第 4 章 运算符

4.1 运算符介绍

4.1. 1运算符介绍

运算符是一种特殊的符号，用以表示数据的运算、赋值和比较等。

1) 算术运算符

2) 赋值运算符

3) 关系运算符 [比较运算符]

4) 逻辑运算符

5) 位运算符 [需要二进制基础]

6) 三元运算符

4.2 算术运算符

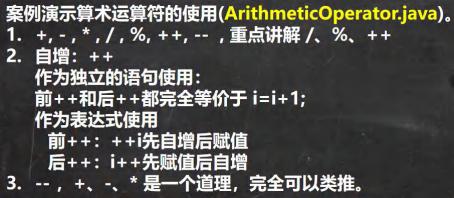
4.2. 1介绍

算术运算符是对数值类型的变量进行运算的，在 Java程序中使用的非常多。

4.2.2算术运算符一览



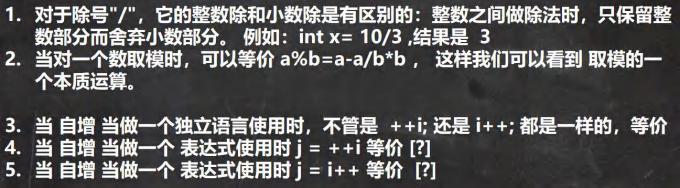
4.2.3案例演示



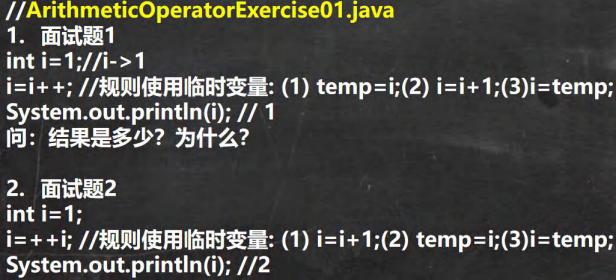
|  |
| --- |
| /\*\*  \* 演示算术运算符的使用  \*/  public class ArithmeticOperator {  //编写一个 main 方法  public static void main(String[] args) {  // /使用  System.out.println(10 / 4); //从数学来看是 2.5, java 中 2  System.out.println(10.0 / 4); //java 是 2.5  // 注释快捷键 ctrl + /, 再次输入 ctrl + / 取消注释  double d = 10 / 4;//java 中 10 / 4 = 2, 2=>2.0  System.out.println(d);// 是 2.0  // % 取模 ,取余  // 在 % 的本质 看一个公式!!!! a % b = a - a / b \* b  // - 10 % 3 => - 10 - (- 10) / 3 \* 3 = - 10 + 9 = - 1  // 10 % -3 = 10 - 10 / (-3) \* (-3) = 10 - 9 = 1  // - 10 % -3 = (- 10) - (- 10) / (-3) \* (-3) = - 10 + 9 = - 1 |

|  |
| --- |
| System.out.println(10 % 3); //1  System.out.println(- 10 % 3); // - 1  System.out.println(10 % -3); //1  System.out.println(- 10 % -3);//- 1  //++的使用  //  int i = 10;  i++;//自增 等价于 i = i + 1; => i = 11  ++i;//自增 等价于 i = i + 1; => i = 12  System.out.println("i=" + i);//12  /\*  作为表达式使用  前++：++i 先自增后赋值  后++：i++先赋值后自增  \*/  intj = 8;  //int k = ++j; //等价 j=j+1;k=j;  int k = j++; // 等价 k =j;j=j+1;  System.out.println("k=" + k + "j=" + j);//8 9  }  } |

