1 静态代码块不能放在方法里面只能放在类里面。

2 先执行父亲的构造代码块和构造方法再执行子类的构造代码块和构造方法。但是同一个类中构造代码块优先于构造函数。

3 方法是最小的封装。方法多了用类来封装，类多了用包来封装。封装分为这三种级别。

4 java底层封装的是c++。

5 在实际开发中final一般只用于修饰成员变量使之变成常量。

6 同名的类的区分方法是放在不同的包里面，util包是最常用的，lang包是不需要导包的。

7 protected只用于修饰成员变量，是为子类准备的。

8 匿名内部类的唯一作用是简化书写。

9 ctrl+鼠标左键查看方法的源码。

10 Object提供的equals()比较的还是两个对象的地址值。

11 重写方法一定要复制粘贴。返回值和参数类型必须与原方法相同。

12 字符串不能用==（比较的是地址值），基本数据类型可以用==

13 系统可以自动写equals方法，右键-sourse-hashcode and equals。自动写toString同理。

14 equals默认比较所有属性，要是只想比较某些属性的话在13中只勾选你想要的即可。

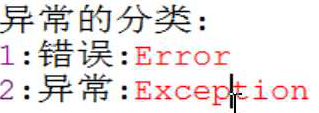
15 打印对象时的默认输出格式（地址）（在打印对象的时候即使不调用toString()也会自动调用）：



16 系统中定义的类已经重写过toString方法。只有我们的自定义类中才需要重写toString。

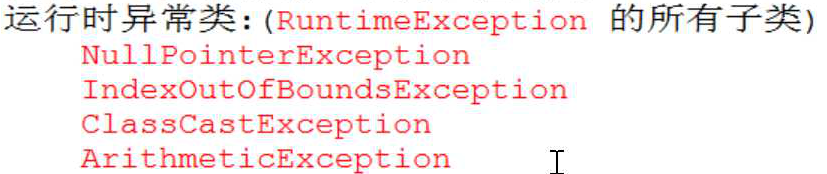
17 ctrl+1+回车可以根据右边补全左边。

18 异常的分类：



错误是无法解决的，我们不关心。

19 **异常分为运行时异常**（例如除数为0），这是由于代码不严谨导致的，直接修改代码即可，不要处理异常。**还有编译时异常**（例如FileReader），这种异常要处理。今天的重点是编译时异常。



20 所有的异常都对应一个类，所以异常是有继承体系的。

21 catch捕捉异常之后程序可以往下进行。

22 当发生异常时，系统会自动new一个相应的异常对象并抛出。接下来看是否能接受这个异常对象（看看有没有人引用这个对象）。若没有人要这个抛出的异常对象的话就要抛出给jvm，就会停止运行程序。

