1. 标识符的命名规范有哪些？

2. Java 的基本数据类型有哪几种，分别是什么。并指出各自占用的内存空间大小

3. 说明基本数据类型变量之间自动类型提升的运算规则。

4. 说明基本数据类型变量之间强制类型转换的使用规则和强转注意的问题。

5. 常用的命令行操作有哪些指令？至少说出 5 个。

1. 标识符的命名规范有哪些？

包名：xxxyyyzzz

类名、接口名：XxxYyyZzz

变量名、方法名：xxxYyyZzz

常量名：XXX_YYY_ZZZ

2. Java 的基本数据类型有哪几种，分别是什么。并指出各自占用的内存空间大小

byte 1 字节 short int long

float 4 double 8

char 2

boolean

3. 说明基本数据类型变量之间自动类型提升的运算规则。

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

4. 说明基本数据类型变量之间强制类型转换的使用规则和强转注意的问题。

使用()。

可能造成精度损失。

5. 常用的命令行操作有哪些指令？至少说出 5 个。

javac 文件名.java
java 类名

cd 路径
cd..
cd/
md
rd
del
dir

3.变量使用的注意点：

- Java中每个变量必须先声明，后使用
- 使用变量名来访问这块区域的数据
- 变量的作用域：其定义所在的一对{ }内
- 变量只在其作用域内才效
- 同一个作用域内，不能定义重名的变量

4.2 自动类型转换(只涉及7种基本数据类型)

当容量小的数据类型变量与容量大的数据类型变量做运算时，结果为容量大的数据类型。

byte、short、char ---> int ---> long ---> float ----> double

特别的，byte、short、char之间做运算，结果为int型。

### 4.3 强制类型转换(只涉及7种基本数据类型）：

是自动类型提升运算的逆运算，将容量大的数据类型的变量转换为容量小的数据类型的变量。

需要使用强转符：()。

### 4.4 String与8种基本数据类型间的运算

1.String属于引用数据类型中的类，不是基本数据类型

2.使用一对""来声明

3.String可以与8种基本数据类型的变量做连接(+)运算，运算的结果为：String型

练习：

```
int num = 10;

String s = num;//编译不通过

String s1 = num + "";//编译通过

System.out.println(s1);//"10"

int nu = (int)s1;//编译不通过

int nu1 = Integer.parseInt(s1);//编译通过

int nu2 = nu1 + 1;

System.out.println(nu2);//11
```

## 2.二进制的使用说明：

2.1 计算机底层的数据都是使用二进制保存的

2.2 对于整数来讲，计算机底层都是使用整数对应的补码来保存的。

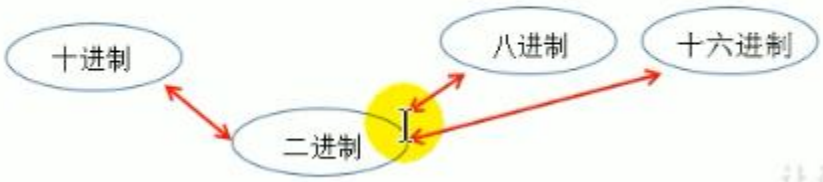
> 对于正数来讲，原码、反码、补码三码合一。

> 对于负数来讲，原码：直接将一个数值换成二进制数。最高位是符号位

负数的反码：是对原码位取反，只是最高位（符号位）确定为1。
负数的补码：其反码加1。

实现二进制与十进制之间的转换：


3.进制间的转换：




## 2.4.3 运算符：比较运算符

| 运算符 | 运算 | 范例 | 结果 |
|---|---|---|---|
| == | 相等于 | 4==3 | false |
| != | 不等于 | 4!=3 | true |
| < | 小于 | 4<3 | false |
| > | 大于 | 4>3 | true |
| <= | 小于等于 | 4<=3 | false |
| >= | 大于等于 | 4>=3 | true |
| instanceof | 检查是否是类的对象 | “Hello” instanceof String | true |

- 比较运算符的结果都是boolean型，也就是要么是true，要么是false。
- 比较运算符“==”不能误写成“=”。

## 2.4.3 运算符：比较运算符

思考1：

```
boolean b1 = false;
//区分好==和=的区别。
if(b1==true)
    System.out.println("结果为真");
else
    System.out.println("结果为假");
```

## 2.4.4 运算符：逻辑运算符

&—逻辑与　　|—逻辑或　　！—逻辑非
&& —短路与　　|| —短路或　　^ —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 |

```
//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
```

```
// | 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 ||
```

