

第一周学习笔记

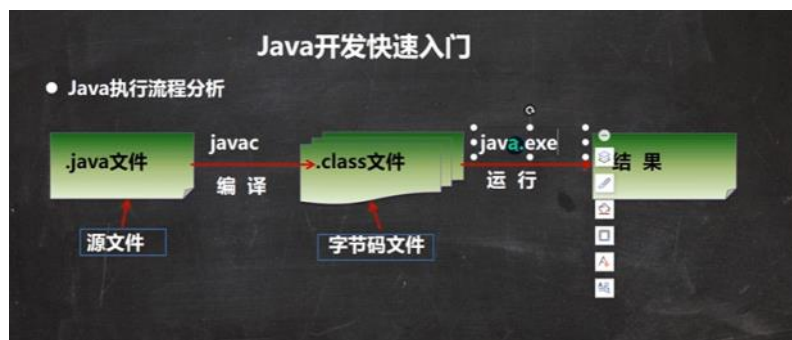
2021年4月21日 20:30

P1-P12

2021年4月19日

13:29

学习并使用 `javac`、`java` 命令，了解两个命令各自的使用场景



编译与运行

主讲人：韩顺平老师

Java开发快速入门

● 什么是编译

```
javac Hello.java
```

1. 有了java源文件，通过编译器将其编译成JVM可以识别的字节码文件。
2. 在该源文件目录下，通过javac编译工具对Hello.java文件进行编译。
3. 如果程序没有错误，没有任何提示，但在当前目录下会出现一个Hello.class文件，该文件称为字节码文件，也是可以执行的java的程序。

Java开发快速入门

● 什么是运行

1. 有了可执行的java程序(Hello.class字节码文件)
2. 通过运行工具java.exe对字节码文件进行执行,本质就是.class装载到jvm 机执行

● java程序开发注意事项

对修改后的Hello.java源文件需要重新编译，生成新的class文件后，再进行执行,才能生效。

主讲人：韩顺平老师

Java开发注意事项和细节说明

1. Java源文件以 .java 为扩展名。源文件的基本组成部分是类（class），如本类中的Hello类。
2. Java应用程序的执行入口是main()方法。它有固定的书写格式：
`public static void main(String[] args) {...}`
3. Java语言严格区分大小写。
4. Java方法由一条条语句构成，每个语句以“`;`”结束。
5. 大括号都是成对出现的，缺一不可。[习惯，先写 `{}` 再写代码]
6. 一个源文件中最多只能有一个public类。其它类的个数不限。[演示]
7. 如果源文件包含一个public类，则文件名必须按该类名命名！
8. 一个源文件中最多只能有一个public类。其它类的个数不限，也可以将main方法写在非public类中，然后指定运行非public类，这样入口方法就是非public的main方法

P19

1. 文件名按照public类命名

◦ ◦ ◦ ◦

P20

快速学习技术或知识点经验

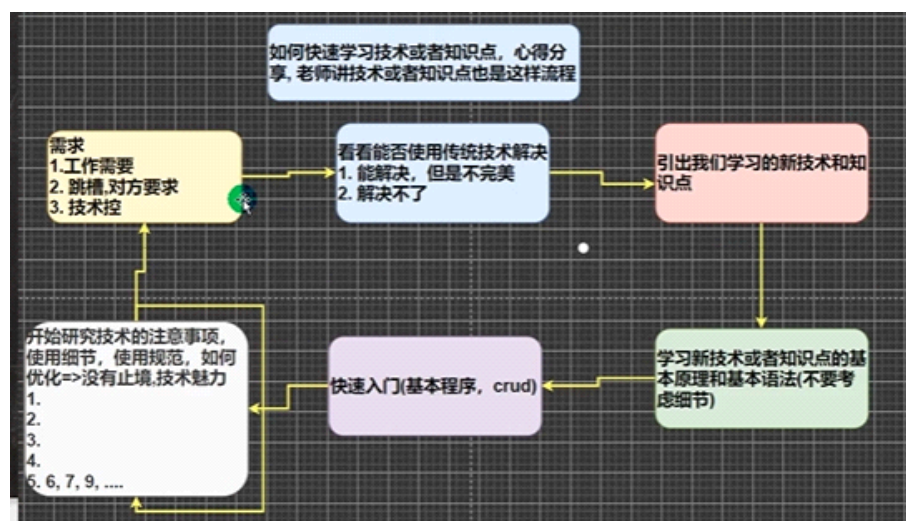
1.需求驱动：学习任务

2.学习技术的基本语句和基本语法（不考虑细节）

3.快速入门案例（基本案例，增删改查）

4.开始研究技术的使用细节、规范与如何优化

优化⇒永无止境、技术魅力



P21转义字符-22易犯错误

2021年4月21日 18:15

```
7 // \t : 一个制表位, 实现对齐的功能
8 System.out.println("北京\t天津\t上海");
9 // \n : 换行符
10 System.out.println("jack\nsmith\nmary");
11 // \\ : 一个\
12 System.out.println("C:\\Windows\\System32\\cmd.exe");
13 // \" : 一个"
14 System.out.println("老韩说:\"要好好学习java,有前途\"");
15 // \' : 一个'
16 System.out.println("老韩说:\'要好好学习java,有前途\'");
17
18 // \r : 一个回车 System.out.println("韩顺平教育\r北京");
19 // 解读
20 // 1. 输出 韩顺平教育
21 // 2. \r表示回车
22 System.out.println("韩顺平教育\r\n北京"); // 北京平教育
23
24 }
25 }
```

