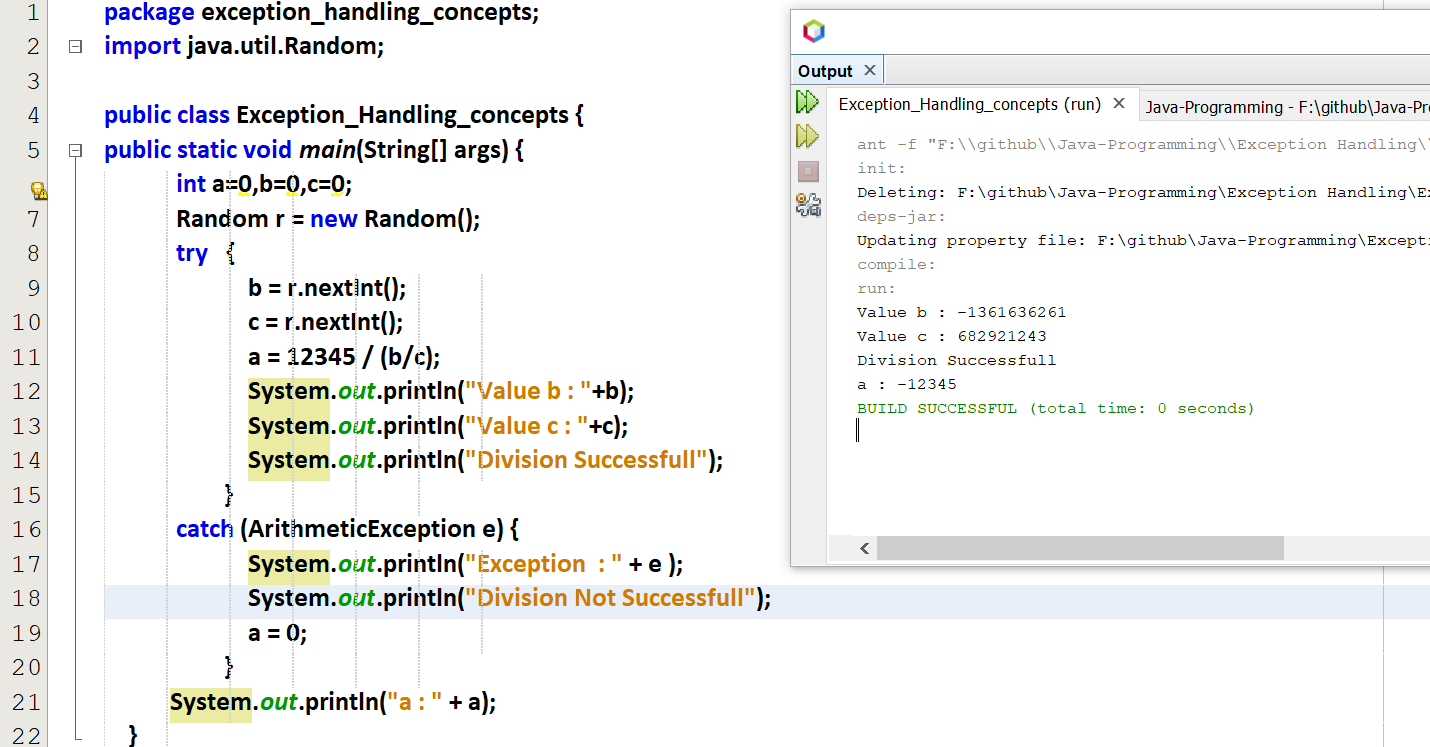
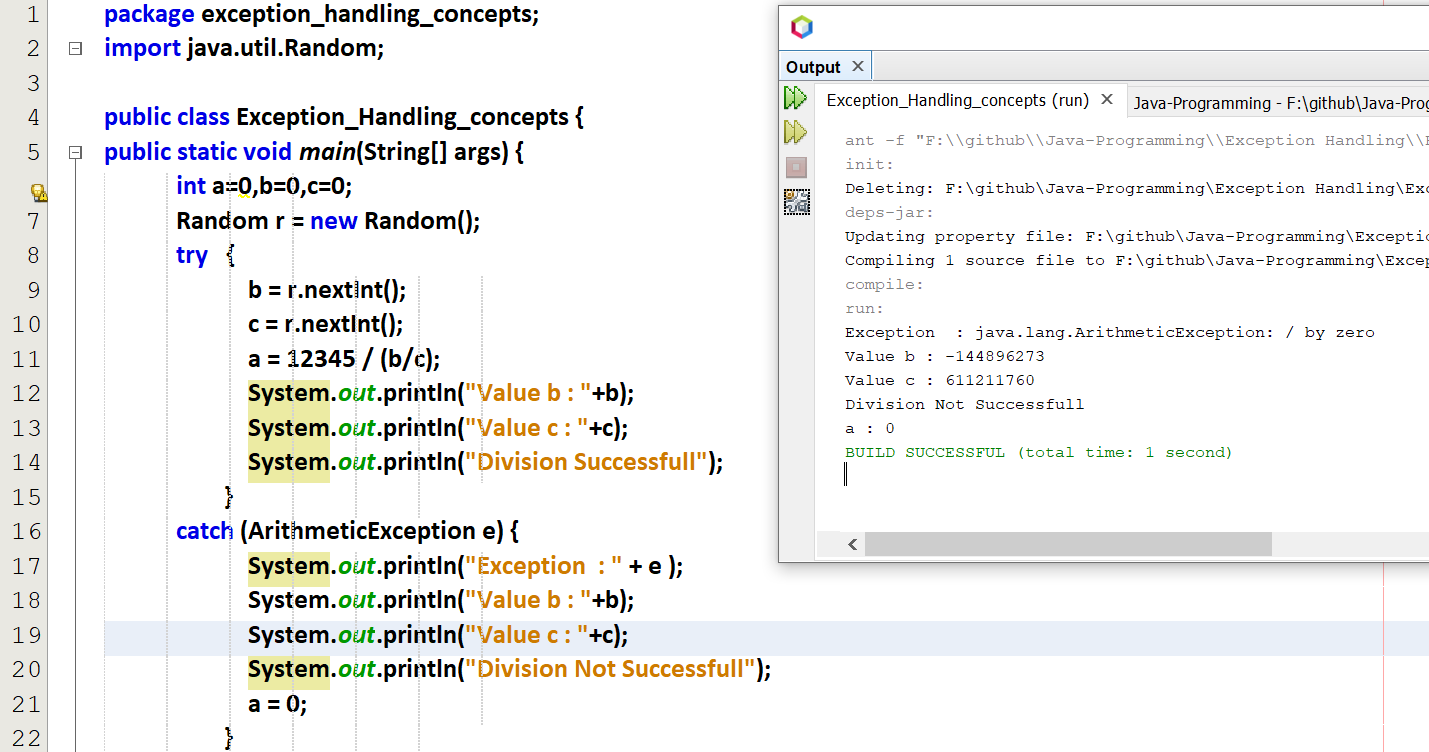
# Exception Handling

# try and catch

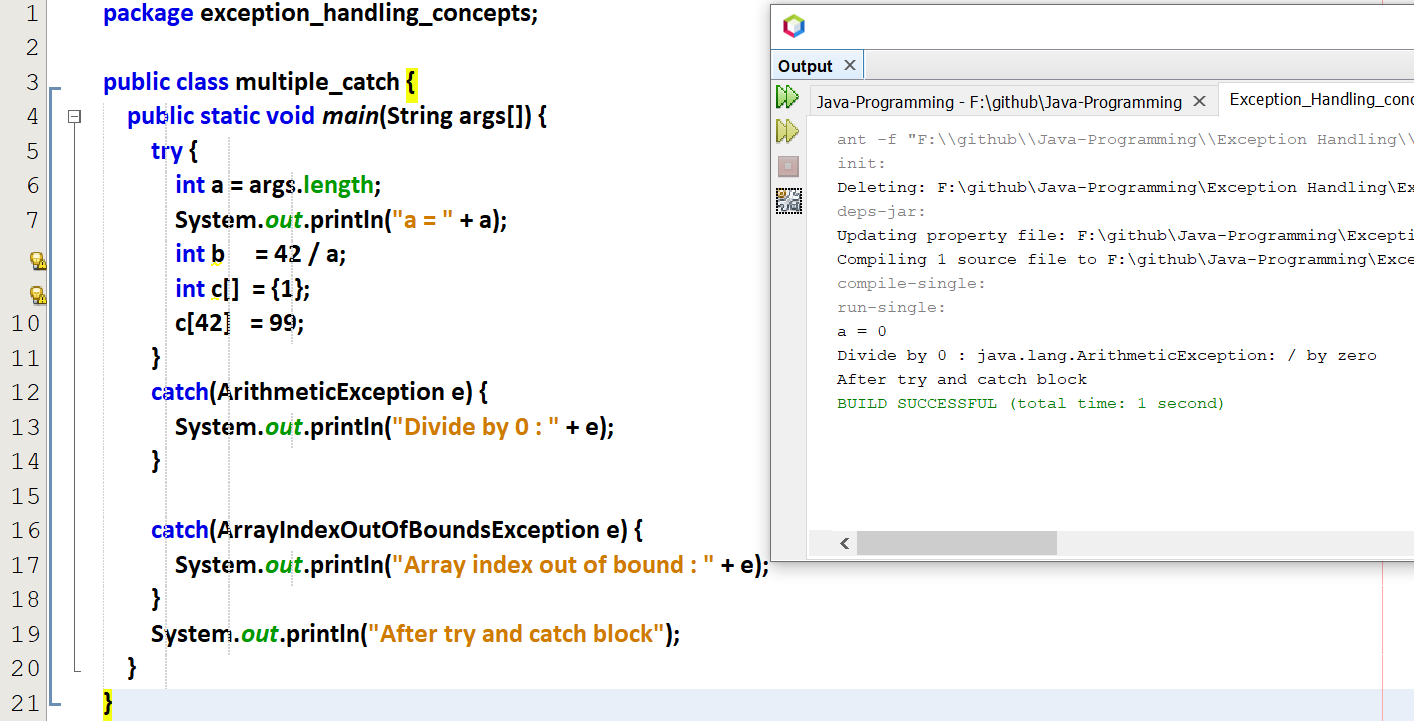
Division is Successful (b>c) [Exception Not caught]



