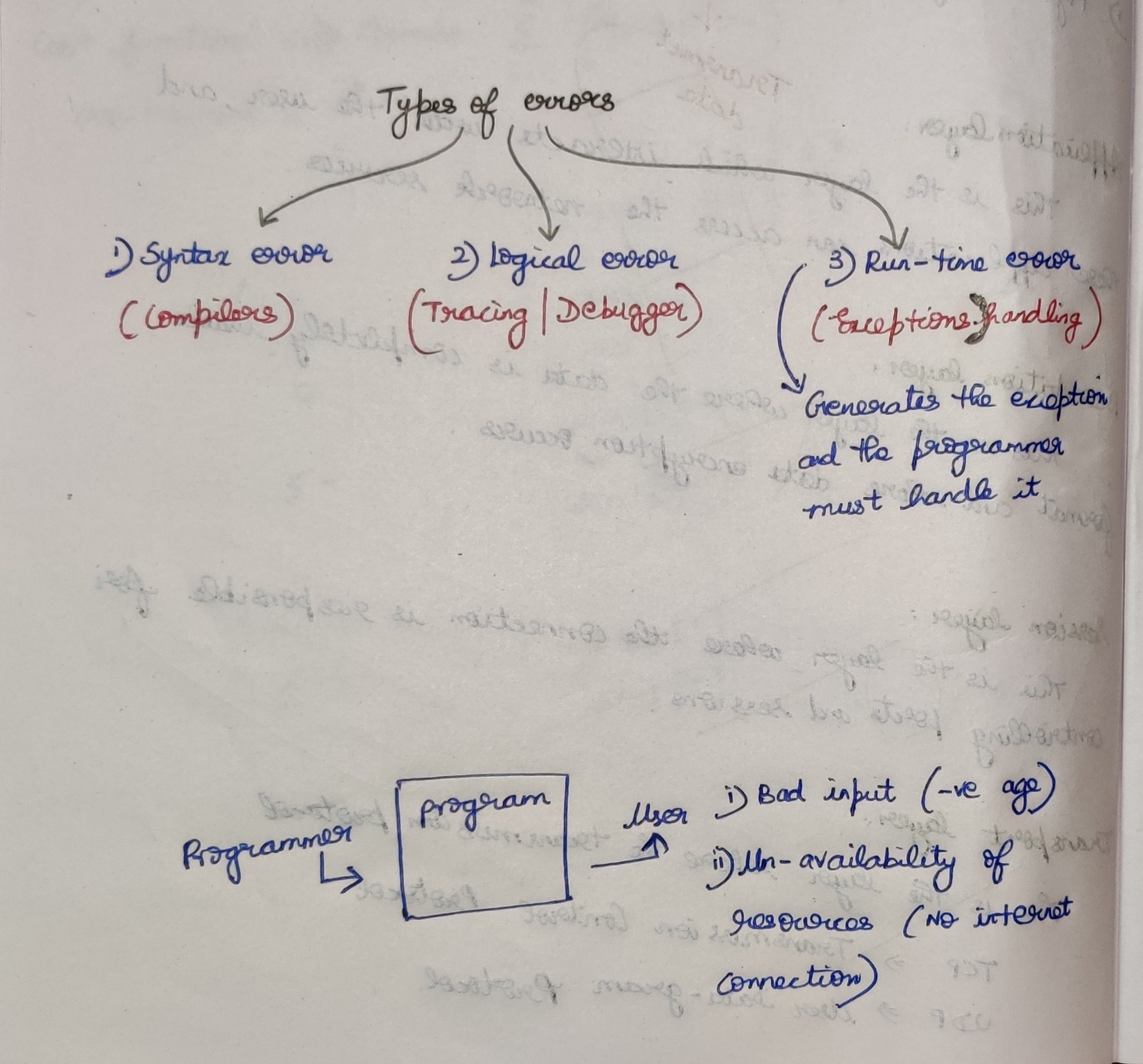
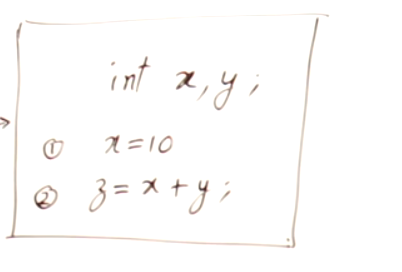
# **Reason for Exception Handling**



## **Syntax error**



* Using z without declaring it.
* Using y without initializing it.  
  ***Compiler can detect the syntax error.***

## **Logical error**

* I want r = -b/(2\*a) but I wrote r = -b/2\*a
* for(int i=1;i<A.length;i++) { sout(A[i]); }  
  I want to print the array elements form 0th index, but according to the code the array elements starts from 1st index.

***Tracer, Debugger can detect the logical error.***

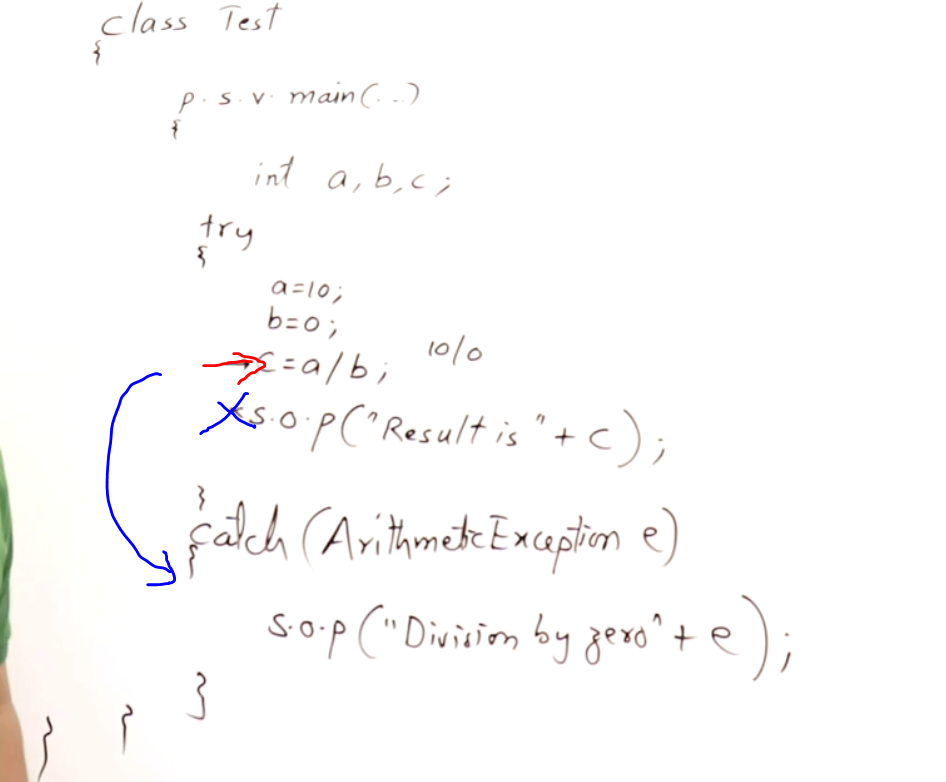
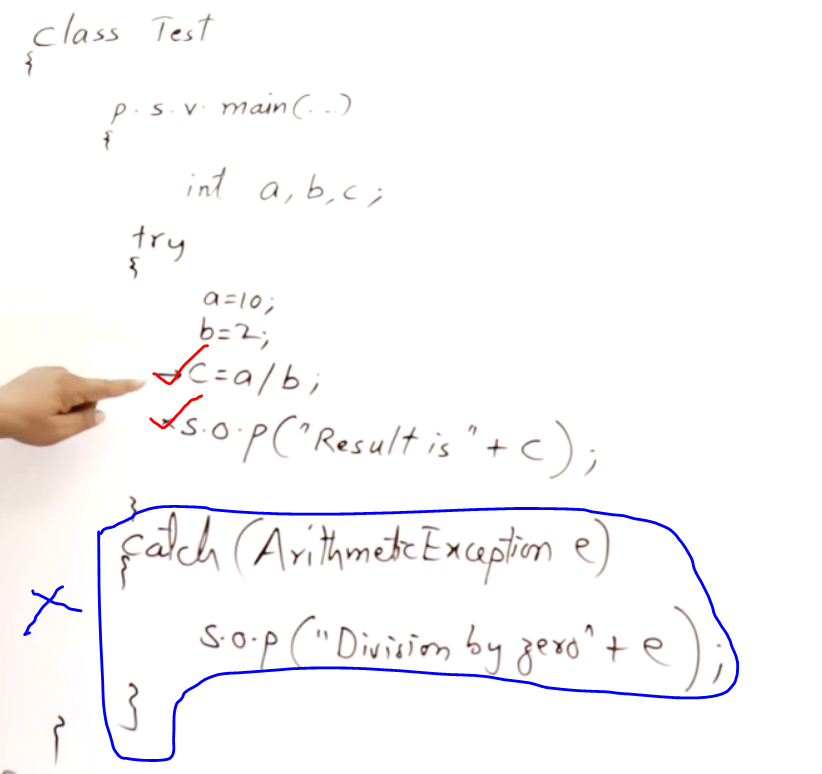
## **Run-time error**

* A user enters might give his/her age as a -ve value.   
  ***[ Bad Input ]***
* A user might access the server without having internet connection.   
  ***[Un-availability of Resources]***

***Exception Handling can detect the run-time error and handles it.***

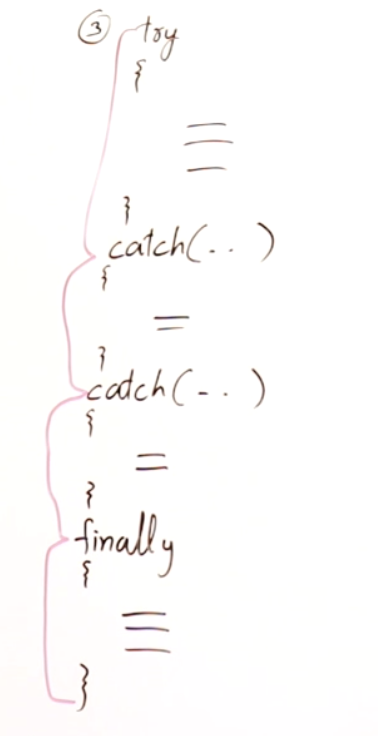
Programmer 🡪 My program works perfectly except in the case of bad input and un-availability of resources.  
If user gives bad-input and un-availability of resources, the user is giving an indirect message to the programmer that I committed the mistake. So from the mistakes done by the user, the programmer must understand it and should handle it (exception-handling).

# **try and catch**

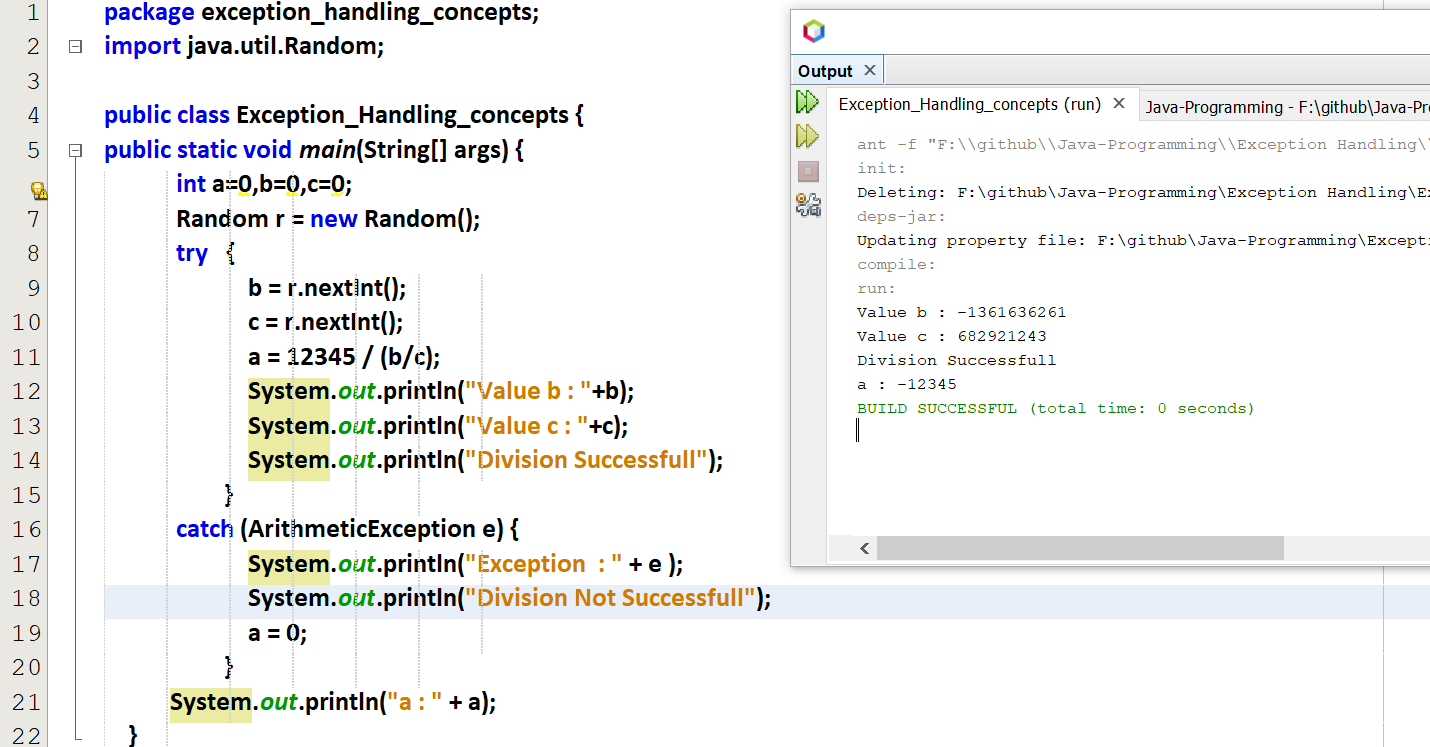
1st catch 🡪 sub-class  
2nd catch 🡪 super class  
If it is swapped, then the 1st catch(super-class) will shadow/hide the remaining catches(sub-classes) under them.