初学java易犯错误

1. 找不到文件

```
D:\javacode>javac ChangeCha.java
javac: 找不到文件: ChangeCha.java
用法: javac <options> <source files>
-help 用于列出可能的选项
```

解决方法:源文件名不存在或者写错, 或者当前路径错误

2. 主类名和文件名不一致

```
D:\javacode>javac ChangeCharExer01.java
ChangeCharExer01.java:2: 错误: 类ChangeCharExer0是公共的,
文件中声明
public class ChangeCharExer0 {
```

解决方法:声明为public的主类应与文件名一致, 否则编译失败

3. 缺少分号

```
D:\javacode>javac ChangeCharExer01.java
ChangeCharExer01.java:7: 错误: 需要';'
System.out.println("书名\t
```

解决方法: 编译失败, 注意错误出现的行数, 再到源代码中指定位置改错。

1. 举例

1. 1-» l
2. 0-» o
3. 中英文混用; ; " "
4. 拼写错误: void-» viod
5. 不好修改的是业务、环境错误

P23注释P24多行p25文档注释

2021年4月21日 20:31

注释 (comment)

用于注解解释程序的文字就是注释，注释提高了代码的阅读性（可读性）；注释是一个程序员必须要具有的良好编程习惯。将自己的思想通过注释先整理出来，再用代码去体现。

- java中的注释类型

1. 单行注释 `//...//`
2. 多行注释 `/*...*/`
3. 文档注释 `/**`

```
    * * *  
    */
```

- 使用细节

- 1) 被注释的文字，不会被JVM(java虚拟机) 解释执行
- 2) 多行注释里面不允许有多行注释嵌套

文档注释：

```
/**  
  
    * * *  
    */
```

注释内容可以被JDK提供的工具javadoc所解析，生成一套以网页文形式体现的该程序的说明文档，一般写在类

- 基本格式
- 如何生成对应的文档注释
- 应用实例

Javadoc -d 文件夹名 -xx -yy Dem03.java java文件

java标签名{@author,@version etc...}

P25代码规范p26dos原理p27路径详解

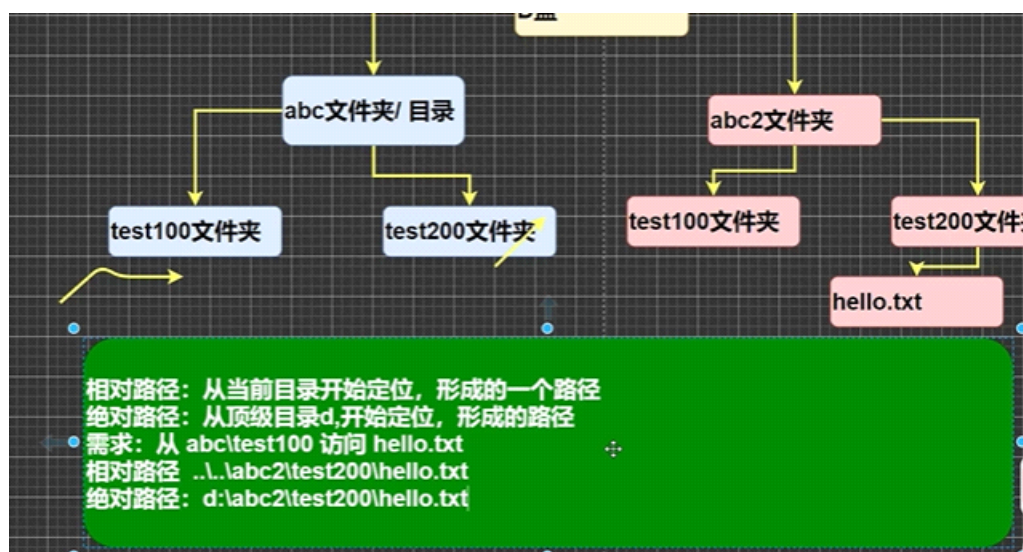
2021年4月22日 21:03

Java代码规范

1. 类、方法的注释，要以javadoc的方式来写。
2. 非JavaDoc的注释(单行与多行)，往往是给代码的维护者看的，着重告诉读者为什么这样写，如何修改，注意什么问题等
3. 使用tab操作，实现缩进，默认整体向右边移动，用shift+tab整体向左移
4. 运算符和=两边习惯性各加一个空格。比如：2 + 4 * 5 + 345 - 89
5. 源文件使用utf-8编码
6. 行宽度不要超过80字符
7. 代码编写次行风格和行尾风格

DOS命令(了解)

- DOS介绍
Dos: Disk Operating System 磁盘操作系统, 简单说一下windows的目录结构。[原理图]
- 相关的知识补充: 相对路径, 绝对路径
- 常用的dos命令
 1. 查看当前目录有什么
`dir` `dir d:\abc2\test200`
 2. 切换到其他盘下: 盘符号 `cd`
案例演示: 切换到 c盘 `cd /D c:`
 3. 切换到当前盘的其他目录下 (使用相对路径和绝对路径演示)
案例演示: `cd d:\abc2\test200` `cd ../../abc2\test200`
 4. 切换到上一级:
案例演示: `cd ..`
 5. 切换到根目录: `cd \`
案例演示: `cd \`



