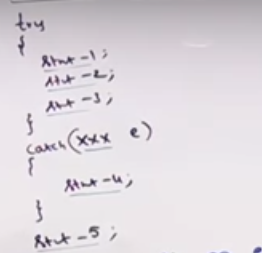
Control flow in Try catch:

Scenarios:



**Scenario1: If no exception occurred?**

Answer: statement1,2,3 and 5 will execute but statement 4 won’t be executed.

This program will get normal termination since all the statements are executed perfectly.

**Scenario 2: If exception occurs at statement 2 mean?**

Answer: Generally, if inside the try block exception occurs and eventhough we have corresponding catch block, remaining try block statement won’t be executed.

So, statement 1, 4 and 5 will be executed but statement 3 won’t be executed. This program will be considered as normal termination since compiler checking at end of final statement (Statement 5).

**Scenario 3: If exception occurs at statement 2 and there is no corresponding catch block**

**Answer:** Statement 1 only will be executed and remaining code won’t be executed and program terminates abnormally.

Scenario: 4

If exception is raised at statement 4 and 5 mean which is abnormal termination.

Please don’t think like that exception will raise only in try block. Exception can occurs in any where which mean outside try block and inside the catch block also.

Methods to print exception Message in console:

1.e.printStackTrace() -

2.e.toString()

3.e.getMessage()

We can apply these methods in any exception and error object and these methods are available in Throwable class.

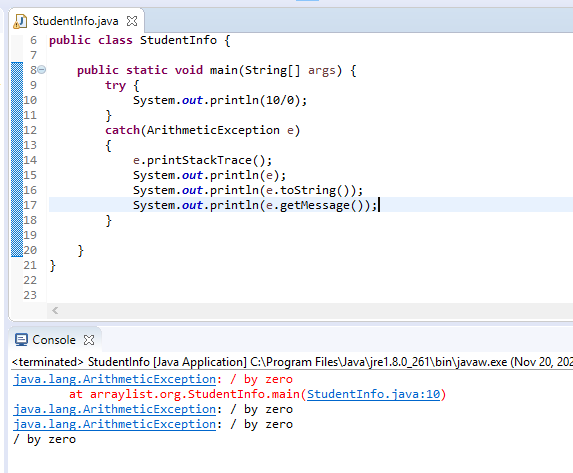
Difference between these methods:

e.printStackTrace()----print the name of the exception,description and stacktrace

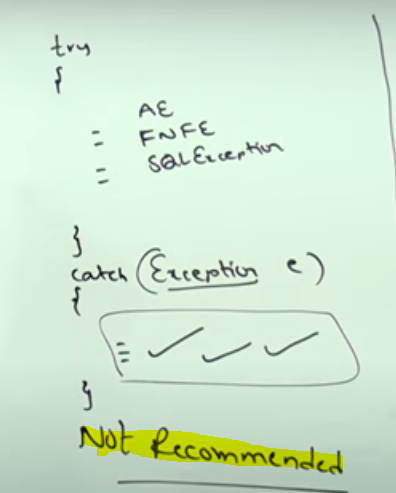
sop( e) and sop(e.toString) ----print the name of the exception and description only & won’t print stacktrace

e.getMessage()---Only print the description… It will not print name of the exception and description

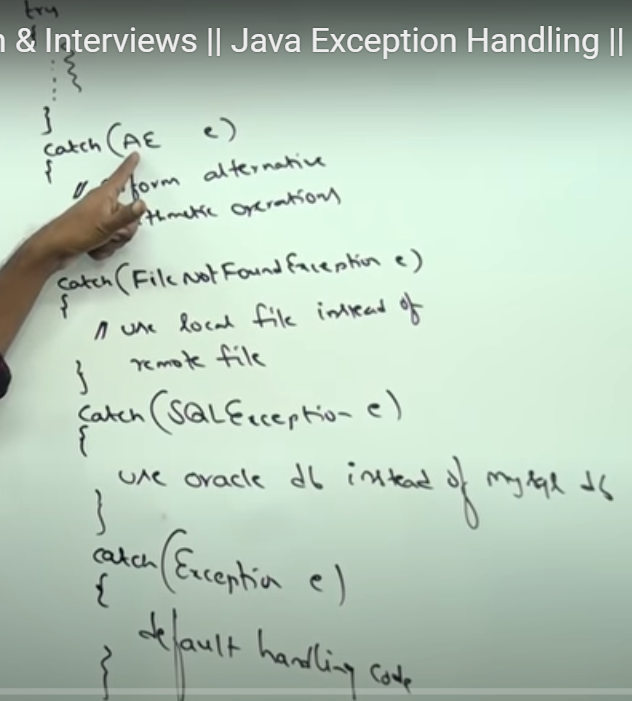
Example program:

