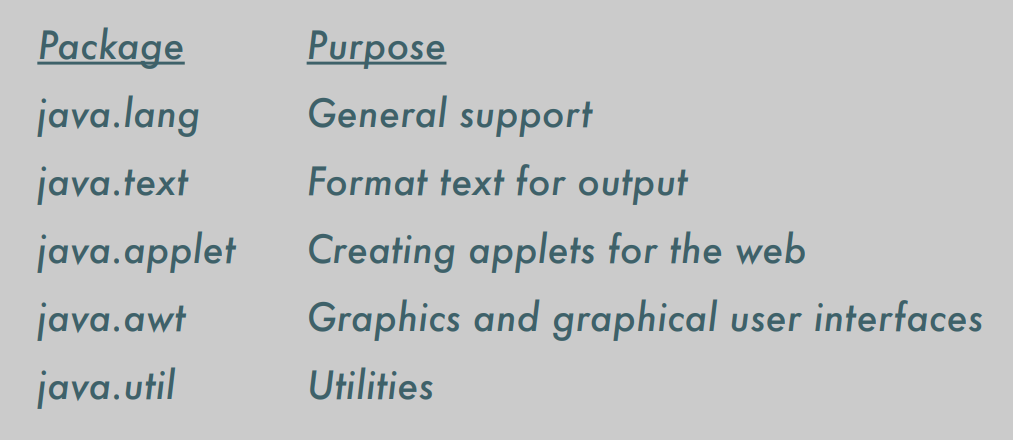


在STRING中，标点符号与空格也占了character

CH2.

Class library 我们用来开发程序所用的class的集合 ，例如Java standard class library

而这个Java standard class library中的Class们又被细分成packages



The import declaration

要用package中的class的话，你可以只输入package（java.text）中的一个class(decimalformat)



输入其全部class



Java.lang中的所有class会自动应用到所有程序

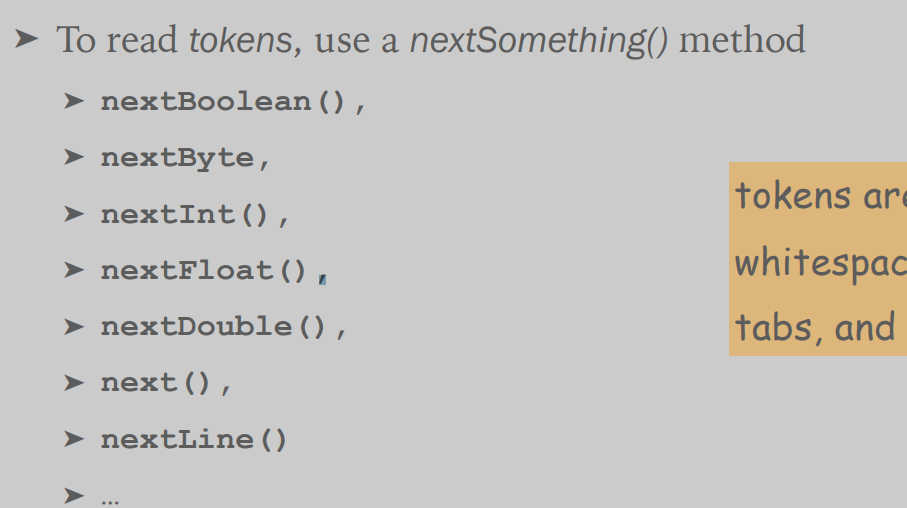
在开局class前就要import

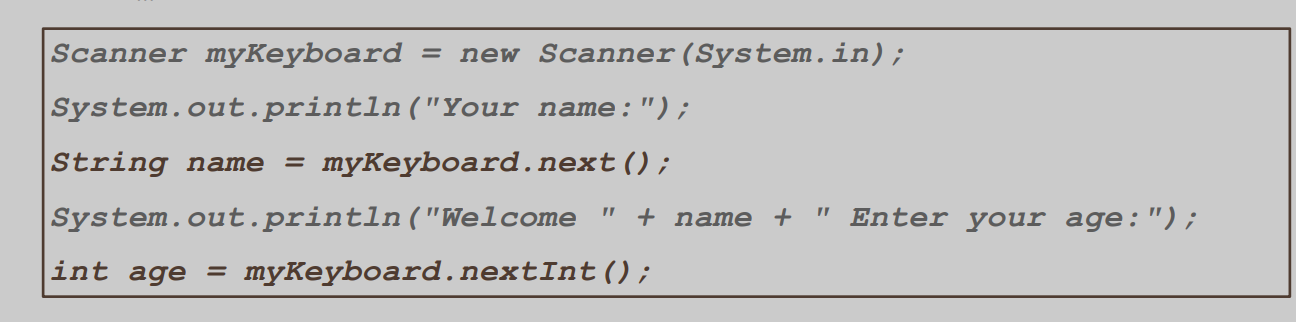
import java.util.Scanner; 加入util scanner package

Scanner myKeyboard = new Scanner(System.in)