P29-31常用dos命令

2021年4月22日 21:42

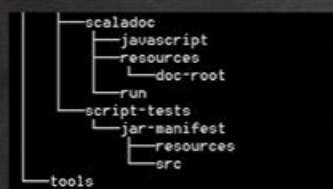
- dir: 查看当前目录包含内容
- cd (change directory) 切换

/D改变根目录 (d盘换c盘)

```
: \javacode>cd /D c:  
: \>
```

- 相对, 绝对
- 案例演示: `cd d:\abc2\test200 cd ../../abc2\test200`

6. 查看指定的目录下所有的子级目录



7. 清屏

8. 退出DOS

9. 说明: 因为小伙伴后面使用dos 非常少, 所以对下面的几个指令, 老韩给大家演示下, 大家了解即可 (md,rd,copy,del,echo,type,move)

6.tree tree c:\

7.cls

8.exit

9.md创建文件rd删除目录copy拷贝文件del删除文件echo输入内容到文件

type/move剪切

P31-33作业+34内容梳理

2021年4月22日 22:03

本周作业。

1. 编写hello, world程序 [Homework01.java]
2. 将个人的基本信息（姓名、性别、籍贯、住址）打印到控制台上输出。各条信息分别占一行。 [Homework02.java]
3. JDK,JRE,JVM的关系 [Homework03.java]
4. 环境变量path配置及其作用 [Homework04.java]
5. Java编写步骤 [Homework05.java]
6. Java编写7个规范 [Homework06.java]
7. 初学者java易犯错误 [Homework07.java]

之后的作业先自己写，再看视屏讲解

为什么需要变量

- 变量是程序的基本组成单位

不论是使用哪种高级程序语言编写程序,变量都是其程序的基本组成单位, 比如:

//变量有三个基本要素(类型+名称+值)

```
class Test{
    public static void main(String []args){
        int a=1;//定义了一个变量, 类型int整型,名称a,值1
        int b=3;//定义了一个变量, 类型int整型,名称b,值3
        b=89;//把89值赋给 b变量
        System.out.println(a);//输出a变量的值
        System.out.println(b);//输出b变量的值
    }
}
```

变(变化)量(值)的介绍

- 概念

变量相当于内存中一个数据存储空间的表示, 你可以把变量看做是一个房间的门牌号, 通过门牌号我们可以找到房间, 而通过变量名可以访问到变量(值)。

- 变量使用的基本步骤

1) 声明变量

● int a;

2) 赋值

a = 60; //应该这么说: 把60赋给a

3) 使用 System.out.println(a);

//也可以一步到位[int a = 60; 通常我们是一步完成]

变量快速入门

变量使用入门案例

看演示并对代码进行说明, 演示记录 人的信息的代码

```
//1. 定义变量
int age = 20;
double score = 88.6;
char gender = '男';
String name = "jack";
```

变量使用注意事项

1. 变量表示内存中的一个存储区域 [不同的变量, 类型不同, 占用的空间大小不同, 比如: int 4 个字节, double 就是 8个字节]
2. 该区域有自己的名称[变量名]和类型[数据类型]
3. 变量必须先声明, 后使用, 即有顺序
4. 该区域的数据可以在同一类型范围内不断变化
5. 变量在同一个作用域内不能重名
6. 变量=变量名+值+数据类型, 这一点请大家注意。变量三要素

P39 + 使用 p40数据类型

2021年4月23日 16:26

程序中 + 号的使用

1. 当左右两边都是数值型时，则做加法运算
2. 当左右两边有一方为字符串，则做拼接运算
3. 课堂测试题 1min

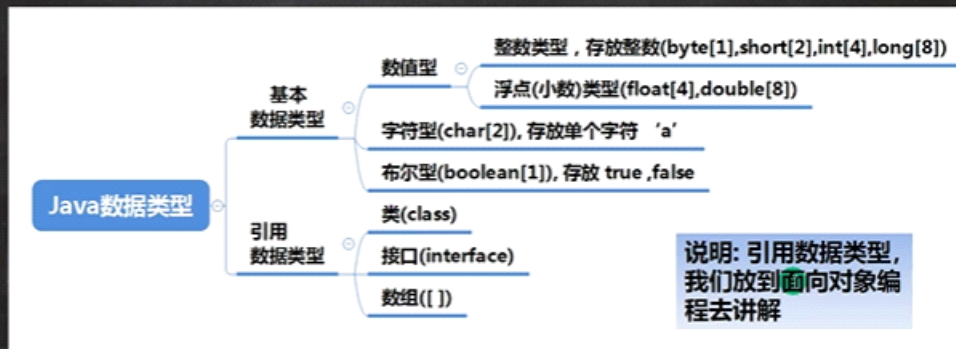
#下面代码输出什么?

```
System.out.println(100 + 98); //198
System.out.println("100" + 98); //10098

System.out.println(100 + 3 + "hello"); //103hello
System.out.println("hello" + 100 + 3); //hello1003
```

