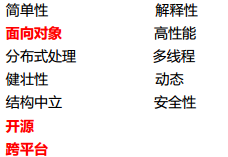
# Java SE基础笔记

### Java入门

#### 1.1 Java简介

#### 1.2 Java的特点



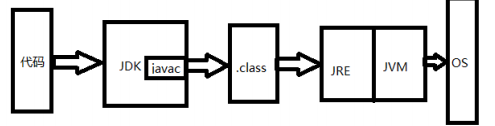
#### 1.3 JDK的安装以及环境变量的配置

##### 1.3.1 JDK、JRE与JVM之间的联系

JDK = JRE + tools

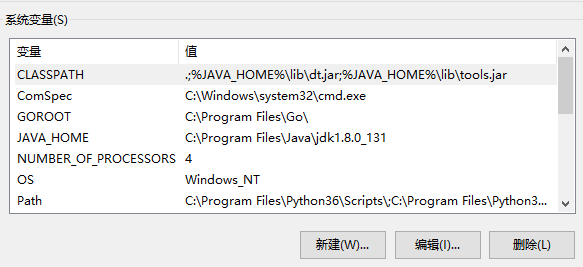
JRE = JVM + 类库

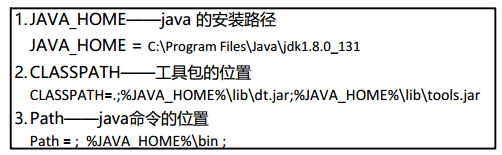
##### 1.3.2 java的运行原理



##### 1.3.3 java 环境变量的配置



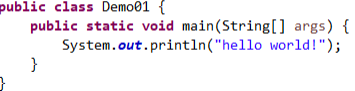




注意：classpath中最开始是一个点，表示本目录。

Javac xxx.java java xxx

##### 1.3.4 HelloWorld入门程序



#### 1.4 Java语言开发工具

##### 1.4.1 文本开发工具

EditPlus

NodePad++

Txt

Sublime

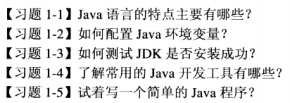
##### 1.4.2 集成开发工具

Eclipse

IDEA

MyEclipse

#### 1.5 习题



### Java基本数据类型

#### 2.1 注释

注释分为：单行注释，多行注释，文档注释

package scnu.edu/cn, 
这 罡 文 档 注 
@author Gyn 且 
这 罡 一 个 h 巳 11 。 艹 r 程 序 
这 罡 多 行 注 
一 个 文 档 中 只 有 一 个 些 bl 类 
而 且 这 个 些 bl 类 的 名 称 和 文 档 名 相 同 
public class DemoØ1 { 
这 罡 单 行 注 
/ / m 毓 n 函 罡 程 序 的 入 囗 ， 程 序 从 这 里 开 始 执 行 
public static void main (String[ ] 引 、 gs ） { 
控 台 输 出 语 甸 ， 打 印 出 “ h 巳 且 。 rld ” 
System out ． println ("hello 到 orld ！ “ ） 丿 

#### 2.2 关键字与标识符

##### 2.2.1 关键字

关键字为预定义的名称。

abstract 
continue 
for 
instanceof 
transient 
assert 
default 
goto 
int 
package 
strictfp 
synchronized 
try 
boolean 
interface 
private 
this 
break 
finally 
implements 
long 
protected 
static 
volatile 
import 
public 
throws 
While 