4.2.4细节说明



4.2.5面试题



4.2.6自增，自减课堂练习, 看看输出什么

|  |
| --- |
| //练习  public class ArithmeticOperatorExercise01 {  //编写一个 main 方法  public static void main(String[] args) { |



|  |
| --- |
| // int i = 1;//i->1  // i = i++; //规则使用临时变量: (1) temp=i;(2) i=i+1;(3)i=temp;  // System.out.println(i); // 1  // int i= 1;  // i=++i; //规则使用临时变量: (1) i=i+1;(2) temp=i;(3)i=temp;  // System.out.println(i); //2  //  // 测试输出  int i1 = 10;  int i2 = 20;  int i = i1++;  System.out.print("i="+i);//10  System.out.println("i2="+i2);//20  i = --i2;  System.out.print("i="+i);//19  System.out.println("i2="+i2);//19  }  } |

4.2.7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\*(华氏温度- 100),请求出华氏温度对应的摄氏温度。

[234.5]

3) 代码

|  |
| --- |
| //课堂练习  public class ArithmeticOperatorExercise02 {  //编写一个 main 方法  public static void main(String[] args) {  //1.需求:  //假如还有 59 天放假，问：合 xx 个星期零 xx 天  //2.思路分析  //(1) 使用 int 变量 days 保存 天数  //(2) 一个星期是 7 天 星期数 weeks： days / 7 零 xx 天 leftDays days % [7](#bookmark1197)  //(3) 输出  //3.走代码  int days = 25911;  int weeks = days / 7;  int leftDays = days % 7;  System.out.println(days + "天 合" + weeks + "星期零"  + leftDays + "天");  //1.需求 |

|  |
| --- |
| //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\*(华氏温度- 100),请求出华氏温度对应的摄氏温度  //  //2 思路分析  //(1) 先定义一个 double huaShi 变量保存 华氏温度  //(2) 根据给出的公式，进行计算即可 5/9\*(华氏温度- 100)  // 考虑数学公式和java 语言的特性  //(3) 将得到的结果保存到 double sheShi  //3 走代码  double huaShi = 1234.6;  double sheShi = 5.0 / 9 \* (huaShi - 100);  System.out.println("华氏温度" + huaShi  + " 对应的摄氏温度=" + sheShi);  }  } |

4.3 关系运算符(比较运算符)

4.3. 1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

4.3.2关系运算符一览



4.3.3案例演示

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

|  |
| --- |
| //演示关系运算符的使用  //  public class RelationalOperator {  //编写一个 main 方法  public static void main(String[] args) {  int a = 9; //老韩提示: 开发中，不可以使用 a, b  int b = 8;  System.out.println(a > b); //T  System.out.println(a >= b); //T  System.out.println(a <= b); //F  System.out.println(a < b);//F |

|  |
| --- |
| System.out.println(a == b); //F  System.out.println(a != b); //T  boolean flag = a > b; //T  System.out.println("flag=" + flag);  }  } |

4.3.4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"="

4.4 逻辑运算符

4.4. 1介绍

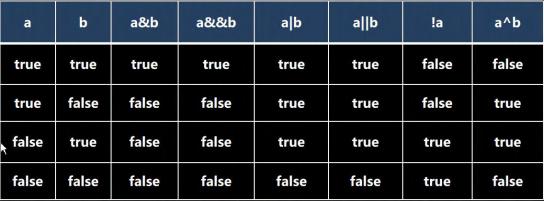
用于连接多个条件（多个关系表达式），最终的结果也是一个 boolean值。

4.4.2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



 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

4.4.3&& 和 & 基本规则

|  |  |  |
| --- | --- | --- |
| 名称 | 语法 | 特点 |
| 短路与&& | 条件 1&&条件 2 | 两个条件都为 true ，结果为 true,否则 false |
| 逻辑与& | 条件 1&条件 2 | 两个条件都为 true ，结果为 true,否则 false |