## 2.4.4 运算符：逻辑运算符

练习：请写出每题的输出结果

```
int x = 1;
int y=1;

if(x++==2 & ++y==2){
        x =7;
}
System.out.println("x="+x+",y="+y);
```

```
int x = 1,y = 1;

if(x++==2 && ++y==2){
        x =7;
}
System.out.println("x="+x+",y="+y);
```

```
int x = 1,y = 1;

if(x++==1 | ++y==1){
        x =7;
}
System.out.println("x="+x+",y="+y);
```

```
int x = 1,y = 1;

if(x++==1 || ++y==1){
        x =7;
}
System.out.println("x="+x+",y="+y);
```



## 2.4.4 运算符：逻辑运算符

【面试题】程序输出：

```
1.  class Test {
2.     public static void main (String [] args) {
3.          boolean x=true;
4.          boolean y=false;
5.          short z=42;
6.           //if(y == true)
7.          if((z++==42)&&(y=true))z++;
8.          if((x=false) || (++z==45)) z++;
9.
10.         System. out.println("z="+z);
11.         }
12.  }
```

结果为：

z= 46

## 2.4.5 运算符：位运算符

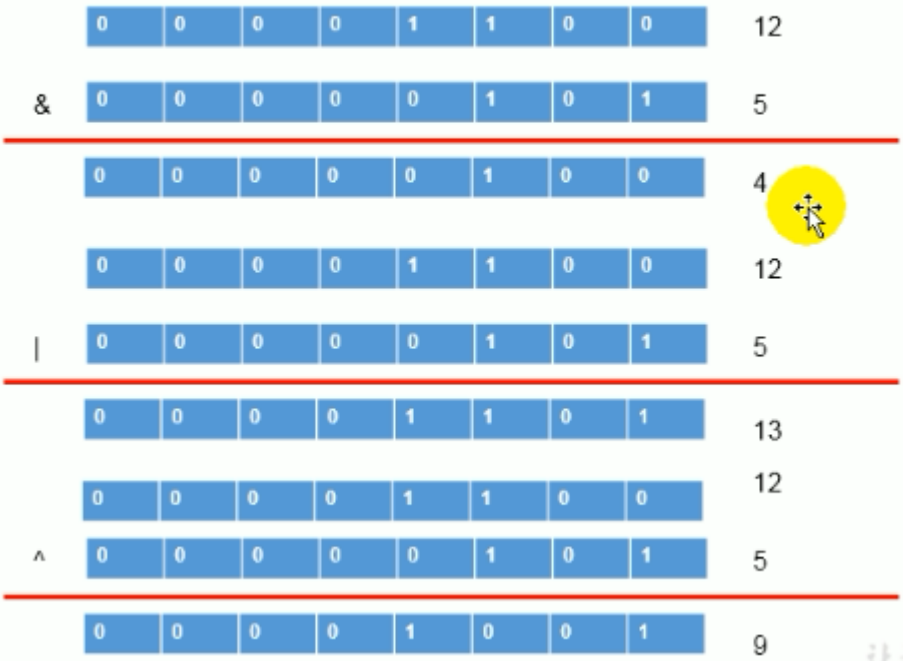
注意：无<<<

| 位运算符 | | |
|---|---|---|
| 运算符 | 运算 | 范例 |
| << | 左移 | 3 << 2 = 12 --> 3*2*2=12 |
| >> | 右移 | 3 >> 1 = 1 --> 3/2=1 |
| >>> | 无符号右移 | 3 >>> 1 = 1 --> 3/2=1 |
| & | 与运算 | 6 & 3 = 2 |
| \| | 或运算 | 6 \| 3 = 7 |
| ^ | 异或运算 | 6 ^ 3 = 5 |
| ~ | 取反运算 | ~6 = -7 |

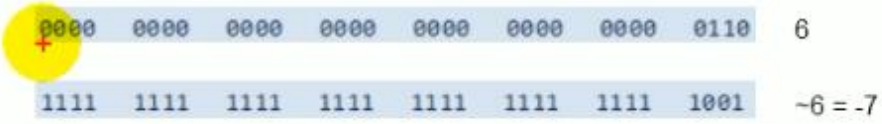
- 位运算是直接对整数的二进制进行的运算

```
/*
5 - 位运算符的使用：<<  >>  >>>  !  &  ^  ~

说明：
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。

*/
```

## 2.4.5 运算符：位运算符




## 2.4.5 运算符：位运算符



## 2.4.6 运算符：三元运算符

- 格式：
  - (条件表达式)?表达式1：表达式2；
    - 为true，运算后的结果是表达式1；
    - 为false，运算后的结果是表达式2；
  - 表达式1和表达式2为同种类型
  - 三元运算符与if-else的联系与区别：
    1）三元运算符可简化if-else语句
    2）三元运算符要求必须返回一个结果。
    3）if后的代码块可有多个语句

练习：获取两个数中的较大数
获取三个数中的较大数

```
/*
6- 三元运算符的使用：(条件表达式)? 表达式1 : 表达式2;

说明：
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。
*/
```