##### 2.2.2 标识符

1. 标识符，自定义的名称。类似数学中的未知数x，可以随意定义为a,b,x等。

2. 标识符的命名规则：

可以由数字，字母，下划线和$组成，数字不能开头。

java中区分大小写

不能和关键字重名

没有长度限制

3. 标识符命名规范：

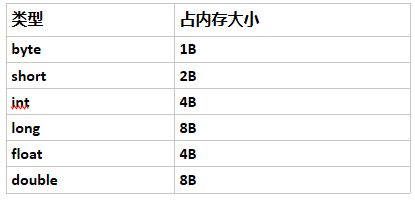


#### 2.3 基本数据类型

##### 2.3.1 数据类型

short 
long 
float 
double 
char 
boolean 
class 
interface 

##### 2.3.2 数据类型所占内存



##### 2.3.3 数据类型示例

//数据类型定义示例：

package scnu.edu.cn;

public class Demo04 {

public static void main(String[] args) {

byte bytValue01 = 7;

char chrValue01= 'h';

char chrValue02 = '爱';//在java中char其实是占2B的。

int intValue01 = 20;

float fValue01 = 3.14F;

double dValue01 = 3.14;

boolean boolValue01 = false;

boolean boolValue02 = true;

short shortValue01 = 30;

long lValue = 30000L;

System.out.println("bytValue01 = "+bytValue01);

System.out.println("chrValue01 = "+chrValue01);

System.out.println("chrValue02 = "+chrValue02);

System.out.println("intValue01 = "+intValue01);

System.out.println("fValue01 = "+fValue01);

System.out.println("dValue01 = "+dValue01);

System.out.println("boolValue01 = "+boolValue01);

System.out.println("boolValue02 = "+boolValue02);

System.out.println("shortValue01 = "+shortValue01);

System.out.println("lValue = "+lValue);

}

}

#### 2.4 数据类型的转换

Java 语言中的数据类型转换有两种：

***自动类型转换*** : 编译器自动完成类型转换，不需要在程序中编写代码。

***强制类型转换*** : 强制编译器进行类型转换，必须在程序中编写代码。

##### 2.4.1 隐式数据类型转换

转换规则：从存储范围小的类型到存储范围大的类型。  
具体规则为：（byte，short，char)→int→long→float→double

示例：

byte b1 = 100;

int n = b1;

##### 2.4.2 显式数据类型转换

转换规则:从存储范围大的类型到存储范围小的类型。

具体规则为：double→float→long→int→（short，char，byte）

示例代码：

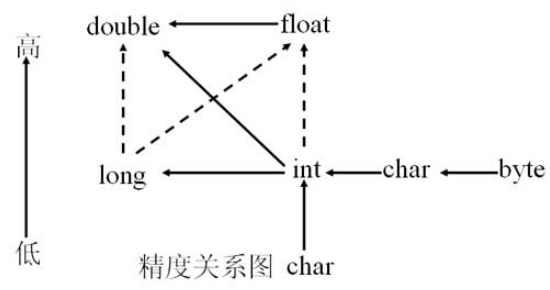
int n = 123;

byte b = (byte)n;

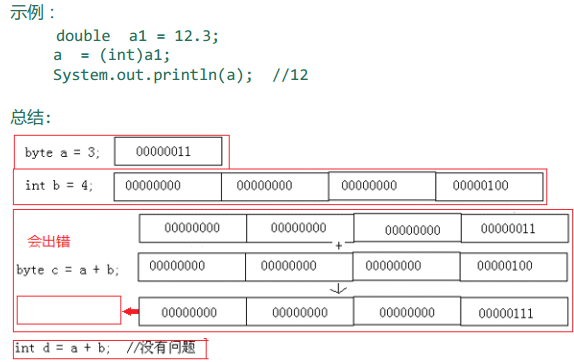
int m = 1234;

byte b1 = (byte)m;

##### 2.4.3 数据类型的精度关系



##### 2.4.4 题目讲解



示例：

public class Demo05 {

public static void main(String[] args) {

/\*

\* 自动类型转换，也叫自动类型提升

\*/

byte bytValue01 = 7;

int intValue01 = bytValue01;

int intValue02 = 34;

double dValue01 = intValue02;

float fValue01 = intValue02;

System.out.println(bytValue01);

System.out.println(intValue01);

System.out.println(intValue02);

System.out.println(dValue01);

System.out.println(fValue01);