Mykeyboard这里是个对象，随便什么名字都可以

键盘由system.in的东西表示了





CHA3

FLOW CONTROL 流控制

Sequence顺序:除非有特定说明，语句的执行都是顺序/线性的

顺序定义：一句接着一句

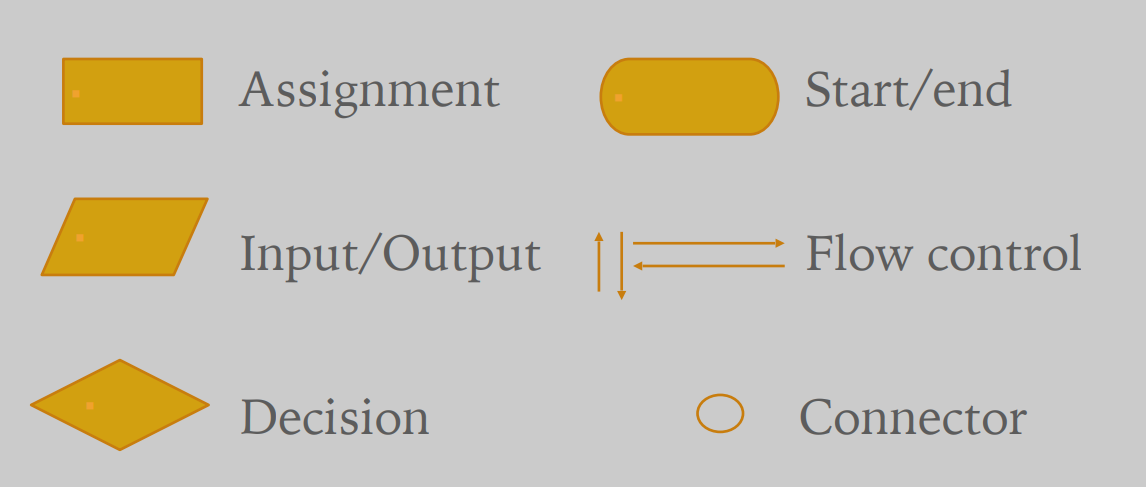
CONDITIONAL STATEMENT条件语句：取决于条件（condition）,这个语句可能执行或不执行

Repetition statement(loop)重复语句：一个语句被一遍又一遍的执行直到某个环境变成true or false

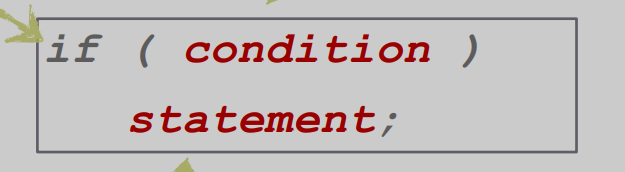
这些决定基于boolean expression (也叫condition)的值是true or false

语句的执行顺序叫做flow control

FLOW CHART:用图形的方式来表达算法（algorithm）



CONDITIONAL STATEMENT:

1:IF STATEMENT  
例如

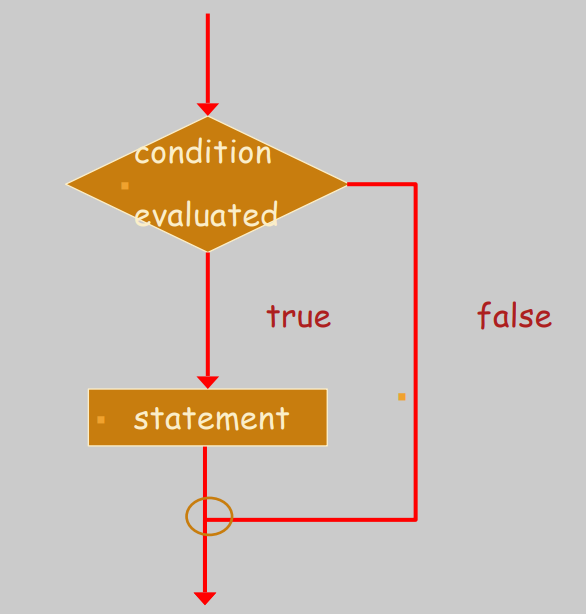
if (sum >= 100)

delta = 5;