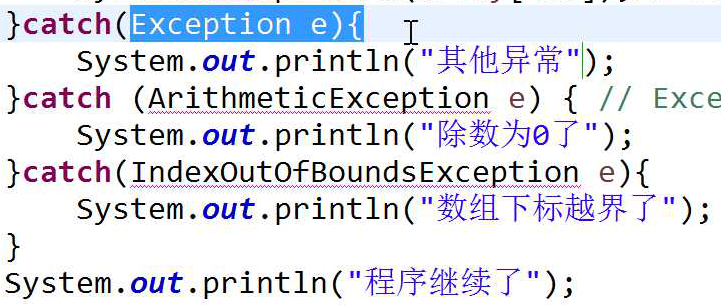
23 异常的捕捉中也用到多态，即父异常也行。

24 写出try之后alt+反斜杠自动补全catch。

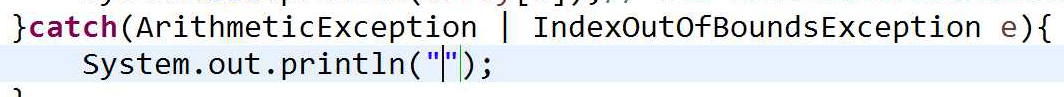
25 只有发生异常才会进入catch，一个try可以对应多个catch。

26 try语句里面有多个语句发生异常时，一旦有一个语句发生异常，try中的后面的语句都不会再执行。

27 当前面的异常是后面异常的父类时，会报错。所以**父异常应该放在最后。**



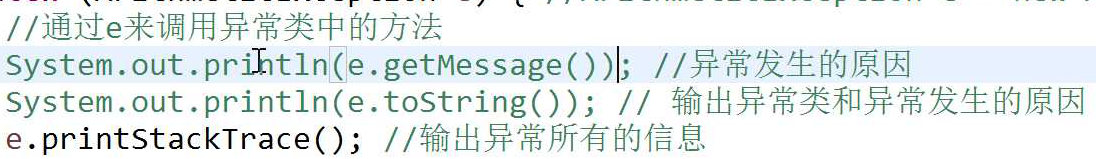
28 catch语句中可以写多个异常：



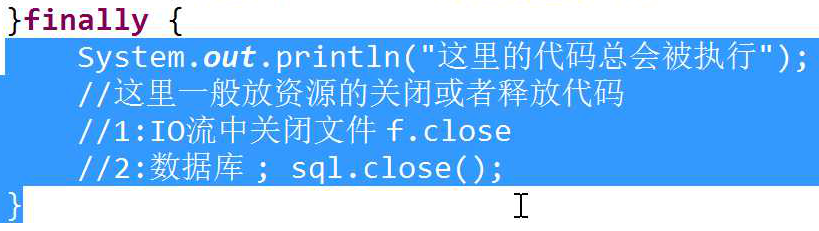
**这是异常里面的或，不是普通的或。**

29 异常是类，类中有方法，并且子类异常可以使用父类异常里的方法。