//byte,char,short之间不能相互转化

//char chValue01 = bytValue01;//会报错

/\*

\* 强制类型转换

\*/

//第一题

float fValue02 = 3.14F;

float fValue03 = (float)3.14;

//第二题

byte b1 = 3;

int a = 4;

byte b2 = 4;

//byte b3 = (byte)(b1+b2) //int 到byte 出错

byte b3 = (byte)(b1 + b2);

byte b4 = 3+4; //不会出错。

//byte b5 = a+b1; //出错 int --》byte

//第三题

byte b6 = (byte)130;

System.out.println(b6); //-126

//字符串是最高级的

System.out.println("hello"+'a'+1); //helloa1

System.out.println('a'+1+"hello"); //98hello

System.out.println("5+5="+5+5); //5+5=55

System.out.println(5+5+"=5+5"); //10=5+5

}

}

byte b6 = (byte)130的分析过程：

A:首先取130的二进制：10000010 → 00000000 00000000 00000000 10000010

这个值，同时是原码，补码和反码。

B:强制转换，做截取操作，得到的是byte类型 → 10000010。

注意：这个值是补码，因为计算机是按补码运算的。

C:由补码求原码

符号位 数值位

补码 1 0000010

反码 1 0000001

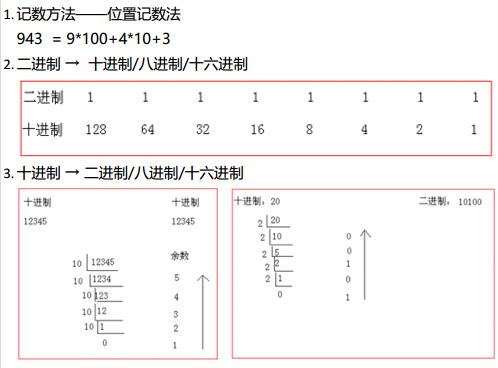
原码 1 1111110 → - 126

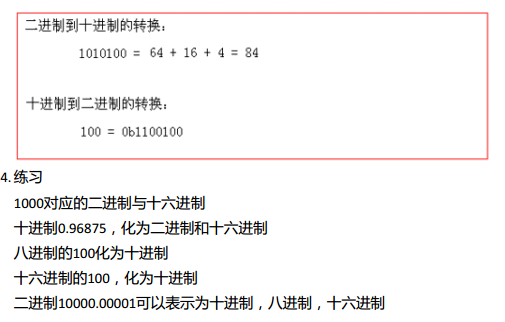
#### 2.5 习题

1. 练习数据类型的转换

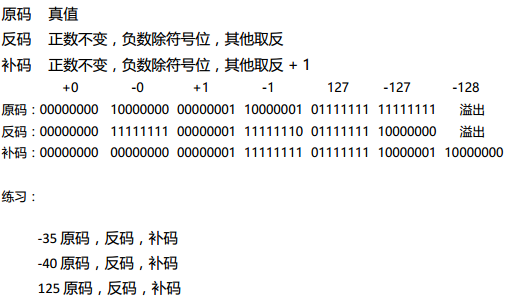
#### 2.6 补充知识

##### 2.6.1 进制转换





##### 2.6.2 数据表示法



### 程序控制语句

#### 3.1 运算符和表达式

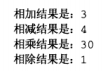
##### 3.1.1 算术运算符及表达式

算术运算符，就是小学学到的加减乘除运算，算术运算符包括（+ - \* / %）五种。

加，减，乘，除，余

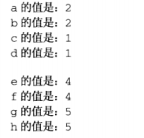
测试代码：

System.***out***.println(3+4);  
System.***out***.println(3-4);  
System.***out***.println(3\*4);  
*//4\*100/3 = 133,double/100*System.***out***.println( ((**double**)(4\*1000/3)/1000));  
*//4/3的余数*System.***out***.println(4%3);