All the exceptions will be handled by their corresponding catch blocks.

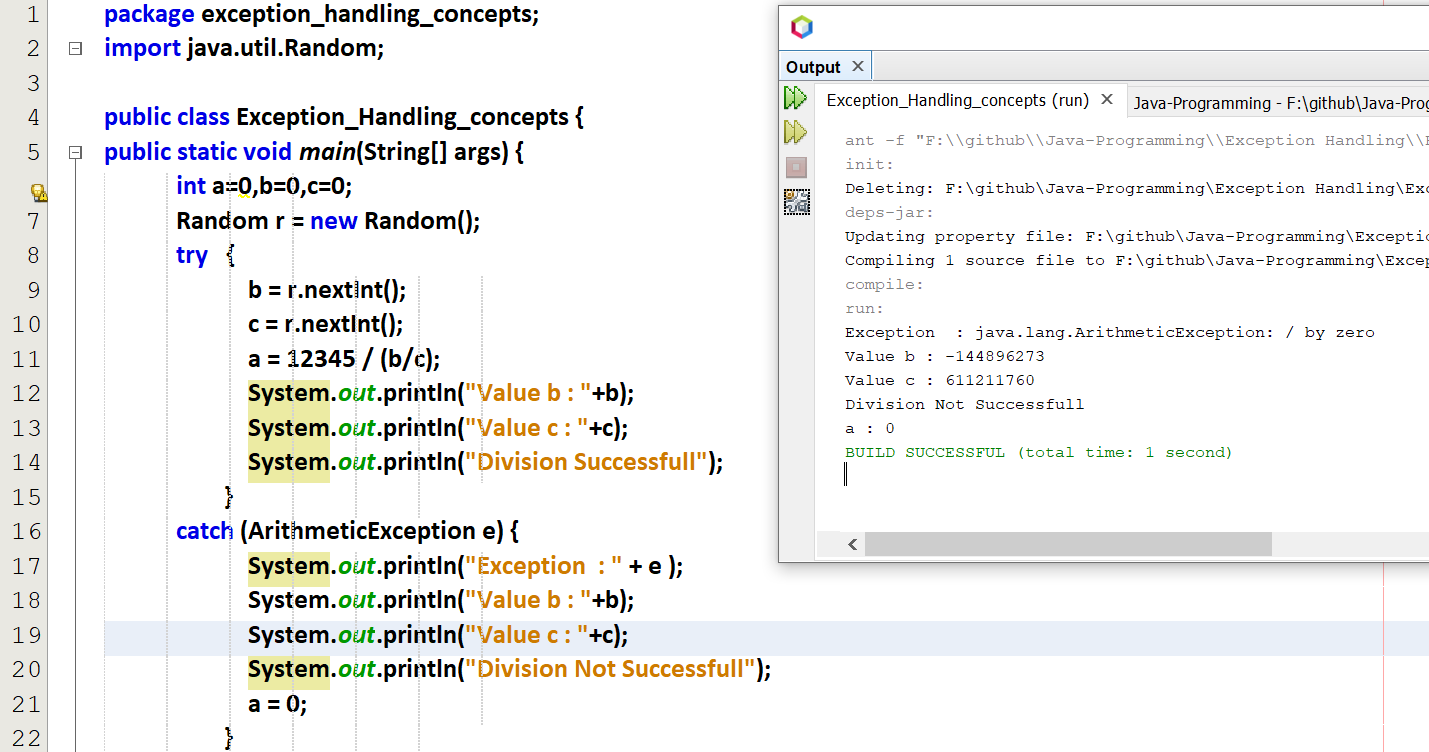


If the exception is caught/not finally will definielty gets executed.

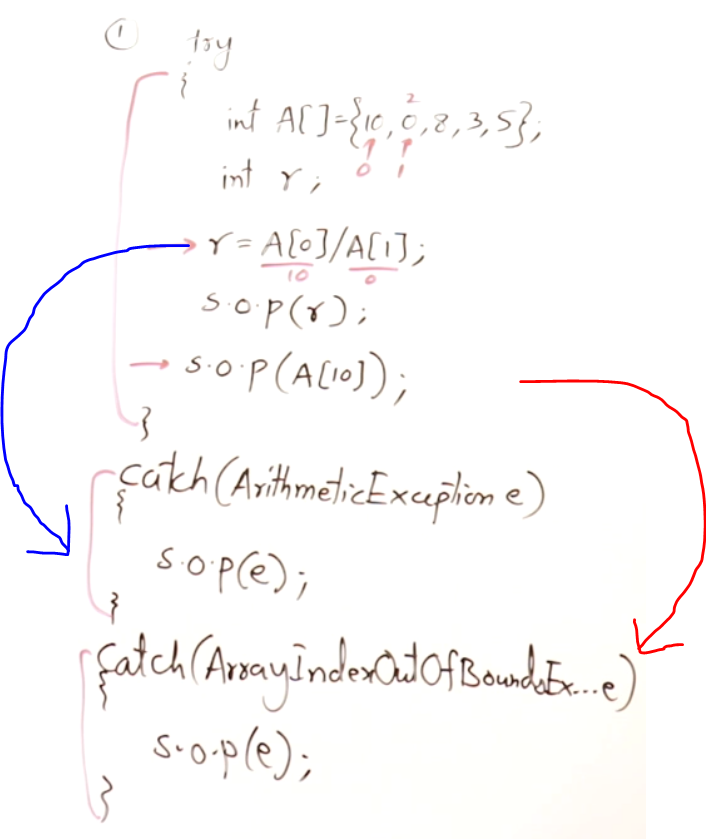
Division is Successful (b>c) [Exception Not caught]

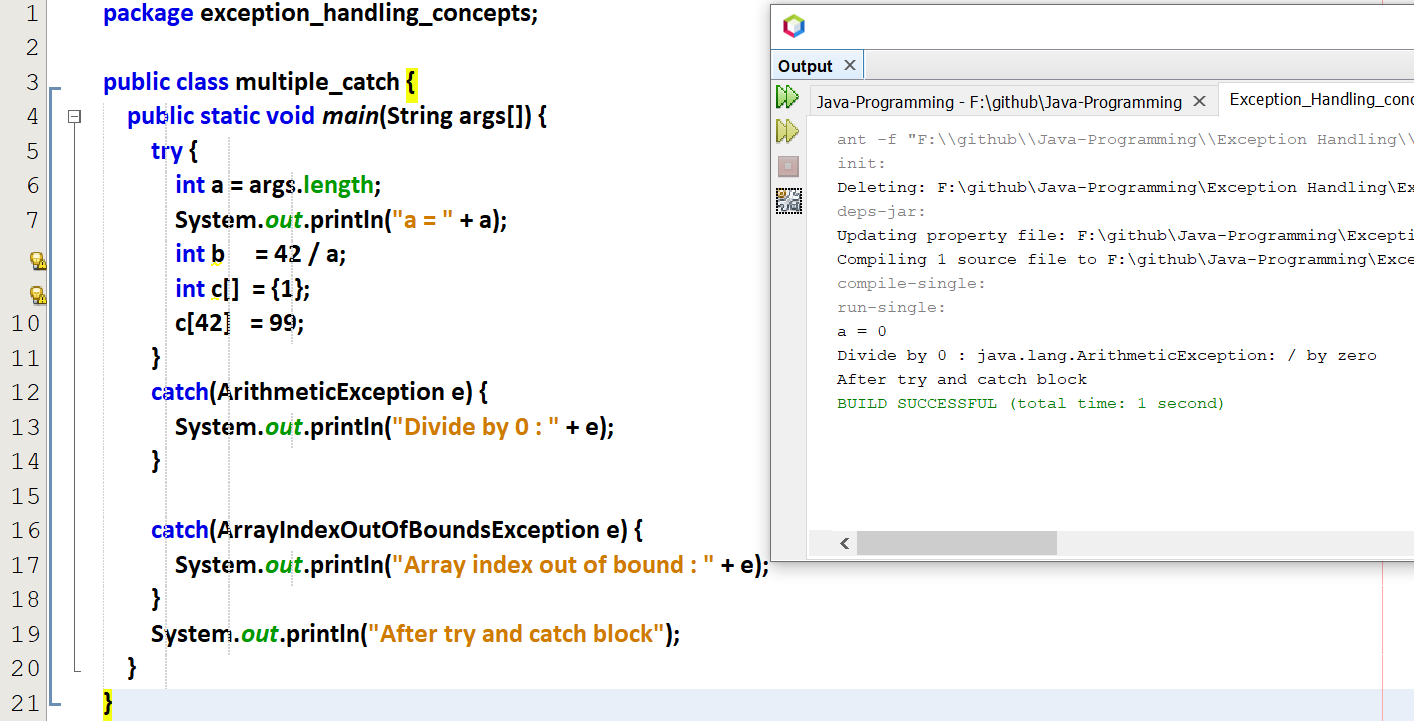


Division is Not Successful (b<c) [Exception caught]



# **Multiple catch with a single try**

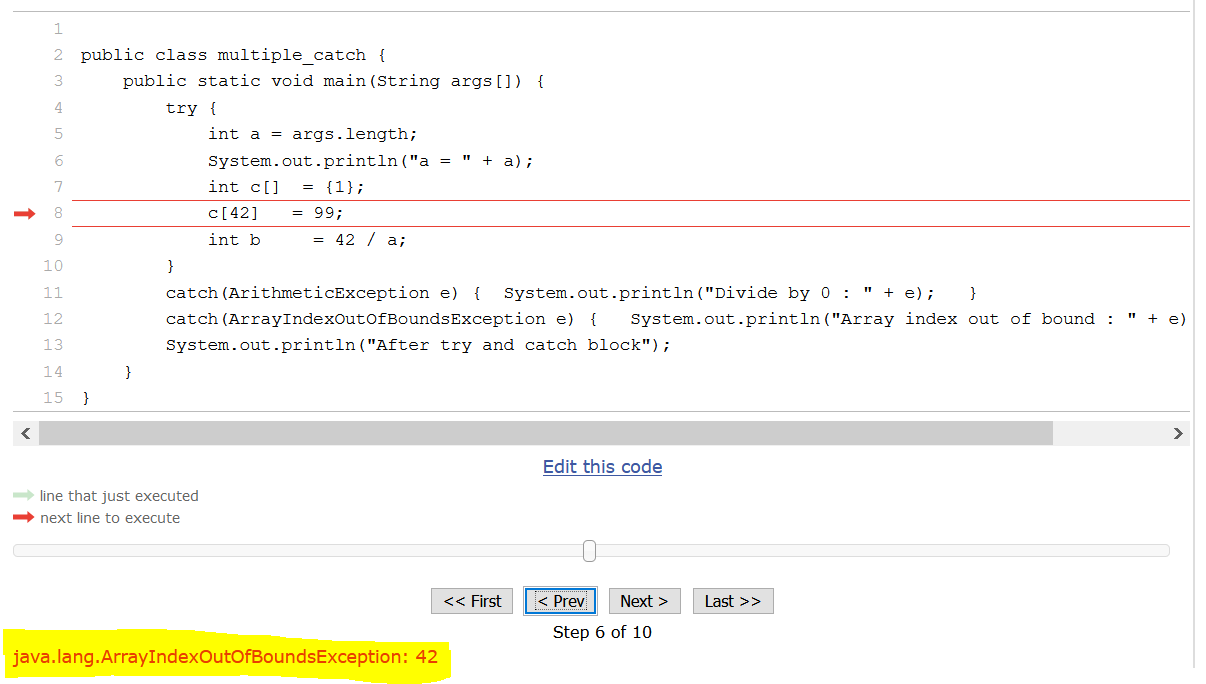




By default when I run the above code in Net-beans, String args[] = 0. So a=0. So only getting Division by 0 exception.  
And array index out of bound exception is not raised.



