

1. "&"和"&&"的异同

2. 程序输出

```
1. class Test4 {  
2.     public static void main (String [] args) {  
3.         boolean x=true;  
4.         boolean y=false;  
5.         short z=40;  
6.           
7.         if((z++==40)&&(y=true)){z++;}  
8.         if((x=false) || (++z==43)) {z++;}  
9.           
10.        System.out.println("z="+z);  
11.    }  
12. }
```

结果为:

3. 使用三元运算符或者 if-else 获取三个数中的较大数的实现

4. 编写程序, 声明 2 个 double 型变量并赋值。判断第一个数大于 10.0, 且第 2 个数小于 20.0, 打印两数之和。

5. 如何从控制台获取 String 和 int 型的变量, 并输出? 使用代码实现

2. 程序输出

```
1. class Test4 {  
2.     public static void main (String [] args) {  
3.         boolean x=true;  
4.         boolean y=false;  
5.         short z=40;  
6.           
7.         if((z++==40)&&(y=true)){z++;}  
8.         if((x=false) || (++z==43)) {z++;}  
9.           
10.        System.out.println("z="+z);  
11.    }  
12. }
```

结果为:

3. 使用三元运算符或者 if-else 获取三个数中的较大数的实现

```
int num1 = 10;  
int num2 = 20;  
int num3 = 2;  
int max = (num1 > num2)? num1 : num2;  
int maxx = (max > num3)? max : num3;  
  
if(num1 >= num2 && num1 >= num3){  
    sysout(num1);  
}else if(num2 >= num1 && num2 >= num3){  
    sysout(num2);  
}else{  
    sysout(num3);  
}
```

4. 编写程序，声明 2 个 double 型变量并赋值。判断第一个数大于 10.0，且第 2 个数小于 20.0，打印两数之和。

```
double d1 = 12.1, d2 = 32.1;
if(d1 > 10.0 && d2 < 20.0){
    sysout(d1 + d2);
}
```

5. 如何从控制台获取 String 和 int 型的变量，并输出？使用代码实现

```
import java.util.Scanner;
Scanner s = new Scanner(System.in);
String name = s.next();
```

```
int age = s.nextInt();
没有 nextChar().
```

3. 比较运算符（关系运算符）：== > < >= <= != instanceof

【典型代码】

```
int num1 = 10;
int num2 = 20;

System.out.println(num1 == num2); // false
System.out.println(num1 = num2); // 20
```

【特别说明的】

1. 比较运算符的结果都是 boolean 类型

2. 区分 == 和 =

比如：

```
boolean b = false;
```

```
if(b = true){ // 执行语句 }
```

I

4. 逻辑运算符: & && || ! ^

【典型代码】

```
boolean b1 = true;

boolean b2 = false;


System.out.println("b1 & b2 : " + (b1 & b2));

System.out.println("b1 && b2 : " + (b1 && b2));

System.out.println("b1 | b2 : " + (b1 | b2));

System.out.println("b1 || b2 : " + (b1 || b2));

System.out.println("!b1 : " + (!b1));

System.out.println("b1 ^ b2 : " + (b1 ^ b2));
```


【特别说明的】

```
//区分& 与 &&

    //如果符号左端是true,二者没有区别,都执行符号右边的运算

    //如果符号左端是false,&:继续执行符号右边的运算。&&: 不再执行符号右端的运算

    //开发中,通常都使用 &&

// 区分|| 与 || 

    //如果符号左端是false时,二者没有区别,都执行符号右边的运算

    //如果符号左端是true时,|会继续执行符号右端的运算,||不再执行符号右端的运算

    //开发中,通常都使用 ||
```

【面试题】 你能否写出最高效的2 * 8的实现方式? 2 << 3 或 8 << 1

【特别说明的】

1. 位运算符操作的都是整型数据
2. <<:在一定范围内,每左移一位,相当于*2
 >>:在一定范围内,每右移一位,相当于/2
3. >>> :无符号右移,不管此整数是正数还是负数,高位都拿0补。