数据类型

每一种数据都定义了明确的数据类型，在内存中分配了不同大小的内存空间(字节)。



✓ 上图说明

1. java 数据类型分为两大类 基本数据类型, 引用类型
2. 基本数据类型有8中 数值型 [byte, short, int, long, float, double] char, boolean
3. 引用类型 [类, 接口, 数组]

字符串String属于类

背诵下来

P41-45数值型（整型、浮点数使用）

2021年4月23日 16:33

整数类型

● 整型的类型

类 型	占用存储空间	范围
byte [字节]	1字节	-128 ~ 127 为啥存放的范围是这个=>二进制(二进制我们详解)
short [短整型]	2字节	$-(2^{15}) \sim 2^{15}-1$ -32768 ~ 32767
int [整型]	4字节	$-2^{31} \sim 2^{31}-1$ -2147483648 - 2147483647
long [长整型]	8字节	$-2^{63} \sim 2^{63}-1$

整数类型

● 整型的使用细节 IntDetail.java

1. Java各整数类型有固定的范围和字段长度，不受具体OS[操作系统]的影响，以保证java程序的可移植性。
2. Java的整型常量（具体值）默认为 int 型，声明long型常量须后加 `L` 或 `l`
3. java程序中变量常声明为int型，除非不足以表示大数，才使用long
4. bit: 计算机中的最小存储单位。byte: 计算机中基本存储单元, 1byte = 8 bit。 [二进制再详细说，简单举例一个 byte 3 和 short 3]

思考题: long 类型，有几个 bit

[8 * 8 = 64 bit]

long n = 3; // 内存中

浮点类型

● 浮点型的分类

类 型	占用存储空间	范围
单精度float	4字节	-3.403E38 ~ 3.403E38
双精度double	8字节	-1.798E308 ~ 1.798E308

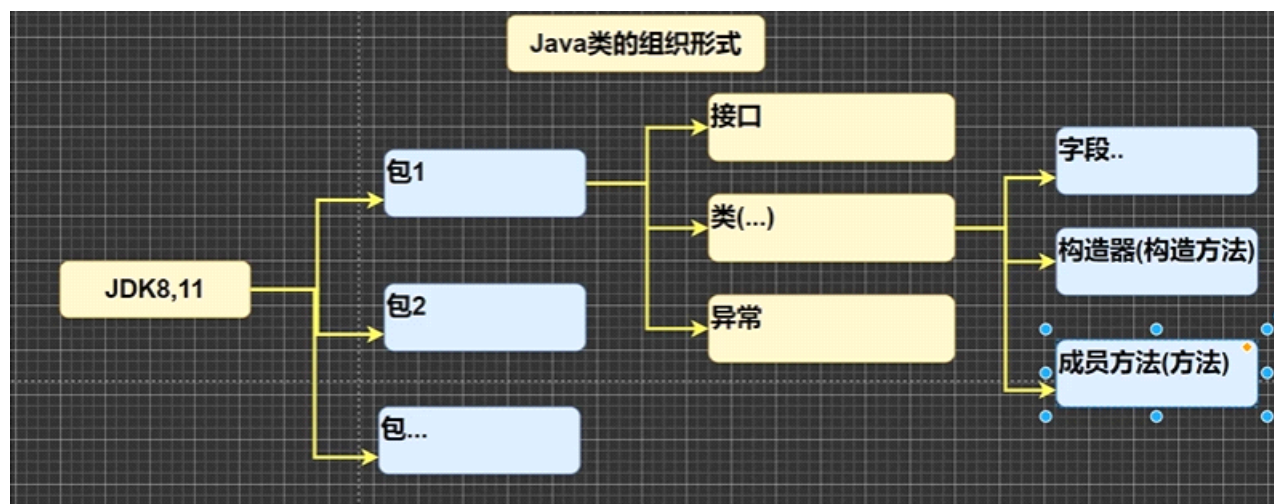
● 说明一下

1. 关于浮点数在机器中存放形式的简单说明, 浮点数=符号位+指数位+尾数位
2. 尾数部分可能丢失，造成精度损失(小数都是近似值)。

浮点类型

● 浮点型使用细节 FloatDetail.java

1. 与整数类型类似，Java 浮点类型也有固定的范围和字段长度，不受具体OS的影响。[float 4 个字节 double 是 8个字节]
2. Java 的浮点型常量(具体值)默认为double型，声明float型常量，须后加 `f` 或 `F`
3. 浮点型常量有两种表示形式
十进制数形式: 如: 5.12 512.0f .512 (必须有小数点)
科学计数法形式: 如: 5.12e2 [5.12*10的2次方] 5.12E-2 [5.12/10的2次方]
4. 通常情况下，应该使用double型，因为它比float型更精确。[举例说明]
double num9 = 2.1234567851;
float num10 = 2.1234567851F;
double num7 = 2.7;
double num8 = 8.1 / 3;
if (Math.abs(num7 - num8) < 0.00001) {
System.out.println("相等~~~");
}
5. 浮点数使用陷阱: 2.7 和 8.1 / 3 比较



按照包» 类» 方法/或者直接检索

字符类型(char)