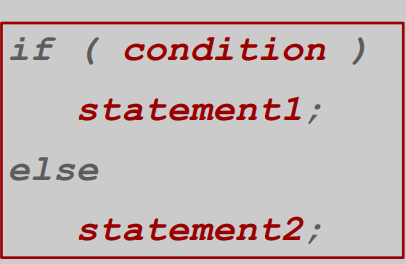
IF: reserved word

Condition: true or false

Statement:如果condition true，执行，false,跳过

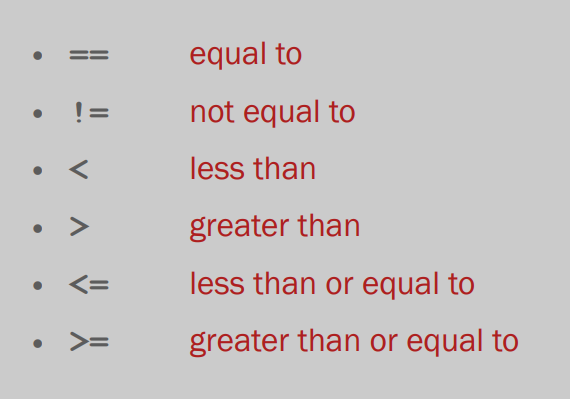


2. IF ELSE STATEMENT



Condition true:执行1，否则执行2

RELATIONAL OPERATORS关系符号



注意==是数学的等号，而=是赋值

比较浮点数的注意点：

当比较两个浮点数是否相等时（float or double）,不能使用==号，因为浮点数表达的不是一个精确的值（approximated）

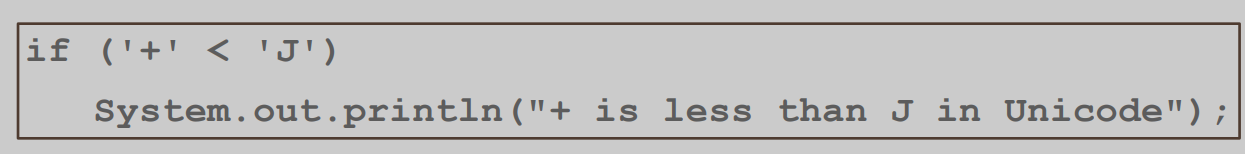
浮点数只能做到相对接近



我们可以用关系符号haracter,结果基于unicode表



把left digit 和right digit连起来组成一个自然数，比如n是110



比较string:不能用==等比较符号来比较string

equals() 来决定是否两个string 同一内容

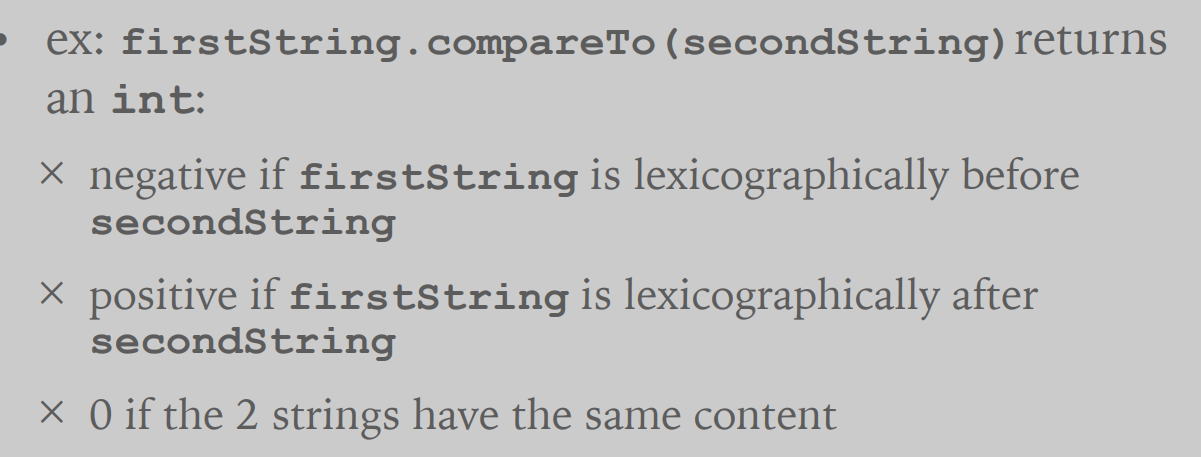
ex: firstString.equals(secondString)