除此之外，java中还有递增和递减的操作。++，--。

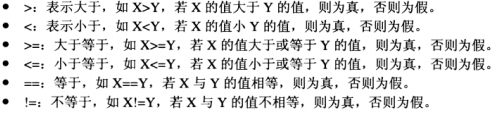
// int b = ++a; //b=4,a=4  
// a=a+1  
// b=a;  
 **int** c = a++;//c=3 a=4  
// c=a;  
// a=a+1;



##### 3.1.2 关系运算符及表达式

程序设计中，操作数之间会用到关系运算符，来判断真假。返回值是布尔类型。

Java中的关系运算符有六种：

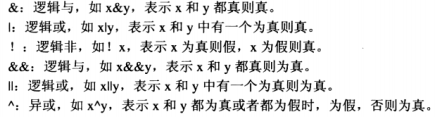


测试代码：

##### 3.1.3 逻辑运算符及表达式

逻辑运算符计算的对象是布尔类型，用于对多个条件进行判断。

Java中逻辑运算符六种：



注意：&&和||是短路操作，而&和|不是短路操作。

短路操作，意思就是需不需要对两边都进行计算，不进行计算就是短路。总是对两边进行计算就是不短路。

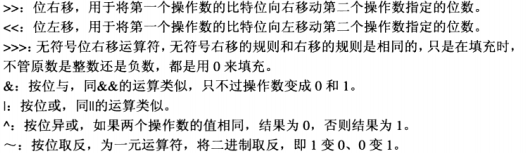
测试代码：

**boolean** x=**true**;  
**boolean** y=**false**;  
System.***out***.println(x&y);*//false*System.***out***.println(x|y); *//true |*System.***out***.println(!x); *//false*System.***out***.println(x^y); *//true*System.***out***.println(x&&y); *//false*System.***out***.println(x||y); *//true ||*

##### 3.1.4 位运算符及表达式

位运算符是按照二进制的形式对操作数进行运算。在位运算符运算中，操作数必须是整数。

Java中的位运算符一共有七个：



测试代码：

8>>2 –》2

8>>3 🡪1

##### 3.1.5 赋值运算符及表达式

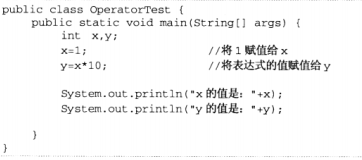
赋值运算符就像是给刚出生的小孩起名字一样，将这个孩子和名字划上等号。

Java中赋值运算符就一个：=（等号）

但是还有一些特殊情况，也习惯性的划分到赋值运算符中

+=，-=，\* =，/=，…

测试代码：



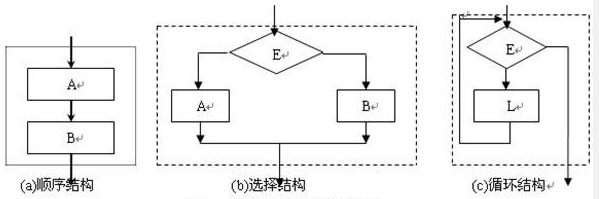
##### 3.1.6 总结



##### 3.1.7 运算符的优先级

#### 3.2 程序控制结构

一个人在思考的时候，总是进行逻辑思维的判断。Java中也有很多，比如顺序结构，分支结构，循环结构，使得程序既能顺序执行，又能够遇到某些条件进行跳转，还可以对某些操作重复执行。



#### 3.3 顺序结构

比如，办理银行业务，首先取一个号码，然后开始排队，不允许插队，直到业务办理完成为止。

测试代码：打印123

#### 3.4 条件结构

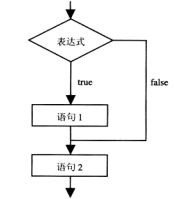
有时候，我们对待一件事情，会考虑，如果这样，结果会怎么样，如果那样，结果又会怎么样？