4.4.4&& 和 & 案例演示

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

|  |
| --- |
| /\*\*  \* 演示逻辑运算符的使用  \*/  public class LogicOperator01 {  //编写一个 main 方法  public static void main(String[] args) {  //&&短路与 和 & 案例演示  int age = 50;  if(age > 20 && age < 90) { |



|  |
| --- |
| System.out.println("ok100");  }  //&逻辑与使用  if(age > 20 & age < 90) {  System.out.println("ok200");  }  //区别  int a = 4;  int b = 9;  //对于&&短路与而言，如果第一个条件为 false ,后面的条件不再判断  //对于&逻辑与而言，如果第一个条件为 false ,后面的条件仍然会判断  if(a < 1 & ++b < 50) {  System.out.println("ok300");  }  System.out.println("a=" + a + " b=" + b);// 4 10  }  } |

4.4.5&& 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， 我们使用的基本是使用短路与&&, 效率高

4.4.6|| 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,否则为 false |
| |逻辑或 | 条件 1|条件 2 | 只要有一个条件成立，结果为 true,否则为 false |

4.4.7|| 和 | 案例演示

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

|  |
| --- |
| //演示| || 使用  public class LogicOperator02 {  //编写一个 main 方法  public static void main(String[] args) {  //||短路或 和 |逻辑或 案例演示  //|| 规则: 两个条件中只要有一个成立，结果为 true,否则为 false  //| 规则: 两个条件中只要有一个成立，结果为 true,否则为 false  int age = 50;  if(age > 20 || age < 30) {  System.out.println("ok100");  }  //&逻辑与使用  if(age > 20 | age < 30) { |



|  |
| --- |
| System.out.println("ok200");  }  //看看区别  //(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  int a = 4;  int b = 9;  if( a > 1 || ++b > 4) { // 可以换成 | 测试  System.out.println("ok300");  }  System.out.println("a=" + a + " b=" + b); //4 10  }  } |

4.4.8|| 和 | 使用区别

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 ||

4.4.9! 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

4.4.10 ! 案例演示

案例演示 !运算符的使用(InverseOperator.java)。

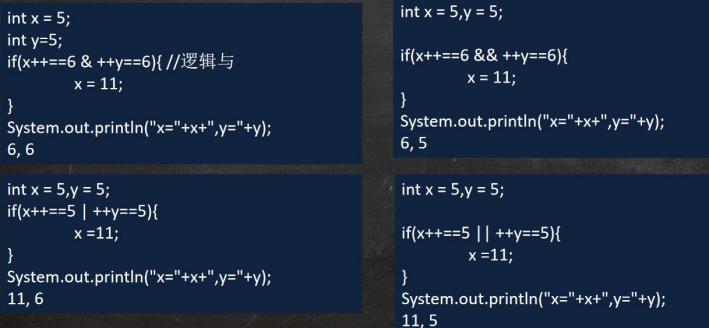
|  |
| --- |
| //!和^案例演示  public class InverseOperator {  //编写一个 main 方法  public static void main(String[] args) {  //! 操作是取反 T->F , F -> T  System.out.println(60 > 20); //T  System.out.println(!(60 > 20)); //F  //a^b: 叫逻辑异或，当 a 和 b 不同时，则结果为 true, 否则为 false  boolean b = (10 > 1) ^ ( 3 > 5);  System.out.println("b=" + b);//T  }  } |

4.4.11 ^ 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) ); // ?

4.4.12 练习题 1 请写出每题的输出结果



4.4.13 练习题 2 请写输出结果



4.5 赋值运算符

4.5. 1介绍

赋值运算符就是将某个运算后的值，赋给指定的变量。

4.5.2赋值运算符的分类

 基本赋值运算符 = int a = 10;

 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 = a + b; ]

a -= b; [等价 a = a - b; ]

4.5.3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

4.5.4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; 其他类推

4) 复合赋值运算符会进行类型转换。

byte b = 2; b+=3; b++;

|  |
| --- |
| //演示赋值运算符的使用  public class AssignOperator { |

|  |
| --- |
| //编写一个 main 方法  public static void main(String[] args) {  int n1 = 10;  n1 += 4;// n1 = n1 + 4;  System.out.println(n1); // 14  n1 /= 3;// n1 = n1 / 3;//4  System.out.println(n1); // 4  //复合赋值运算符会进行类型转换  byte b = 3;  b += 2; // 等价 b = (byte)(b + 2);  b++; // b = (byte)(b+1);  }  } |

