1.java的关键字（keyword）有多少个？

   51+2个保留字=53个关键字(java的关键字都是小写的！！)

2.java的保留字(reserve word)有多少个？问题：分别是什么？

     2个保留字

     Java语言的的保留字是指预留的关键字

   1).const  有道释义：n. 常量，常数

            用于修改字段或局部变量的声明。它指定字段或局部变量的值是常数，不能被修改

   2).goto   有道释义：vi. 转到

         指定跳转到标签，找到标签后，程序将处理从下一行开始的命令。

3.java的关键字分别是什么，作用是什么？

   1).访问修饰符的关键字（共3个）

|  |  |  |
| --- | --- | --- |
| 关键字 | 意思 | 备注，常用 |
| public | 公有的 | 可跨包，（默认选择） |
| protected | 受保护的 | 当前包内可用 |
| private | 私有的 | 当前类可用 |

    2).定义类、接口、抽象类和实现接口、继承类的关键字、实例化对象（共6个）

|  |  |  |
| --- | --- | --- |
| 关键字 | 意思 | 备注，常用 |
| class | 类 | public class A(){}  花括号里有已实现方法体，类名需要与文件名相同 |
| interface | 接口 | public interface B(){}  花括号里有方法体，但没有实现，方法体句子后面是英文分号“:”结尾 |
| abstract | 声明抽象 | public abstract class C(){} 介于类与接口中间，可以有也可以没有已经实现的方法体 |
| implements | 实现 | 用于类或接口实现接口public class A  interface B(){} |
| extends | 继承 | 用于类继承类 public class A extends D(){} |
| new | 创建新对象 | A a=new A();  A表示一个类 |

   3).包的关键字（共2个）

|  |  |  |
| --- | --- | --- |
| 关键字 | 意思 | 备注，常用 |
| import | 引入包的关键字 | 当使用某个包的一些类时，仅需类名 然后使用ctrl+shift+o或者选定类名（类或属性或方法）按住ctrl+单击 即可自动插入类所在的包。如：JFrame 快捷键之后自动加入  **import** javax.swing.JFrame; |
| package | 定义包的关键字 | 将所有有关的类放在一个包类以便查找修改等。如：**package** javake.flycat.draw002; |

   4).数据类型的关键字（共12个）

|  |  |  |
| --- | --- | --- |
| 关键字 | 意思 | 备注，常用 |
| byte | 字节型 | 8bit |
| char | 字符型 | 16bit |
| boolean | 布尔型 | -- |
| short | 短整型 | 16bit |
| int | 整型 | 32bit |
| float | 浮点型 | 32bit |
| long | 长整型 | 64bit |
| double | 双精度 | 64bit |
| void | 无返回 | public void A(){}   其他需要返回的经常与return连用 |
| null | 空值 |  |
| true | 真 |  |
| false | 假 |  |

   5).条件循环（流程控制）（共12个）

|  |  |  |
| --- | --- | --- |
| 关键字 | 意思 | 备注，常用 |
| if | 如果 | if(){} 如果小括号里面怎么怎么样 花括号就怎么怎么样 |
| else | 否则，或者 | 常与if连用，用法相同 |
| while | 当什么的时候 | while 怎么样就do什么    while(){} |
| for | 满足三个条件时 | for ( ; ; ){} |
| switch | 开关 | switch(表达式) { case 常量表达式1:语句1; .... case 常量表达式2:语句2; default:语句; } default就是如果没有符合的case就执行它,default并不是必须的. case后的语句可以不用大括号. switch语句的判断条件可以接受int,byte,char,short,不能接受其他类型. |
| case | 返回开关里的结果 |  |
| default | 默认 |  |
| do | 运行 | 长与while连用 |
| break | 跳出循环 |  |
| continue | 继续 | 中断本次循环，并并开始下一次 |
| return | 返回 | return 一个返回值类型 |
| instanceof | 实例 | 一个二元操作符，和==，>，<是同一类的。测试它左边的对象是否是它右边的类的实例，返回boolean类型的数据 |

   6).修饰方法、类、属性和变量（共9个）

|  |  |  |
| --- | --- | --- |
| 关键字 | 意思 | 备注，常用 |
| static | 静态的 | 属性和方法都可以用static修饰，直接使用类名.属性和方法名。  只有内部类可以使用static关键字修饰，调用直接使用类名.内部类类名进行调用。   static可以独立存在。静态块 |
| final | 最终的不可被改变的 | 方法和类都可以用final来修饰  final修饰的类是不能被继承的  final修饰的方法是不能被子类重写。  常量的定义：  final修饰的属性就是常量。 |
| super | 调用父类的方法 | 常见**public** **void** paint(Graphics g){  **super**.paint(g); ··· } |
| this | 当前类的父类的对象 | 调用当前类中的方法（表示调用这个方法的对象）  this.addActionListener(al):等等 |
| native | 本地 |  |
| strictfp | 严格,精准 |  |
| synchronized | 线程,同步 |  |
| transient | 短暂 |  |
| volatile | 易失 |  |