Division is Not Successful (b<c) [Exception caught]



## Multiple Catch with a single try

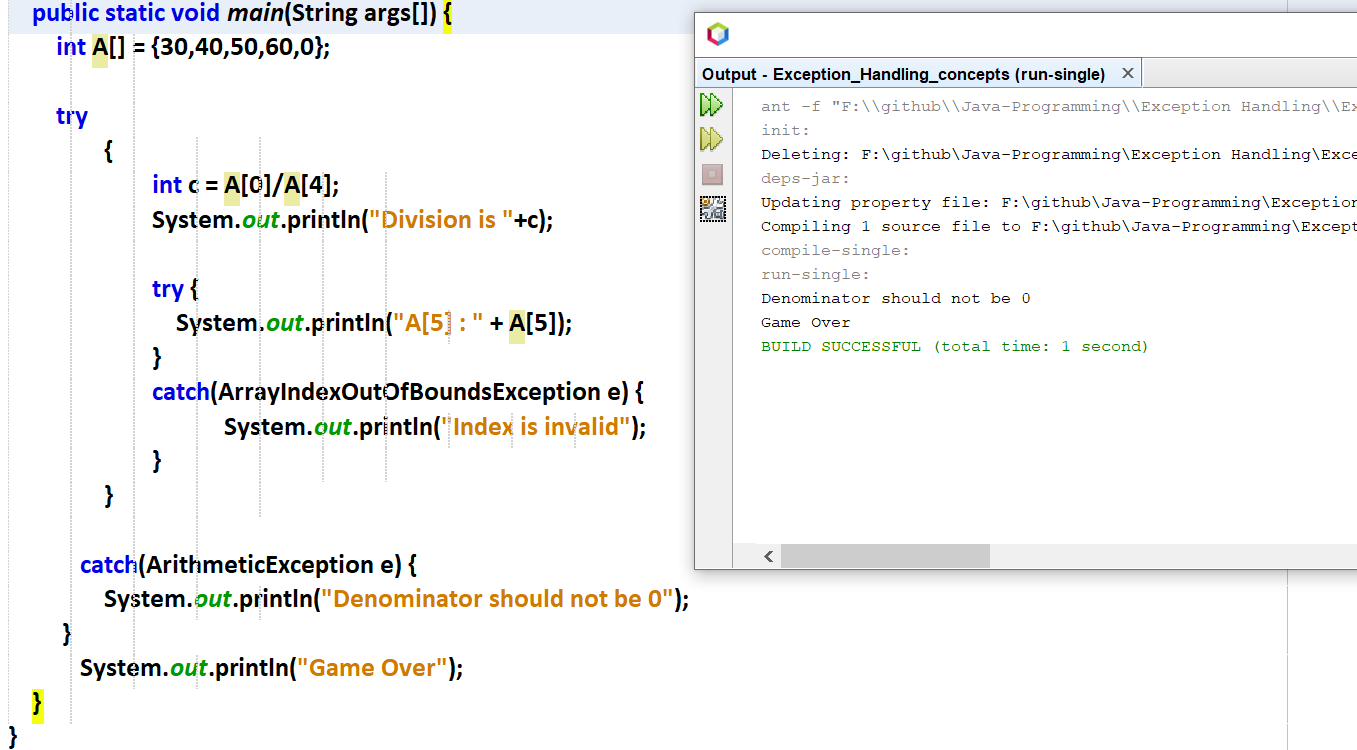


By default when I run the above code in Net-beans, String args[] = 0. So a=0. So only getting Division by 0 exception.  
And array index out of bound exception is not raised.

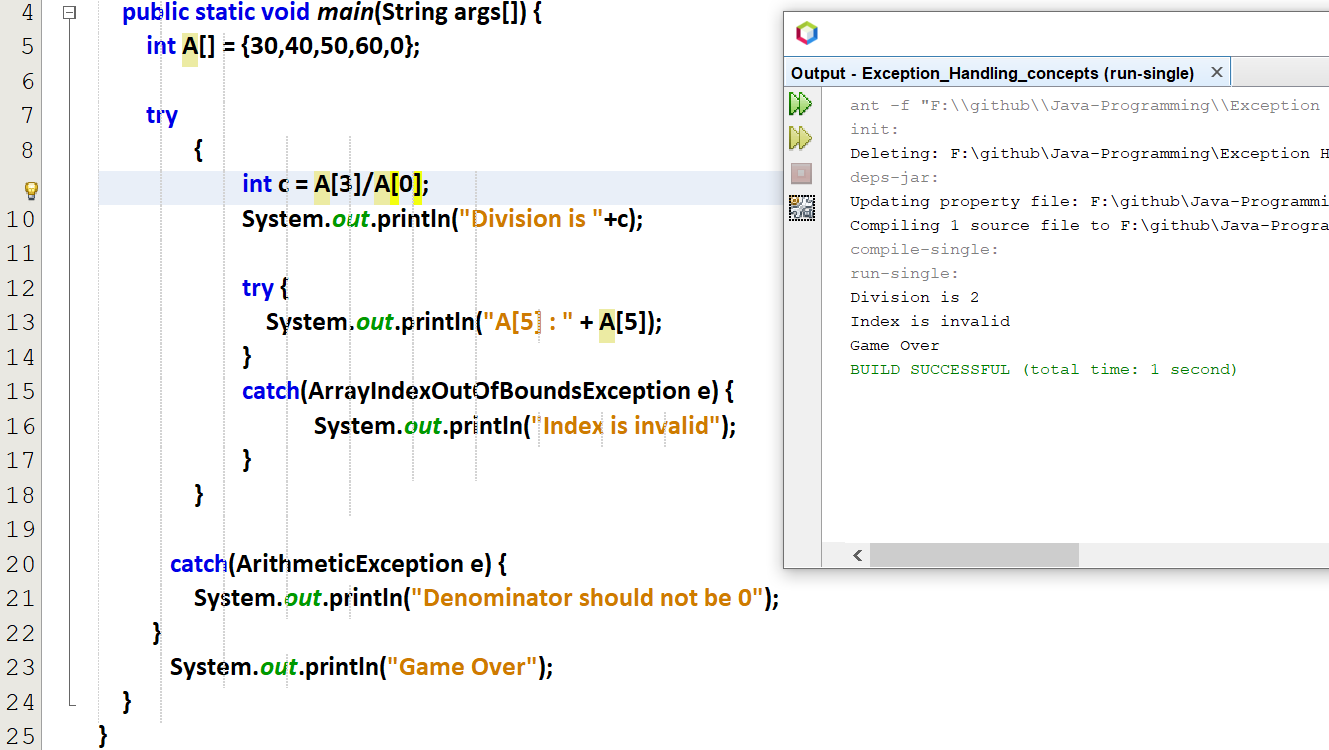


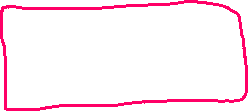
When I give an array of 5 integers as input, so args length=5 , so division exception is not found.  
And array index out of bound exception is raised.