4.6 三元运算符

4.6. 1基本语法

条件表达式 ? 表达式 1: 表达式 2;

运算规则：

1. 如果条件表达式为 true ，运算后的结果是表达式 1；

2. 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**]

4.6.2案例演示 TernaryOperator.java

|  |
| --- |
| //三元运算符使用  public class TernaryOperator {  //编写一个 main 方法  public static void main(String[] args) {  int a = 10;  int b = 99;  // 解读  // 1. a > b 为 false  // 2. 返回 b--, 先返回 b 的值,然后在 b- 1  // 3. 返回的结果是 99  int result = a > b ? a++ : b--;  System.out.println("result=" + result);  System.out.println("a=" + a);  System.out.println("b=" + b);  }  } |

4.6.3使用细节 TernaryOperatorDetail.java

1) 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)

2) 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;

|  |
| --- |
| //三元运算符细节  public class TernaryOperatorDetail {  //编写一个 main 方法  public static void main(String[] args) {  //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/或者强制转换)  int a = 3;  int b = 8;  int c = a > b ? (int)1. 1 : (int)3.4;//可以的  double d = a > b ? a : b + 3;//可以的，满足 int -> double  }  } |

4.6.4课堂练习

案例：实现三个数的最大值 TernaryOperatorExercise.java

|  |
| --- |
| public class TernaryOperatorExercise {  //编写一个 main 方法  public static void main(String[] args) { |

|  |
| --- |
| //案例：实现三个数的最大值  int n1 = 553;  int n2 = 33;  int n3 = 123;  //思路  //1. 先得到 n1 和 n2 中最大数 , 保存到 max1  //2. 然后再 求出 max1 和 n3 中的最大数，保存到 max2  int max1 = n1 > n2 ? n1 : n2;  int max2 = max1 > n3 ? max1 : n3;  System.out.println("最大数=" + max2);  //使用一条语句实现, 推荐使用上面方法  //老师提示: 后面我们可以使用更好方法, 比如排序  // int max = (n1 > n2 ? n1 : n2) > n3 ?  // (n1 > n2 ? n1 : n2) : n3;  // System.out.println("最大数=" + max);  //  int abcclass = 10;  int n = 40;  int N = 50;  System.out.println("n=" + n);//40  System.out.println("N=" + N);//50  //? abc 和 aBc 是两个不同变量 |

|  |
| --- |
| int abc = 100;  int aBc = 200;  //int a b = 300;  //int a-b= 10;  int goto1 = 10;  }  } |

4.7 运算符优先级

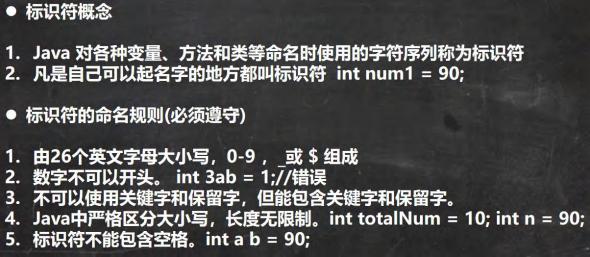
1) 运算符有不同的优先级，所谓优先级就是表达式运算中的运算顺序。如右表，上一行运算符总优先于下一行。

2) 只有单目运算符、赋值运算符是从右向左运算的。

3) 一览表, 不要背，使用多了，就熟悉了.



4.8 标识符的命名**规则**和**规范**



4.8. 1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 |

4.8.2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.