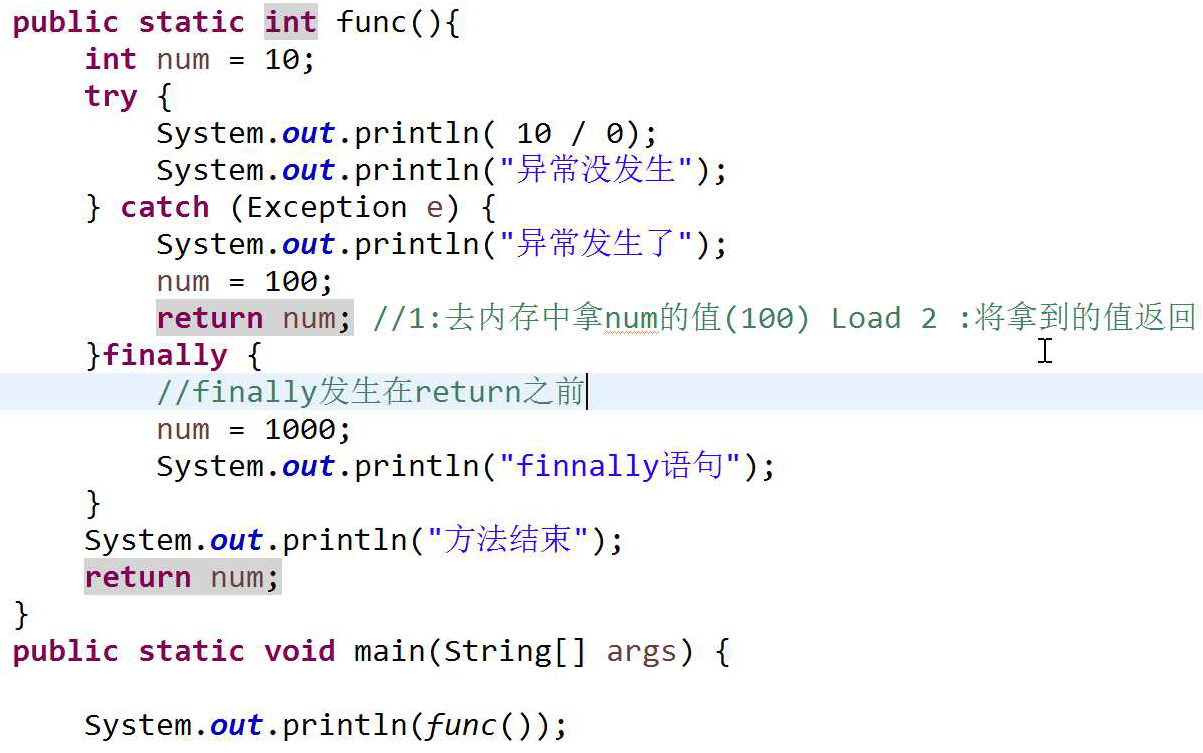
30 调用异常类中的常见方法：



31 finally中一般放资源的关闭代码（资源的释放代码）。



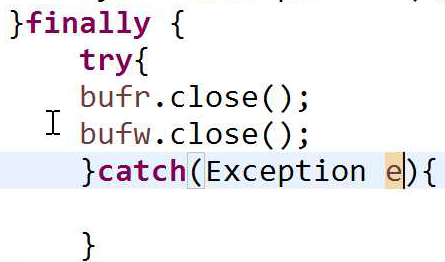
32 面试题



打印的是100而不是1000，因为return分好几步执行（与汇编有关）。先return拿到100再执行finally将内存中的num改成1000，再返回num。由于已经从内存中拿到了100，所以改成1000也没用了。

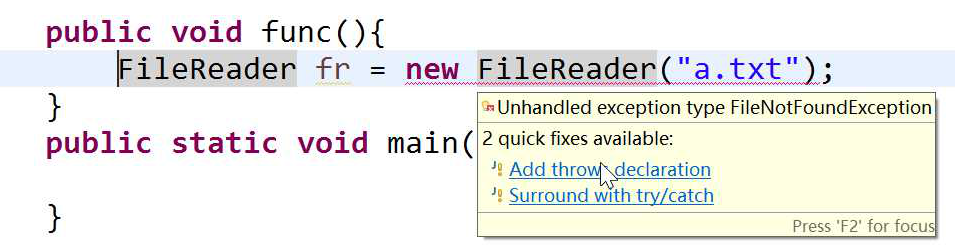
33 抛出异常而不处理的话程序出错会直接崩掉。

34 异常的嵌套：



35 异常会让代码变得复杂。

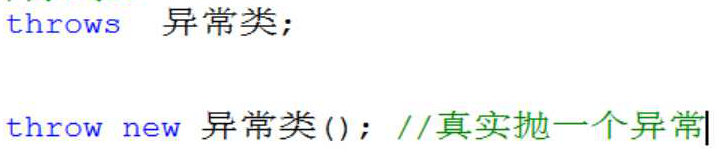
36 系统可以自动加try catch，只需要在红色横线报错的时候点击选项就可以自动加try catch。