然后会得到一个boolean值，true相等，false不等

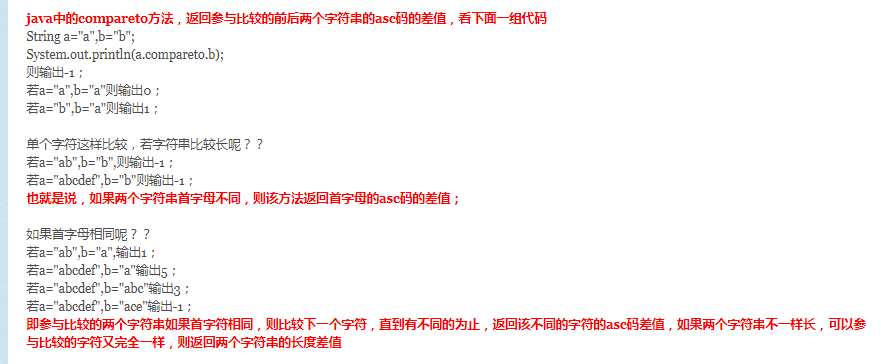


Ignorecase就不用管大小写

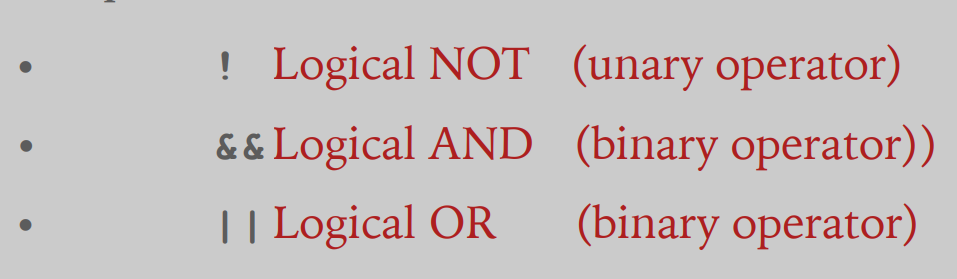
compareTo()比较一个string通过unicode码组出来的数大小



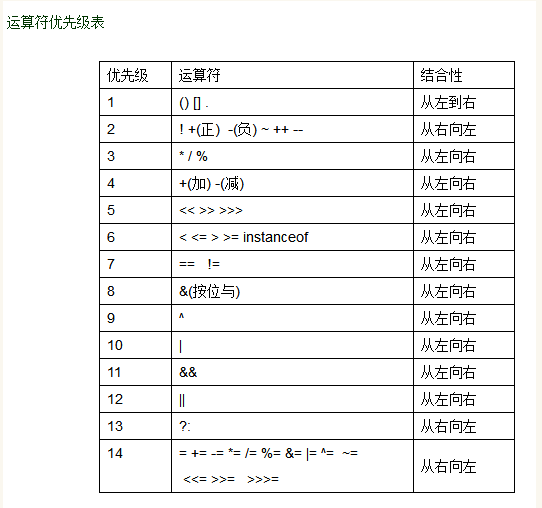
这个compare to 的比较方法



4.逻辑符号



感叹号跟在否定前面比如！A

****

所有的逻辑运算符优先级都比rational operator(><)或者arithmetic operator(+-\*/)低

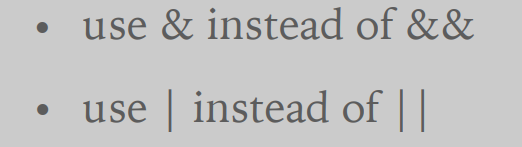
NOT优先级更高

SHORT-CIRCUIT EVALUATION短路求值（lazy evaluation）

比如and关系第一个false我们就不用知道第二个

Or关系第一个true我们就不用知道第二个

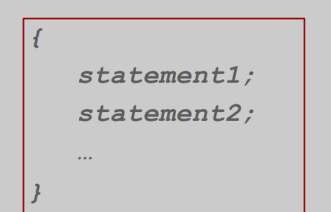