After division by 0 exception is caught, the program ends.

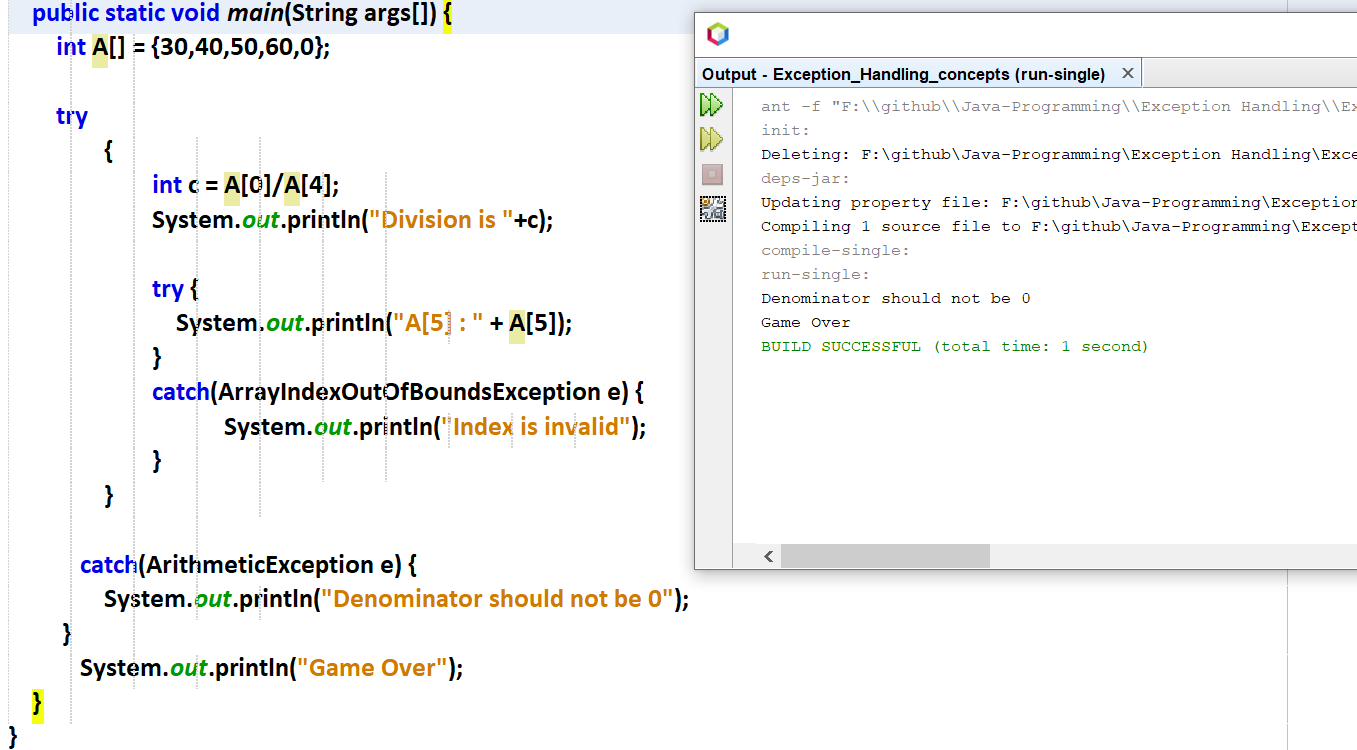


After array index out of bound exception is caught, the program ends.

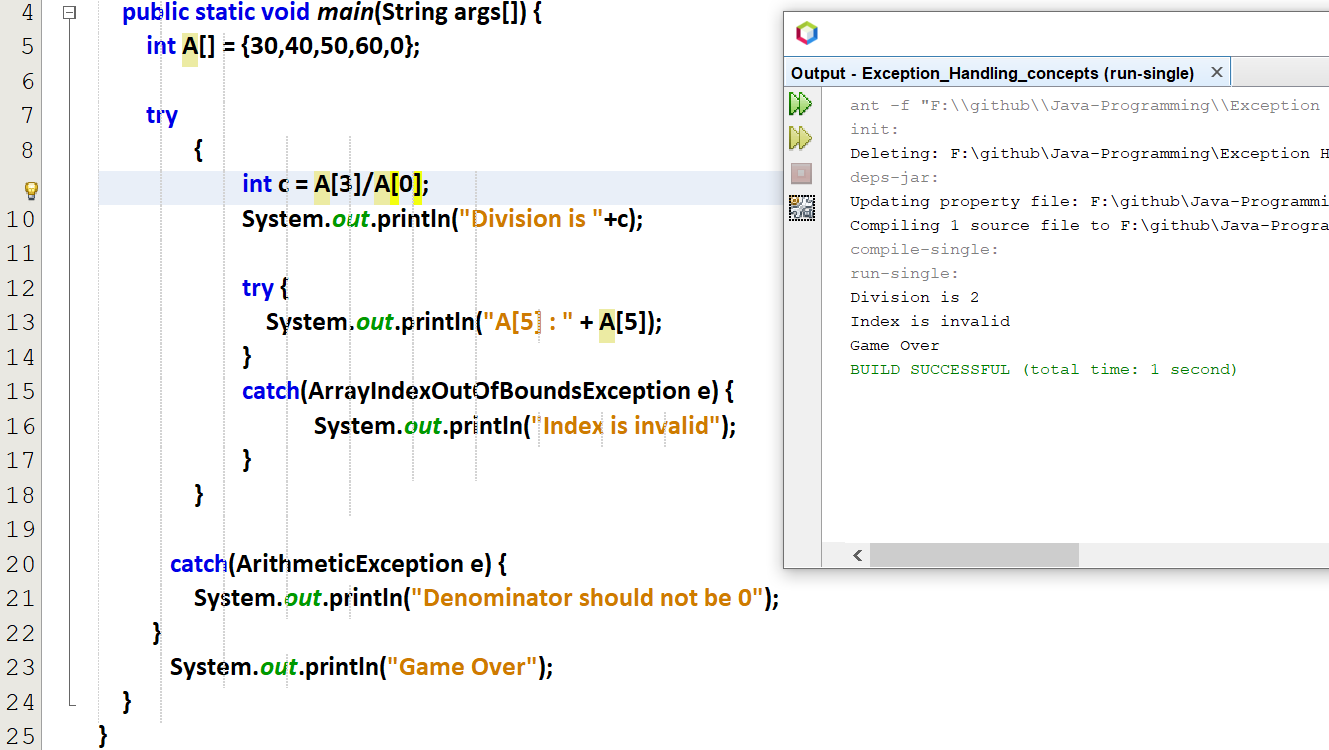


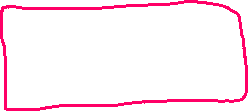
When I give an array of 5 integers as input, so args length=5 , so division exception is not found.  
And array index out of bound exception is raised.

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





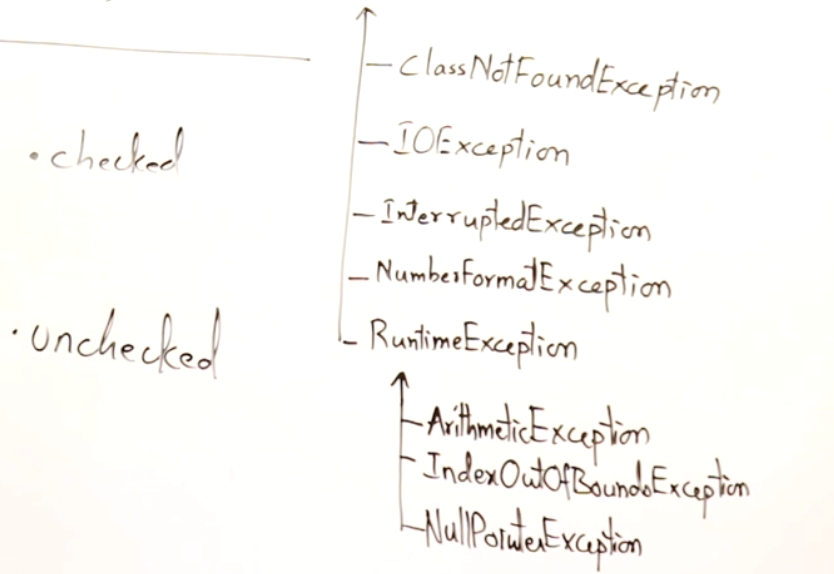




# **Checked and Unchecked Exception**

object is the mother class of all JAVA classes

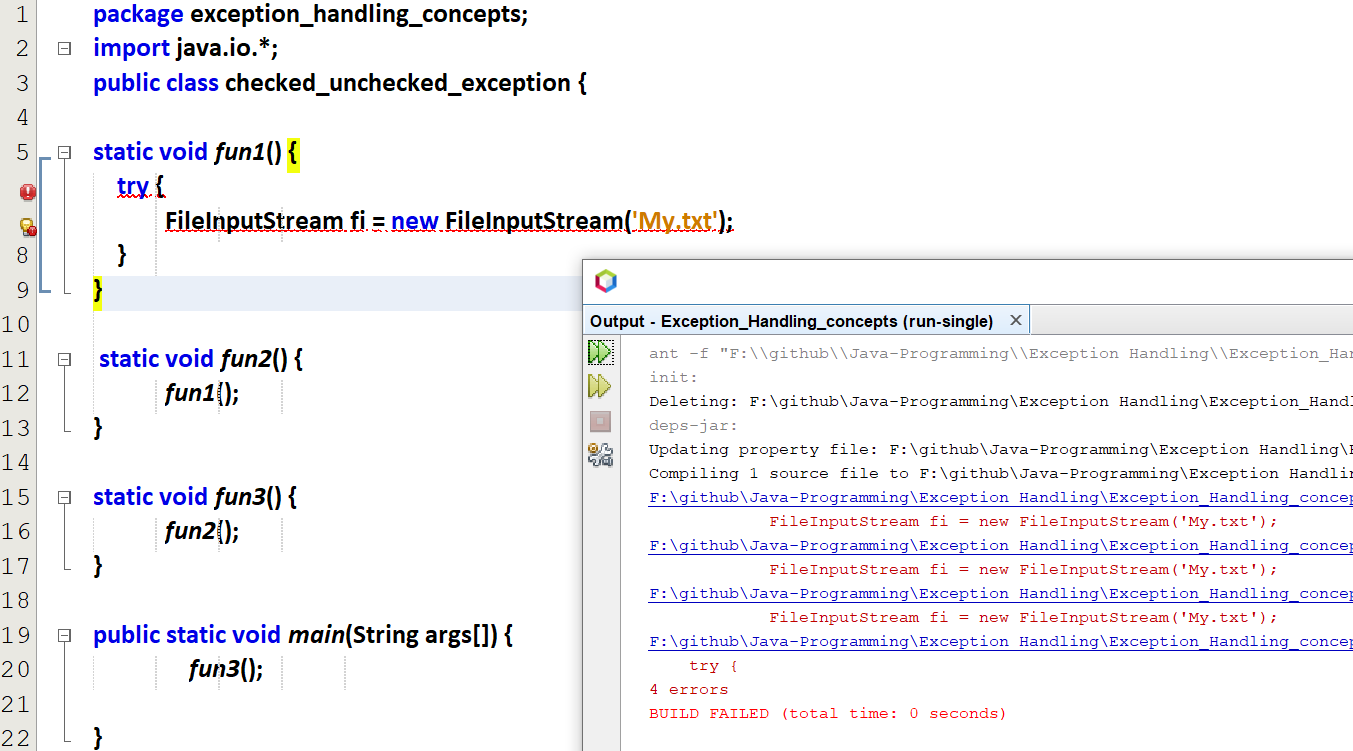
I/O exception 🡪 File not found exception  
Interrupted Exception 🡪 If a thread stops abnormally   
NumbersFormat Exception 🡪 If I give a number as a string