在程序中遇到这种情况，就需要条件结构语句进行处理：

##### 3.4.1 If的基本语法格式 ：

###### 1. if语法格式

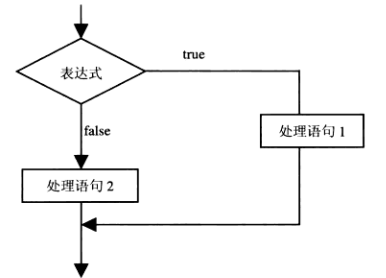
示例图：



**if**(expression){  
 *//语句1*}  
  
*//语句2*

###### 2. if-else语法格式

示意图：

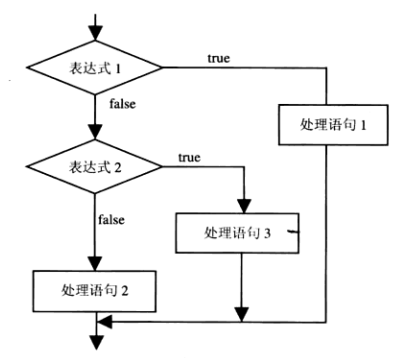


**if**(expression){  
 *//语句1*}**else**{  
 *//语句2*}

**if**(a<=b){  
 *//语句1* System.***out***.println(**"a<=b"**);  
}**else**{  
 System.***out***.println(**"a>b"**);  
}

###### 3. if-else if-else语法格式

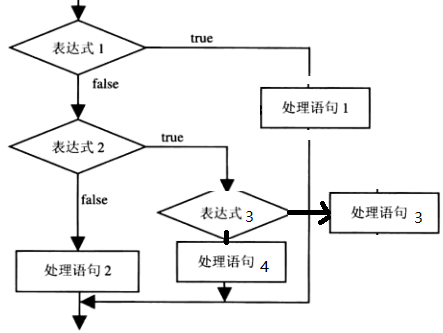
示意图：



**if**(expression){  
 *//语句1*}**else if**(expression){  
 *//语句3*}**else**{  
 *//语句2*}

###### 4. if语句的嵌套

示意图



**if**(expression){  
 *//1*}**else if**(expression2){  
 **if**(expression3){  
 *//3* }**else**{  
 *// 4* }  
}**else**{  
 *//2*}

##### 3.4.2 测试代码

判断两个人年龄的大小

（使用if格式和if-else格式）

**int** a = 3;  
**int** b =4;  
**if**(a>b){  
 System.***out***.println(**"a大"**);  
}**else**{  
 System.***out***.println(**"b大"**);  
}

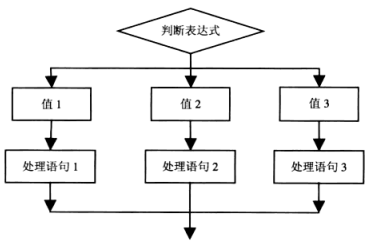
判断成绩优秀还是良好或者不及格

（使用if—else if—else格式以及嵌套）

Scanner input = **new** Scanner(System.***in***);  
 System.***out***.println(**"ENter a num:"**);  
 **int** score = input.nextInt();  
  
**if**(score>=90){  
 *//优秀* System.***out***.println(**"优秀"**);  
}**else if**(score<90&&score >=70){  
 *//良好* System.***out***.println(**"良好"**);  
}**else if**(score <70&&score >=60){  
 *//及格* System.***out***.println(**"及格"**);  
}**else**{  
 *//不及格* System.***out***.println(**"不及格"**);  
}

##### 3.4.3 switch语句

示意图：



测试代码：

打印星期

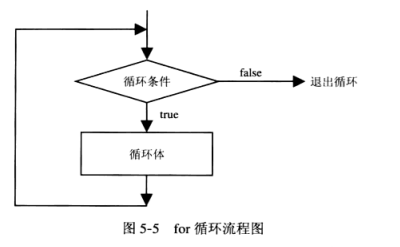
**switch** (value){  
 **case** 1:  
 System.***out***.println(**"1"**);  
 **break**;  
 **case** 2:  
 **case** 3:  
 **case** 4:  
 System.***out***.println(**"4"**);  
 **break**;  
 **case** 5:  
 System.***out***.println(**"5"**);  
 **break**;  
 **case** 6:  
 System.***out***.println(**"6"**);  
 *//没有break会执行7* **case** 7:  
 System.***out***.println(**"7"**);  
 **break**;  
 **default**:  
 System.***out***.println(**"你输入数有误"**);  
}

