1. 标识符由数字、字母、下划线\_、美元符$，不能以数字开头，不能是关键字，区分大小写。

2. byte、short、char在运算的时候，都会提升为int，再进行运算。

3. 常量计算 byte a = 3 + 4; java存在常量优化机制，编译时，会将3和4计算出一个7的结果，并且会自动判断该结果是否在byte取值范围内，在:编译通过，不在:编译失败。

4. 数组定义格式：int[] type 或 int type[]

5. 创建数组动态初始化：int[] arr = new int[3]，定义了三个长度的数组，下标从0开始，不赋值默认0，arr[0]=1进行赋值，int[] arr进栈并保存指向堆的引用，new int[3]进堆，并生成地址值。

静态初始化：int[] arr = new int[]{1, 2, 3} 或 int[] arr = {1, 2, 3}

6. 栈内存：存放局部变量及堆中的地址引用，如果是方法只有在调用时从方法区进入栈内存中，先进后出；

堆内存：new出来的内容都会进入堆内存，并且会存在地址值；

方法区：字节码文件（.class文件->类文件）加载时进入的内存及包含类中的方法；

7. 产生随机数：Random r = new Random(); int i = r.nextInt(5) + 5; 随机产生[0, 5)之间的数据，生成出来的再加5，为最终结果。

8. 方法重载：方法名相同，参数个数/类型/顺序不同，跟返回值无关。

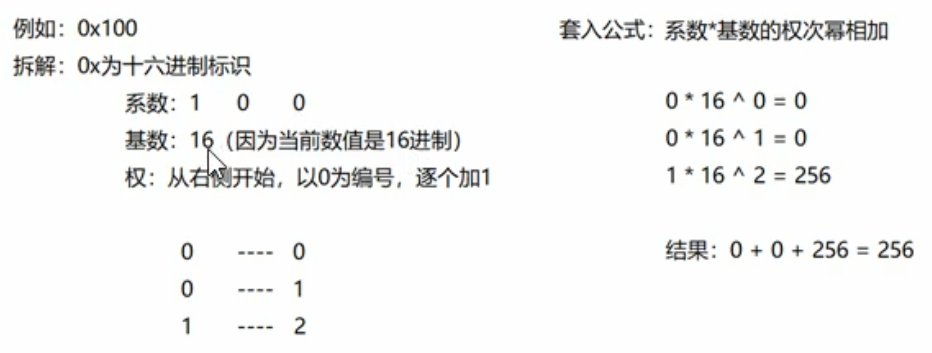
9. 十进制：Java中，数值默认都是10进制，不需要加任何修饰；

二进制：数值前面以0b开头，b大小写都可以；

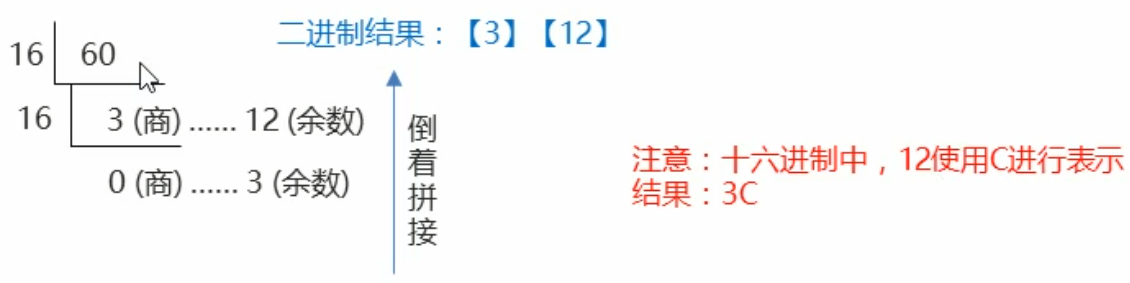
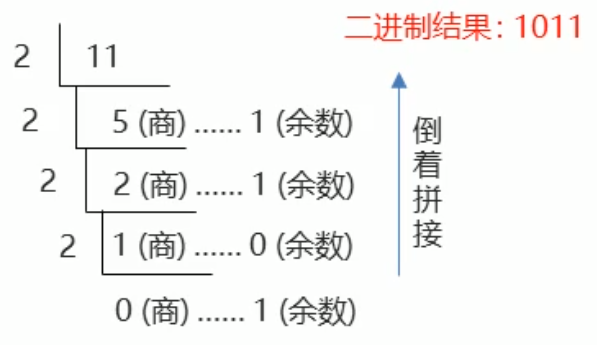
八进制：数值前面以0开头；