# **Nested try and catch (try within a try**)



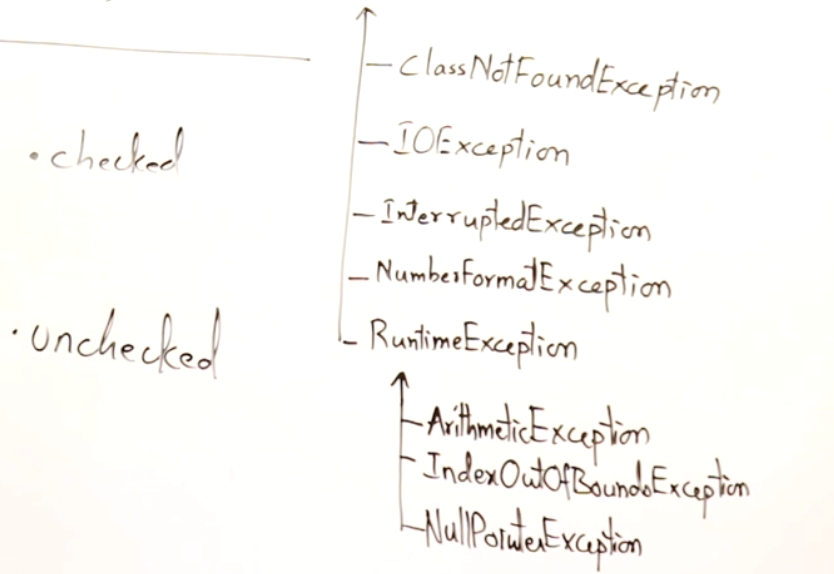






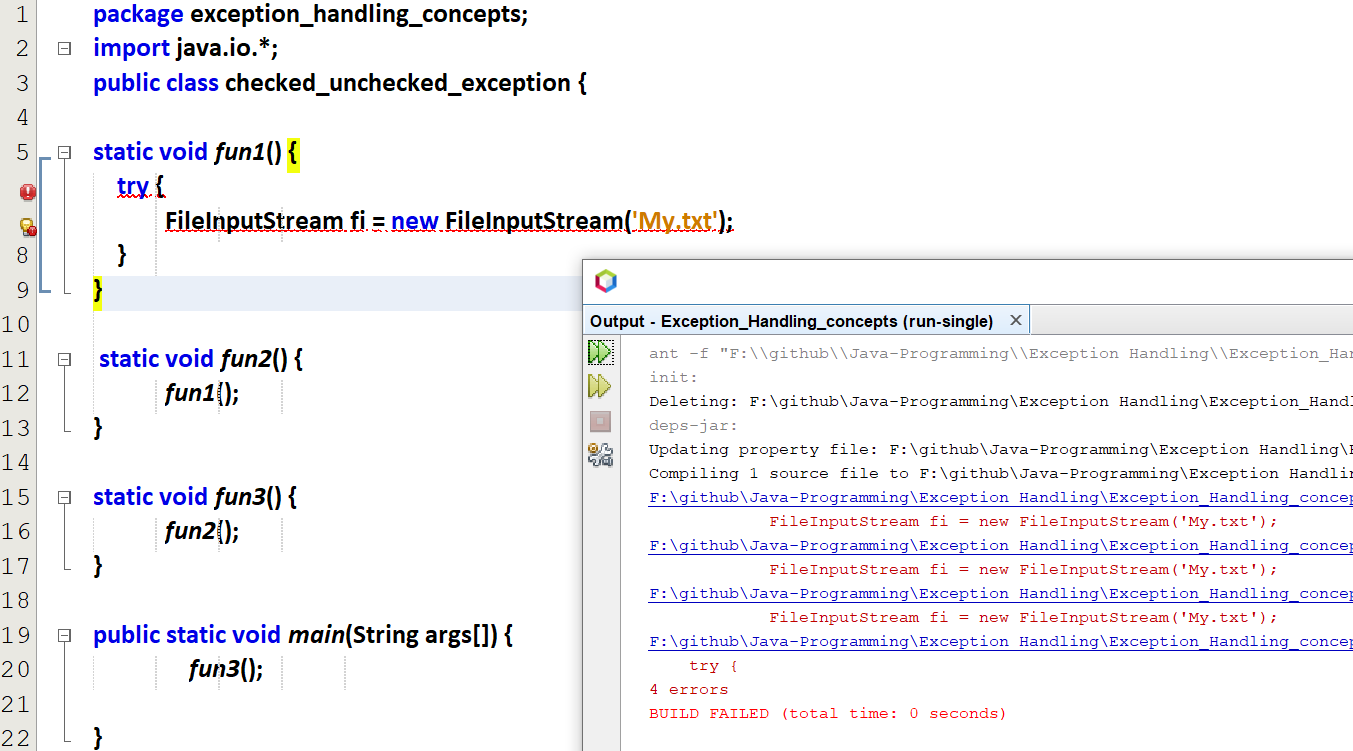
object is the mother class of all JAVA classes

I/O exception 🡪 File not found exception  
Interrupted Exception 🡪 If a thread stops abnormally   
NumbersFormat Exception 🡪 If I give a number as a string



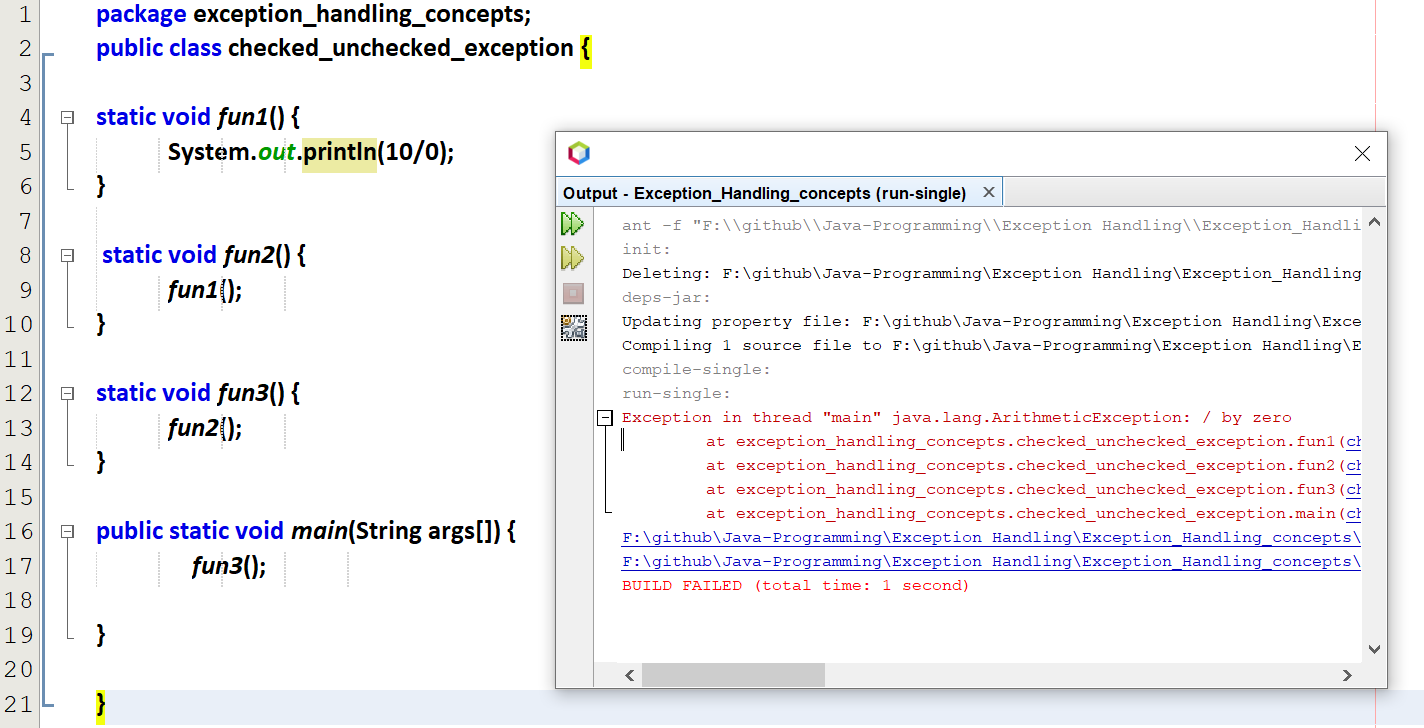


Checked exception :   
It is necessary to handle this exception using try and catch. We cannot escape from this exception. JAVA will force to handle checked exception using try and catch. Trying to open a file

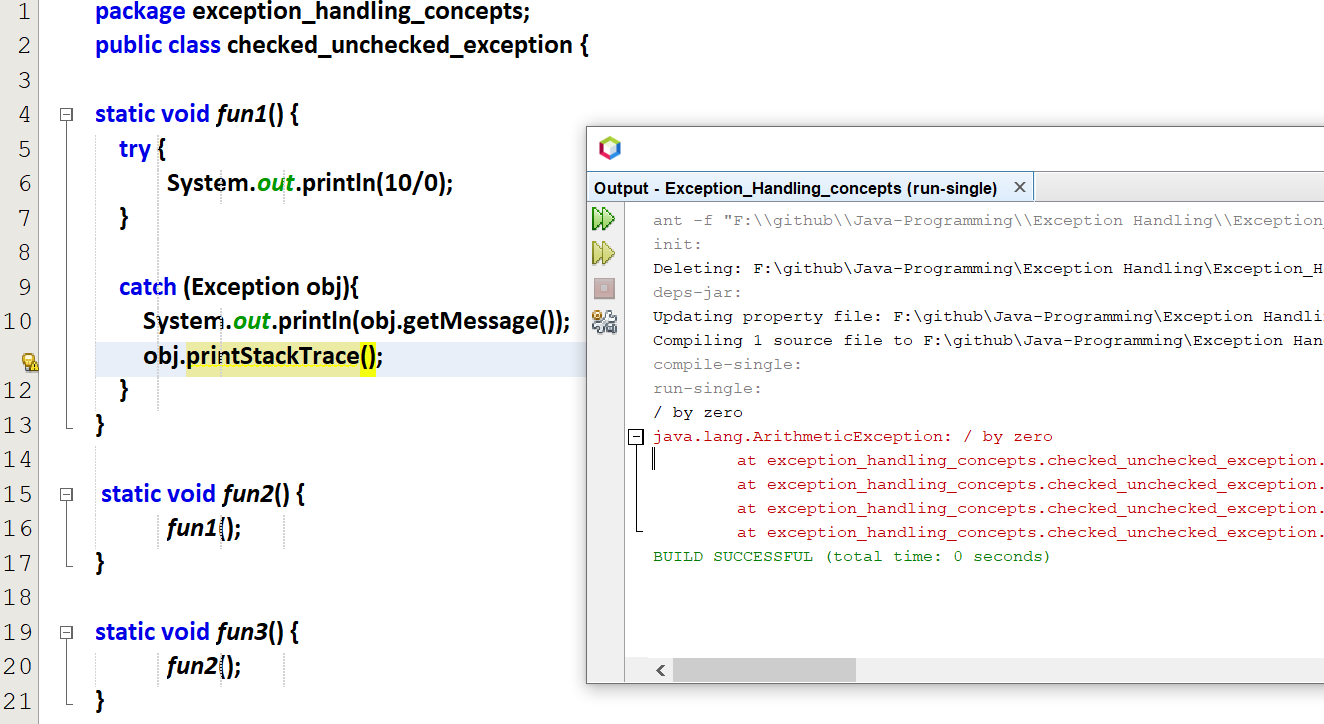


Java will definitely raise an error for the un-checked exception.

Un-Checked exception :  
It in Not mandatory to handle this exception. We can may/may-not handle this exception. JAVA will not force to handle unchecked exception.