## **Checked Exception**

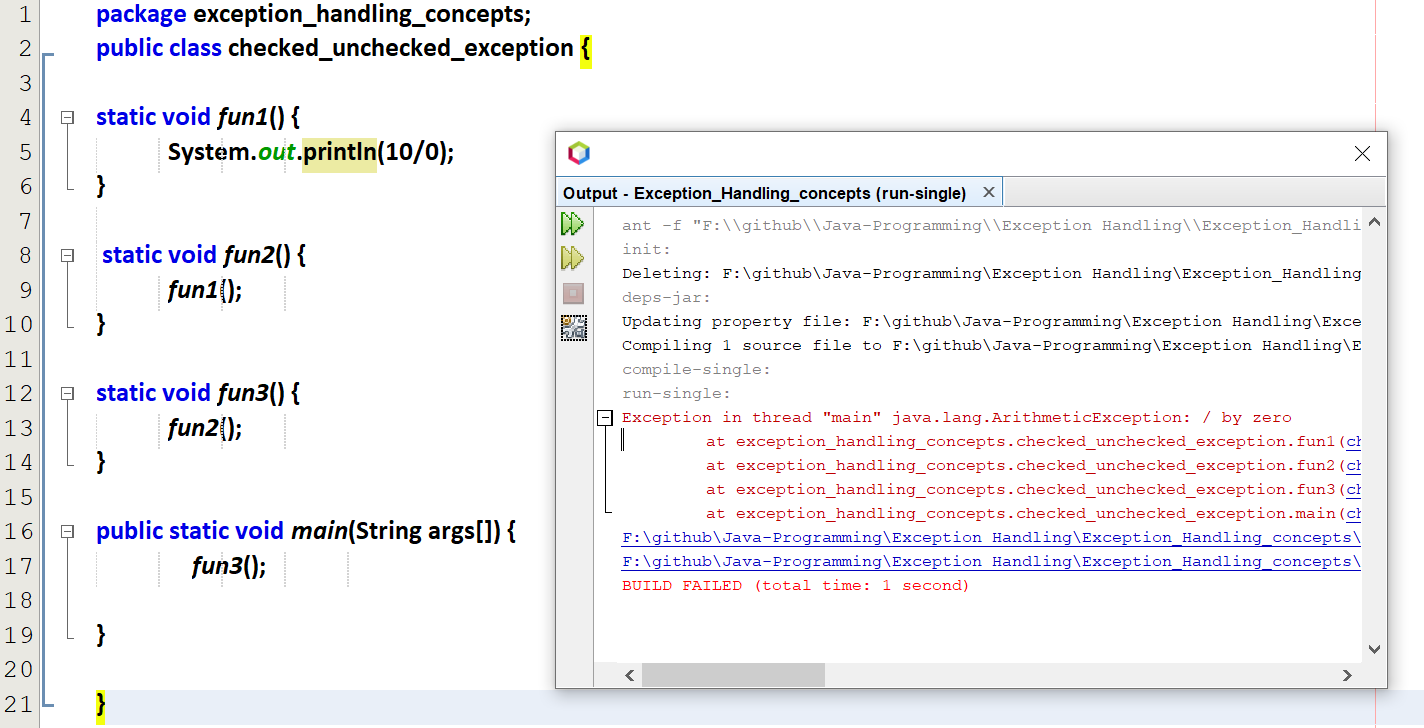
It is necessary to handle this exception using try and catch. We cannot escape from this exception. JAVA will force to handle checked exception using try and catch. Trying to open a file



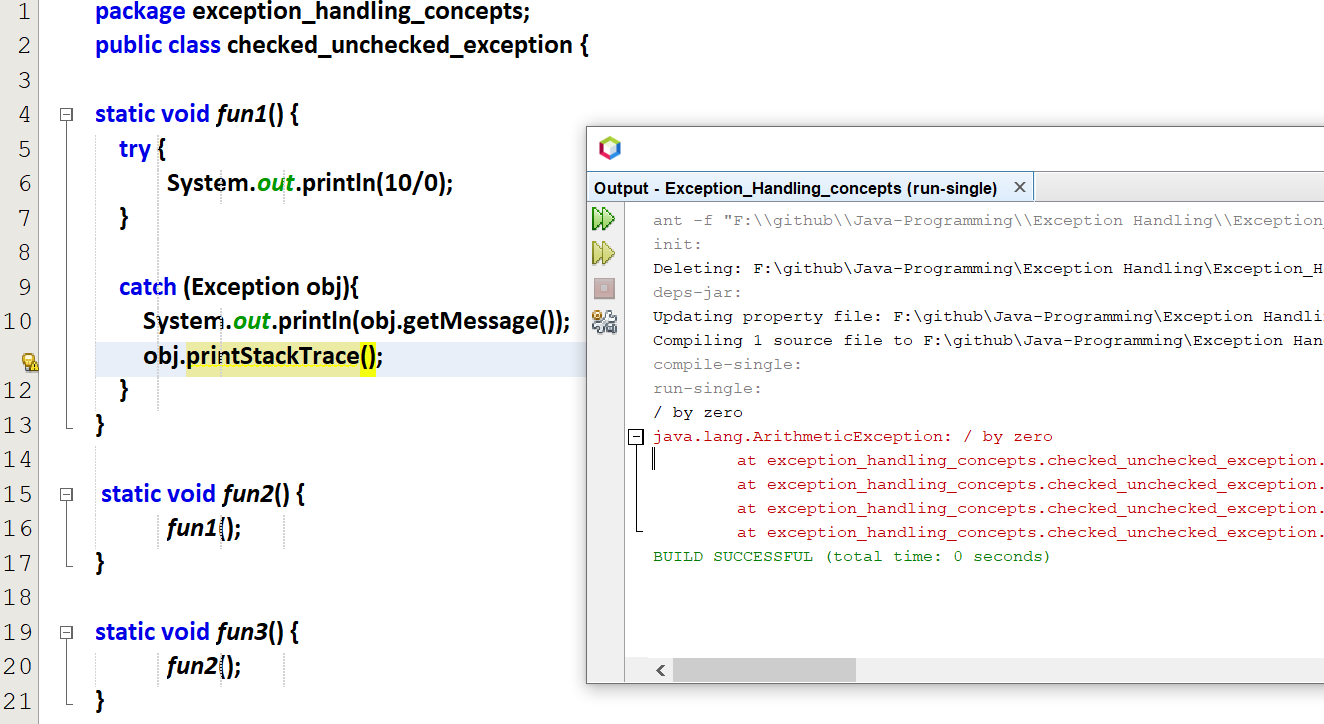
Java will definitely raise an error for the un-checked exception.

## **Un-Checked Exception**

It is Not mandatory to handle this exception. We can may/may-not handle this exception. JAVA will not force to handle unchecked exception.   
If we handle, then the program will run smoothly.  
If not, then the program crashes.

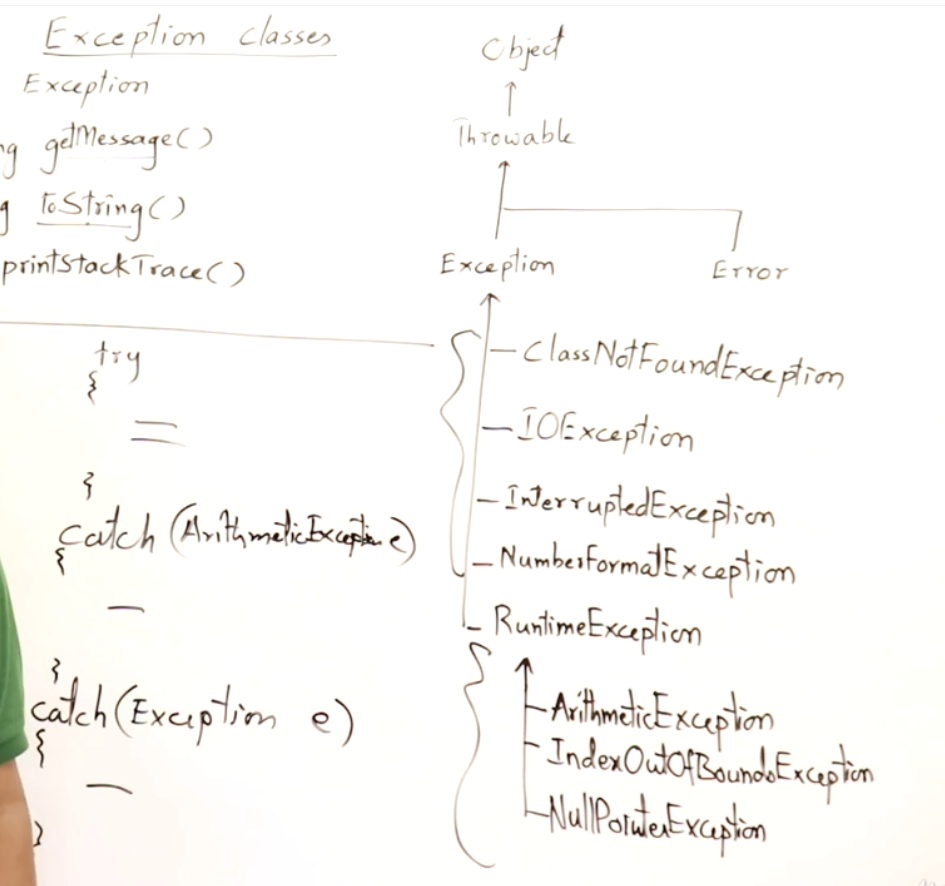


Handle the above un-checked exception using try and catch



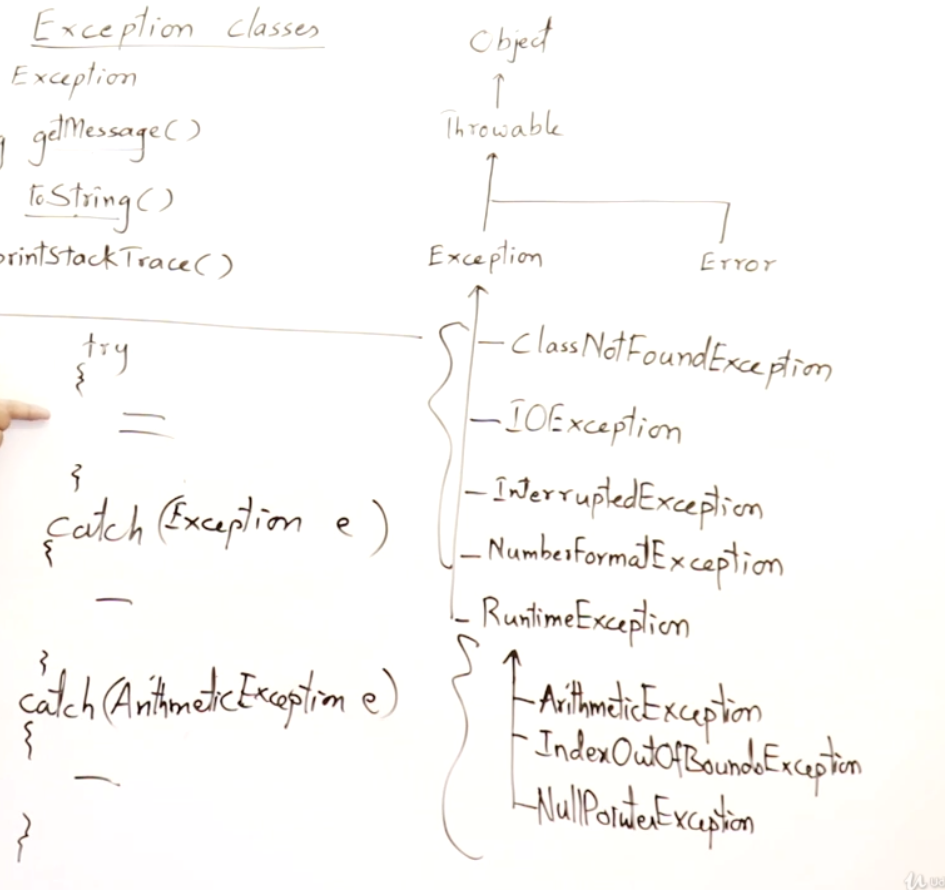
The stack of functions are also displayed.

# **Hierarchy of Exception in Multiple catch blocks**





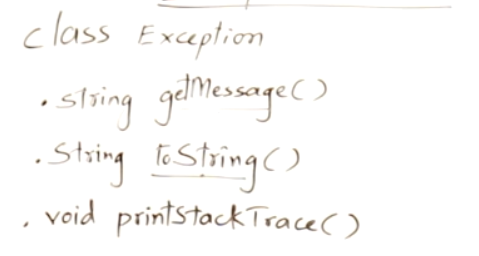
It will check for Arithmetic Exception, if no Arithmetic Exception it will look for the other exceptions in Exception parent class.