- 字符类型使用细节
 1. 字符常量是用单引号(' ')括起来的单个字符。例如：
char c1 = 'a'; char c2 = '中'; char c3 = '9';
 2. Java中还允许使用转义字符 '\ ' 来将其后的字符转变为特殊字符型常量。例如：char c3 = '\n'; // '\n'表示换行符
 3. 在java中，char的本质是一个整数，在输出时，是unicode码对应的字符。
<http://tool.chinaz.com/Tools/Unicode.aspx>
 4. 可以直接给char赋一个整数，然后输出时，会按照对应的unicode 字符输出 [97-》a]
 5. char类型是可以进行运算的，相当于一个整数，因为它都对应Unicode码。

```

// Example code from the slide
char n1 = 'a' + 'b';
char char1 = 'a'+2;
System.out.println(n1);
System.out.println(char1);

char c1 = 'a';
char c2 = '\t';
char c3 = '韩';
char c4 = 97;
System.out.println((int)c1);
System.out.println((int)c2);
System.out.println((int)c3);
System.out.println((int)c4);
    
```

字符类型(char)

- 字符类型本质探讨
 1. 字符型 存储到 计算机中，需要将字符对应的码值（整数）找出来，比如'a'
存储：'a' ==> 码值 97 ==> 二进制(110 0001) ==> 存储
读取：二进制(110 0001) ==> 97 ==> 'a' ==> 显示
 2. 字符和码值的对应关系是通过字符编码表决定的(是规定好)
- 介绍一下字符编码表 [sublime测试]

ASCII (ASCII 编码表 一个字节表示，一个128 个 字符，实际上一个字节可以表示256个字符,只用128个)

Unicode (Unicode 编码表 固定大小的编码 使用两个字节来表示字符，字母和汉字统一都是占用两个字节，这样浪费空间)

utf-8 (编码表，大小可变的编码 字母使用1个字节，汉字使用3个字节)

gbk (可以表示汉字，而且范围广，字母使用1个字节，汉字2个字节)

gb2312 (可以表示汉字，gb2312 < gbk)

big5 码(繁体中文，台湾，香港)

Unicode编码介绍(了解)

1. Unicode的好处：一种编码，将世界上所有的符号都纳入其中。每一个符号都给予一个独一无二的编码，使用 Unicode 没有乱码的问题。
2. Unicode 的缺点：一个英文字母和一个汉字都占用2个字节，这对于存储空间来说是浪费。
3. 2的16次方是 65536，所以最多编码是65536个字符。
4. 编码0-127的字符是与ASCII的编码一样.比如 'a' 在ASCII码是 0x61，在 unicode码是 0x0061，都对应97. 因此 Unicode码兼容 ASCII码。

UTF-8编码介绍(了解)

1. UTF-8 是在互联网上使用最广的一种 Unicode 的实现方式 (改进)
2. UTF-8 是一种变长的编码方式。它可以使用 1-6 个字节表示一个符号，根据不同的符号而变化字节长度。
3. 使用 大小可变的编码 字母占1个字节，汉字占3个字节

布尔类型：boolean

● 基本介绍

1. 布尔类型也叫boolean类型，boolean类型数据只允许取值true和false，无null
2. boolean类型占1个字节。
3. boolean 类型适于逻辑运算，一般用于程序流程控制[这个后面会详细介绍]：
 - ✓ if条件控制语句；
 - ✓ while循环控制语句；
 - ✓ do-while循环控制语句；
 - ✓ for循环控制语句

● 案例演示：

Boolean01.java

```
boolean pass = true;
if (pass) {
    System.out.println("通过考试");
} else {
    System.out.println("没有通过考试~");
}
```



基本数据类型转换

● 自动类型转换

✓ 介绍

当java程序在进行赋值或者运算时，精度小的类型自动转换为精度大的数据类型，这个就是**自动类型转换**。

✓ 数据类型按精度(容量)大小排序为(背，规则)



✓ 看一个基本案例 **AutoConvert.java**

```
int a = 'c';//ok
double d = 80;//ok
```

基本数据类型转换

● 自动类型转换注意和细节

1. 有多种类型的数据混合运算时，系统首先自动将所有数据转换成容量最大的那种数据类型，然后再进行计算。

2. 当我们把精度(容量)大 的数据类型赋值给精度(容量)小 的数据类型时，就会报错，反之就会进行自动类型转换。

3. (byte, short) 和 char之间不会相互自动转换。

```
byte b = 10;
char c = b;
```

```
double d3 = 100;
int num3 = 1.1;
```

4. byte, short, char 他们三者可以计算，在计算时首先转换为int类型。

5. boolean 不参与转换

6. 自动提升原则：表达式结果的类型自动提升为 操作数中最大的类型

```
byte b = 10;
char c;
short s;
short s;
```

看老师演示 **AutoConvertDetail.java**

```
//boolean b = 1;
boolean b = true;
int num1 = (int)b;
```


P55-57强制类型转换

2021年4月24日

22:35

基本数据类型转换

● 强制类型转换

✓ 介绍

自动类型转换的逆过程，将容量大的数据类型转换为容量小的数据类型。使用时要加上强制转换符 ()，但可能造成精度降低或溢出，格外要注意。

● ✓ 案例演示 ForceConvert.java

```
int i = (int)1.9;  
System.out.println(i);
```

```
int j = 100;  
byte b1 = (byte)j;  
System.out.println(b1);
```