#### 3.5 循环结构

在程序中有时候需要对某一段代码进行反复的执行，这个时候就需要循环语句了

##### 3.5.1 for循环

示意图：



测试代码：

简单for循环 打印自然数

**for** (**int** i = 0; i < 10; i++) {  
 System.***out***.println(i);  
}  
*//int i =0;  
//i<10;  
//sout  
//i++  
//i<10  
//sout  
//...*

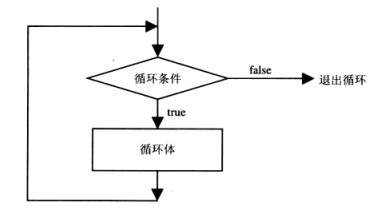
**int** i=0;  
**for**(;i<10;i++){  
   
}  
**for**(;i<10;){  
 i++;  
}  
**for**(;;){  
 **if**(i<10){  
 i++;  
 *//sout* }  
}

复杂for循环，嵌套。打印九九乘法表

**for**(**int** i =1;i<10;i++){  
 **for**(**int** j=1;j<=i;j++){  
 System.***out***.print(j+**" \* "**+i+**" = "**+(i\*j)+**" "**);  
 }  
 System.***out***.println();  
}

##### 3.5.2 while循环

示意图：

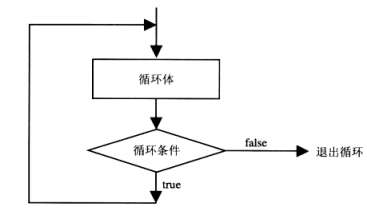


测试代码：

简单while循环，打印自然数

##### 3.5.3 do-while循环

示意图：



简单while循环，打印自然数

#### 3.6 中断控制

彩票开奖时，会使用摇号机随机选择几个号码，在选择完成后，机器就会停止。

程序中同样也有中断控制，比如循环到某个条件退出，或者循环到某个条件跳过。

##### 3.6.1 continue—跳过

在用于循环中时，可以中断本次循环，开始下一阶段的循环。

测试代码：

遇5跳过

##### 3.6.2 break

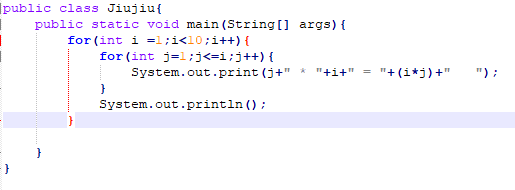
Break在switch中已经使用过了。作用时中断整个循环。

测试代码：

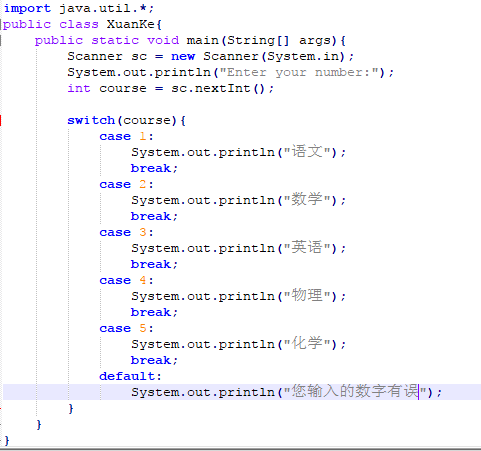
遇6中断。

##### 3.7 习题

1. 打印九九乘法表

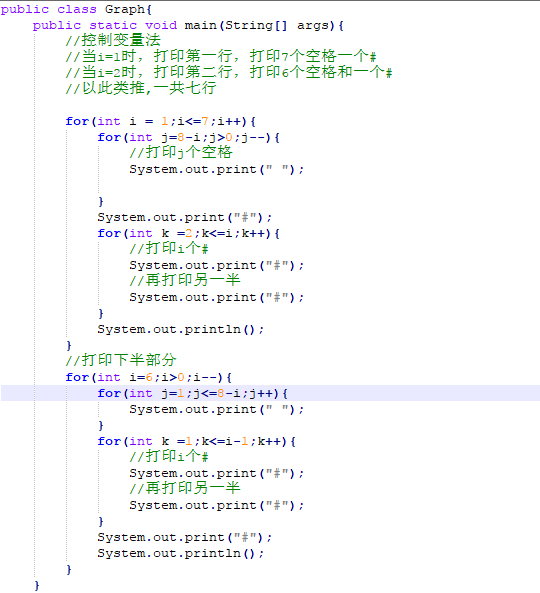


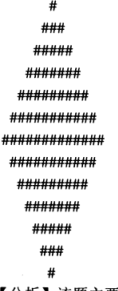