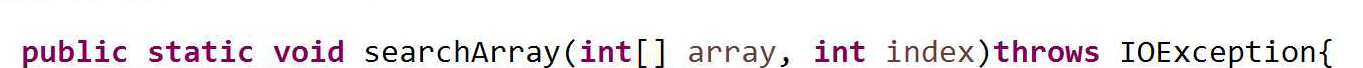
37 throws一般用在方法上，表示该方法有可能产生异常。

38 真是开发中要写try而不是throws。

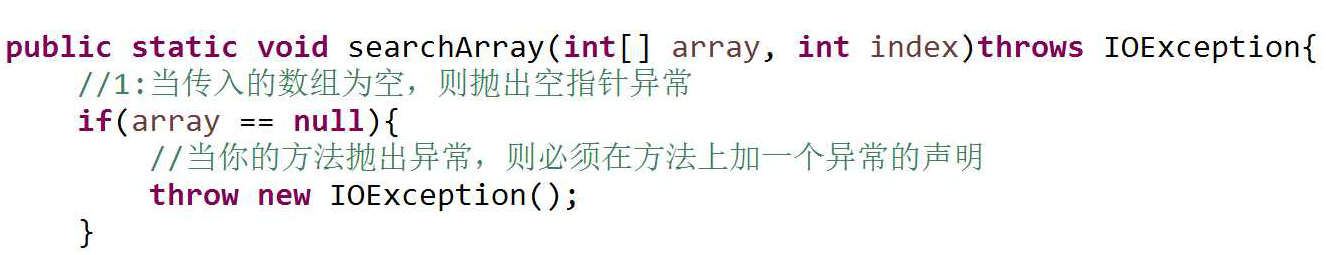
39 区分throw和throws。



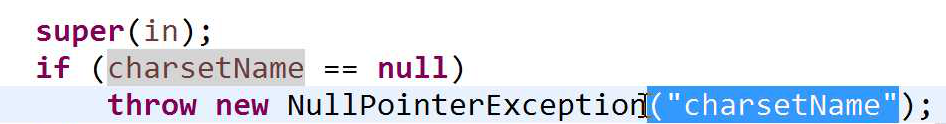
40 注意抛出异常的声明位置：



41 自己抛出的异常也要在方法上声明（除非是运行时异常，不用抛出）：

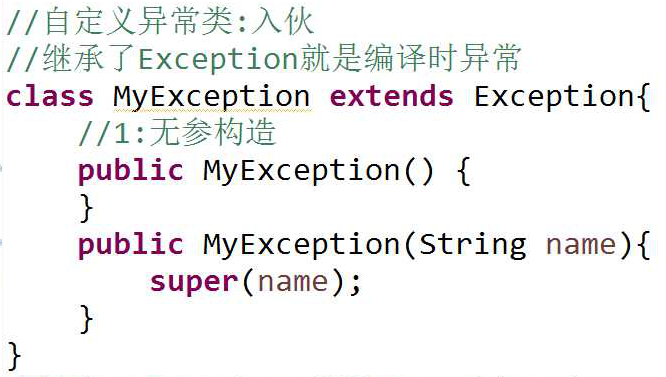


42 异常中还可以传入参数：



43 自定义异常：

自定义异常类要继承其他系统异常，表示你自己的身份以及异常的类型。



Super(name)是因为父类知道怎么处理参数。有参构造并传入参数的结果如下：



要只想得到参数不想得到其他的信息，用getMessage()方法。

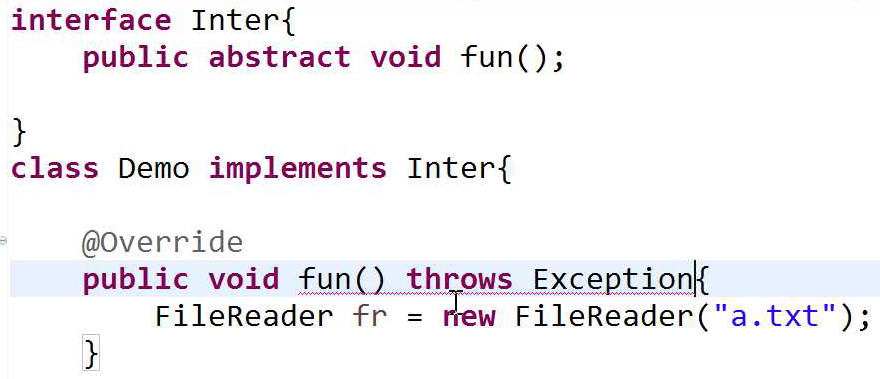
44 自定义方法并自定义抛出异常的写法（当然也可以抛出系统异常，即非自定义异常。）：