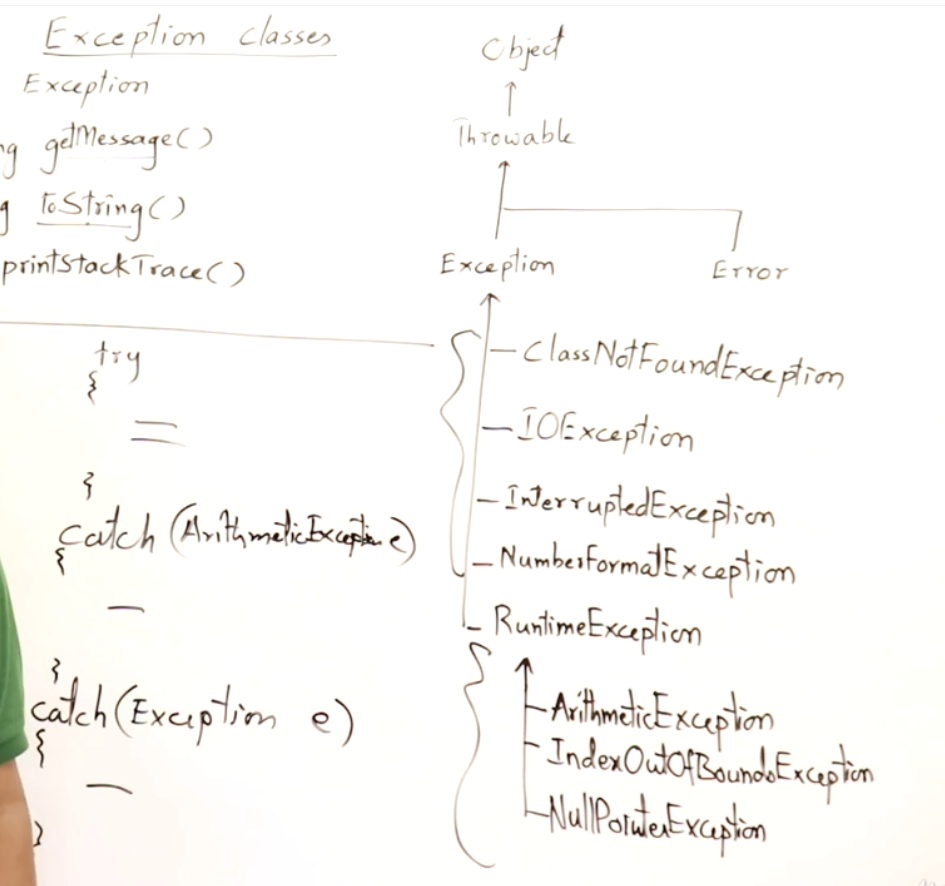


Handle the above un-checked exception using try and catch



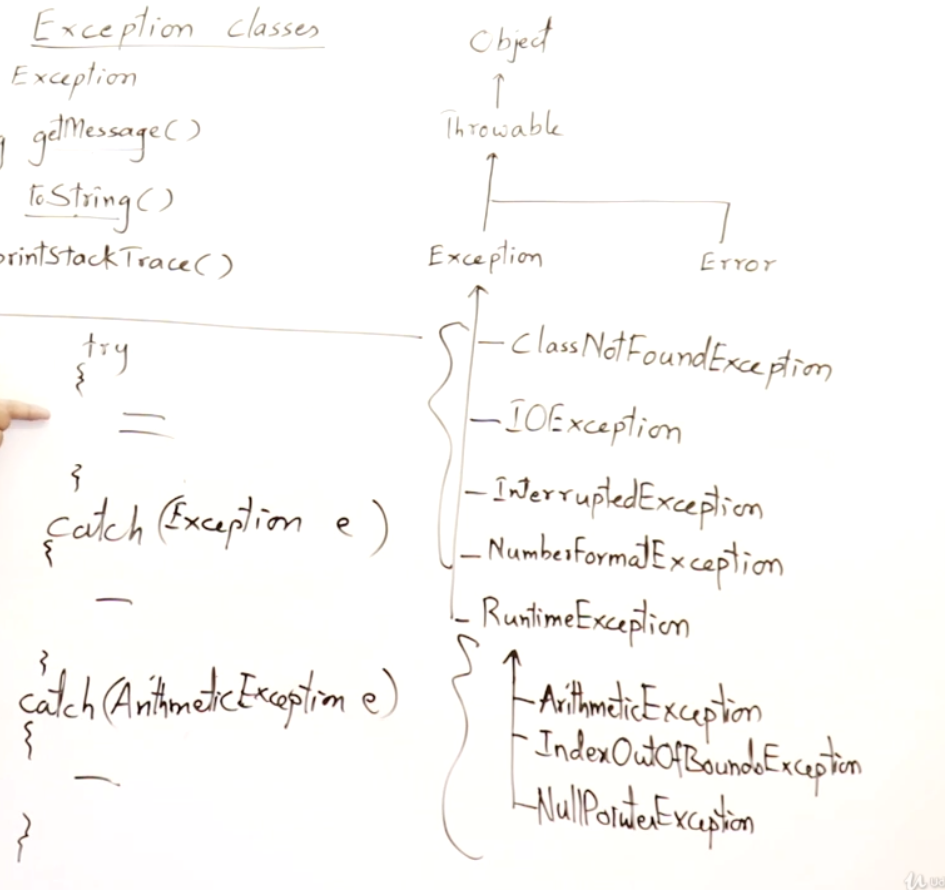
The stack of functions are also displayed.

# Hierarchy of Exception in Multiple catch blocks





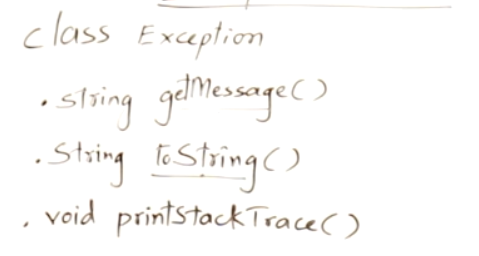
It will check for Arithmetic Exception, if no Arithmetic Exception it will look for the other exceptions in Exception parent class.

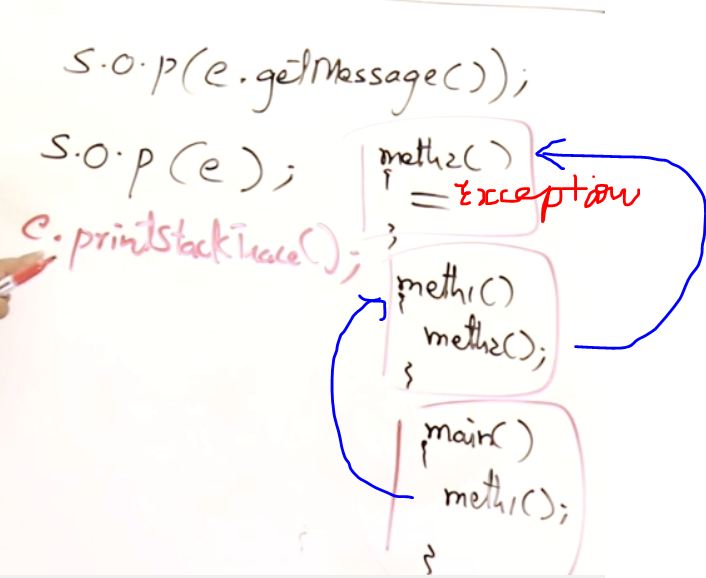




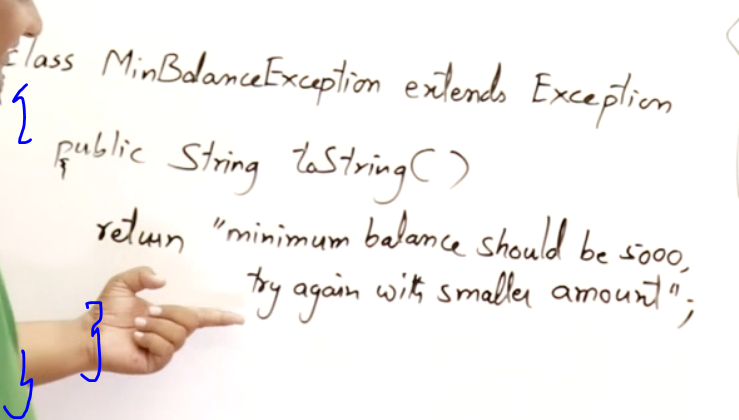
Compiler will raise an error, first catch 🡪 sub class, next catch 🡪 super class