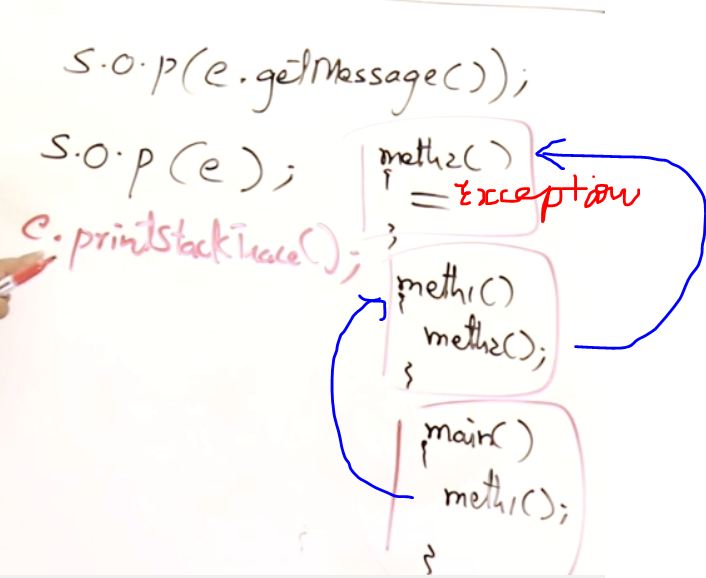




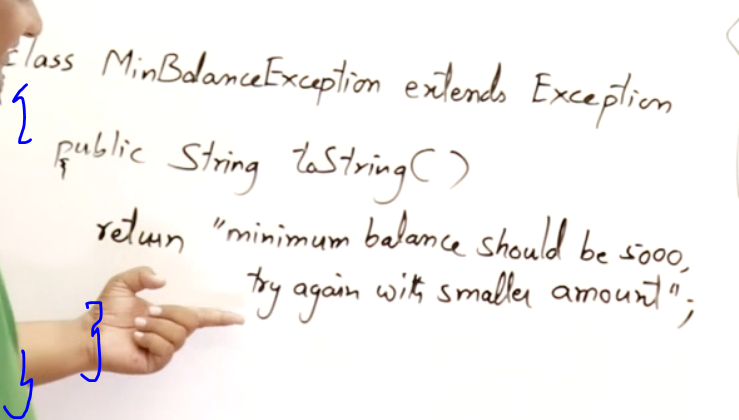
Compiler will raise an error, first catch 🡪 sub class, next catch 🡪 super class

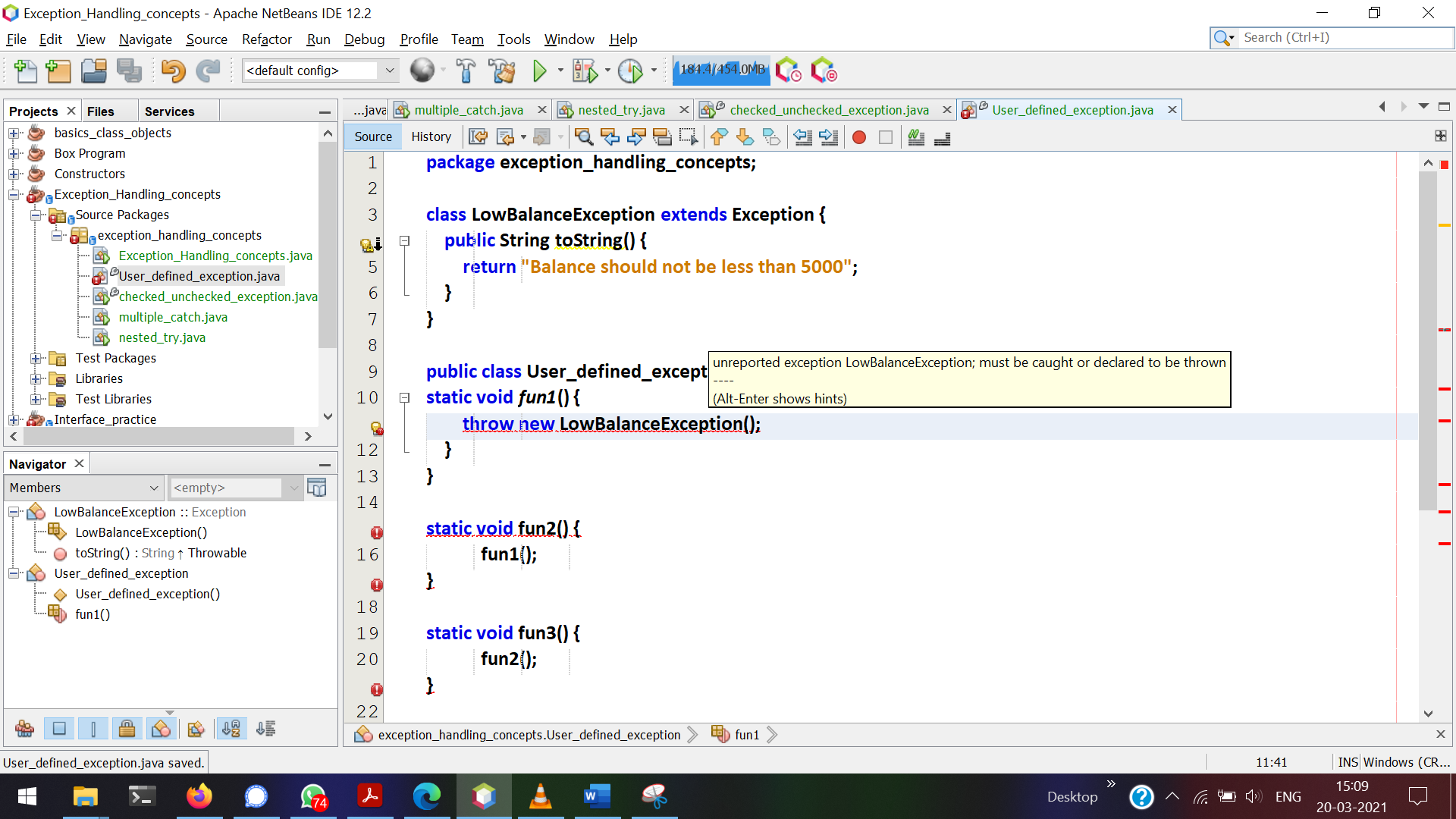
# **Important methods in Exception class**





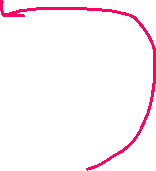
getmessage() 🡪 Message about the exception. sout(obj.getmessage()).  
ToString() 🡪 Message about the exception. sout(obj) 🡪 automatically toString() will be called.  
printStackTrace() 🡪 Sequence of method calls when the exceptions were raised.



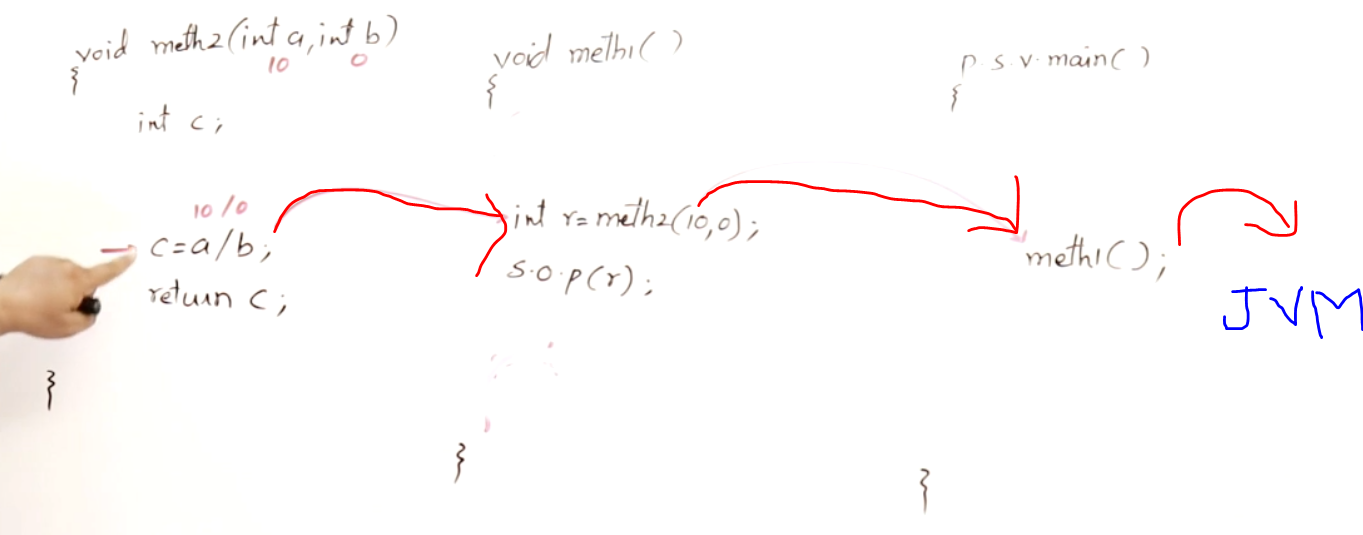


LowBalanceException is a checked exception since it extends from Exception.   
So this can be handled via try and catch or throws