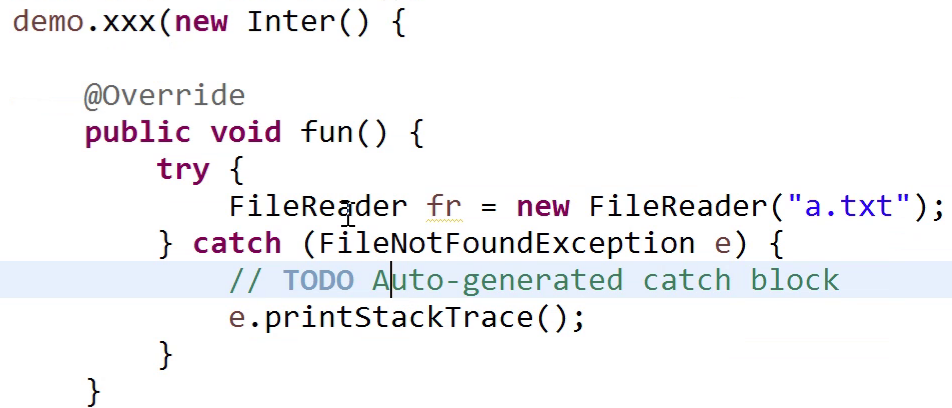
45 自定义异常的好处是可以通过名字看出异常的含义。

46 抛出异常的注意事项：

子类只能抛出父类异常的子集。父类没有抛出异常，子类决不能抛出异常。此时子类出现异常的话只能try catch处理掉。例如接口里面的方法未抛出异常，则实现该接口的类也不能抛出异常，只能自己处理掉。