## 2.4.6 运算符：三元运算符

尚硅谷

### 运算符的优先级

- 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。
- 只有单目运算符、三元运算符、赋值运算符是从右向左运算的。

| | |
|---|---|
| | . () {} ; , |
| R—>L | ++ -- ~ !(data type) |
| L—>R | * / % |
| L—>R | + - |
| L—>R | << >> >>> |
| L—>R | < > <= >= instanceof |
| L—>R | == != |
| L—>R | & |
| L—>R | ^ |
| L—>R | \| |
| L—>R | && |
| L—>R | \|\| |
| R—>L | ? : |
| R—>L | = *= /= %= |
| | += -= <<= >>= |
| | >>>= &= ^= \|= |

高

低

## 2.5 程序流程控制

- 流程控制语句是用来控制程序中各语句执行顺序的语句，可以把语句组合成能完成一定功能的小逻辑模块。

- 其流程控制方式采用结构化程序设计中规定的三种基本流程结构，即：
  - 顺序结构
  - 分支结构
  - 循环结构

## 2.5 程序流程控制




顺序

分支

## 2.5.1 程序流程控制：顺序结构

# 2.5.1 顺序结构

●顺序结构

Java中定义成员变量时采用合法的前向引用。如：

```
public class Test{
        int num1 = 12;
        int num2 = num1 + 2;
}
```

错误形式：

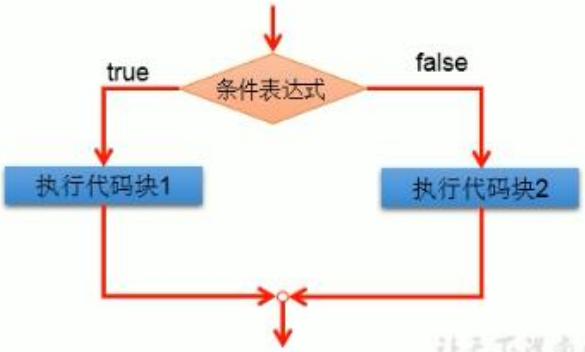
```
public class Test{
        int num2 = num1 + 2;
        int num1 = 12;
}
```


## 2.5.2 程序流程控制：if-else结构

**if语句三种格式：**

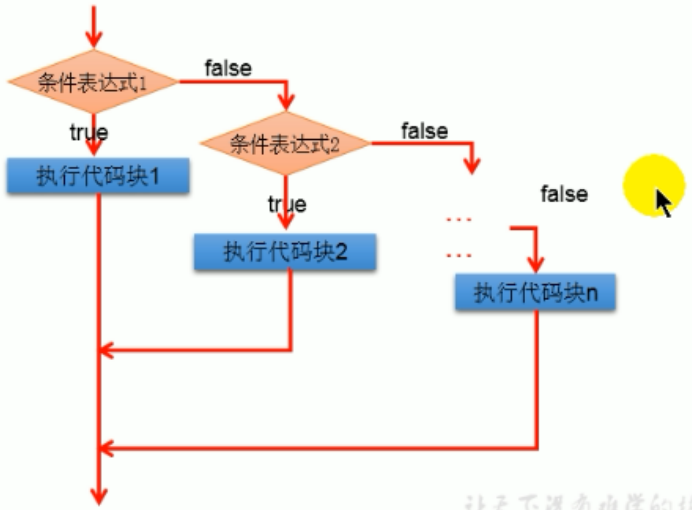
1. if(条件表达式){
        执行代码块；
    }


2. if(条件表达式){
        执行代码块1;
    }
    else{
        执行代码块2;
    }



## 2.5.2 程序流程控制：if-else结构

3. if(条件表达式1){
   执行代码块1;
   }
   else if (条件表达式2){
   执行代码块2;
   }
   ……
   else{
   执行代码块n;
   }






## 2.5.2 程序流程控制：if-else结构

### 分支结构：if-else使用说明

- 条件表达式必须是布尔表达式（关系表达式或逻辑表达式）、布尔变量
- 语句块只有一条执行语句时，一对{}可以省略，但建议保留
- if-else语句结构，根据需要可以嵌套使用
- 当if-else结构是“多选一”时，最后的else是可选的，根据需要可以省略
- 当多个条件是“互斥”关系时，条件判断语句及执行语句间顺序无所谓
  当多个条件是“包含”关系时，“小上大下 / 子上父下”

```
/*
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
*/
```

# if语句练习1

1)对下列代码，若有输出，指出输出结果。

```
int x = 4;
int y = 1;
if (x > 2) {
    if (y > 2)
        System.out.println(x + y);
        System.out.println("atguigu");
} else
    System.out.println("x is " + x);
```

2)

```
boolean b = true;
//如果写成if(b=false)能编译通过吗？如果能，结果是？
if(b == false)
        System.out.println("a");
else if(b)
        System.out.println("b");
else if(!b)
        System.out.println("c");
else
        System.out.println("d");
```