基本数据类型转换

✓ 强制类型转换细节说明

// ForceConvertDetail.java

1. 当进行数据的大小从 大 → 小，就需要使用到强制转换
2. 强转符号只针对于最近的操作数有效，往往会使用小括号提升优先级

```
//int x = (int)10*3.5+6*1.5;  
int y = (int)(10*3.5+6*1.5);  
System.out.println(y);
```

3. char类型可以保存 int的常量值，但不能保存int的变量值，需要强转

```
char c1 = 100;  
int m = 100;  
char c2 = m;  
char c3 = (char)m;  
System.out.println(c2);
```

4. byte和short类型在进行运算时，当做int类型处理。

基本数据类型转换-练习题

判断是否能够通过编译

1. short s = 12; //
s = s-9;
2. byte b = 10;
b = b + 11;
b = (byte)(b+11);
3. char c = 'a';
int i = 16;
float d = .314F;
double result = c + i + d;
4. byte b = 16;
short s = 14;
short t = s + b;

基本数据类型转换-练习题

判断是否能够通过编译

1. short s = 12; //ok
s = s-9; //错误 int -> short
2. byte b = 10; //ok
b = b + 11; //错误 int->byte
b = (byte)(b+11); //正确，使用强转
3. char c = 'a'; //ok
int i = 16; //ok
float d = .314F; //ok
double result = c + i + d; //ok float->double
4. byte b = 16; //ok
short s = 14; //ok
short t = s + b; //错误 int -> short

基本数据类型和String类型的转换

● 介绍

在程序开发中，我们经常需要将基本数据类型转成String 类型。或者将String类型转成基本数据类型。

● 基本类型转String类型

● 语法：将基本类型的值+"" 即可

案例演示：StringToBasic.java

```
int n1 = 100;
float n2 = 1.1f;
double n3 = 3.4;
boolean b1 = true;
String str1 = n1 + "";
String str2 = n2 + "";
String str3 = n3 + "";
String str4 = b1 + "";
System.out.println(str1 + " " + str2 + " " + str3 + " " + str4);
```

韩原平
教育 英

● String类型转基本数据类型

语法：通过基本类型的包装类调用parseXX方法即可

案例演示：StringToBasic.java

```
Integer.parseInt("123");
Double.parseDouble("123.1");
Float.parseFloat("123.45");
Short.parseShort("12");
Long.parseLong("12345");
Boolean.parseBoolean("true");
Byte.parseByte("12");
```

基本数据类型和String类型的转换

● 注意事项

案例演示：StringToBasicDetail.java

1. 在将String 类型转成 基本数据类型时，要确保String类型能够转成有效的数据，比如 我们可以把 "123"，转成一个整数，但是不能把 "hello" 转成一个整数
2. 如果格式不正确，就会抛出异常，程序就会终止，这个问题在异常处理章节中，会处理

P60-62作业与小结

2021年4月25日 13:40

算术运算符

● 算术运算符一览

运算符	运算	范例	结果
+	正号	$+7$	7
-	负号	$b=11; -b$	-11
+	加	$9+9$	18
-	减	$10-8$	2
*	乘	$7*8$	56
/	除	$9/9$	1
%	取模(取余)	$11\%9$	2
++	自增 (前): 先运算后取值	$a=2; b=++a;$	$a=3; b=3$
++	自增 (后): 先取值后运算	$a=2; b=a++;$	$a=3; b=2$
--	自减 (前): 先运算后取值	$a=2; b-- -a$	$a=1; b=1$
--	自减 (后): 先取值后运算	$a=2; b=a--$	$a=1; b=2$
+	字符串相加	"hsp"+"edu"	"hsp edu"

算术运算符

● 案例演示

案例演示算术运算符的使用(**ArithmeticOperator.java**)。

1. $+$, $-$, $*$, $/$, $\%$, $++$, $--$, 重点讲解 $/$ 、 $\%$ 、 $++$
2. 自增: $++$
 - 作为独立的语句使用:
前 $++$ 和后 $++$ 都完全等价于 $i=i+1$;
 - 作为表达式使用
前 $++$: $++i$ 先自增后赋值
后 $++$: $i++$ 先赋值后自增
3. $--$, $+$ 、 $-$ 、 $*$ 是一个道理, 完全可以类推。

```

System.out.println(10/4);
double d = 10/4;
System.out.println(d);

System.out.println(10%3);
System.out.println(-10%3);
System.out.println(10%-3);
System.out.println(-10%-3);

int i = 10;
//i++;
++i;
System.out.println(i);
int j = 20;
//int k = ++j;
int k = j++;
System.out.println(k);
System.out.println(i);
  
```

算术运算符

● 课堂练习2

//**ArithmeticOperatorExercise02.java**

1. 假如还有59天放假, 问: 合xx个星期零xx天
2. 定义一个变量保存华氏温度, 华氏温度转换摄氏温度的公式为: $5/9 * (\text{华氏温度} - 100)$, 请求出华氏温度对应的摄氏温度。[234.5]

关系运算符(比较运算符)