47 接口中的抽象方法在重写的时候可能出现异常，但由于接口没有抛出异常，故匿名内部类也不能抛出异常，只能在方法内部自己解决掉：

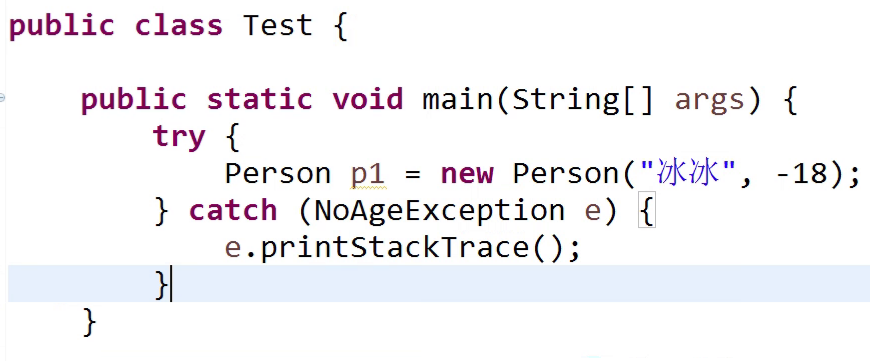


48 由程序员造成的叫异常，不是程序员造成的叫错误。

49 构造方法也是方法，也可以抛出异常：

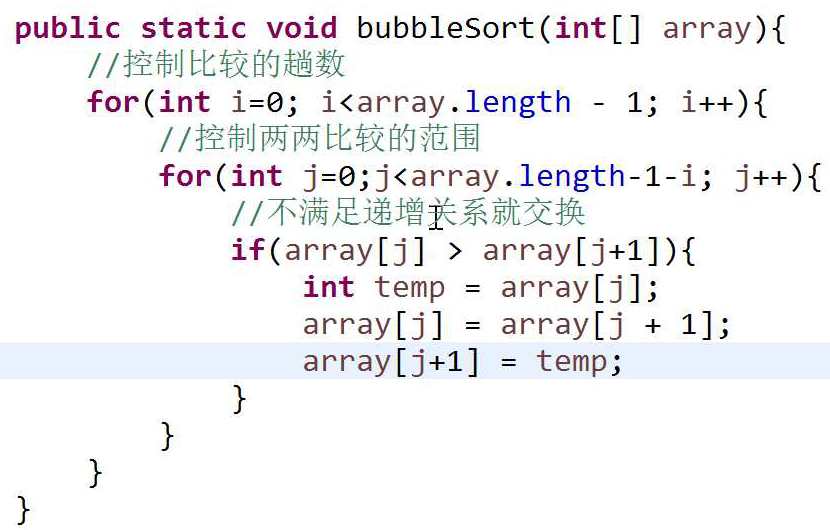


当new对象的时候会调用构造方法，所以new对象时会抛出异常：

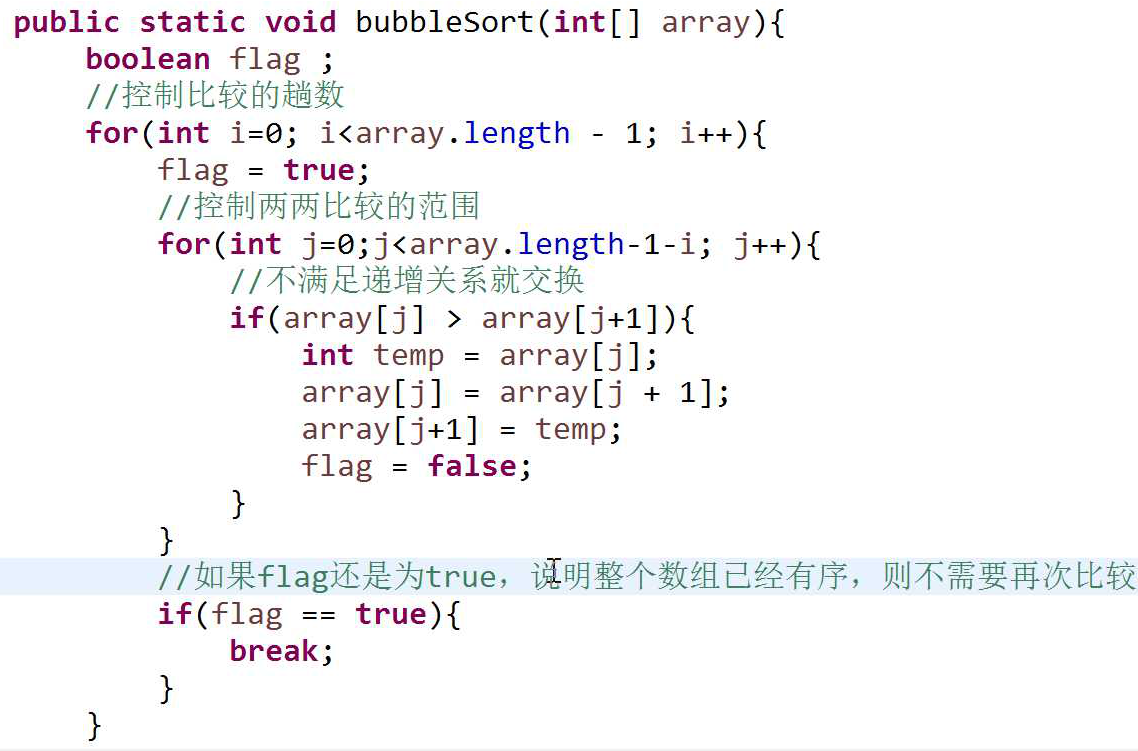


50 nextInt()和nextLine()报的是不同的异常。

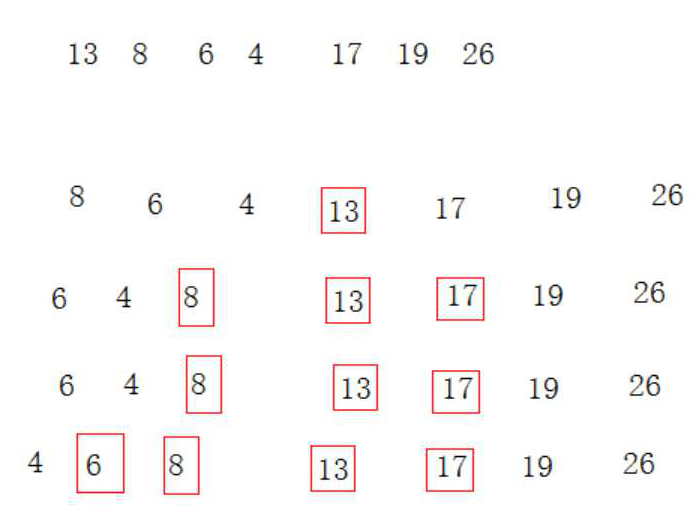
51 冒泡排序面试用的较多。冒泡排序是两两从小到大排序，每次挑出来的是最右边的最大的那个数。