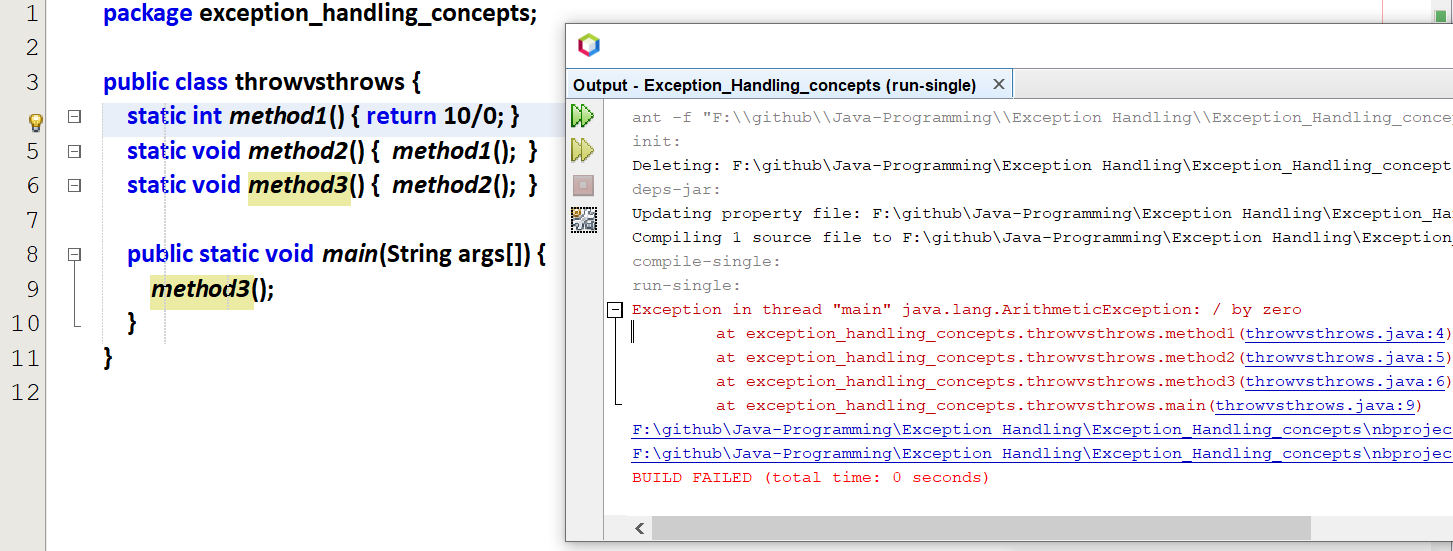
# **Propagation of Exception**

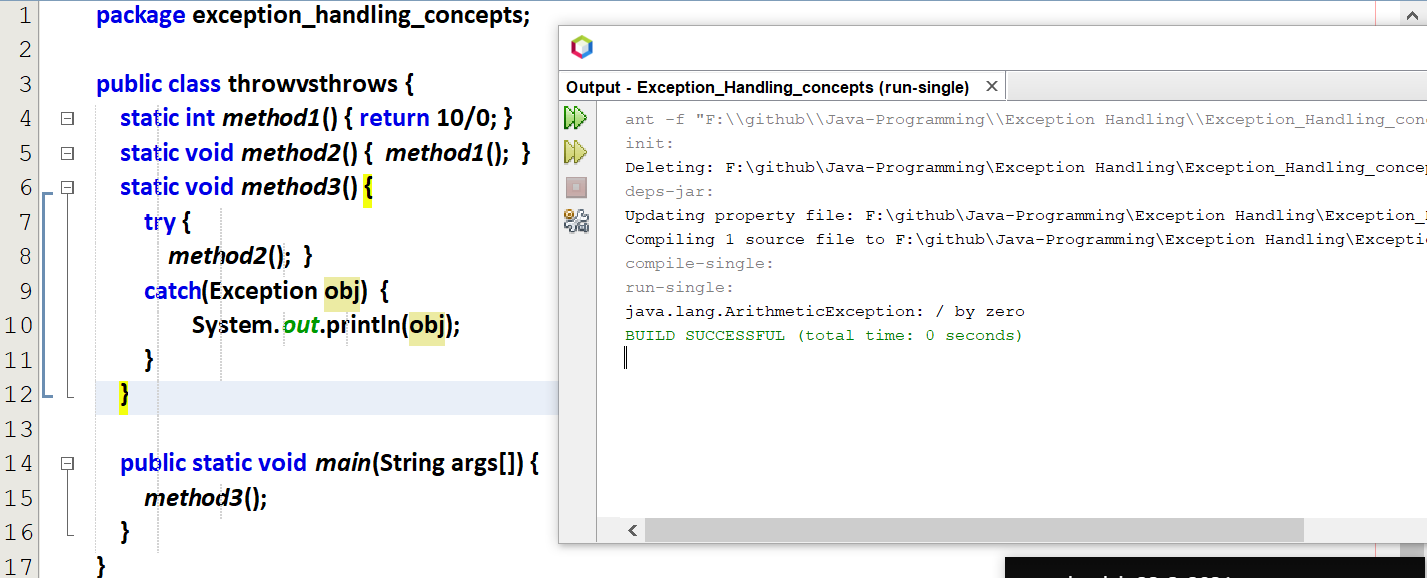


If try and catch not available in

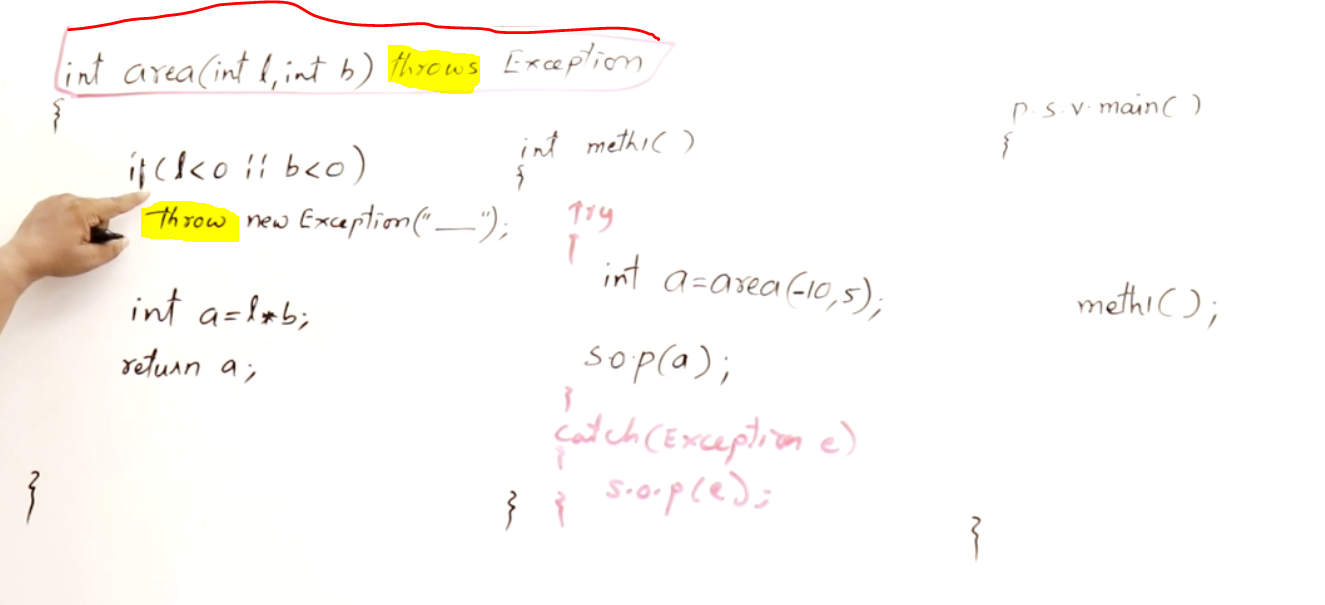
Method-2’s exception will be handled by method-1.  
Method-1’s exception will be handled by main().  
main’s exception will be handled by JVM and then the program crashes.

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???

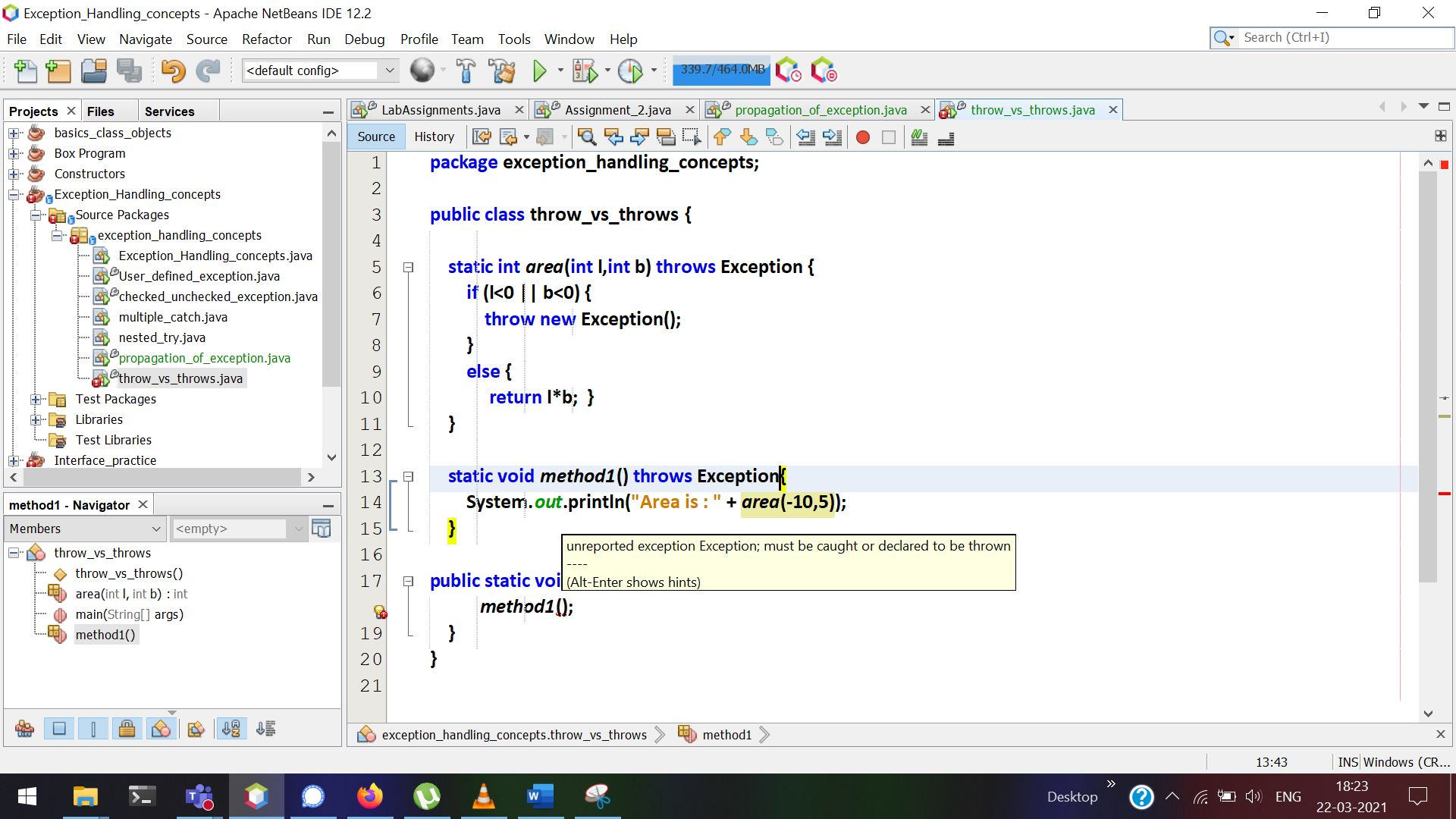
# **Throw vs Throws**

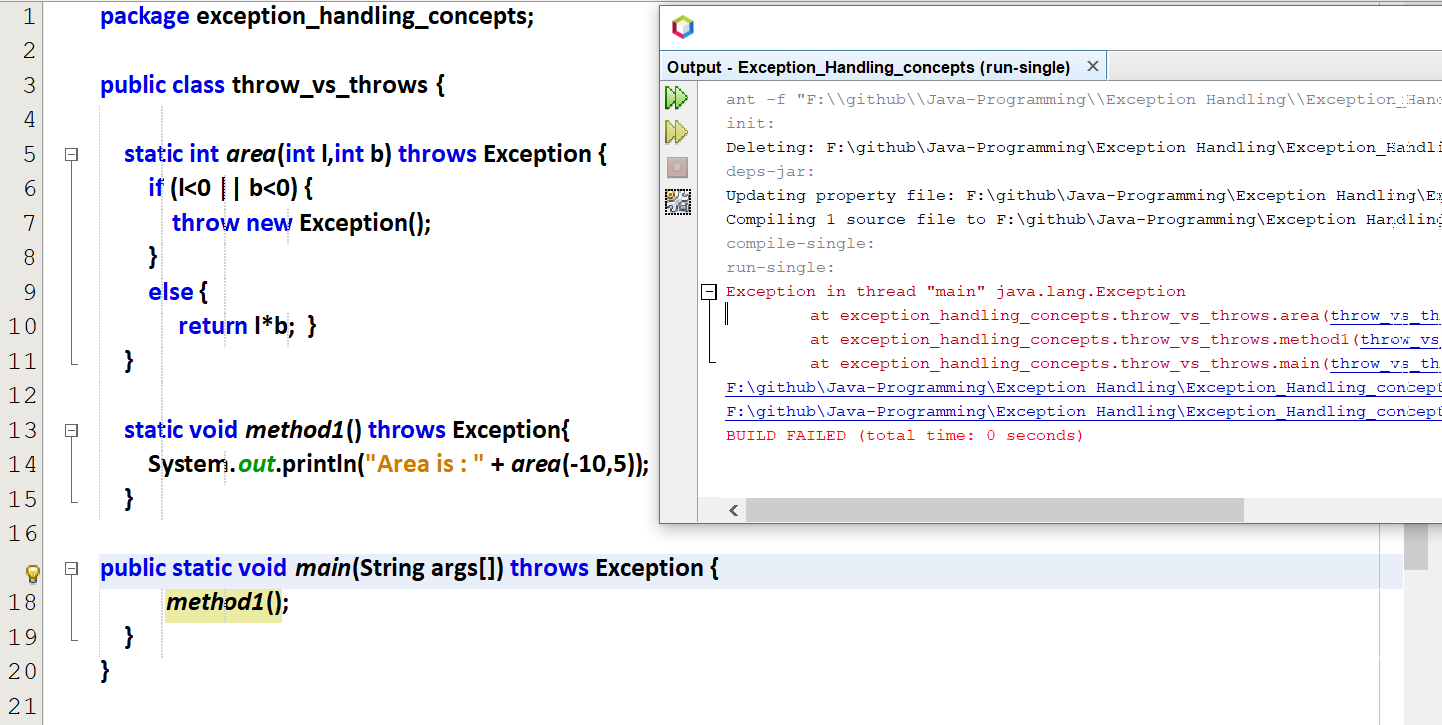


If I want to use the area value in area(), we can use try and catch.  
If I don’t want to use the area value in area(), and want to access the area value in meth1() and main(), then we can use throw()  
area() throws an exception and it will be handled my method-1 using try can catch.

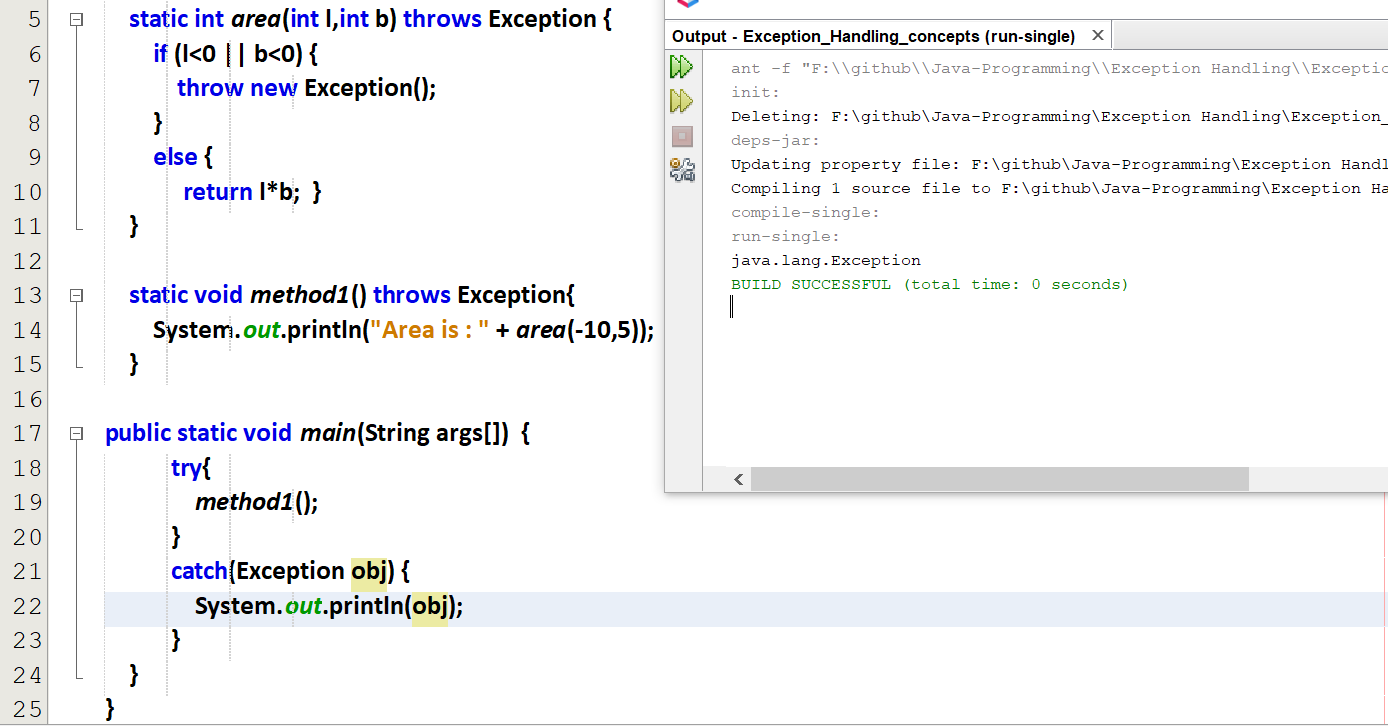
If we don’t want to handle in method1(), that exception can also be handled in main() also using try and catch.





