Java Code基本理论

1. Java变量

>.什么是变量？

变量是指内存中开辟存储空间

>.变量能干什么？

用于存放运算过程中需要的数据。

! 程序 = 数据结构 + 算法。

>.int a = 5; int b = 6; int c = a + b;

变量a、b和c指代内存中的3块存储整数的空间，

c用来存储ad两个整数之和。

1. 变量的声明
2. 未经声明的变量不能使用

* 当需要使用一个变量时，必须对该变量声明

a = 5; //没有声明

* 变量的声明规范:

数据类型 变量名; int a; String username;

!不同数据类型决定了存储空间的结构。

* 变量没有声明，大多数是因为拼写错误造成。

1. 一条语句可以声明多个同类型的变量

* 如果多个变量的类型完全一致，则可以放在同一条语句中声明

int a =5;int b = 6; <==> int a = 6,b = 5;中间用逗号隔开。

int a,b= 6;//这里a没有被赋值，而b = 6;

1. 变量的命名
2. 命名需要符合标识符语法要求

* 变量的命名必须符合Java标识符的规则:

>.由字母、数字、”\_”和”$”组成;

>.首字母不能以数字开头;

>.Java大小写敏感

>.不能使用Java保留(goto)和关键字

>.中文可以使用，但不建议

!在编程中，安装路劲尽量不要出现中文。

* 以下变量名正确的是？

123 123go \_$$$$ \_$^j interface

中国 break $132go

1. 命名需要见名知意，且符合Java规范

* 命名一定需要见名知意
* 采用小驼峰(帕斯卡命名法) setName 、getName

1. 变量的初始化
2. 未经初始化的变量不能使用(重)

* Java语言规定变量在使用之前必须初始化

int a , b = 10;

int c = a + b;//a无值 不能使用

1. 在声明时初始化

* 格式: 变量类型 变量名称 = 初始值

int sum;

sum = 0;//在使用之前初始化

1. 在第一次使用变量时初始化
2. 变量的访问
3. 可以对变量的值进行存取、操作

* 变量是存放数据的空间，可以对其赋值、更改或操作。
* 对变量的操作可以理解为其存储的数据操作。

int a = 100; a = a + 200;//将变量a的值改变后重新赋值

1. 变量的操作必须与类型匹配(重)

* 变量在声明时指定了类型，Java编译器会检测对该变量的

操作是否类型匹配，若不匹配，会产生编译错误。

int salary; salary = 1500.00;//1500.00是double类型。

double d = 123.456; int n = d % 2;//d%2为double

int d = 5 / 3; //d= 1 int d = 5.0 / 3;//编译错误需要强转

1. Java基本类型

8种基本类型:

* 整型: byte 、 short 、int 、long
* 浮点型: float 、double
* 字符型: char
* 布尔型: boolean

8种基本类型的存储空间和使用场景:

|  |  |  |
| --- | --- | --- |
| 类型名称 | 字节空间 | 使用场景 |
| byte | 1字节(8位) | 存储字节数据(常用) |
| short | 2字节(16位) | 兼容性问题 |
| int | 4字节(32位) | 存储普通整数(常用) |
| long | 8字节(64位) | 存储的长整数且在值后加L或l(非必须) |
| float | 4字节(32位) | 存储浮点数且必须在值后加F或f |
| double | 8字节(64位) | 存储双精度浮点数 |
| char | 2字节(16位) | 存储一个字符且必须用单引号’’赋值 |
| boolean | 1字节(8位) | 存储逻辑变量(true、false) |

从小到大排序:byte-short-int-long-float-double

-char

1. int类型

最常用的数据类型，范围值为-2147483648~2147483647(-2^31~2^31-1)

* 整数直接量是int类

>.直接量(Iteral)就是直接写出的整数 int a = 100;

>.整数的直接量默认为int，如果直接写出的整数超过了int的存储范围，则会产生编译错误。

>.通常整数都是十进制写出的，但是有时候也会用16进制(0x或者0X),或8进制(以0开头)。

* 整型数据的除法运算中的取整

>.两个整数相除，会舍弃小数部分(不是4舍5入)。

* 运算时要防止溢出

两个整数进行运算时，其结果可能会超出整数的范围而溢出—重头开始。

byte b = 127; b = b+1; b为-128

1. long类型

在表示数值时，int不够存储，可以采用long，最大范围为:

(-2^63~2^63-1)。

注意:如果要表示long，需要以L或者l结尾。

* 使用long类型进行较大整数的运算

>.对于较大的整数运算(超出int范围)，使用long型。

//如果运算结果为long，超出int范围用long

long l5 = 1000\*365\*24\*360000000L;