4.9 关键字

关键字的定义和特点 (**不用背**)

定义：被 Java语言赋予了特殊含义，用做**专门用途的字符串**（单词）

特点：关键字中所有字母都为小写





4. 10 保留字

4.10.1 介绍

Java 保留字：现有 Java 版本**尚未使用**，但**以后版本可能会作为关键字使用**。自己命名标识符时要避免使用这些保留

字 byValue、cast 、future 、 generic 、 inner 、 operator 、 outer 、 rest 、 var 、 goto 、const

4. 11 键盘输入语句

4.11.1 介绍

在编程中，需要接收用户输入的数据，就可以使用键盘输入语句来获取。Input.java , 需要一个 扫描器(对象), 就是

Scanner

4.11.2 步骤 ：

1) 导入该类的所在包, java.util.\*

2) 创建该类对象（声明变量）

3) 调用里面的功能

4.11.3 案例演示：

要求：可以从控制台接收用户信息，【姓名，年龄，薪水】

|  |
| --- |
| import java.util.Scanner;//表示把java.util 下的 Scanner 类导入  public class Input {  //编写一个 main 方法  public static void main(String[] args) {  //演示接受用户的输入  //步骤  //Scanner 类 表示 简单文本扫描器，在java.util 包  //1. 引入/导入 Scanner 类所在的包  //2. 创建 Scanner 对象 , new 创建一个对象,体会  // myScanner 就是 Scanner 类的对象  Scanner myScanner = new Scanner(System.in);  //3. 接收用户输入了， 使用 相关的方法  System.out.println("请输入名字");  //当程序执行到 next 方法时，会等待用户输入~~~  String name = myScanner.next(); //接收用户输入字符串  System.out.println("请输入年龄");  int age = myScanner.nextInt(); //接收用户输入 int  System.out.println("请输入薪水");  double sal = myScanner.nextDouble(); //接收用户输入 double  System.out.println("人的信息如下:"); |

|  |
| --- |
| System.out.println("名字=" + name  + " 年龄=" + age + " 薪水=" + sal);  }  } |

4. 12 进制(程序员的基本功)

4.12.1 进制介绍

对于整数，有四种表示方式：

二进制：0, 1 ，满 2 进 1. 以 0b 或 0B 开头。

十进制：0-9 ，满 10 进 1。

八进制：0-7 ，满 8 进 1. 以数字 0 开头表示。

十六进制：0-9 及 A(10)-F(15) ，满 16 进 1. 以 **0x 或** **0X** 开头表示。此处的 A-F 不区分大小写。

4.12.2 举例说明 BinaryTest.java

|  |
| --- |
| //演示四种进制  //  public class BinaryTest {  //编写一个 main 方法  public static void main(String[] args) {  //n1 二进制  int n1 = 0b1010;  //n2 10 进制  int n2 = 1010;  //n3 8 进制 |

|  |
| --- |
| int n3 = 01010;  //n4 16 进制  int n4 = 0X10101;  System.out.println("n1=" + n1);  System.out.println("n2=" + n2);  System.out.println("n3=" + n3);  System.out.println("n4=" + n4);  System.out.println(0x23A);  }  } |

4. 13 进制的图示





4. 14 进制的转换(基本功)