十六进制：数值前面以0x开头，x大小写都可以；

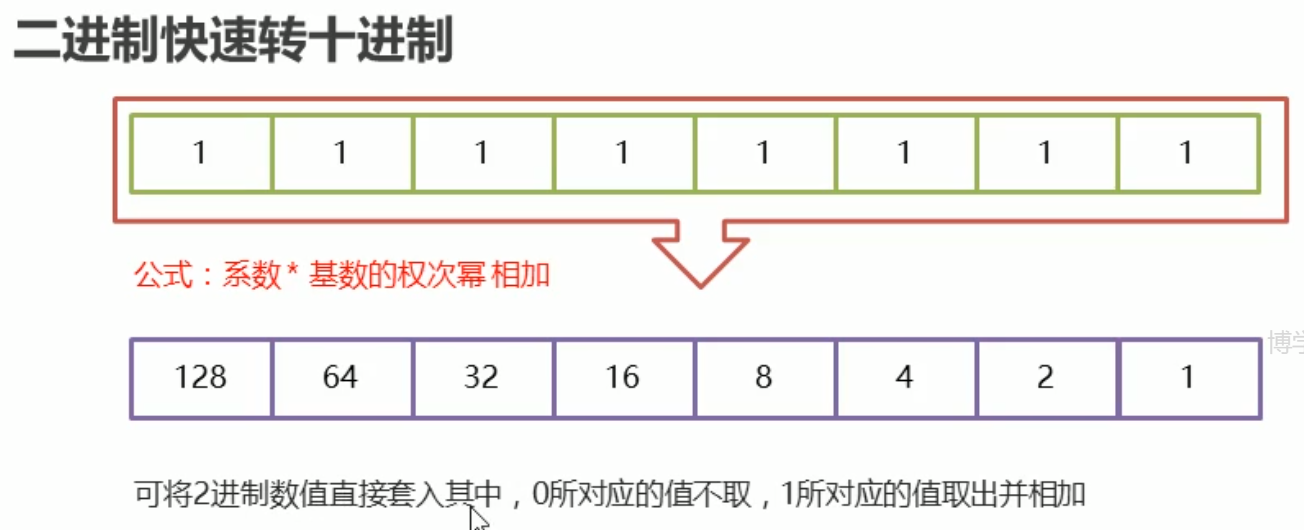
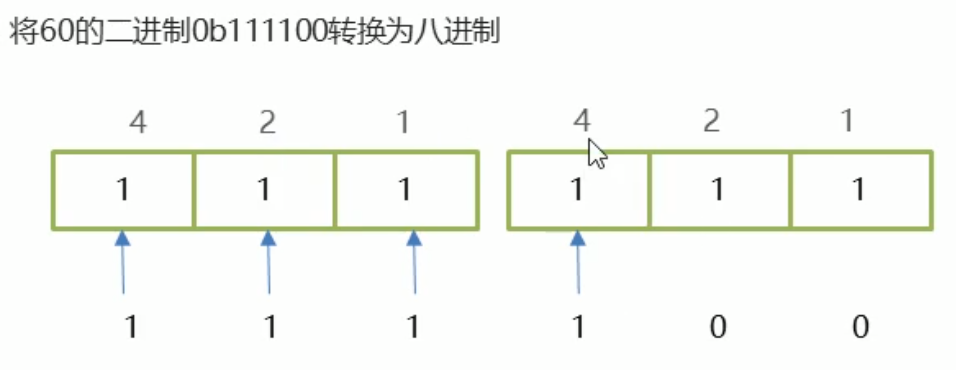
10. 任意进制转换为十进制：