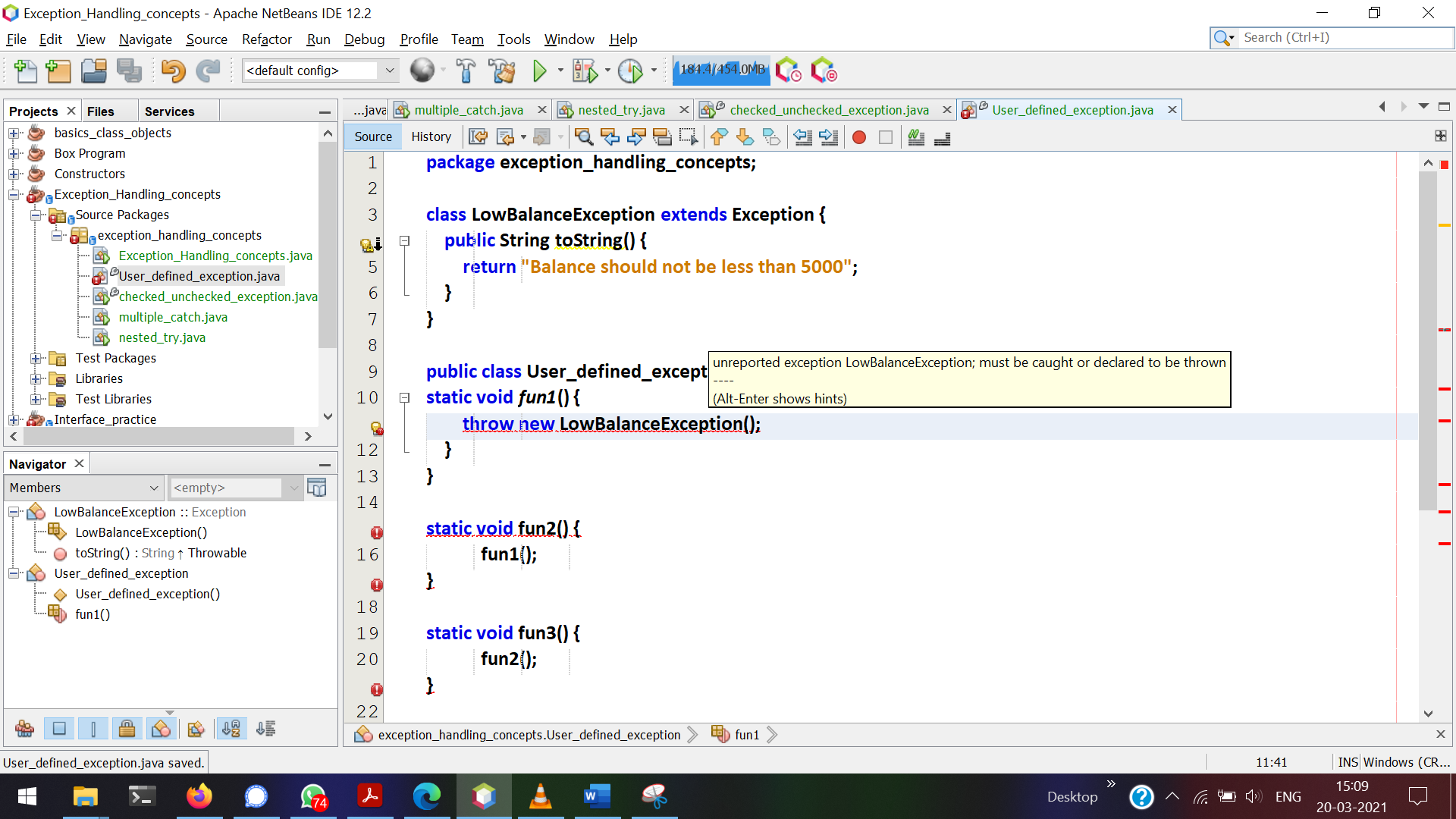
# Important methods in Exception class





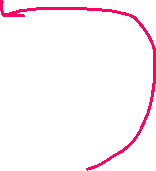
getmessage() 🡪 Message about the exception. sout(obj.getmessage()).  
ToString() 🡪 Message about the exception. sout(obj) 🡪 automatically toString() will be called.  
printStackTrace() 🡪 Sequence of method calls when the exceptions were raised



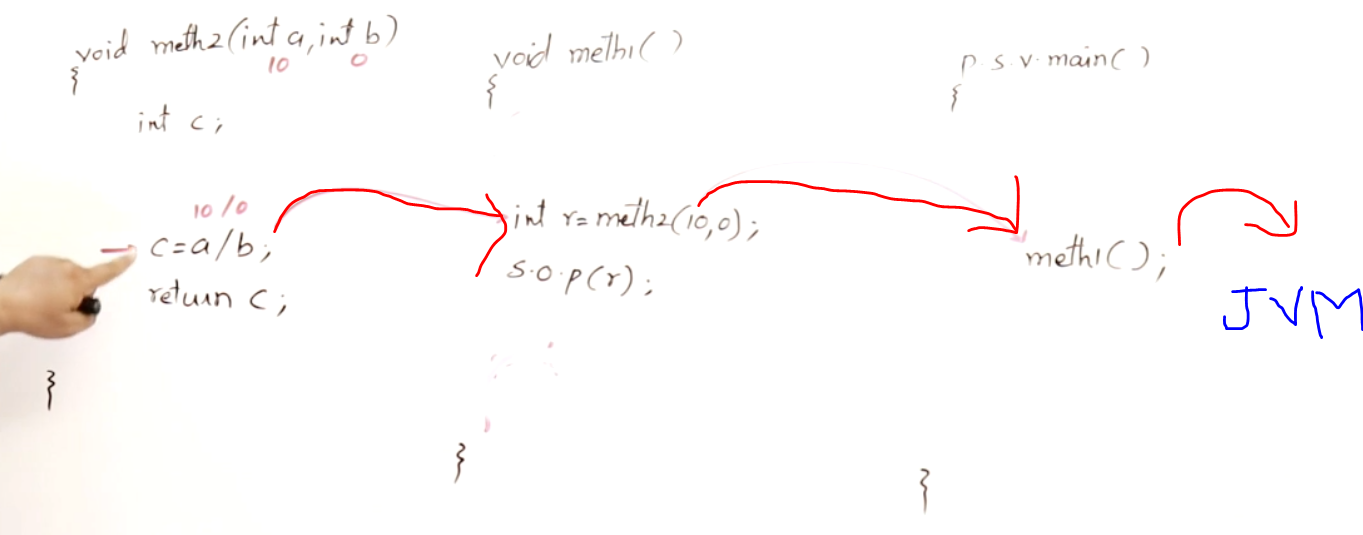


LowBalanceException is a checked exception since it extends from Exception.   
So this can be handled via try and catch or throws





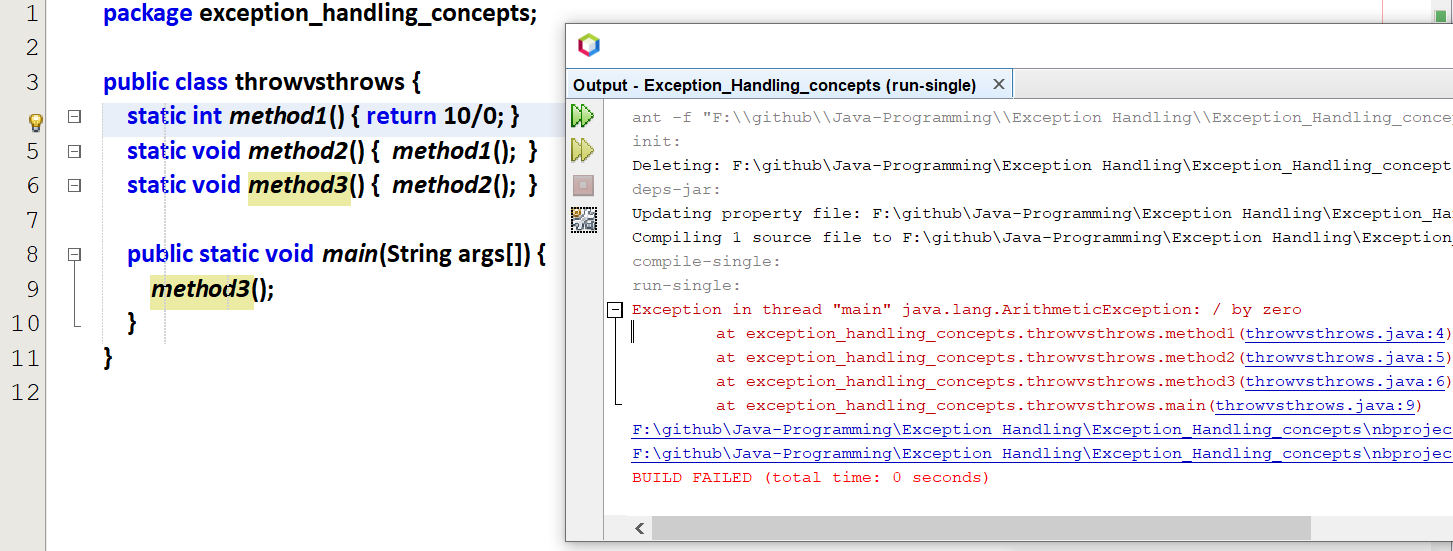
# Propagation of Exception



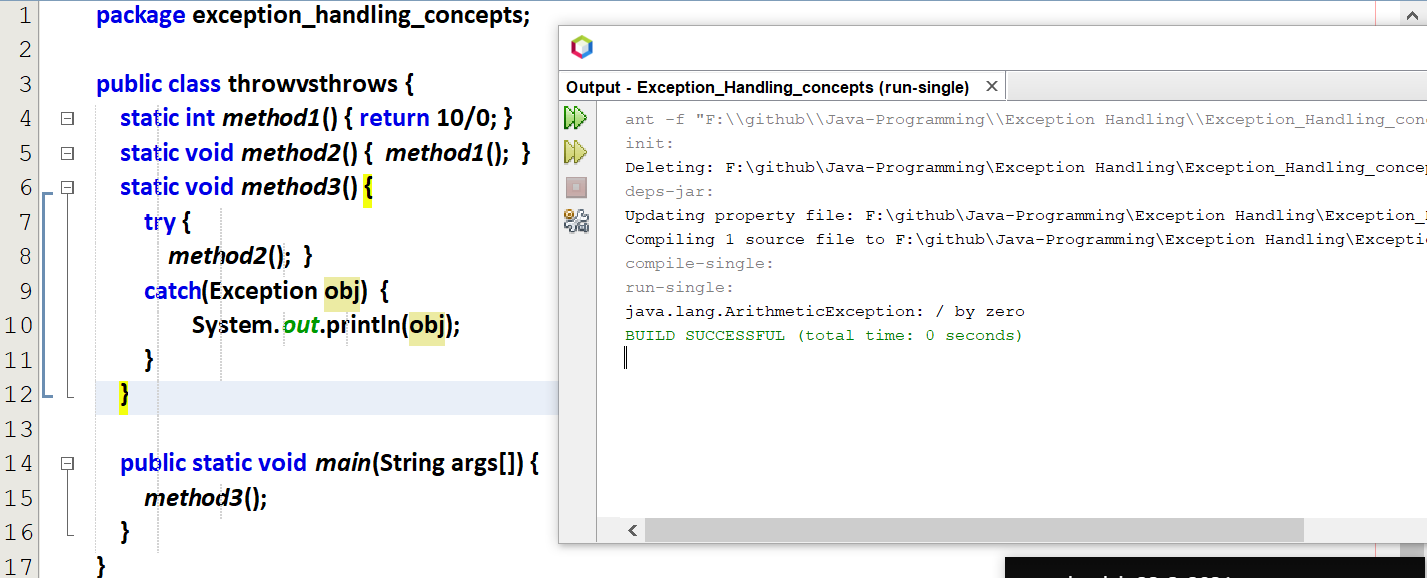
If try and catch not available in

Method-2’s exception will be handled by method-1.  
Method-1’s exception will be handled by main().  
main’s exception will be handled by JVM and then the program crashes.

