## Java SE 知识点汇总

## 运算符考核点

|  |  |
| --- | --- |
| & | | 是逻辑运算符也是位运算符。  其实不要想这么多，两边是boolean型 就是逻辑运算。两边是int 就是位运算。  位与( & ) 、位或( | )、位非( ~ )、位异或( ^ )。从表面上看似乎有点像逻辑运算符，但逻辑运算符是针对两个关系运算符来进行逻辑运算（比如条件1&条件2），而位运算符主要针对两个二进制数的位进行逻辑运算。 |
|  | Java中的位运算\移位运算类似的考核题目需要先转换为补码，然后进行运算，再把运算完的补码转换为原码，最后把2进制的原码转换为10进制数字 |
| 优先级 | ()级别最高，=赋值级别最低，  单目优先级【例如++】>双目运算【例如a\*b】>三目运算 |
| ，&与&&，|与||的区别： | &&和&都是表示与，区别是&&只要第一个条件不满足，后面条件就不再判断。而&要对所有的条件都进行判断。 ||和|都是表示“或”，区别是||只要满足第一个条件，后面的条件就不再判断，而|要对所有的条件进行判断。 |
|  | 位与：第一个操作数的的第n位于第二个操作数的第n位如果都是1，那么结果的第n为也为1，否则为0  位或操作：第一个操作数的的第n位于第二个操作数的第n位 只要有一个是1，那么结果的第n为也为1，否则为0  位异或：第一个操作数的的第n位于第二个操作数的第n位 相反，那么结果的第n为也为1，否则为0  位非：操作数的第n位为1，那么结果的第n位为0，反之。 |
| 位移运算符 | value << num（左移运算符<<）：丢弃最高位，0补最低位  如对int型移动33位，实际上只移动了32=1位  value >> num（右移运算符>>）：  符号位不变，左边补上符号位  value >>> num （无符号右移运算符>>>）  忽略了符号位扩展，0补最高位 |
|  | 浮点型不能做switch的参数。 |
|  |  |

## 编码字符集

|  |  |
| --- | --- |
| 十六进制与十进制的换算： | 0x123 —> 1 16^2 + 2 \* 16 + 3 = 291 |
| 原码、反码和补码 | 正数：它的原码、反码、补码相同 负数：反码符号位不变，其余位数取反, 补码:反码+1 |
| 转码 | 硬转码写法为：  **String show = new String(oldString.getBytes(“GBK”)， “UTF-8”)** |
|  | unicode是一个定长的编码标准，每个字符都是2个字节，也就是1个char类型的空间。所以 char c = ‘中’ 合法  Java是以unicode字符集作为编码的  ( JVM内部使用的是UTF-16编码）  Unicode是编码字符集，而UTF-8就是字符编码，即Unicode规则字库的一种实现形式 |

## 异常

|  |  |
| --- | --- |
| 可以创建异常类的实例对象通过throw语句抛出 | **public void doMethod() {**  **System.out.println("123");**  **throw new RuntimeException("1234");**  **}** |
| finally | return语句并不是函数的最终出口，如果有finally语句，这在return之后还会执行finally（**return的值会暂存在栈里面**，等待finally执行后再返回） |
| finally与return | 情况三：**如果finally中没有return语句，但是改变了要返回的值**，这里有点类似与引用传递和值传递的区别，分以下两种情况，：  如果return的数据是基本数据类型，则在finally中对该基本数据的改变不起作用，try中的return语句依然会返回进入finally块之前保留的值。  如果return的数据是引用数据类型，而在finally中对该引用数据类型的属性值的改变起作用，try中的return语句返回的就是在finally中改变后的该属性的值。 |
|  |  |