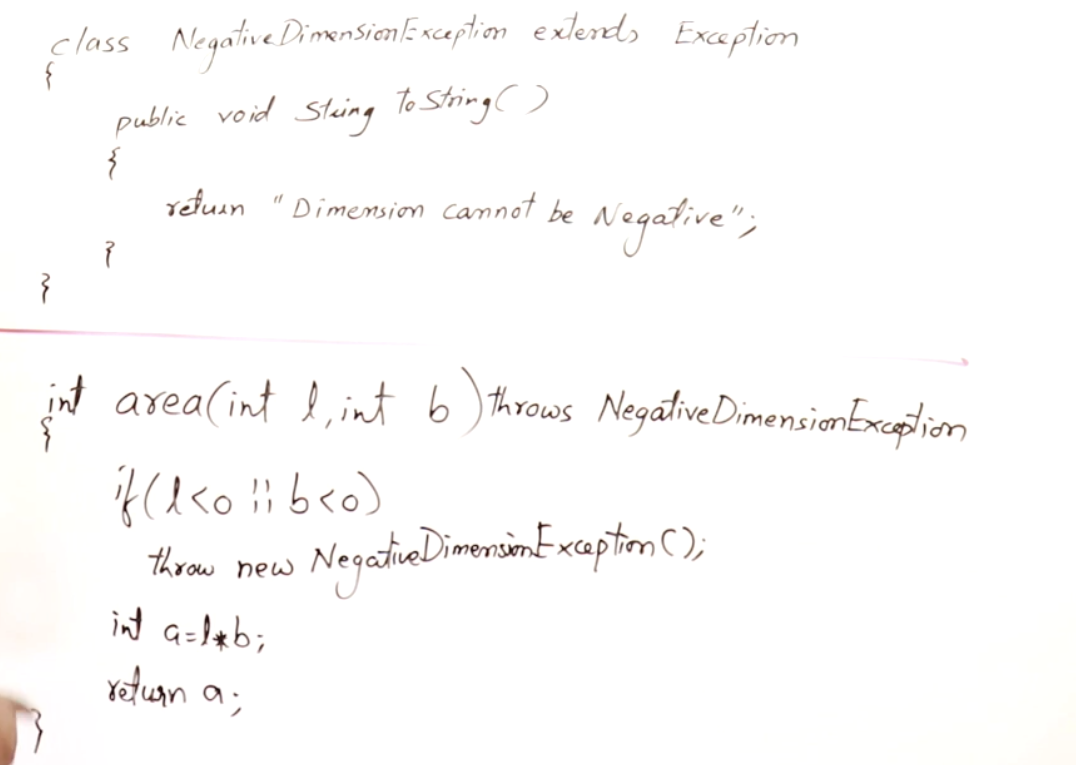


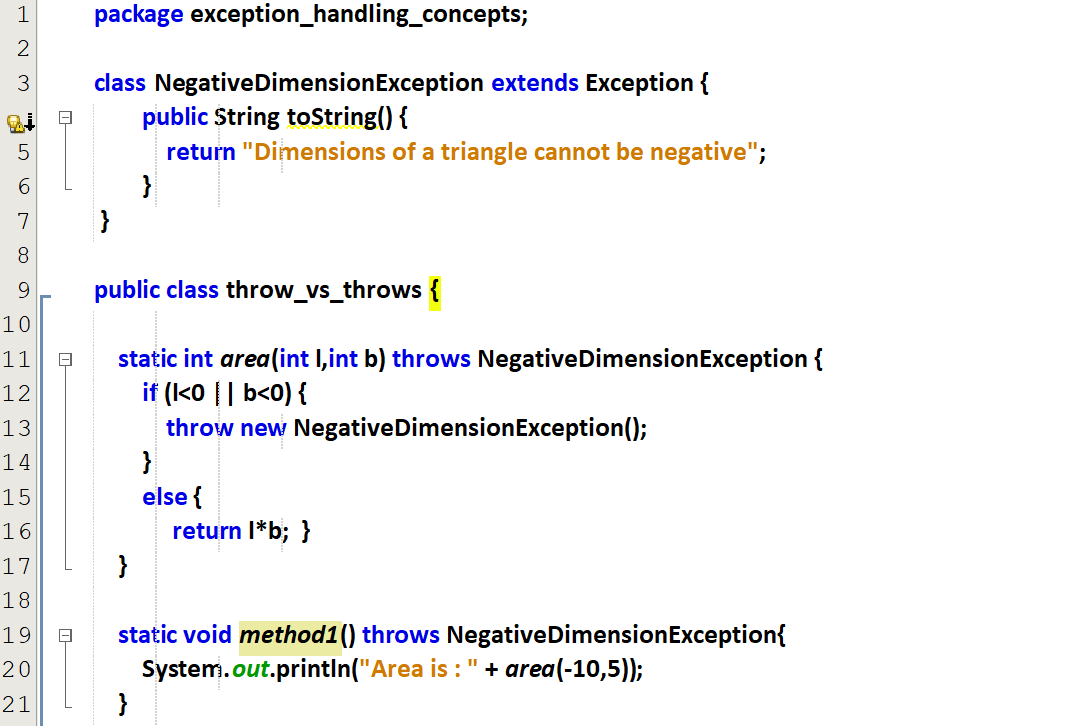
Now JVM cannot handle this exception and got crashed.

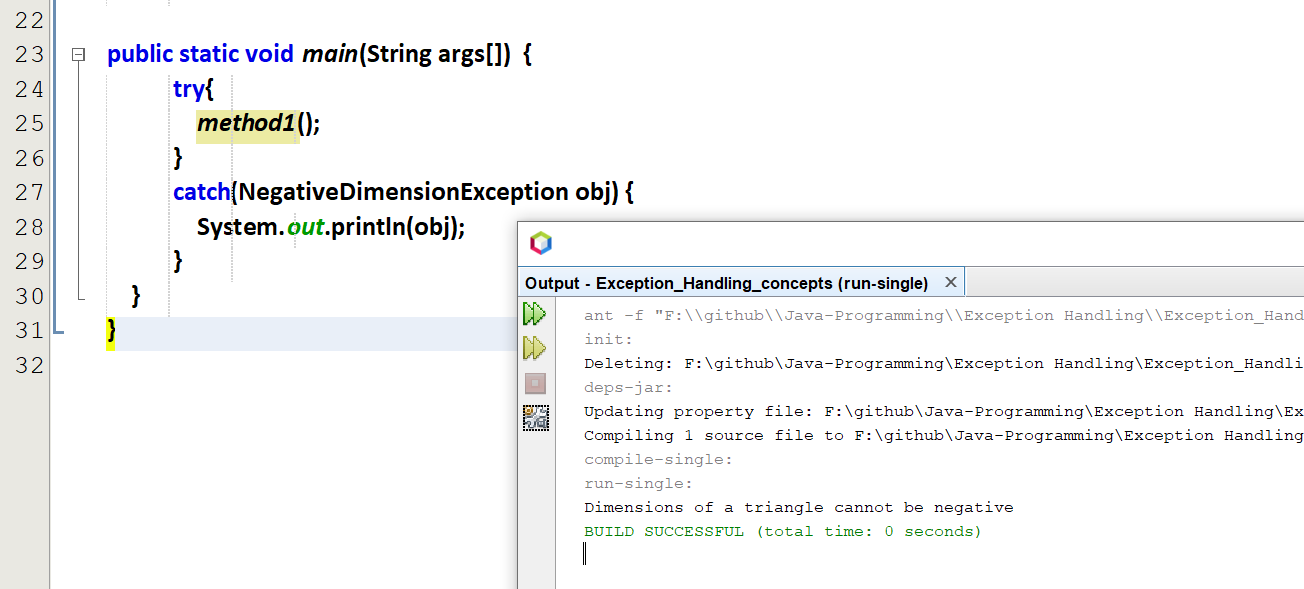
This exception can be handled in main()  