Exceptions will be propagated to a calling method, its calling method and its calling method and so on.

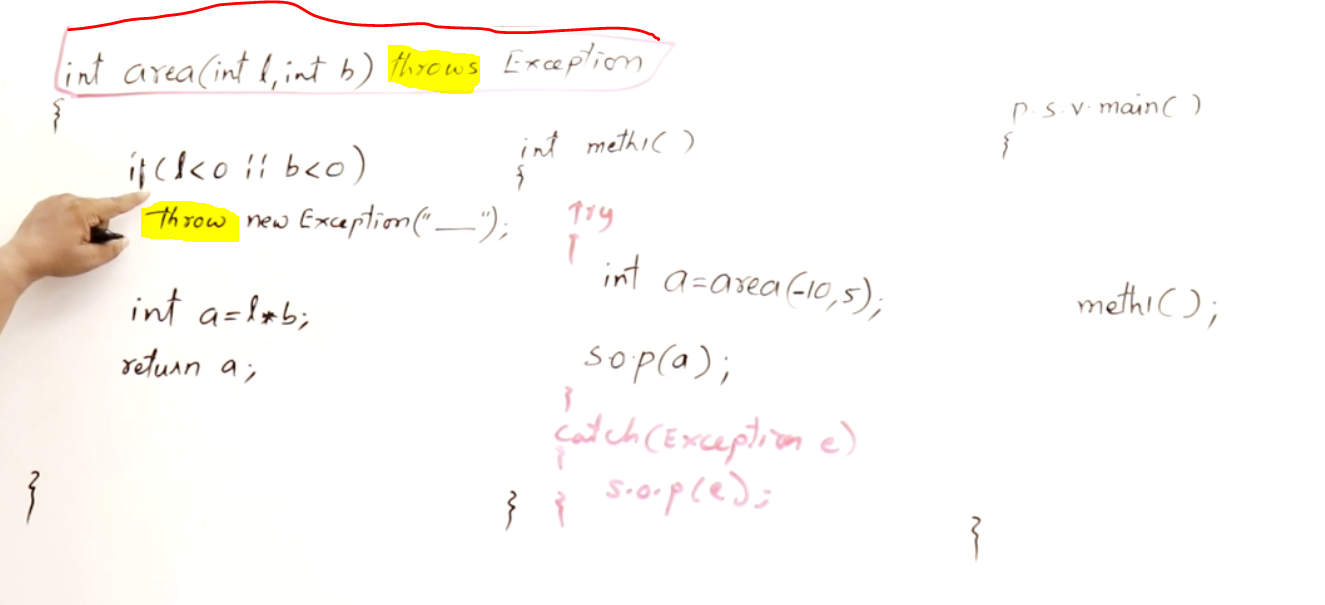


This exception can be handled in any of the methods call



If we want to throw an exception, then How to throw it???

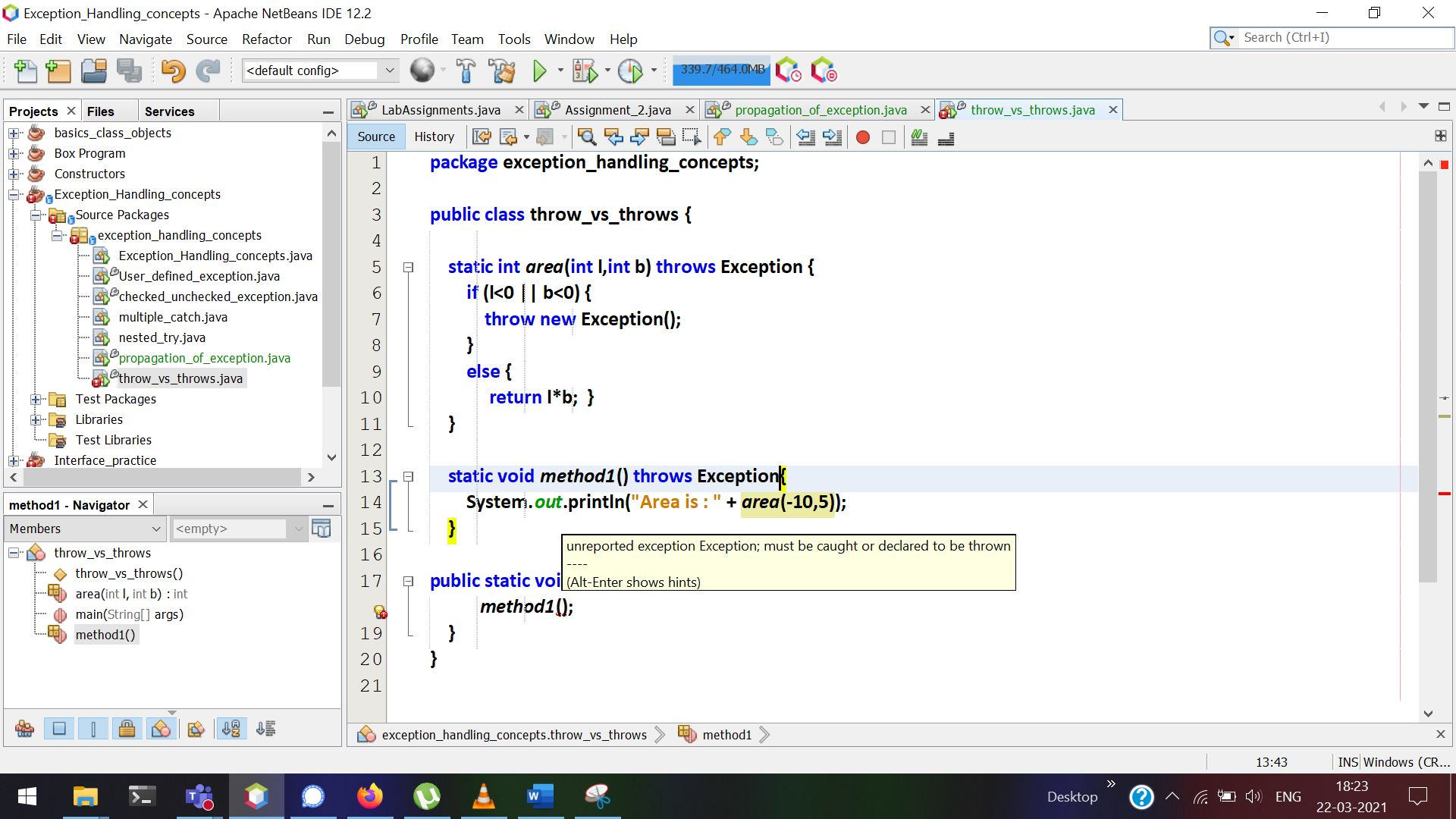
# Throw vs Throws

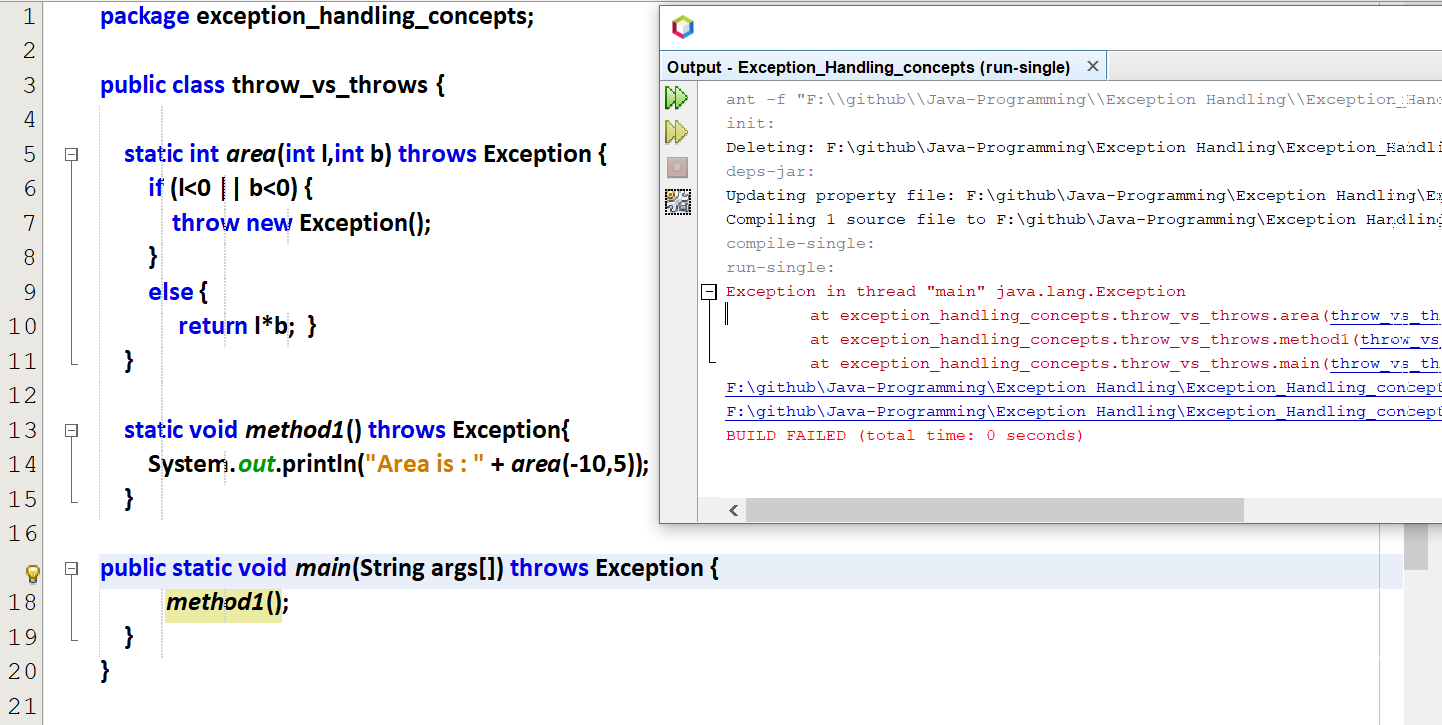


If I want to use the area value in area(), we can use try and catch.  
If I don’t want to use the area value in area(), and want to access the area value in meth1() and main(), then we can use throw()  
area() throws an exception and it will be handled my method-1 using try can catch.