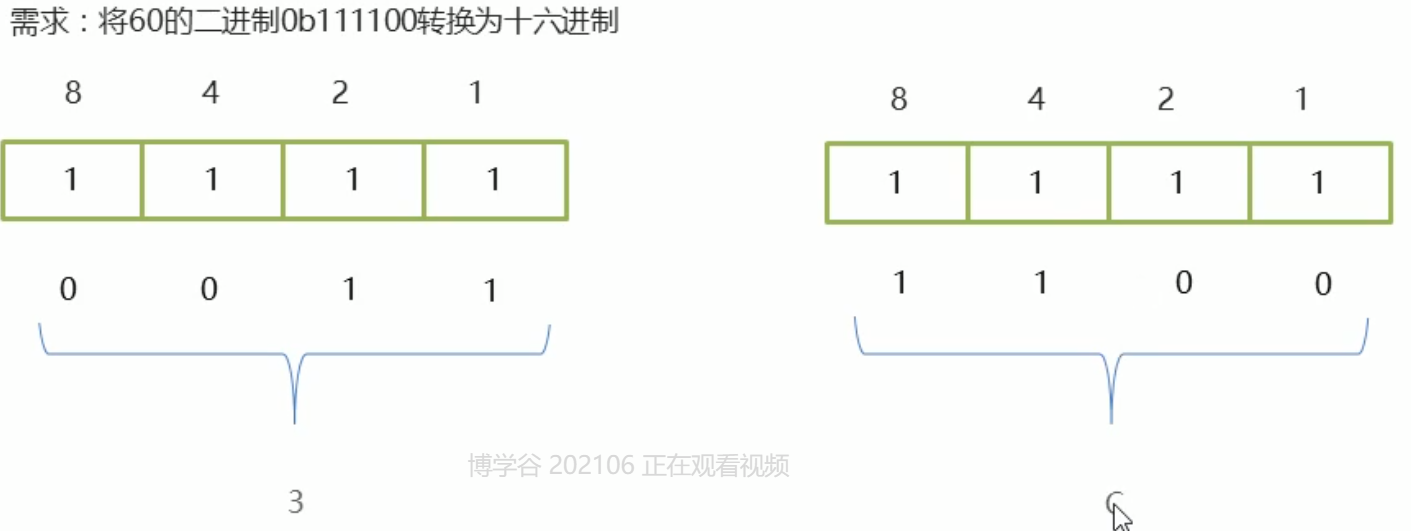


11. 十进制转换为任意进制：



12. 二进制转其他进制：



13. 位移运算：

<<：有符号左移运算，左边符号位丢弃，右边补齐0；

>>：有符号右移运算，根据符号位，补齐左边；

>>>：无符号右移，无论符号位是0还是1，都补0；

14. 异或^：一个数被另一个数异或两次，结果还是本身；如：10^20^20=10;

15. 二维数组：int[][] arr=new int[3][4]; 创建了包含3个一维数组并且数组元素4个的二维数组，赋值：int[0][0]=11，或int[][] arr={{11, 22, 33}, {44, 55, 66}}; 遍历：

**for** (**int** i=0; i<array.**length**; i++) {  
 **for** (**int** j=0; j<array[i].**length**; j++) {  
 System.***out***.println(array[i][j]);  
 }  
}

16. 集合和数组的区别：

①数组的长度不可变，集合的长度可变；

②数组可以存基本数据类型和引用数据类型；集合只能存引用数据类型，如果存基本数据类型，需要存对应的包装类；

17. Collection接口：包含List(有序，可重复)和Set(无序，不可重复)单列结合，继承了Iterator接口，可以使用增强for循环。

18. Iterator迭代器：next()获取当前位置的元素，将迭代器对象移向下一个索引的位置；

19. for使用场景：操作索引使用普通for; 删除元素使用迭代器; 只是遍历使用增强for;

20. 栈：先进后出，队列：先进先出；

21. 数组：通过地址值和索引定位，查询速度快；添加/删除元素时，后面每个元素后移/前移，增删效率低；

链表：每个元素称为结点，结点又是一个独立的对象，结点以存储具体的数据和下一个结点的地址组成，第一个结点称为head头结点(不存储值)，没有数据结点指向空地址，前一个结点记录后一个结点的地址值形成链表。增删快，查询需要head头结点往后查，所以查询慢；

22．ArrayList源码：当new创建时长度为0的数组，只有调用add方法才会创建一个长度为10的数组并且初始化为null，数组名为elementData，底层定义了变量size，功能：要操作的索引和元素的个数，当10元素的数组添加满之后，就需要自动扩容，扩容到1.5倍长度，会将旧数组的元素拷贝到新数组中，然后对新数组进行操作；