COMPLETE EVALUATION



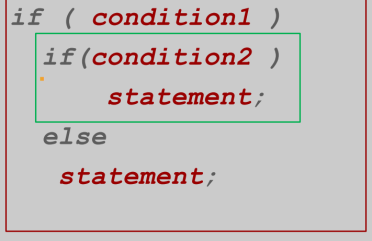
这样就没有短路求值的说法，符号两边都会求值，比较麻烦，所以很少用

5.compound statement

许多语句可以被集合在一个compound statement里（也叫做block）

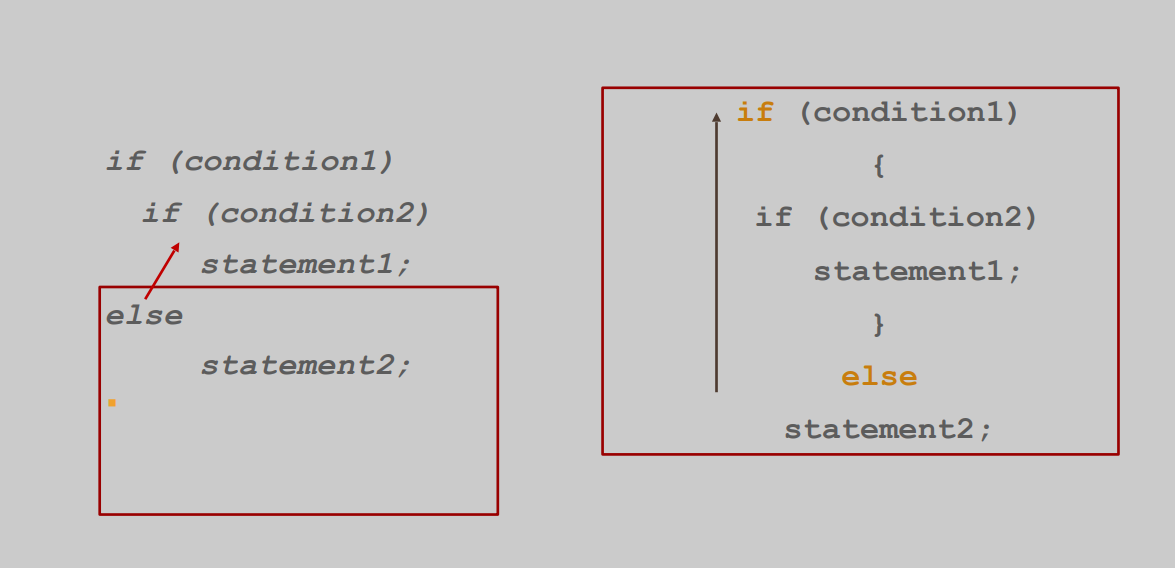


6. nested if statements （if里面还有if）





Else跟随着上一个if语句

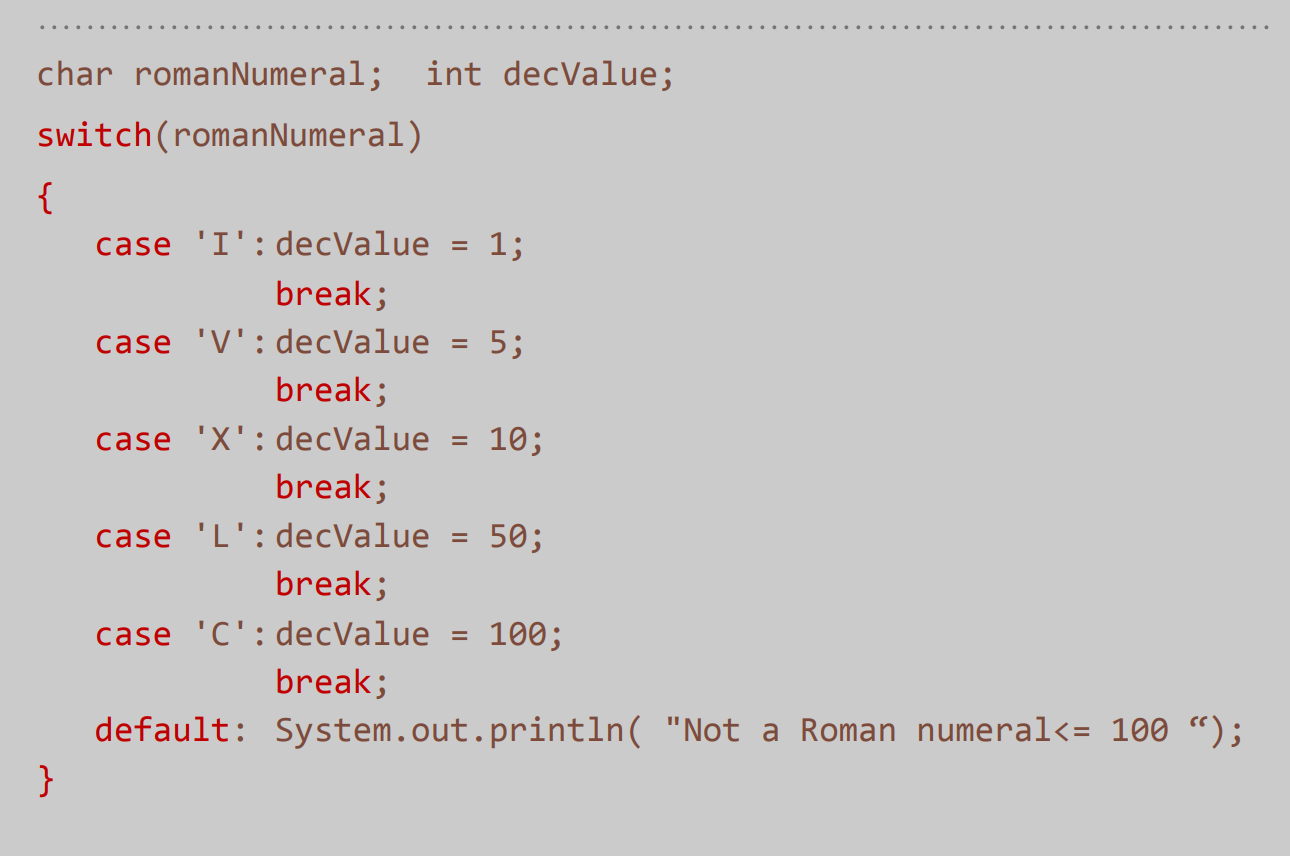


这里用花括号组成了一个compound statement所以排除在外

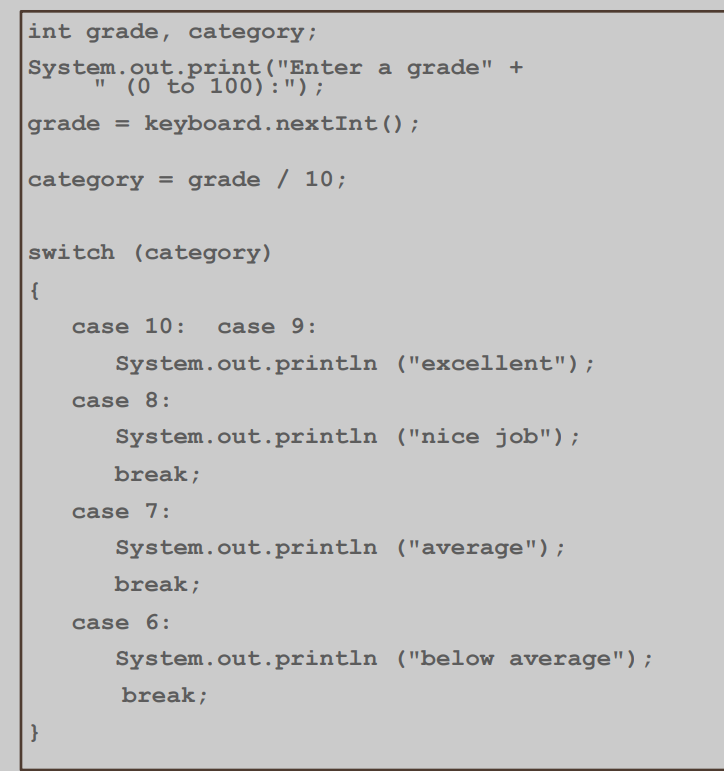
7. switch statement

用来替代if else if else无限循环

只能用来求char int short byte(不能求float和double因为他们无法相等，不能求string因为string无法相等要用equal to)