4.14.1 进制转换的介绍

. 第一组：

1) 二进制转十进制

2) 八进制转十进制

3) 十六进制转十进制

. 第二组：

1) 十进制转二进制

2) 十进制转八进制

3) 十进制转十六进制

. 第三组

1) 二进制转八进制

2) 二进制转十六进制

. 第四组

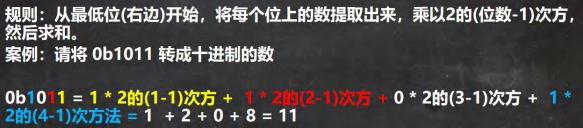


4

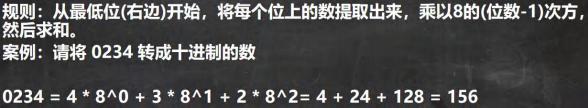
1) 八进制转二进制

2) 十六进制转二进制

4. 15 二进制转换成十进制示例



4. 16 八进制转换成十进制示例



4. 17 十六进制转换成十进制示例

规则：从最低位(右边)开始，将每个位上的数提取出来，乘以 16 的(位数- 1)次方，然后求和。

案例：请将 0x23A 转成十进制的数

0x23A = 10 \* 16^0 + 3 \* 16 ^ 1 + 2 \* 16^2 = 10 + 48 + 512 = 570

课堂练习：请将

0b110001100 转成 十进制

02456 转成十进制

0xA45 转成十进制

4. 18 十进制转换成二进制

规则：将该数不断除以 2 ，直到商为 0 为止，然后将每步得到的余数倒过来，就是对应的二进制。

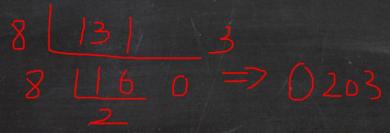
案例：请将 34 转成二进制 = 0B00100010



4. 19 十进制转换成八进制

规则：将该数不断除以 8 ，直到商为 0 为止，然后将每步得到的余数倒过来，就是对应的八进制。

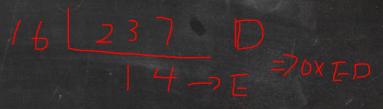
案例：请将 131 转成八进制 => 0203



4.20 十进制转换成十六进制

规则：将该数不断除以 16 ，直到商为 0 为止，然后将每步得到的余数倒过来，就是对应的十六进制。

案例：请将 237 转成十六进制 => 0xED



课堂练习：(一定练习，使用计算器/用程序去验证)

123 转成 二进制 => ?

678 转成八进制 => ?

8912 转成十六进制 => ?

4.21 二进制转换成八进制

规则：从低位开始,将二进制数每三位一组，转成对应的八进制数即可。

案例：请将 ob11010101 转成八进制

ob11(3)010(2)101(5) => 0325

4.22 二进制转换成十六进制

规则：从低位开始，将二进制数每四位一组，转成对应的十六进制数即可。

案例：请将 ob11010101 转成十六进制

ob1101(D)0101(5) = 0xD5

课堂练习：请将

0b11100101 转成 八进制

0b1110010110 转成 十六进制

4.23 八进制转换成二进制

规则：将八进制数每 1 位，转成对应的一个 3 位的二进制数即可。

案例：请将 0237 转成二进制

02(010)3(011)7(111) = 0b10011111

4.24 十六进制转换成二进制

规则：将十六进制数每 1 位，转成对应的 4 位的一个二进制数即可。

案例：请将 0x23B 转成二进制

0x2(0010)3(0011)B(1011) = 0b001000111011

课堂练习：请将

01230 转成 二进制

0xAB29 转成二进制

4.25 位运算的思考题

1) 请看下面的代码段，回答 a,b,c,d,e 结果是多少?

public static void maiin(String []args)

{

int a= 1>>2; // 1 向右位移 2 位

int b=- 1>>2;//算术右移

int c= 1<<2;//算术左移

int d=- 1<<2;//

int e=3>>>2;//无符号右移

//a,b,c,d,e 结果是多少

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

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

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

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

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

}

2) 请回答在java 中，下面的表达式运算的结果是: (位操作) ， 晚上练习

~2=? // 按位取反

2&3=?// 2 按位与 [3](#bookmark1198)

2|3=?

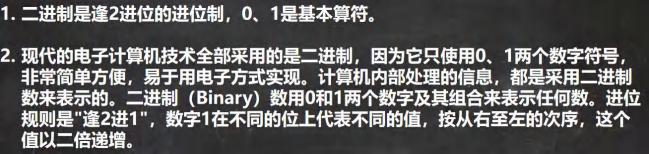
~-5=?