● 介绍

1. 关系运算符的结果都是boolean型，也就是要么是true，要么是false
2. 关系表达式 经常用在 if结构的条件中或循环结构的条件中

关系运算符(比较运算符)

● 关系运算符一览

运算符	运算	范例	结果
<code>==</code>	相等于	<code>8==7</code>	false
<code>!=</code>	不等于	<code>8!=7</code>	true
<code><</code>	小于	<code>8<7</code>	false
<code>></code>	大于	<code>8>7</code>	true
<code><=</code>	小于等于	<code>8<=7</code>	false
<code>>=</code>	大于等于	<code>8>=7</code>	true
<code>instanceof</code>	检查是否是类的对象	<code>"hsp" instanceof String</code>	true

关系运算符(比较运算符)

● 案例演示

案例演示关系运算符的使用(`RelationalOperator.java`)。

```
int a = 9; //老韩提示: 开发中, 不可以使用 a, b a1, bc n1, n2
int b = 8;
System.out.println(a>b);
System.out.println(a>=b);
System.out.println(a<=b);
System.out.println(a<b);
System.out.println(a==b);
System.out.println(a!=b);
boolean flag = a>b;
```

关系运算符

● 细节说明

- 1) 关系运算符的结果都是boolean型，也就是要么是true，要么是false。
- 2) 关系运算符组成的表达式，我们称为关系表达式。 `a > b`
- 3) 比较运算符"`==`"不能误写成"`=`"

逻辑运算符

● 逻辑运算符一览

✓ 分为两组学习

1) 短路与 `&&` , 短路或 `||` , 取反 `!`

2) 逻辑与 `&` , 逻辑或 `|` , `^` 逻辑异或

a	b	a&b	a&&b	a b	a b	!a	a^b
true	true	true	true	true	true	false	false
true	false	false	false	true	true	false	true
false	true	false	false	true	true	true	true
false	false	false	false	false	false	true	false

逻辑运算符

● 逻辑运算符一览

✓ 说明逻辑运算规则:

1. `a&b` : `&` 叫逻辑与: 规则: 当a 和 b 同时为true ,则结果为true, 否则为false
2. `a&&b` : `&&` 叫短路与: 规则: 当a 和 b 同时为true ,则结果为true,否则为false
3. `a|b` : `|` 叫逻辑或, 规则: 当a 和 b , 有一个为true ,则结果为true,否则为false
4. `a||b` : `||` 叫短路或, 规则: 当a 和 b , 有一个为true ,则结果为true,否则为false
5. `!a` : 叫取反, 或者非运算。当a 为true, 则结果为false, 当 a 为false是, 结果为true
6. `a^b`: 叫逻辑异或, 当 a 和 b 不同时, 则结果为true, 否则为false

逻辑运算符

● `&&` 和 `&` 基本规则

名称

短路与`&&`

逻辑与`&`

语法

条件1`&&`条件2

条件1`&`条件2

特点

两个条件都为true, 结果为true

两个条件都为true, 结果为true

● `&&` 和 `&` 案例演示

案例演示`&&` 和 `&` 运算符的使用(`LogicOperator01.java`)。

● `&&` 和 `&` 使用区别

1. `&&`短路与: 如果第一个条件为false, 则第二个条件不会判断, 最终结果为false, 效率高
2. `&` 逻辑与: 不管第一个条件是否为false, 第二个条件都要判断, 效率低
3. 开发中, 我们使用的基本是短路与`&&`, 效率高

```
// 逻辑案例
double score = 70;
// 示例: 成绩不在60—80之间
if (score >= 60 && score <= 80) {
    System.out.println("yes100");
}
if (score >= 60 & score <= 80) {
    System.out.println("yes200");
}
```

```
int a=10;
int b = 99;
if (a<5 && b++>100){
    System.out.println("哈哈");
}
if (a<5 & b++>100){
    System.out.println("哈哈");
}
System.out.println(a);
System.out.println(b);
```

P73-75短路或逻辑或；非；异或

2021年4月26日 23:25

逻辑运算符

- || 和 | 基本规则

名称	语法	特点
短路或	条件1 条件2	两个条件中只要有一个成立，结果为true！
逻辑或	条件1 条件2	只要有一个条件成立，结果为true

- || 和 | 案例演示
- 案例演示&& || !运算符的使用(LogicOperator02.java)。
- || 和 | 使用区别

普通案例
使用前面的

```
// 案例代码
int a=10;
int b = 99;
if(a>5 || b>=100){
    System.out.println("哈哈");
}
if(a>5 | b>=100){
    System.out.println("哈哈");
}
System.out.println(a);
System.out.println(b);
```

- 1) ||短路或：如果第一个条件为true，则第二个条件不会判断，最终结果为true，效率高
- 2) | 逻辑或：不管第一个条件是否为true，第二个条件都要判断，效率低
- 3) 开发中，我们基本使用 ||

逻辑运算符

- ! 取反 基本规则

名称	语法	特点
!非 (取反)	!条件	如果条件本身成立，结果为false，否则为true

- ! 案例演示
- 案例演示 !运算符的使用(InverseOperator.java)。
- ^ 案例演示

a^b: 叫逻辑异或，当 a 和 b 不同时，则结果为true，否则为false

^逻辑异或，System.out.println((4 < 1) ^ (6 > 3)); // ?