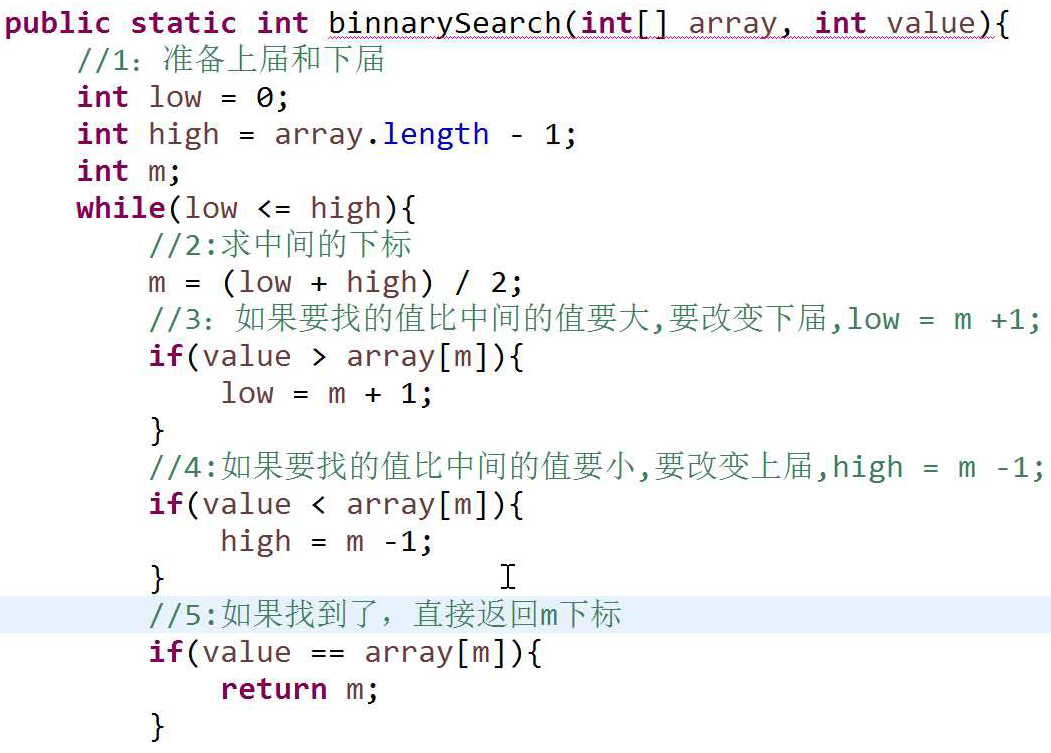
上面这种冒泡排序在有序的时候依然继续排序，效率较低。改进如下：



52 快速排序是最快的排序算法，需要用到递归。

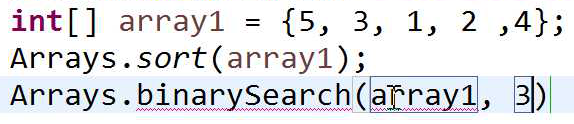


53 查找算法（二分法查找binarySearch，折半查找）：在已经有序的数组中查找。



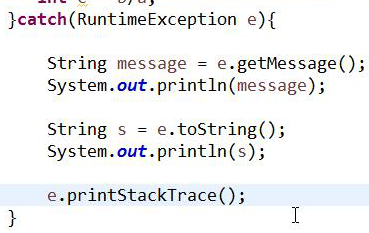


54 排序和查找的api：



55 return是方法的行为而不是循环的行为。

56 System.exit(0)表示结束程序，退出虚拟机。0表示正常退出，其他值表示异常退出。

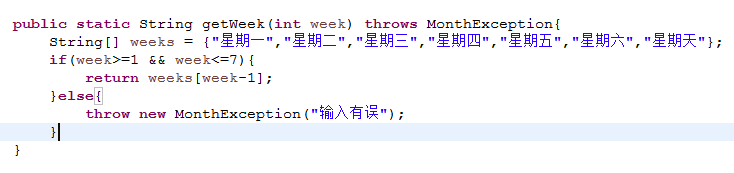
57 三种方法的区别：（注意getMessage()和toString()要打印，printStackTrace()本来就是打印。）  


但是三种同时使用的时候顺序会混乱，因为是多线程，随机执行。



58 int[] arr=null,syso(arr.length)会产生空指针异常，但是int[] arr={}不会产生空指针异常，因为虽然长度为零，但毕竟是一个数组。

59 抛出异常之后方法就中断了，所以没有返回值也行：



60