13&7=?

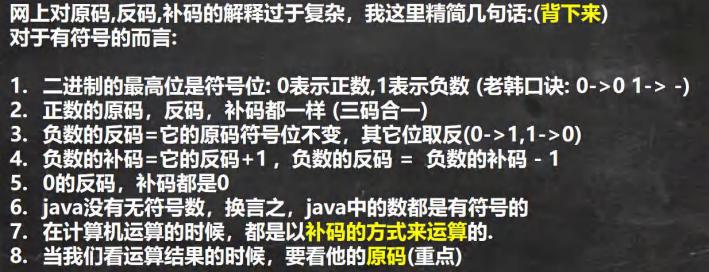
5|4=?

-3^3=?//^ 按位异或

4.26 二进制在运算中的说明

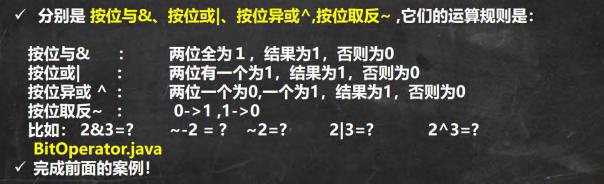


4.27 **原码、反码、补码**(重点 难点)



4.28 位运算符

4.28.1 java 中有 7 个位运算(& 、| 、^ 、~ 、>> 、<<和 >>>)



4.28.2 还有 3 个位运算符 >> 、<< 和 >>> , 运算规则:

1) 算术右移 >>：低位溢出,符号位不变,并用符号位补溢出的高位

2) 算术左移 <<: 符号位不变,低位补 0

3) >>> 逻辑右移也叫无符号右移,运算规则是: 低位溢出，高位补 0

4) 特别说明：没有 <<< 符号

4.28.3 应用案例 BitOperator02.java

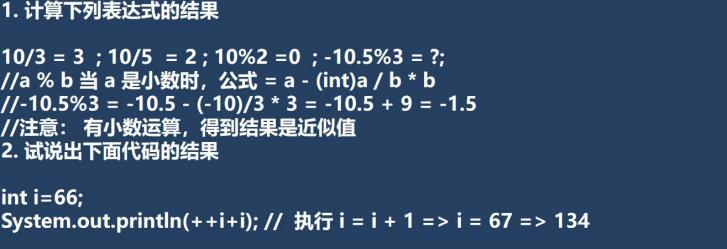
1) int a= 1>>2; //1 => 00000001 => 00000000 本质 1 / 2 / 2 =0

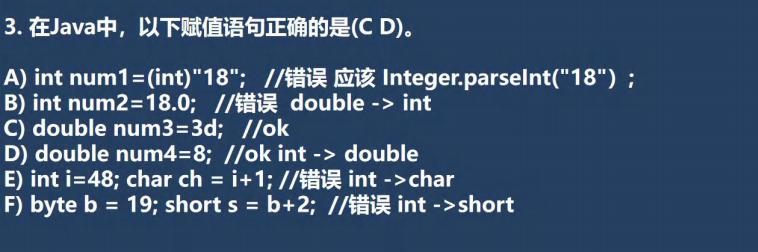
2) int c= 1<<2; //1 => 00000001 => 00000100 本质 1 \* 2 \* 2 = 4

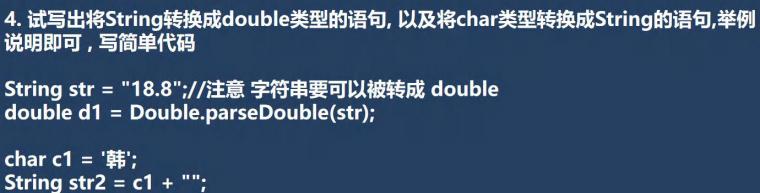
4.28.4 完成前面的案例

建议：掌握老师讲解的即可，不用再深入.

4.29 本章作业









1