SWITCH一个数，如果SWITCH括号内的数=其中一个CASE，那么跳到这个CASE中，一句一句往下执行，直到break结束这个SWITCH一句，如果没有匹配的，就default

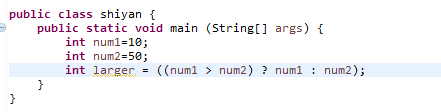


以此图为例，如果得分是96，实际上会显示 excellent

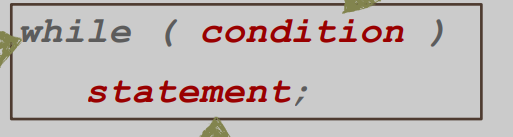
Nice job 因为excellent后没有break

8.conditional operator

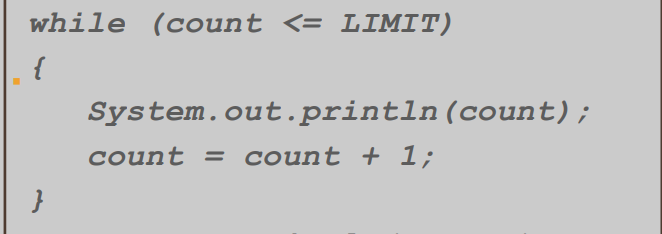
If的简短法condition ? expression1 : expression2如果condition true,那么这段话=expression1，反之=expression2