//这里的结果为int，则超出范围，结果不符合逻辑

long l6 = 1000\*365\*24\*360000000;//运算时的溢出

* 通过时间毫秒数来存储日期和时间

>.JDK提供了一个方法System.currentTimeMillis()这个方法，返回的是1970年1月1日零点到此时此刻所经历的毫秒数，这个经常用于计时操作。

1. double类型

* 使用double进行浮点型的运算

>.浮点数，就是小数，float 和double。

>.double类型的精度值是float的两倍。

double s = 3.14 \* 10 \* 10; s为314.0

* 浮点数直接量是double类型

>.浮点数有两种表示方法:

通常写法: 3.0 3.14 314.0

科学计数法: 1.25E2 1.25e2 1.25E-2

>.默认的小数类型为double，如果需要用float，则在值后需要加F或f;

float f1 = 3.14f;

* double运算时会出现舍入误差(面试)

>.2进制中无法精确的表示1/10，就像10进制中无法表示1/3一样。因此有时候会出现舍入误差。导致无法精确运算。

>.如果需要精确运算则放弃float和double，采用BigDemical类。

1. char类型

本质是一个16位无符号的整数，这个值是对应字符的编码。

Java字符类型采用Unicode(世界通用的定长字符集)编码。

字符的直接量用单引号’中’括起来。也可以使用16进制的形式表示: ’\u4e2d’ <==> ‘中’

* 对char型变量赋值

>.字符直接量: ‘A’ ‘a’ ‘h’.

>.整数直接量: 范围为0~65535

>.Unicode形式:’\u4e2d’ ‘\u0041’16进制Unicode表示

* 使用转义字符

>.对于不方便输出的字符采用转义字符表示,

|  |  |
| --- | --- |
| 转义字符 | 含义 |
| ‘\n’ | 表示回车符 |
| ‘\r’ | 表示换行符 |
| ‘\\’ | 表示\ |
| ‘\” | 表示’(单引号) |
| ‘\”’ | 表示”(双引号) |

1. boolean类型

* boolean类型的关系运算
* >.boolean用于逻辑运算，表示某个条件是否成立，一般用于流程控制。
* >.boolean的值只有两个，一个为true，一个为false。
* >.关系运算指多个变量的大小是否相等等关系。
* Int age = 18;
* boolean age1 = age < 16;
* age1 为 false。

1. 基本类型间的转换(重)

自动类型转换(隐式类型转换)：小 → 大 long l = 12

强制类型转换： 大 → 小

格式：**(需要转换的类型)变量** float f = (float) 3.14;

* 强制转换时的精度丧失和溢出(面试)

int a = 100;

int b = 200;

long c = a+b;//300 a+b结果为int int → long 隐式

long l1 =1024l;

long i =(int) l1;// l1为long，long→int，强制

long l = 1024L \* 1024 \* 1024 \*4;

Int j = (int)l; // l为long,long →int,强制转换但超出int范围

j为溢出，其结果不符合逻辑，不做研究

double pi 3.141592653589;

float f = (float)pi;

f 为3.1415927，double→float时，可能造成精度丢失

* byte、char、short转为int

>.byte，char，从、short在运算时遵循如下规则：

int的直接量可以直接赋值给这三种类型，只要不超过其范围。

在运算时，这3种类型默认先一律转为int再运算。

* 数值运算时自动转换

多种类型进行运算时，会向较大类型转换。

double s1 = 80/100; //0.0,int型除以int型，为0.8，舍弃小数部分，将0.8赋值给s1，为0.0.

三．运算符和表达式

1.算数运算

- 使用%运算

>.java运算符通常除了 + - \* / 之外，还有包括取模和自增（++）和自减（--）运算

>.取模（取余）取余数 5 % 3 = 2 适用于整数、char、浮点数。

- 使用++和 -- 运算符

>.如果写在变量之前，在使用之前+1或-1.

>.如果写在变量之后，在使用之后+1或-1.

1. 关系运算

- 使用关系运算符

>. > < >= <= == !=

1. 逻辑运算

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 变量b1 | 变量b2 | b1 && b2 | b1 || b2 | !b1 |
| false | false | false | false | true |
| false | true | false | true | true |
| true | false | false | true | false |
| true | true | true | true | false |

与运算一假必假 或运算一真必真！

- &&

- ||

- !