2. 编写一个程序，根据序号可以选课



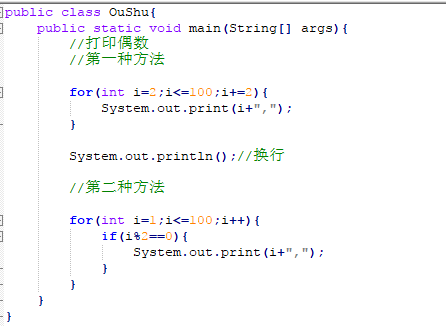


3. 编写一个程序，打印出如下图形状





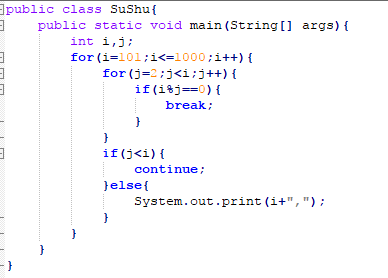
4. 打印出1-100之间的偶数



For(int i=0;i<=100;i+=2){}

For(int i=1;i<100;i++){if(i%2==0(sout)}

5. 判断101-1000之间有多少个素数



For(int i=101;i<1000;i++){}

6.打印出100-999之间的水仙花数，各个位上的三次方之和等于此数

例如153 = 1\*1\*1 + 5\*5\*5 +3\*3\*3=1+125+27=153

7. 一个球从100米高空落下，每次落地后反弹会原来高度的一半，小球十次落地时，共经过了多少米，第十次落地后反弹多高

8. 用一个锅炉烧热水，假设锅炉容量位100，如果流出的水量超过100，就往外流凉水。

### 数组

数组时java中经常用到的一个工具。它是用来存放具有相同数据类型的数据结构。比如，将几个标上数字的小球放在一个篮子里，篮子就是一个数组，小球就是数组中的元素

#### 4.1 数组基础

Java中的数组可以看作一种特殊的对象，也就是把数组看作是一个同种类型变量的集合，在同一个数组中所有的数据属于相同的数据类型，***每一个数据的名字都是数组名和下标组合起来表示。***

##### 4.1.1 创建数组

###### 1. 对数组声明

int[] arrInt;

int[] arrInt01,arrInt02;

###### 2. 为数组分配内存

arrInt = new int[5];

arrInt01=new int[3];

arrInt02=new int[4];

###### 3. 对数组进行初始化

arrInt[0]=1;

arrInt[1]=2;

arrInt[2]=3;

arrInt[3]=4;

arrInt[4]=5;

arrInt01[0]=1;

arrInt01[1]=2;

arrInt01[2]=3;

arrInt02[0]=1;

arrInt02[1]=2;

arrInt02[2]=3;

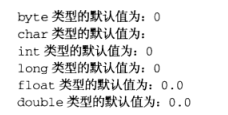
arrInt02[3]=4;

int[] array01 = new int[]{1,2,3,4,5};

int[] array02 =new int[3]{1,2,3};

int[] array03={1,2,3,4,5,6,7,8,9,0];

