

Day3

1、数据类型中的小问题

在定义Long或Float类型变量时，要在末尾加L或F

byte, short在定义时，实际接受的是一个int类型的值

2、byte值的问题

 image-20221119135142411

3、类型转换

byte, short, char--int--long--float--double

long: 8字节, float: 4字节 为什么是long向float转换? 因为整数和小数的底层存储方式不一样, 整数是10字符串, 小数是用科学计数法, float表示范围更广

 image-20221119135714629

4、问题

Java中的char可以表示一个汉字吗?

可以, 因为java中char占两个字节, 采用Unicode编码

5、运算符

算数、赋值、比较、逻辑、位、三目运算符

(1) 算数运算符

 image-20221119141805915

整数相除只能得到整数: 结果是商, 不加余数, 要想得到小数只需要将其中一个数改为浮点数即可; 直接乘一个1.0也可以

字符串做操作结果都是字符串

(2) 自增和自减: 变量才能++和--

单独使用: 放在前面和后面效果一样

参与运算使用: a、放在后面: 先赋值, 再做++或--

b、放在前面: 先++或--, 再赋值

(3) 运算符的优先级

 image-20221119152232427

(4) +的用法

加号, 正号, 字符串连符

字符用单引号, 字符串用双引号

字符 + int, 会输出int, 因为字符在运算时是先转成int再计算

字符串+int, 输出的是字符串, +在这里是字符串连接符号

(5) 赋值运算符

=, +=, -=, *=, /=

扩展的运算符隐含强制类型转换

$s += 1;$ 等价于 $(s\text{的类型})\ s = (s\text{的类型})\ (s + 1)$

```
class OperatorDemo3{
    public static void main(String[] args) {
        //short s = 1;
        //s = s + 1;
        //这里会报错, 因为short参与运算会先转换成int, int相加再赋值给short可能会损失精度

        short s = 1;
        s += 1;
        System.out.println(s); //不报错, 扩展的运算符隐含强制类型转换
    }
}
```

(6) 关系运算符

注意==不要写成=


 image-20221119154954748

(7) 逻辑运算符

&、|、^、!、&&、||

用于连接布尔类型表达式或者值

表达式: 就是用算术运算符把变量或者常量连接起来的式子

 image-20221119164903314

逻辑与&: 有false则为false, 全为true才是true

逻辑或: 有true则为true, 全为false才是false

逻辑异或: 相同为false, 不同为true

逻辑非: 偶数个! 不变, 奇数个改变

&&和&的区别: 最终结果一样, 双与具有短路效果, 左边一旦能够判断结果, 后面就不再处理了

(8) 位运算符

做位运算之前, 要先将数据转换为二进制, 并且是补码

左右两侧是数据则是位运算符, 左右两侧是布尔, 则是逻辑运算符

 image-20221119170618605


```
//位运算符
class OperatorDemo6{
    public static void main(String[] args) {
        int a = 3;
        int b = 4;
        System.out.println(3 & 4); //0
        System.out.println(3 | 4); //7
        System.out.println(3 ^ 4); //7
        System.out.println(~3); //-4
    }
}
```

&位运算：有0则0，全1为1

|位运算符：有1则1，全0为0

^位异或运算符：相同为0，不同为1，，一个数据对另一个数异或两次不变

~位运算符：按位取反，0变1，1变0，+3取反-4

 image-20221119171244778

交换两个数：


```
class Swap2{
    public static void main(String[] args) {
        int a = 10;
        int b = 20;
        System.out.println("a:"+a+",b:"+b);

        //位异或实现
        //左边a,b,a
        //右边a^b
        a = a^b;
        b = a^b; //a^b^b,b=a
        a = a^b; //a^b^a,a=b
        System.out.println("a:"+a+",b:"+b);
    }
}
```

(1) <<: 左移，左边最高位丢弃，右边补0

(2) >>: 右移，最高位是0左边补0，最高位是1，左边补1

(3) >>>: 无符号右移，无论最高位是0还是1，左边都补0

 image-20221119200348142

技巧:

左移：把被移动的数字*2^{移动位数}，例如3 >>2 = 3*2² = 12

右移：把被移动的数字/2^{移动位数}，例如324 >>2 = 24/2² = 6

无符号右移：

正数	和有符号移动相同
负数	按照规则计算即可



面试题

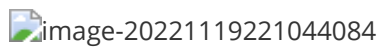
用最有效率的方法计算 2×8

```
class Demo{
    public static void main(String[] args){
        System.out.println(2 << 3);
    }
}
```

(9) 三目运算符

格式： (关系表达式) ? 表达式1 : 表达式2;

如果条件为true，则结果为1，反之为2



6、键盘输入

实现方式：

(1) 导包：放置在class上面

```
import java.util.Scanner
```

(2) 创建键盘录入对象

```
Scanner sc11 = new Scanner(System.in); //创建对象sc
```

(3) 通过对象获取数据

```
int x = sc.nextInt(); //调用对象的方法
```

7、流程控制

顺序结构： 从上往下依次执行

选择结构： 也被称为分支结构，**if** 和 **switch**

表达式结果必须是布尔类型，语句是一条可以省略大括号，多条一定不要省略



```
(1) 单判断
if(表达式){
    语句
}
```

(2) 双判断

某些情况下可以与三元运算符进行转换

```
if(表达式){  
    语句  
}else{  
    语句  
}
```

(3)多判断

一旦某一个else if成立，后面的不再判断

```
if(表达式){  
    语句  
}else if{  
    语句  
}...else{  
    语句  
}
```

循环结构