# **user-defined exception**

Every user-defined exception must be inherited from Exception class

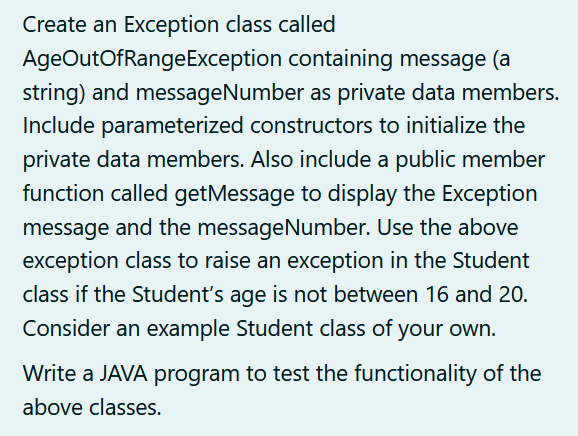
User-defined exception with throws  




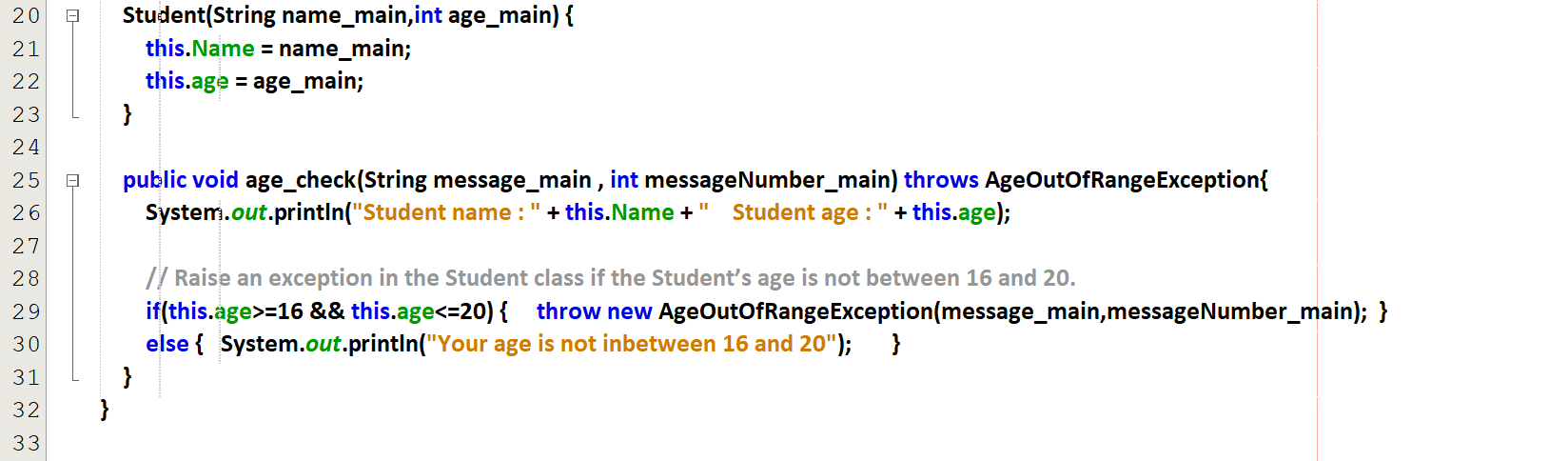
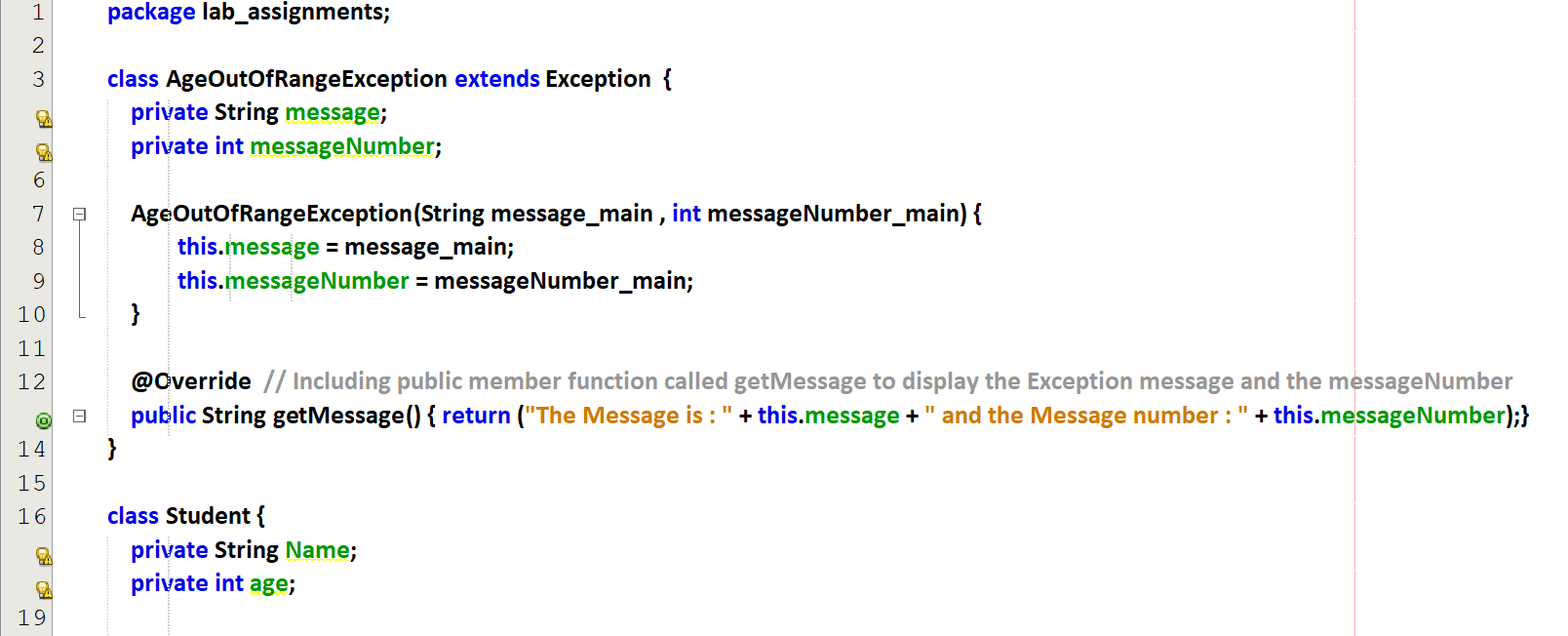
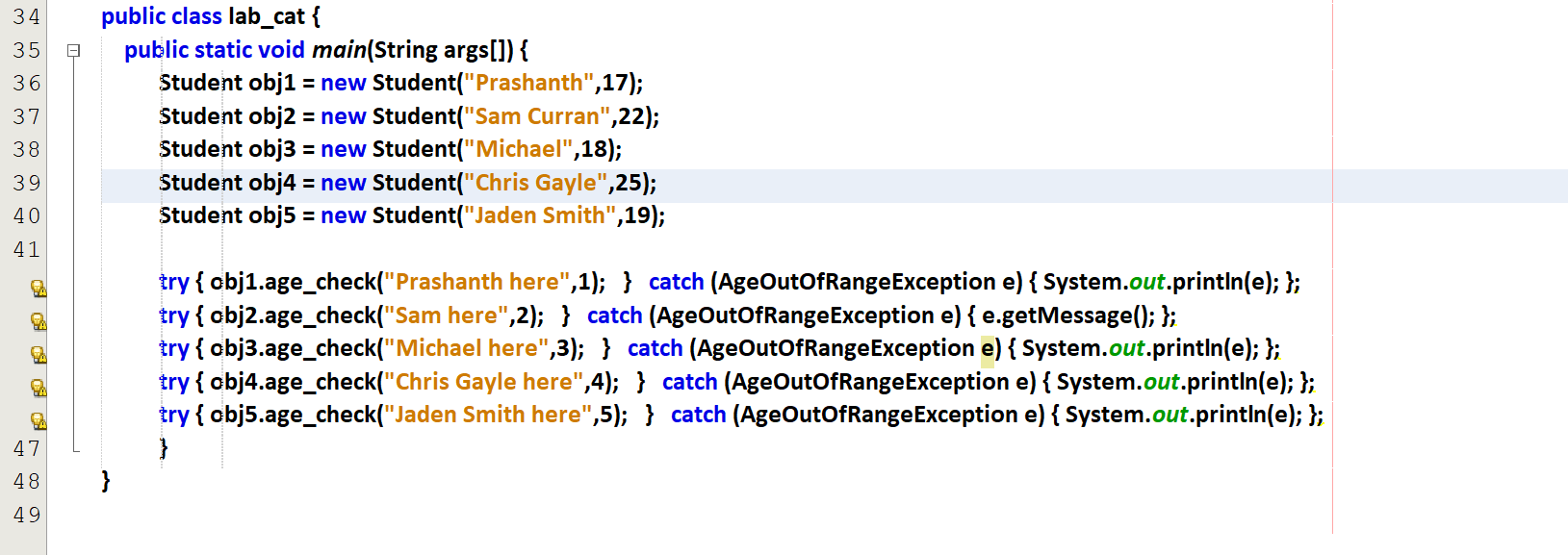


## Example program

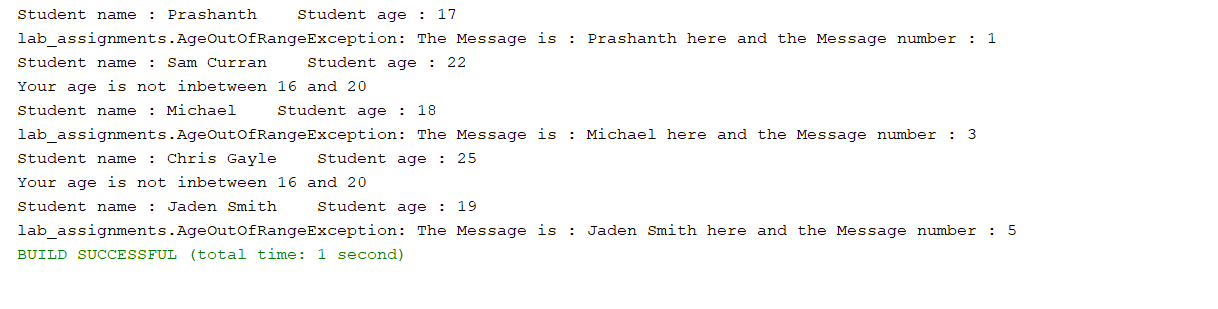
Refer labcat.java



Code

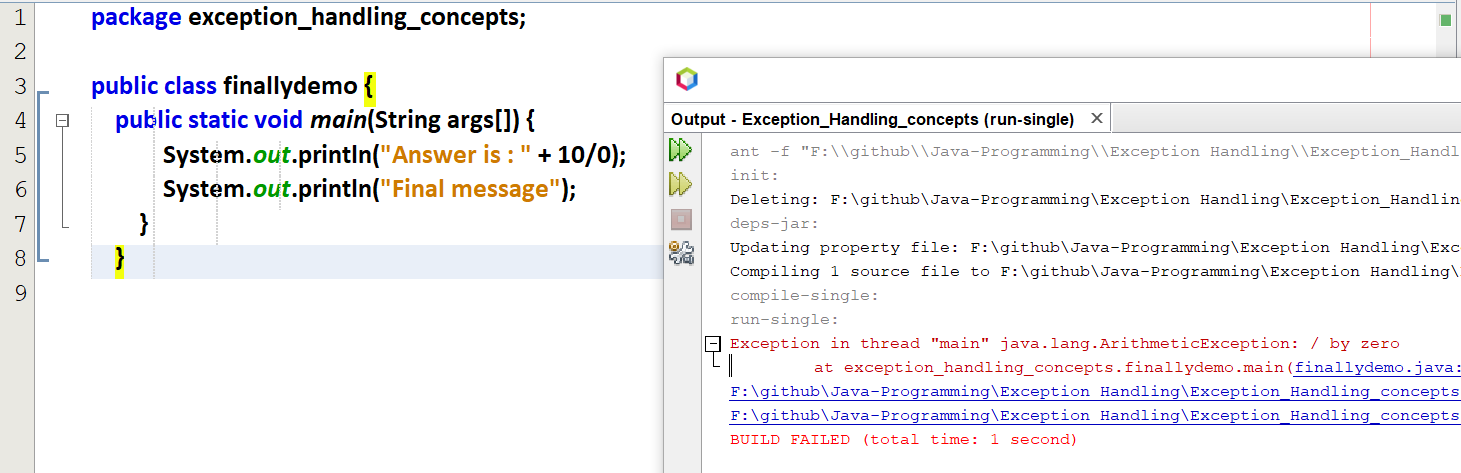
  


Output :



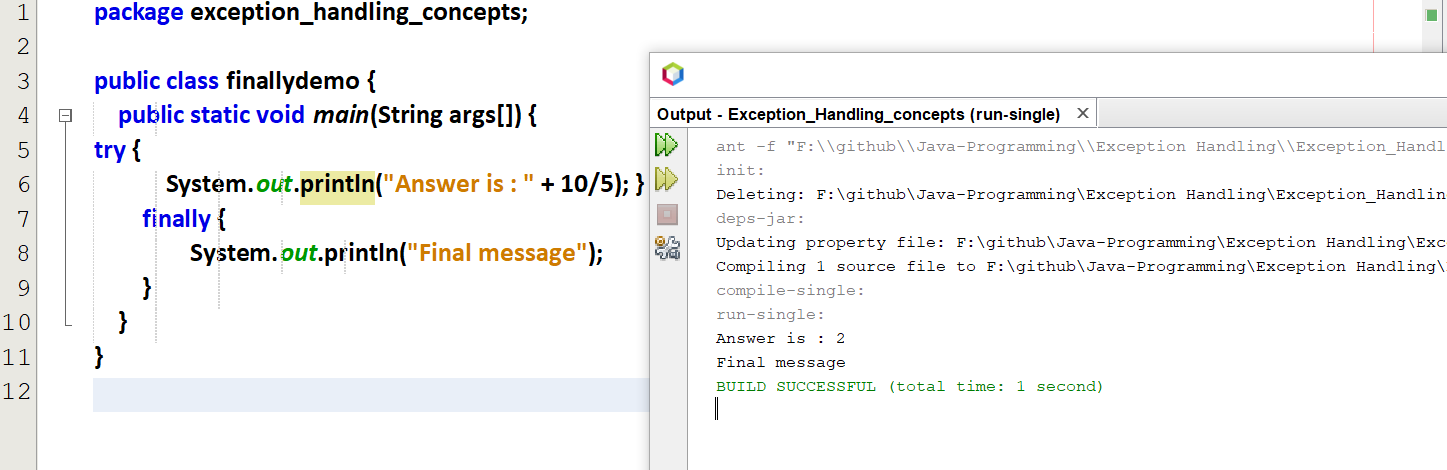
# **finally**

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

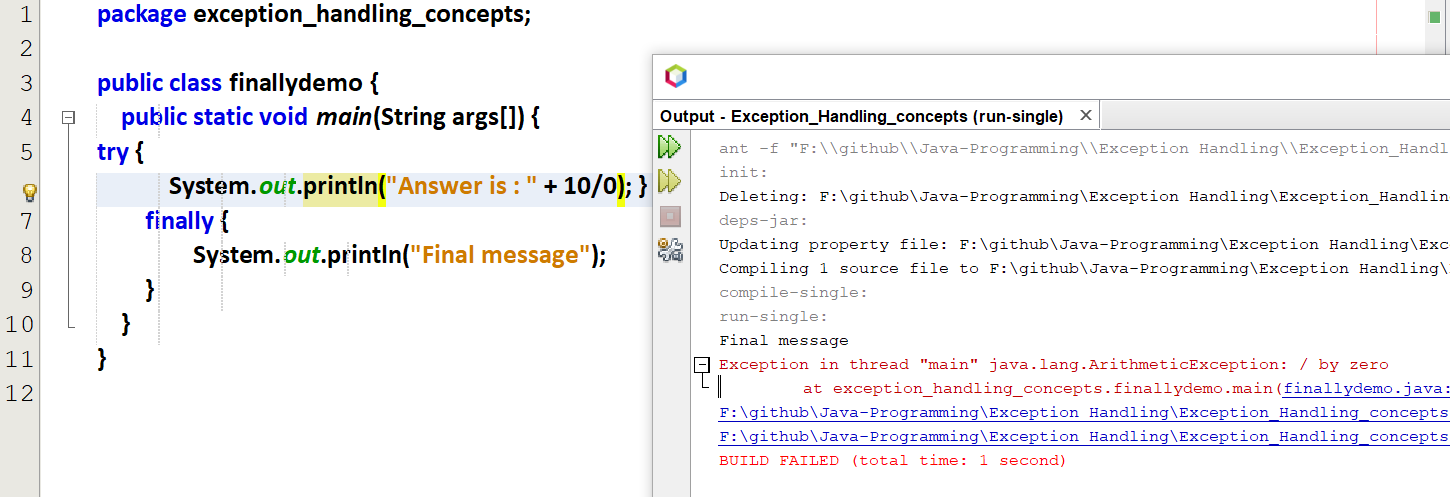


Now “Final message” is not printed after exception.

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