###### 4. 测试数组默认值



##### 4.1.2 数组的使用

注意：数组之间相互赋值，只是数组的标识符的指向改变。

测试代码：

class xxx{

public static void main(String[] args){

int[] array01 = new int[]{1,2,3,4,5};

int[] array02 =new int[]{7,8,9};

array01=array02;

System.out.println(array01.length);

for(int i :array01){

System.out.println(i);

}

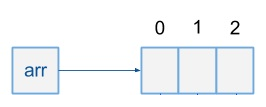
}

}

##### 4.1.3 数组在内存中的存放情况

int[] arr = new int[3];

在以上代码中，arr变量存放了数组对象的引用；如果你创建了空间大小为10的整形数组，情况是一样的，一个数组对象所占的空间在堆上被分配，然后返回其引用；

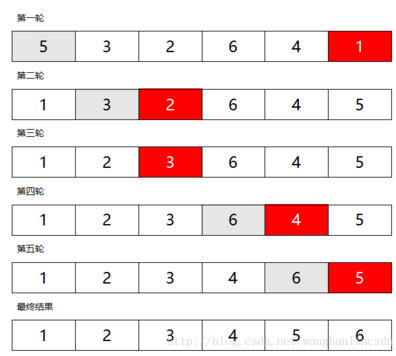


#### 4.2 数组排序

##### 4.2.1 选择排序法

这是最简单的排序，一遍一遍的遍历，每次选出最小的和第一个交换。

示意图：



测试代码

public static void main(String[] args){

int[] array = {1,23,45,66,2,332,8,5};

int min=0;

int temp=0;

for(int i=0;i<array.length;i++){

min = i;

for(int j=i+1;j<array.length;j++){

if(array[min]>array[j]){

min = j;

}

}

if(i!=min){

temp = array[min];

array[min]=array[i];

array[i]=temp;

}

}

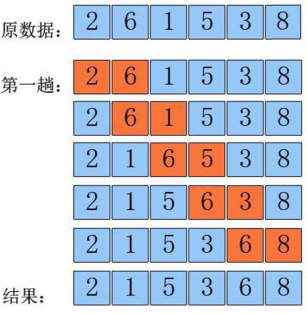
}



##### 4.2.2 冒泡排序法

冒泡排序法，是两两比较并相互交换。

示意图：



测试代码：

for(int i=0;i<array.length-1;i++){//循环次数

for(int j=0;j<array.length-i-1;j++){

if(array[j]>array[j+1]){

temp=array[j];

array[j]=array[j+1];

array[j+1]=temp;

}

}

}

#### 4.3 多维数组

##### 4.3.1 创建多维数组

int[][] two{

{1,2,3,4,5},

{2,3,4,5,6},

{3,4,5,6,7},

{4,5,6,7,8},

{5,6,7,8,9}

};

##### 4.3.2 多维数组的使用

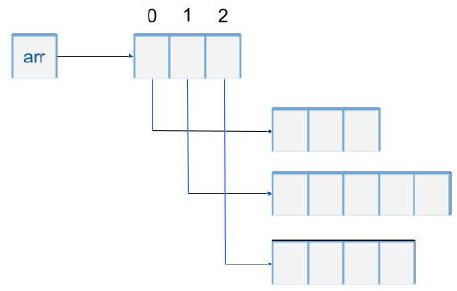
##### 4.3.3 多维数组在内存中的存放情况

int[ ][ ] arr = new int[3][ ];

arr[0] = new int[3];

arr[1] = new int[5];

arr[2] = new int[4];



#### 4.4 习题

1. 声明一个数组，将该数组中存在的重复元素去除，并重新进行排序。

（使用冒泡排序和选择排序）

2. 定义一个双精度的数组，并求其最大值，最小值，平均值以及总和

3. 将一个整型数组的索引为偶数的元素变为原来的两倍

### 字符串

### 面向对象程序设计

### 继承

### 多态

### 接口

### 异常处理

### 第十一章 常用类

### 第十二章 容器