try-catch-finally

结论：

1、不管有木有出现异常，finally块中代码都会执行；

2、当try和catch中有return时，finally仍然会执行；

3、finally是在return后面的表达式运算后执行的（此时并没有返回运算后的值，而是先把要返回的值保存起来，管finally中的代码怎么样，返回的值都不会改变，任然是之前保存的值），所以函数返回值是在finally执行前确定的；

4、finally中最好不要包含return，否则程序会提前退出，返回值不是try或catch中保存的返回值。

举例：

情况1：try{} catch(){}finally{} return;

显然程序按顺序执行。

情况2:try{ return; }catch(){} finally{} return;

程序执行try块中return之前（包括return语句中的表达式运算）代码；

再执行finally块，最后执行try中return;

finally块之后的语句return，因为程序在try中已经return所以不再执行。

情况3:try{ } catch(){return;} finally{} return;

程序先执行try，如果遇到异常执行catch块，

有异常：则执行catch中return之前（包括return语句中的表达式运算）代码，再执行finally语句中全部代码，

最后执行catch块中return. finally之后也就是4处的代码不再执行。

无异常：执行完try再finally再return.

情况4:try{ return; }catch(){} finally{return;}

程序执行try块中return之前（包括return语句中的表达式运算）代码；

再执行finally块，因为finally块中有return所以提前退出。

情况5:try{} catch(){return;}finally{return;}

程序执行catch块中return之前（包括return语句中的表达式运算）代码；

再执行finally块，因为finally块中有return所以提前退出。

情况6:try{ return;}catch(){return;} finally{return;}
程序执行try块中return之前（包括return语句中的表达式运算）代码；
有异常：执行catch块中return之前（包括return语句中的表达式运算）代码；
则再执行finally块，因为finally块中有return所以提前退出。
无异常：则再执行finally块，因为finally块中有return所以提前退出。

最终结论：任何执行try 或者catch中的return语句之前，都会先执行finally语句，如果finally存在的话。
如果finally中有return语句，那么程序就return了，所以finally中的return是一定会被return的，
编译器把finally中的return实现为一个warning。

一般情况下不管try{}catch(){}语句块如何结束，finally保证其所包含的语句块最终被执行，但是存在特殊情况：1.try语句没有被执行到，如在try语句之前就返回了；2.在try块中有System.exit(0)），此时finally中的语句块是不会执行的；
B：System.exit(0)。表示将整个虚拟机里的内容都释放，JVM停止工作，此时程序正常退出；
C和D： finally块中的内容会先于try中的return语句执行，如果finall语句块中也有return语句的话，那么直接从finally中返回了，但是不建议在finally中return。