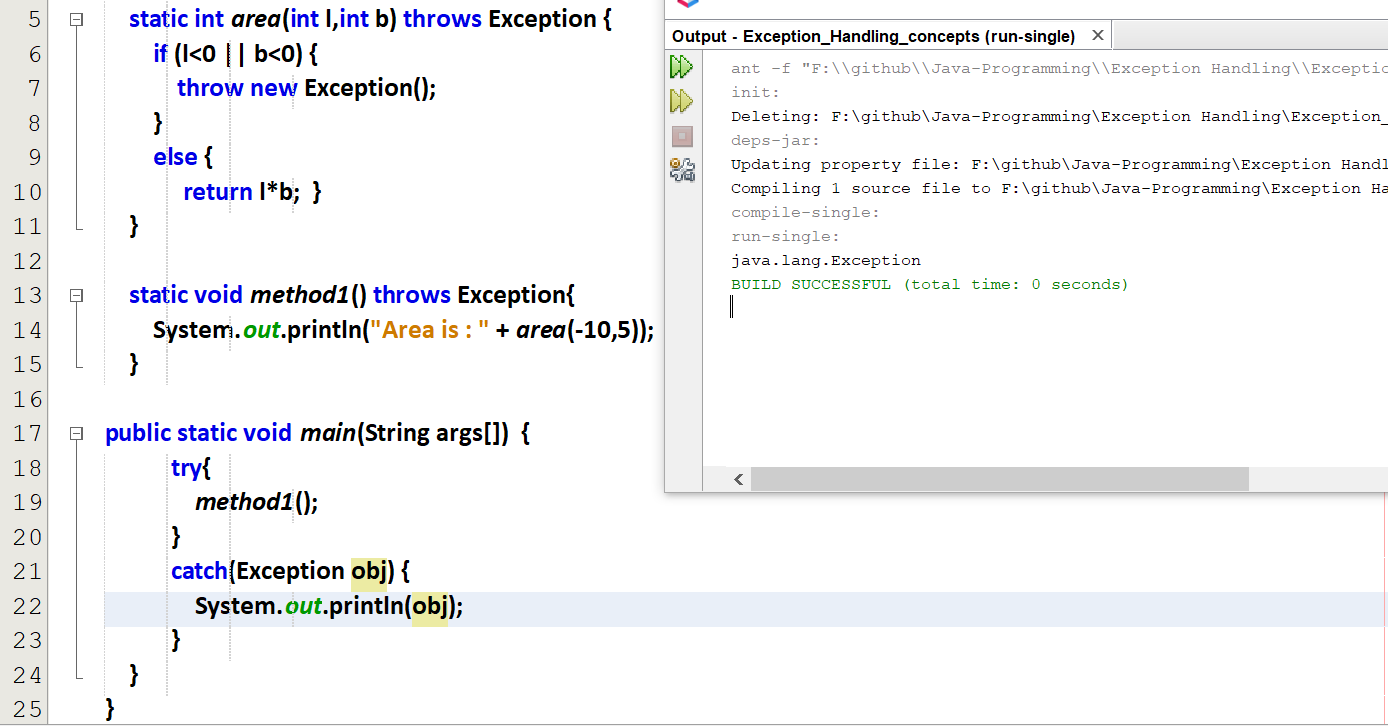
If we don’t want to handle in method1(), that exception can also be handled in main() also using try and catch.





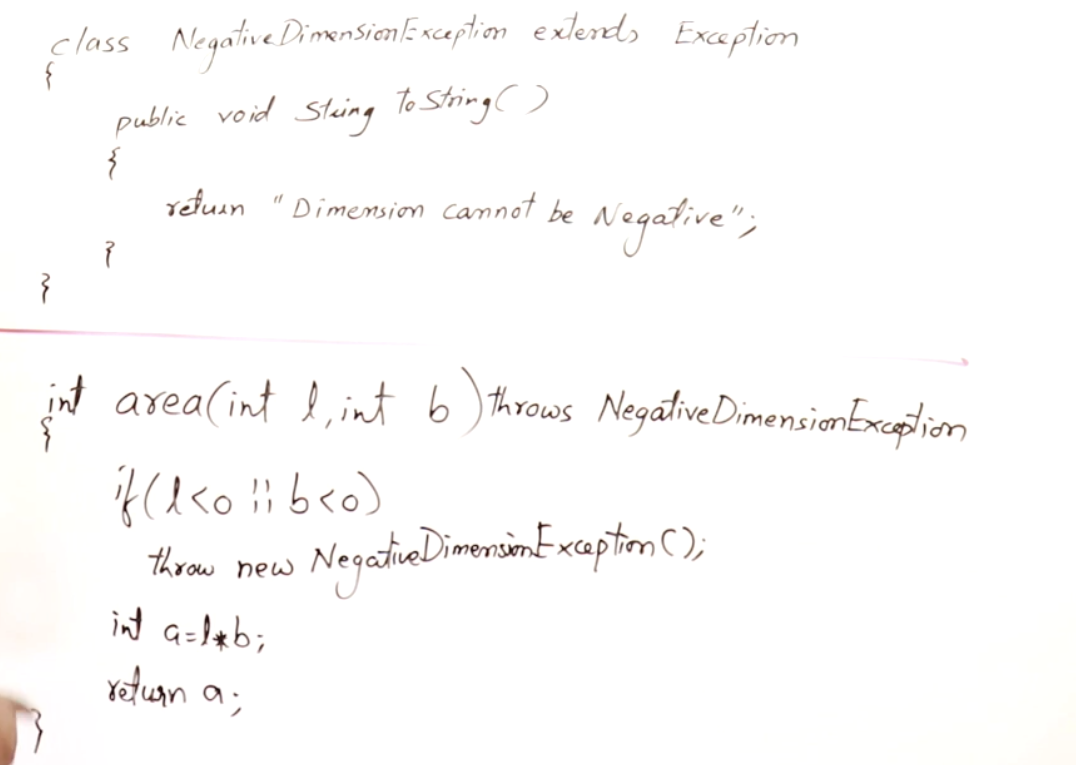


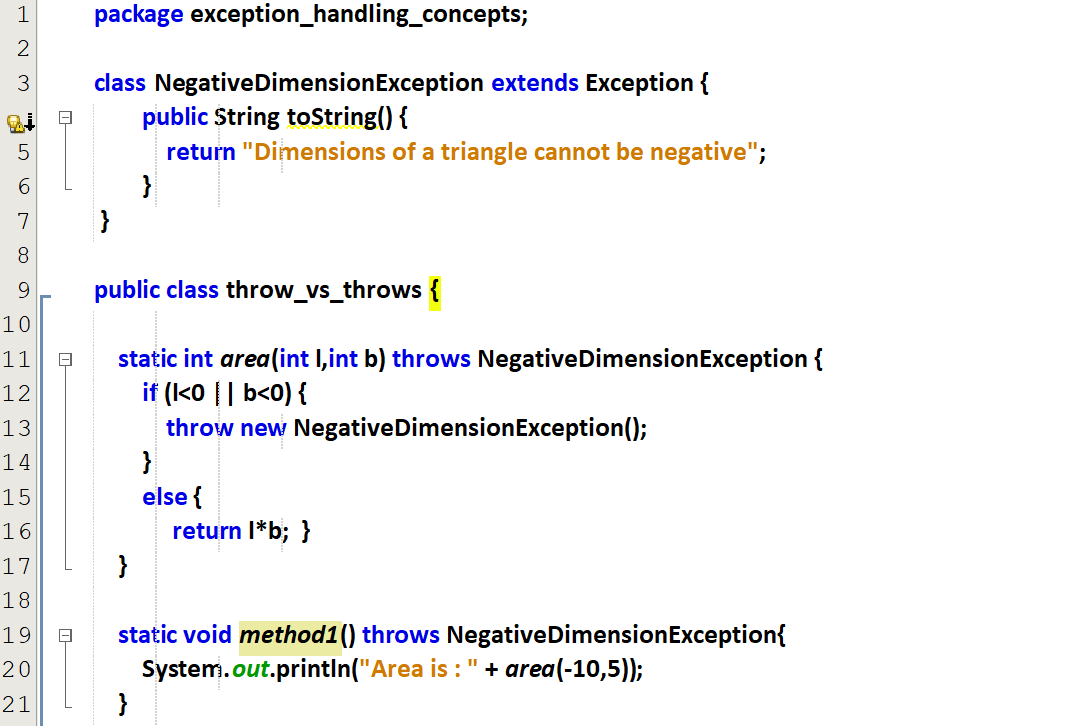
Now JVM cannot handle this exception and got crashed.