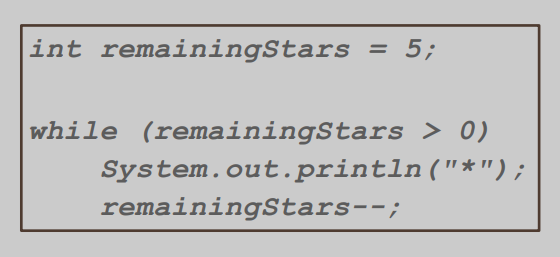
9.The while loop



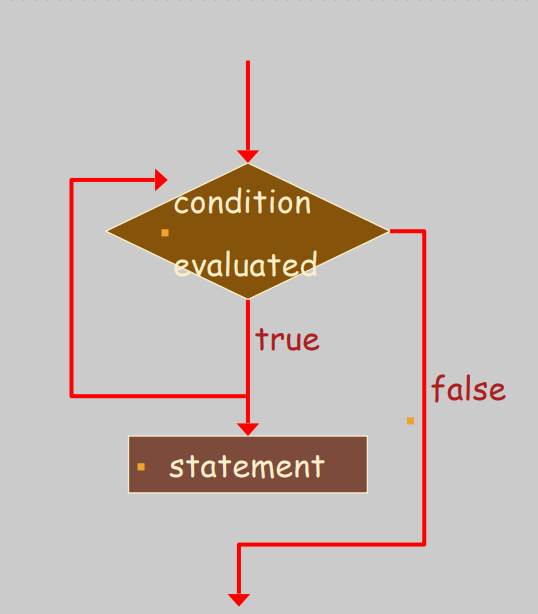
直到这个condition变成false ,整个while语句都会不停重复



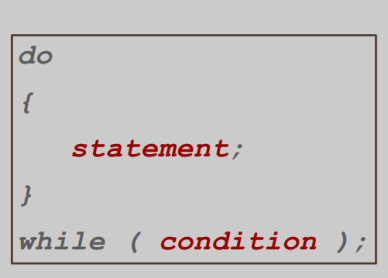
Statement可以加个花括号来进行一段statement来保险



如此图为例，没加花括号那么就会无限打\*号而进入不到下一个remainingstar- -的环节中



10.the do while 循环

直到condition为false，跳出循环

与while语句区别，do while会至少进行一次

典型例子



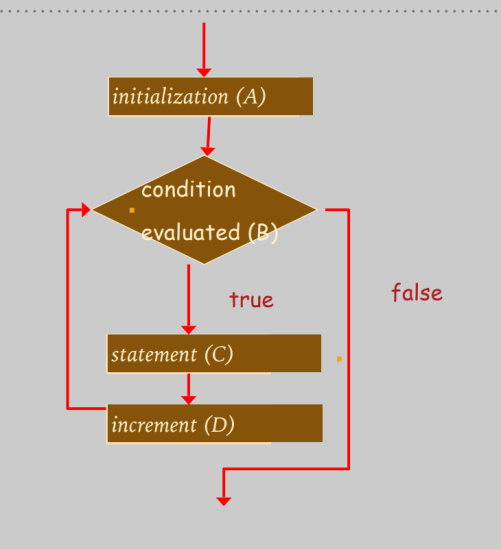
11. the for loop

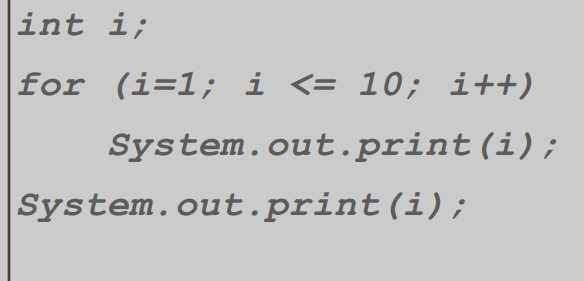


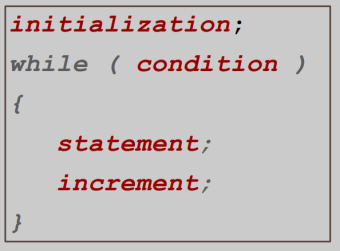
Initialization：在Loop开始前先进行一次，不会循环

Condition：这个Loop会不断进行直到condition变成false，会循环

update（更新）：在每次循环的末尾进行，会循环

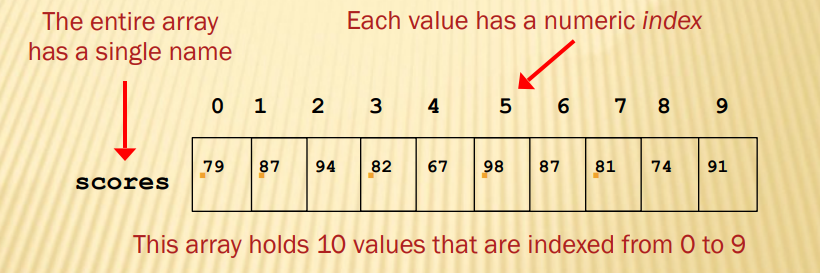




一个for loop等同于以下while loop

CH6

Array：一系列有顺序的同一type的element



Index：编号

10个数就有0-9的编号

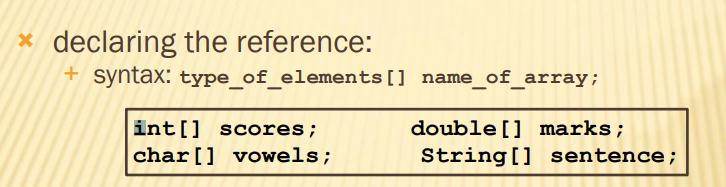
Elesment分类有：

a primitive type： int double char之类的

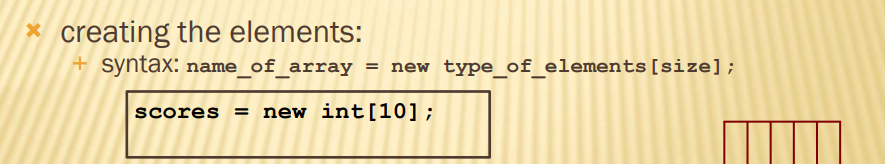
an object reference

array 的语法

1.先定义一个array

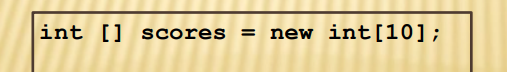


2.创造里面的元素

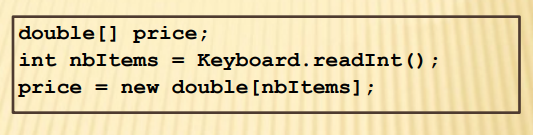


这个size指的是array元素的个数

两种写一起的形式



Array的size必须是一个整数（可以是常量或变量）



Price是这个array, size是nbItems

Array initialization初始值

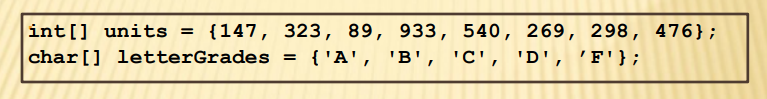
当你创建一个array还没定义元素的时候，每个元素初始值都是0

int, double --> 0

boolean --> false

references --> null

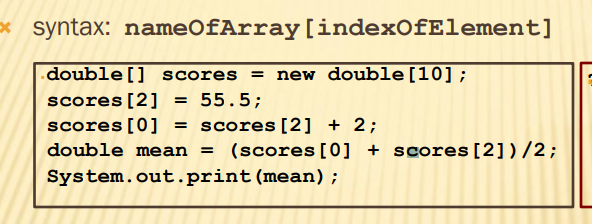
我们可以一步declare+initialize(就是直接写出元素)



Size没有声明（=写出来的nmuber数量）

没有用new

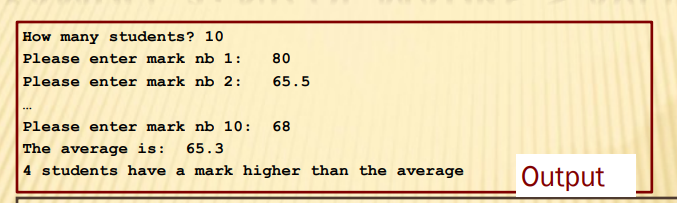
怎么样编辑里面的element



就name【第几个index】

Length 就是长度





**import** java.util.\*;

**public** **class** shiyan {

**public** **static** **void** main(String[] args) {

Scanner keyboard=**new** Scanner(System.***in***);