-”短路逻辑”

>.Java逻辑运算遵循”短路逻辑”的原则：

与运算，当第一个操作数(表达式)为false时，就不会判断剩余操作数。

或运算，当第一个操作数为true时，就不会判断剩余操作数。

1. 赋值运算

- 使用 =

>.int a, b, c; a = b = c = 100 ;

c = 100，且整个表达式的值为100，将其赋值给b，同样 b = ( c = 100)整个表达式的值为100，再将其赋值给a。

- 使用扩展赋值运算符

+= -= \*= /= %=

>.int a = 100; a = a + 100;<==>a += 100;

1. 字符串连接符

-使用 +

>.可以实现字符串连接(相连不运算) +

1. 条件(三目运算)符

- 使用三目

>.boolean 表达式？ 表达式1：表达式2;

int a = 100 , b =200; int flag = a>b?1:-1; //-1

- 使用三目运算符的嵌套

>.条件运算可以嵌套使用，即格式中表达式1和表达式2也可以为三目运算。

四、条件分支

什么是分支结构？

>.任何复杂的顺序逻辑都可以通过“顺序”“分支”“循环”三种基本的程序结构实现

>.根据不同的条件运行不同的语句。

1.if语句

- if语句的执行逻辑

If(逻辑表达式){

语句1;

}

- if语句流程图

- if语句用于处理分支逻辑

收银台v2.0

- if语句块尽量不要省略{}

>.当if语句块中的语句只有一条时，可以不用{ }

>.但建议不要省略，可能造成代码变更时的错误

2.if else语句

- 执行逻辑

If(逻辑表达式){

语句体1；

}else {

语句体2；

}

- 流程图

- 处理分支逻辑

收银台3.0

3.else if语句

- if else的嵌套

if(逻辑表达式){

语句体1;

}else{

If(逻辑表达式){

语句体2;

}

}

- else if的执行逻辑

if(逻辑表达式){

语句体1;

}else if(逻辑体){

语句体2;

}......

else{

语句体3;

}

4.switch-case结构(重)

- 执行逻辑

>.switch-case是一种特殊的分支结构，可以根据一个整数表达式的不同取值，

从不同的分支入口执行。

switch(整数表达式){

case 整型常量值1:

语句体1;

case 整型常量值2:

语句体2;

......

case 整型常量值 n:

语句体n;

defult 整型常量值:

语句;

}

- switch-case和break关键字联合使用

>.break用来跳出当前case的。

- 处理分支

- switch-case的优势

>.效率更高，结构更加清晰

-总结switch-case

>.在整数表达式(byte、short、int、long、char)中，long不能作为表达式传入。

>.case和break联合使用，如果不加break，则程序会一直向下执行，

直到执行到有break的case为止。

>.从JDK1.7开始，switch-case结构支持字符串表达式。

switch (num) {

case 1:

System.out.println("显示全部登记记录");

break;

case 2:

System.out.println("查询登记记录");

break;

case 0:

System.out.println("退出");

break;

}

作业：

1. 输出两个int值的最大值 HomeWork1

用户输入两个int类型的数值，比较打印后的结果

1. 3个数字的排序问题？ HomeWork2

用户输入3个数字，排序后打印。

1. 判断闰年(if else) HomeWork3
2. 完成猜数字。
3. 循环结构

什么是循环？

循环是程序设计语言中反复执行某些代码的一种计算机处理过程，是一组相同或相似的语句被

有规律的重复性执行。

循环的要素：循环体(相同或相似的语句) 循环条件(规律)

1.while语句  
 - 执行逻辑

while(boolean表达式){循环体}

- 处理循环逻辑

- 使用break跳出循环

>.break 跳出所有循环

2.do-while语句

- 执行逻辑

do{

循环体

}while(boolean表达式)

- 处理循环逻辑

- while和do-while的区别(重)

do-while无论条件满足与否都会执行一次循环体

3.for语句(重)

- 考虑循环问题的相同之处

计算0-100和

计算10！

for循环的3要素：循环变量、循环条件、变量改变

- 执行逻辑

for(初始化变量1;循环条件2;变量的改变4)

{

循环体3

}

- 用于处理固定次数循环

- 三个表达式的特殊用法

>.循环变量和变量的改变的位置可以移动。

>.for( ; ; ){} --死循环

>.for循环的3要素可以多样化(3要素的参数可以为多个，且数值随意)

- 在循环中使用break语句

>.在for循环使用break语句时，需要配合条件判断去使用

- 在循环使用continue语句

4.循环问题

- 循环问题定义-“当”循环

>.while

- “直到”循环

>.do -while

- 固定次数问题

>.for

- 循环嵌套问题

>.9\*9乘法表

总结:嵌套循环：外层循环控制行数，内层循环控制列数。

作业：

1. 计算9，99，999，9999，...999999999的和。
2. 计算1+1/2+1/3+...+1/n的和，n是用户输入的。
3. 控制台绘制杨辉三角形，行数为n，n是用户输入的。

扩展：

空心菱形、等腰三角形、空心三角形

1. 数组

>.程序 = 算法(分支、循环) + 数据结构(数组)

>.什么是数据结构?

把数据按照特定的某种结构来保存，设计合理的数据结构是解决问题的关键。

>.什么是数组?