x++==6; 后++先比较在自增，
++x==6; 前++先自增在比较，

逻辑运算符

- 练习题2请写输出结果

```
boolean x=true;
boolean y=false;
short z=46;
if( (z++==46) && (y=true) ) z++;
if( (x=false) || (++z==49) ) z++;
System.out.println("z="+z);
```

→ 50

练习笔记

大小: 1920 * 1080
(40, 30, 37)

韩顺平 著


赋值运算符

- 介绍

赋值运算符就是将某个运算后的值，赋给指定的变量。

- 赋值运算符的分类

- ✓ 基本赋值运算符 =
- ✓ 复合赋值运算符
 - + = , - = , * = , / = , % = 等, 重点讲解一个 + = , 其它的使用是一个道理
 - a + = b; [等价?]
 - a - = b; [等价?]



赋值运算符。

- 案例演示

案例演示赋值运算符的基本使用。AssignOperator.java

- 1) 赋值基本案例 [int num1 = 10]
- 2) + = 的使用案例

- 赋值运算符特点

- 1) 运算顺序从右往左 int num = a + b + c;
- 2) 赋值运算符的左边 只能是变量, 右边 可以是变量、表达式、常量值
int num = 20; int num2 = 78 * 34 - 10; int num3 = a;
- 3) 复合赋值运算符等价于下面的效果
比如: a += 3; 等价于 a = a + 3;
- 4) 复合赋值运算符会进行类型转换。
byte b = 2; b += 3; b++;

P78-81三元运算符，优先级

2021年4月27日 0:16

三元运算符

- 基本语法

条件表达式 ? 表达式1: 表达式2;

运算规则:

- 如果条件表达式为true, 运算后的结果是表达式1;
- 如果条件表达式为false, 运算后的结果是表达式2;

口诀: [一灯大师: 一真大师]

- 案例演示

```
int a = 10;
int b = 99;
int result = a > b ? a++ : b--;
```

三元运算符

- 使用细节 **TernaryOperatorDetail.java**

- 表达式1和表达式2要为可以赋给接收变量的类型(或可以自动转换)
- 三元运算符可以转成if--else 语句

```
int res = a > b ? a++ : --b;
if (a > b) res = a++;
else res = --b;
```

- 课堂练习

案例: 实现三个数的最大值

运算符优先级

1. 运算符有不同的优先级, 所谓优先级就是表达式运算中的运算顺序。如右表, 上一行运算符总优先于下一行。

2. 只有单目运算符、赋值运算符是从右向左运算的。

梳理小结: 小伙伴有一个大致印象, 使用多了, 就熟悉

- 1) 0, 0 等
- 2) 单目运行 ++ --
- 3) 算术运算符
- 4) 位移运算符
- 5) 比较运算符
- 6) 逻辑运算符
- 7) 三元运算符
- 8) 赋值运算符

R→L	++ -- ~ !(data type)
L→R	* / %
L→R	+ -
L→R	<< >> >>> 位移
L→R	< > <= >= instanceof
L→R	== !=
L→R	&&
L→R	^
L→R	
L→R	&&&
L→R	
L→R	? :
R→L	= *= /= %=
	+= -= <<= >>=
	>>>= &= ^= =

标识符的命名规则和规范

- 标识符概念
 1. Java 对各种变量、方法和类等命名时使用的字符序列称为标识符
 2. 凡是自己可以起名字的地方都叫标识符 `int num1 = 90;`
- 标识符的命名规则(必须遵守)
 1. 由26个英文字母大小写, 0-9, `_` 或 `$` 组成
 2. 数字不可以开头。 `int 3ab = 1;`
 3. 不可以使用关键字和保留字, 但能包含关键字和保留字。
 4. Java中严格区分大小写, 长度无限制。 `int totalNum = 10; int n = 90;`
 5. 标识符不能包含空格。 `int a b = 90;`

标识符的命名规则和规范

- 判断下面变量名是否正确
 - hsp //ok
 - hsp12 //ok
 - 1hsp //错误, 数字不能开头
 - h-s // 错误, 不能有 -
 - x h // 错误, 有空格
 - h\$4 // ok
 - class //错误, class 关键字
 - int // 错误, int 是关键字
 - double //错误, double是关键字
 - public //错误, public 是关键字
 - static //错误, static是关键字
 - goto //错误, goto是保留字
 - stu_name //ok

标识符的命名规则和规范

- 标识符命名规范[更加专业]
 1. 包名: 多单词组成时所有字母都小写: `aaa.bbb.ccc` //比如 `com.hsp.crm`
 2. 类名、接口名: 多单词组成时, 所有单词的首字母大写: `XxxYyyZzz` [大驼峰]
比如: `TankShotGame`
 3. 变量名、方法名: 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: `xxxYyyZzz` [小驼峰, 简称 驼峰法]
比如: `tankShotGame`
 4. 常量名: 所有字母都大写。多单词时每个单词用下划线连接: `XXX_YYY_ZZZ`
比如: 定义一个所得税率 `TAX_RATE`
 5. 后面我们学习到 类, 包, 接口, 等时, 我们的命名规范要这样遵守, 更加详细的看文档。