System.***out***.print("How many students:");

**int** studentnumber=keyboard.nextInt();

**double**[]score=**new** **double**[studentnumber];

**for**(**int** i=0;i<score.length;i++)

{

**int** t=i+1;

System.***out***.print("number "+t+" student mark is:");

score[i]=keyboard.nextDouble();//依次输入数据}

}

**double** allscore=0;

**for**(**int** i=0;i<score.length;i++)

allscore=allscore+score[i];//求总和

**double** averagescore=allscore/studentnumber;//求平均值

System.***out***.println("The average score is "+averagescore+"");

**int** numberhigh=0;

**for**(**int** k=0;k<score.length;k++)

**if** (score[k]>averagescore)

numberhigh=numberhigh+1;//求数目

System.***out***.println("There are "+numberhigh+" students higher than average score");

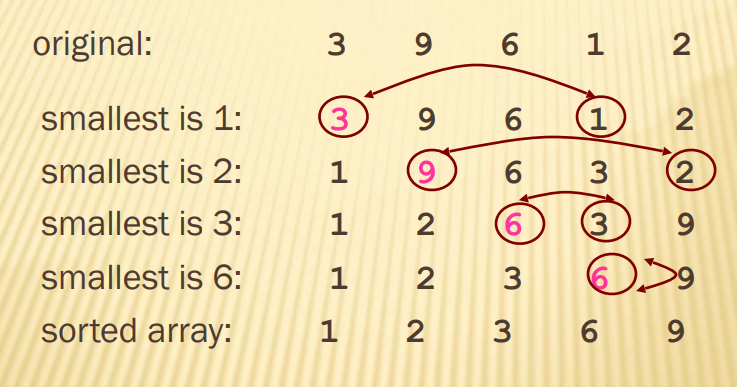
Sorting:把一系列item按特定顺序排好的过程

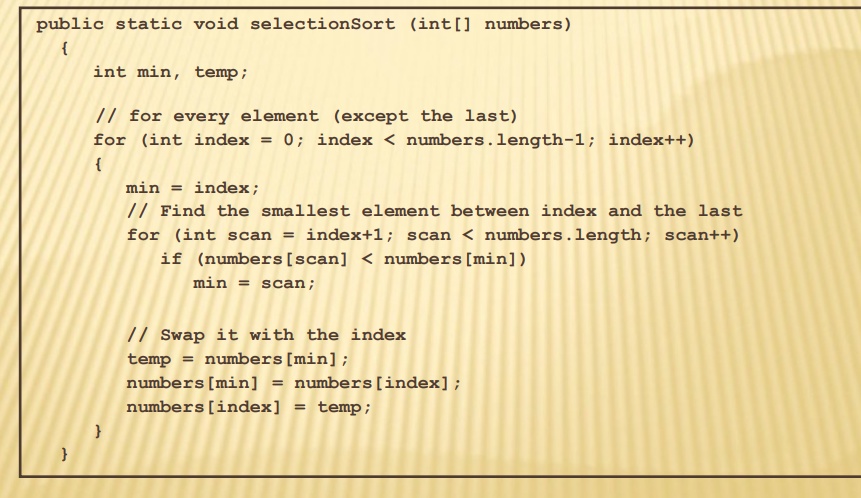
Selection sort:

1.找到Array中最小的值，与第一个数换位置

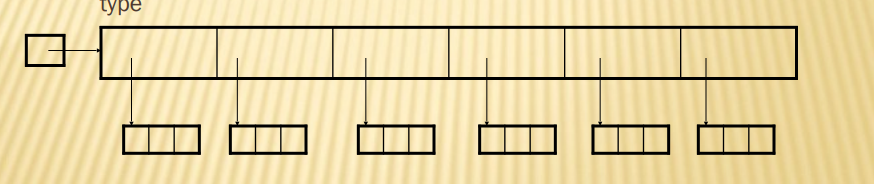
2.找到ARRAY中第二小的值与第二个数换位置

3.重复此过程直到每个值都到了他应该到的地方（有n个数就重复n-1次）





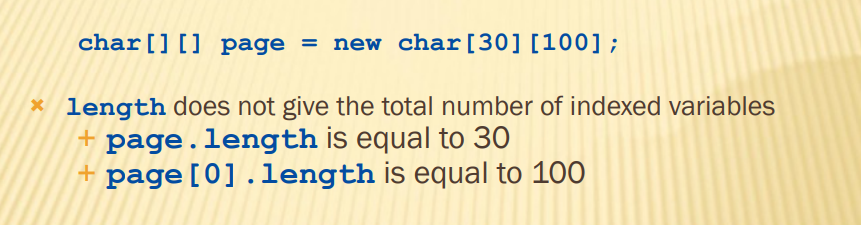
3.multiple array





提取其中的值

.length不会给出整个array的长度



ragged参差不齐的 array