- 如何求一个 0~255 范围内的整数的十六进制值，例如 60 的十六进制表示形式 3C。

```
//方式一：自动实现
String str1 = Integer.toBinaryString(60);
String str2 = Integer.toHexString(60);

//方式二：手动实现
int i1 = 60;
int i2 = i1 & 15;
String j = (i2 > 9)? (char)(i2-10 + 'A')+"": i2+"";

int temp = i1 >>> 4;
i2 = temp & 15;
String k = (i2 > 9)? (char)(i2-10 + 'A')+"": i2+"";
System.out.println(k+""+j);
```

【特别说明的】

1. 条件表达式的结果是boolean类型。
2. 如果表达式是true，则执行表达式1；否则，执行表达式2
3. 表达式1和表达式2必须类型是一致的。
4. 三元运算符可以嵌套使用
5. 结论1：凡是使用三元运算符的都可以改写为使用if-else实现。反之，不成立。
结论2：既可以使用三元运算符，也可以使用if-else，我们应该选择：三元运算符。因为执行效率高。

如何从键盘获取不同类型的变量？使用Scanner类

步骤：

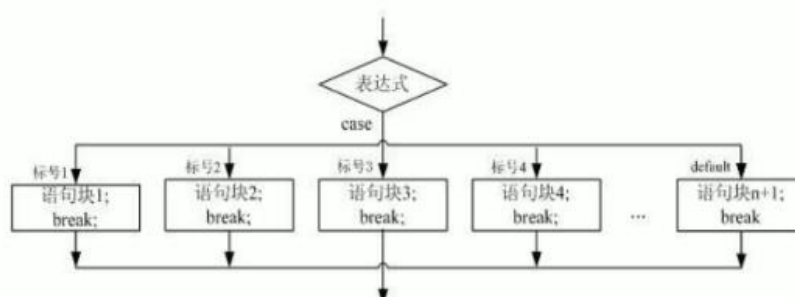
1. 导包：import java.util.Scanner;
2. Scanner类的实例化：Scanner scan = new Scanner(System.in);
3. 调用Scanner类提供的相关方法，获取不同类型的变量

说明：

1. 不能使用Scanner来获取char类型变量，可以获取String型替代
2. 如果要求获取的类型和用户输入的类型不匹配，则报错：InputMismatchException

2.5.3 程序流程控制：switch-case结构

```
switch(表达式){  
case 常量1:  
    语句1;  
    // break;  
case 常量2:  
    语句2;  
    // break;  
... ..  
case 常量N:  
    语句N;  
    // break;  
default:  
    语句;  
    // break;  
}
```



让天下没有难学的技术

20 说明：
21 1. switch中的表达式，不是条件表达式。
22 2. 根据表达式中的值，依次匹配case中的常量，一旦匹配成功，就进入相应case中的执行语句执行。
23 执行完此case以后，如果有break，则跳出当前的switch-case结构。如果没有break，则依次执行下面的case，直到遇到break或
24 到程序执行结束为止。
25 3. break可以使用在switch-case结构中，表示：结束当前switch-case结构的执行。
26 4. switch中表达式的类型，只能是如下的6种类型：byte \ short \ char \ int \ 枚举类对象（jdk5.0） \ String（jdk7.0）
27 */

12 说明：
13 1. 凡是使用switch-case结构能实现的分支语句，都可以转换为if-else。反之，不成立。
14 2. 当满足switch中表达式的数据类型（6种），同时变量的取值也不多时，建议使用switch-case。原因：执行效率高。
15 */

2.5.4 程序流程控制：循环结构

●循环结构

➤在某些条件满足的情况下，反复执行特定代码的功能

●循环语句分类

➤for 循环

➤while 循环

➤do-while 循环

● 循环语句的四个组成部分

- 初始化部分(`init_statement`)
- 循环条件部分(`test_exp`)
- 循环体部分(`body_statement`)
- 迭代部分(`alter_statement`)