相同类型的元素组成的集合，最基本的一种数据结构。

按照线性排列的，除第一个元素没有前驱元素外，及最后一个元素没有后继元素外，其他元素都有一个前驱和后继元素，这样的顺序我们称为线性顺序。

1.数组的定义

- 基本类型数组的定义

>.数据类型[] 数组名 = new 数据类型[数组的长度];<==> int arr[]

int[] arr = new int[5]; //长度：5 类型：int 下标：0~4 元素为0,0,0,0,0.

>.定义数组的注意事项：

- 具备确切的数据类型

- 整体的数组名字 -- 帕斯卡命名法

- 不能省略[]

>.int[] arr;问有没有内存空间? 没有。 new之后才会分配指定大小内存空间

>.int[] arr和int arr[]两种写法都可以。

2.数组的初始化

- 初始化数组

>. 基本类型的数组创建后，其元素初始值：

整型：0 浮点型：0.0 char：空字符 Boolean：false

>. 可以在定义时直接对其初始化

int [] arr = {0,10,20,30,40}; //长度：5 arr[4]:40

这种写法只能用于初始化的时候写，不能分开写。

int[] arr; arr = {0,10,20,30,40}

>. 可以使用以下方式对数组进行声明初始化

int[] arr = new int[]{0,10,20};

总结：定义数组的三种方式：。

//第一种

int[] arr1 = new int[5];

//第二种

int arr2[] = {0,10,20,30,40};

//第三种

int[] arr3 = new int[] {0,10,20,30}

3.数组的访问

- 获取数组的长度

int len = arr.length;

- 通过下标访问数组元素

下标从0开始，最大为length -1

- 遍历数组元素

>.遍历数组元素通常使用for循环，循环变量作为数组的下标。

4.数组的复制

- System.arrayCopy方法用于数组的复制

>.public static void arrayCopy(Object src,int srcpos,Object dest,int destpos,int length)

Object src: 源数组

int srcpos: 源数组中的起始位置

Object dest： 目标数组

int destpos: 目标数组的起始位置

int length: 复制数组的元素的数量--复制的长度

int a[] = {10,20,30,40,50} ;

int a1[] = new int[6];

System.arrayCopy(a,1,a1,0,4);

>.复 制 的 长 度 不 能 越 界( 大 于 其 存 储 长 度)， 否则产生ArrayIndexOutOfBoundsException(数组下标越界异常)

- Arrays.copyof方法用于数组的复制

>.Arrays.copyOf(数组，copy的长度);

int[] a = {10,20,30,40,50};

int[] a1 = Arrays.copyOf(a,6);

>.当复制的长度小于源数组长度，则截取

当复制的长度大于源数组的长度，则扩容

- 数组的扩容

>.int a = new int [length];

a = Arrays.copyOf(a,length+1);将源数组复制给自己并将长度变化，达到扩容的效果。

作业：

1. ）定义一个长度为10的数组，数组元素为0-100内的随机数字。

找出最大值放在最后一位，并删除最后一个元素。

5.数组的排序

>.按照特定的规则对数组元素进行排列

>.对于较多的元素排序的算法至关重要，自行查阅资料整理排序算法。

>.常用排序算法：插入排序、冒泡排序、快速排序等等。

- 冒泡排序算法

>.比较相邻两个元素，如果满足条件，则

- Arrays.sort方法

>.JDK提供了一个方法Arrays.sort方法用于数组的排序

作业：

1> 定义n个长度的数组，元素为100内随机数，使用冒泡排序算法完成排序并打印。

2> 找出2~n范围内所有质数，并每5个换行打印

6.引用类型数组

七．方法(重要)

>.封装特定的逻辑功能 sort；排序 nextInt：输入

>.复用性-简化代码

>.减少代码的重复量，便于程序的维护。

1.方法的定义

- 定义(函数、过程)的功能

>.五个要求：

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

修饰词：public static...修饰词根据需求选择合适的关键词语定义，如果没有，则可以不写。

返回值类型：void... (1.)具体的返回值类型且代码方法体中需要使用return返回具体的返回值

(2.)无返回值用void

方法名： main

参数列表：String[] args ... 参数列表为形式参数，是逻辑体需要用到的变量声明。

方法体：逻辑体

- 定义参数和返回值

>.在方法定义时，需要声明该方法需要的参数变量。

>.在方法调用时，将实际的参数值传递给参数变量

必须保证传递的参数类型和个数符合方法的声明。

>.方法调用完毕后返回的值。

2.方法的调用

- return 语句

>.return; ---返回值类型为void的方法可以使用此结构返回返回值。

- 调用方法参数传递

作业：

1.自定义排序算法并测试，将代码拷贝到笔记中。

2.自定义方法，产生n个随机元素的数组。

总结：1.规定返回值类型

-有返回值：return语句返回返回值，规定其返回值类型

-无返回值：不要return语句，并且返回值类型void

2.定义方法名字 --- 小驼峰命名法

3.设置参数列表 --- 可以为多个可以没有

4.核心代码逻辑