  static 例子：

          public class Test{

                  class A{}   //内部类

            ActionListener al=new ActionListener(···){}   //匿名内部类

          }

   静态块优先于其他方法/类的执行

    7).错误处理（共5个）

|  |  |  |
| --- | --- | --- |
| 关键字 | 意思 | 备注，常用 |
| catch | 处理异常 | 1.try+catch 程序的流程是：运行到try块中，如果有异常抛出，则转到catch块去处理。然后执行catch块后面的语句  2.try+catch+finally 程序的流程是：运行到try块中，如果有异常抛出，则转到catch块,catch块执行完毕后，执行finally块的代码，再执行finally块后面的代码。 如果没有异常抛出，执行完try块，也要去执行finally块的代码。然后执行finally块后面的语句  3.try+finally 程序的流程是：运行到try块中,如果有异常抛出的话，程序转向执行finally块的代码。那末finally块后面的代码还会被执行吗？不会！因为你没有处理异常，所以遇到异常后，执行完finally后，方法就已抛出异常的方式退出了。 这种方式中要注意的是，由于你没有捕获异常，所以要在方法后面声明抛出异常  (来自网上的资料) |
| try | 捕获异常 |  |
| finally | 有没有异常都执行 |  |
| throw | 抛出一个异常对象 | 一些可以导致程序出问题的因素,比如书写错误,逻辑错误或者是api的应用错误等等. 为了防止程序的崩溃就要预先检测这些因素,所以java 使用了异常这个机制.  在java中异常是靠 "抛出" 也就是英语的"throw" 来使用的,意思是如果发现到什么异常的时候就把错误信息 "抛出" |
| throws | 声明一个异常可能被抛出 | 把异常交给他的上级管理，自己不进行异常处理 |

   throw是你执行的动作。比如你觉得可能有异常，那么就抱出去 如：

           String a; if(a == null),

            throw new exception("a为null");

           所以throw是一个抛出去的动作

   throws只用在一个方法的末端，表示这个方法体内部如果有异常，这抛给它的调用者。 如： public void add(int a, int b) throws Exception(); 这个方法表示，在执行这个方法的时候，可能产生一个异常，如果产生异常了，那么谁调用了这个方法，就抛给谁。（来自百度）

   8).不知道是什么（共2个）

|  |  |  |
| --- | --- | --- |
| 关键字 | 意思 | 备注，常用 |
| enum | 枚举，列举型别 |  |
| assert | 断言 |  |

 Enum:代表一组常用常量，可用来代表一类相同类型的常量值如：

性别：

public enum SexEnum {

  male, female;

}

颜色：

public enum Color {

  RED, BLUE,GREEN,BLACK;

}

枚举对象里面的值都必须是唯一的。

**附：**throw 和throws的差别

区别一：

             throw 是语句抛出一个异常；throws 是方法抛出一个异常；

              throw语法：throw <异常对象>

              在方法声明中，添加throws子句表示该方法将抛出异常。

             throws语法：[<修饰符>]<返回值类型><方法名>（[<参数列表>]）[throws<异常类>]

             其中：异常类可以声明多个，用逗号分割。

区别二：

          throws可以单独使用，但throw不能；

区别三：

            throw要么和try-catch-finally语句配套使用，要么与throws配套使用。但throws可以单独使用，然后再由处理异常的方法捕获。

throws E1,E2,E3 只是告诉程序这个方法可能会抛出这些个异常，方法的调用者可能要处理这些异常。而这些异常E1，E2，E3可能是该函数体产生的。

而throw是明确之处这个地方要抛出这个异常。

void doA() throws Exception1, Exception3 {

  try {

    ……

  } catch(Exception1 e) {

    throw e;

  } catch(Exception2 e) {

    System.out.println("出错了");

  }

  if (a != b)

    throw new Exception3("自定义异常");

}

代码块……中可能产生异常Exception1、Exception2和Exception3。

如果产生Exception1异常，则捕捉了之后抛出由该方法的调用者去做处理；

如果产生Exception2异常，则该方法自己做了处理(打印出了说出错了)，所以该方法就不会再向外抛出Exception2异常了，void doA() throws Exception1,,Excpetion3里面的Exception2也就不用写了；

而Exception3异常是该方法的某段逻辑出错，程序员自己作了处理在该段逻辑错误的情况下抛出异常Exception3，则调用者也需要处理。

throw语句用在方法体内,表示抛出异常,由方法体内的语句处理

throws语句用在方法声明后面,表示再抛出异常,由调用这个方法的上一级方法中的语句来处理

throws主要是声明这个方法会抛出这种类型的异常，使其他地方调用它时知道要捕获这个异常。

throw是具体向外抛异常的动作，所以它是抛出一个异常实例。

throws说明你有哪个可能，倾向

throw的话，那就是你把那个倾向变成真实的了

同时：

1）throws出现在方法函数头；而throw出现在函数体；

2）throws表示出现异常的一种可能性，并不一定会发生这些异常；throw则是抛出了异常，执行throw则一定抛出了某种异常；

3）两者都是消极处理异常的方式（这里的消极并不是说这种方式不好），只是抛出或者可能抛出异常，但是不会由函数去处理异常，真正的处理异常由函数的上层调用处理。