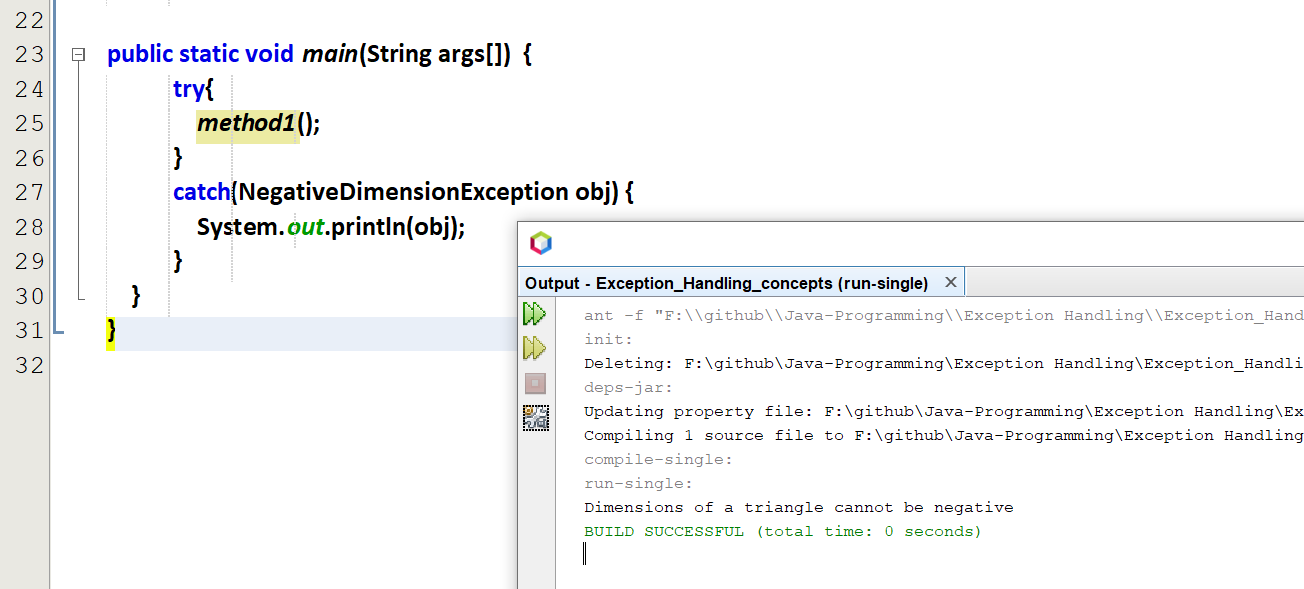
This exception can be handled in main()  


# user-defined exception

Every user-defined exception must be inherited from Exception class

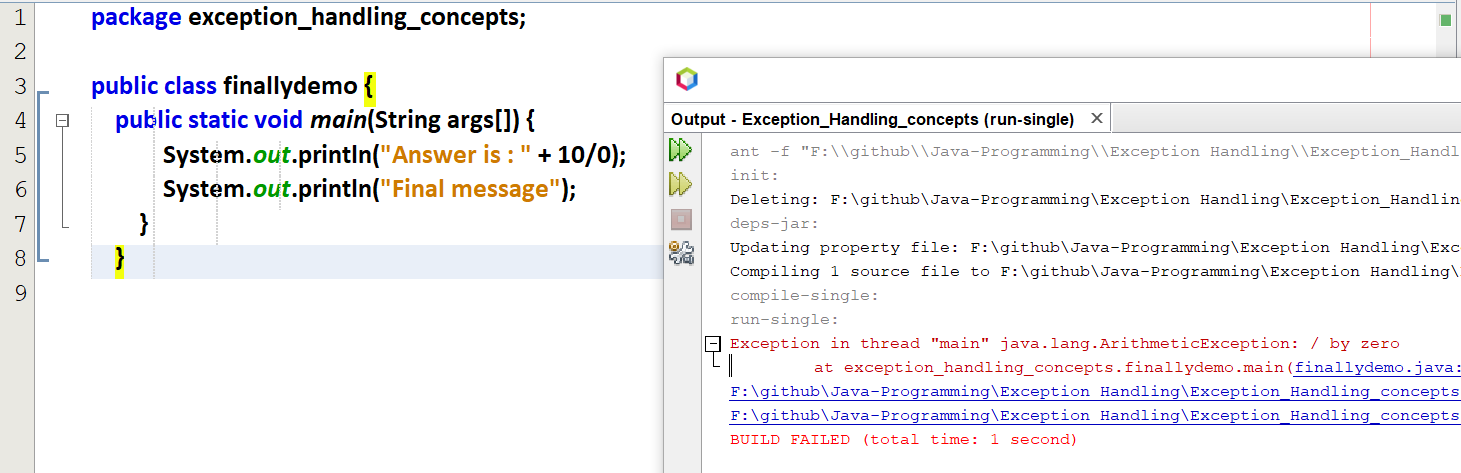
User-defined exception with throws  






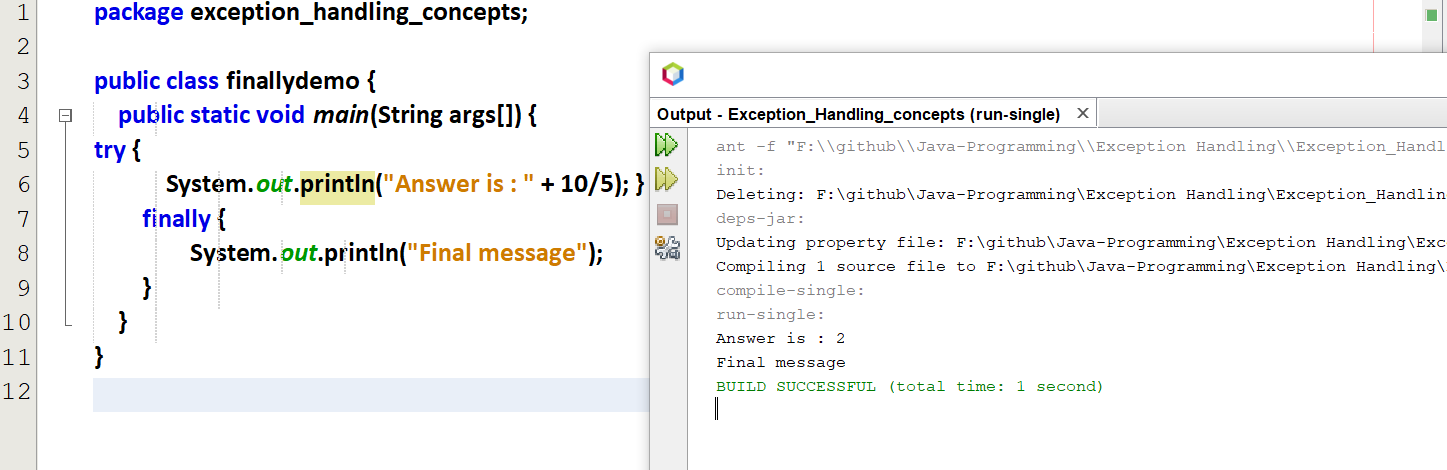
# finally

finally will be get executed if there is an Exception or if there is no exception.

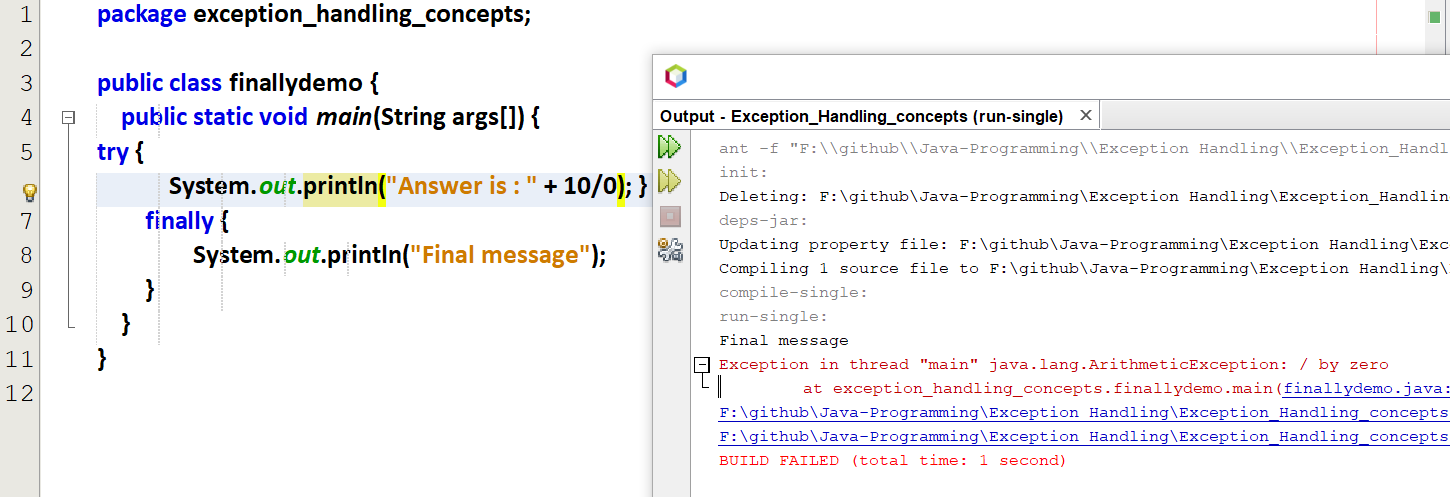


Now “Final message” is not printed after exception.

But I want to print this “Final message” even the exception occurs.



finally is executed without exception.



finally is executed with exception.

With catch()



Is finally() necessary ???

If the exceptions are handled 🡪 finally is not at all necessary  
If the exceptions are not handled 🡪 finally is necessary