  **Try with Multiple catch blocks:**

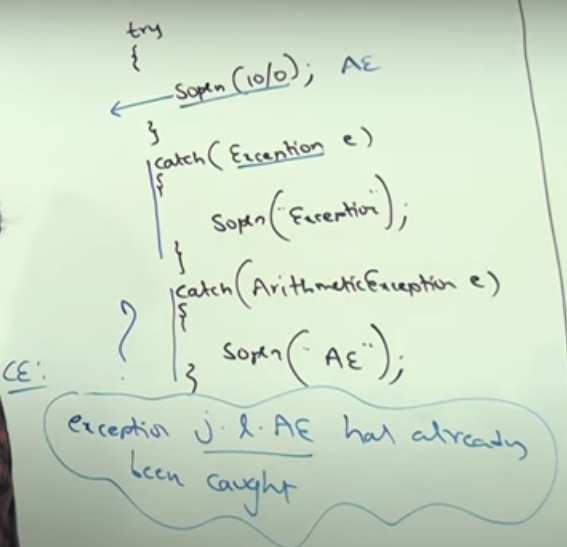
**If try block has multiple exceptions but catch block has only one solution mean that program is not good as given below.**



Below program is recommended:



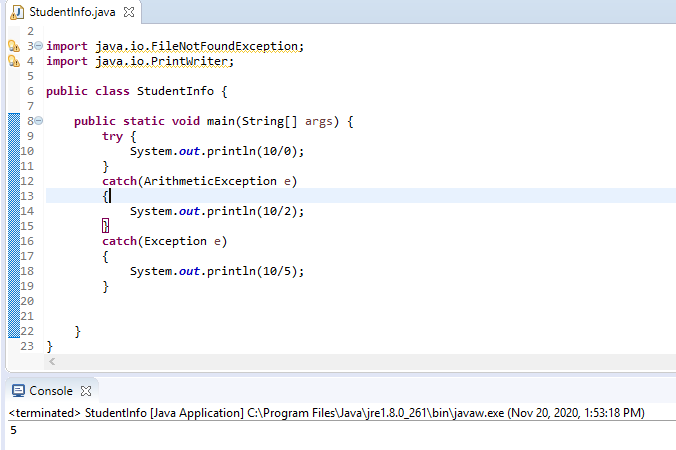
Scenario 1: Arithmetic exception is raised in Try block and multiple exceptions are mentioned but the order is Exception and AE.

 \

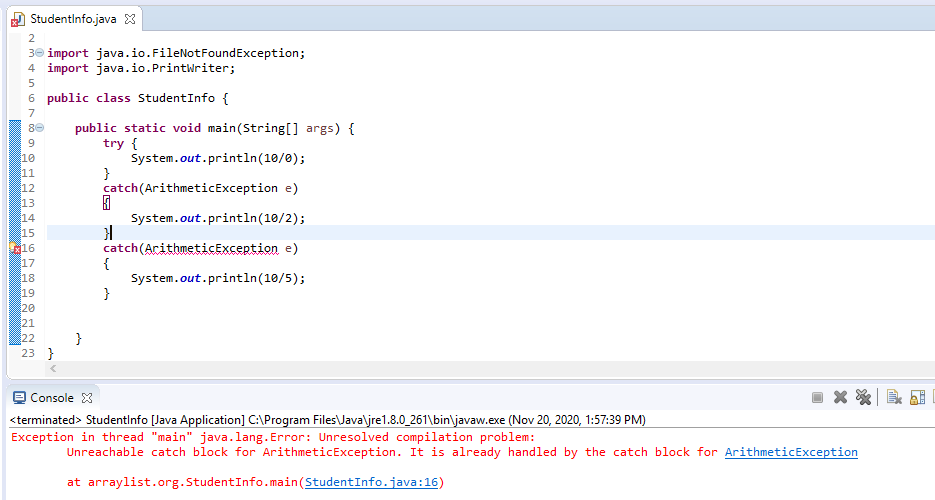
Conclusion: If try with multiple catch block mean the order of the exceptions are important.

Always it should be Child to parent not from parent to child.

**Scenario 2: Child exception to parent exception is perfectly valid.**



**Scenario 3: If same exception solution is provided multiple times in catch block mean we receive compile time error.**

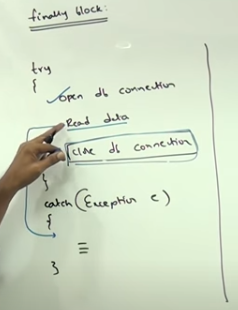


**Finally block:**

Finally, block is having clean up code.

Clean up code mean “Resource deallocation code” like closing the file, db connection….

Clean up code is never recommended to mention in try block. Because, see the below program

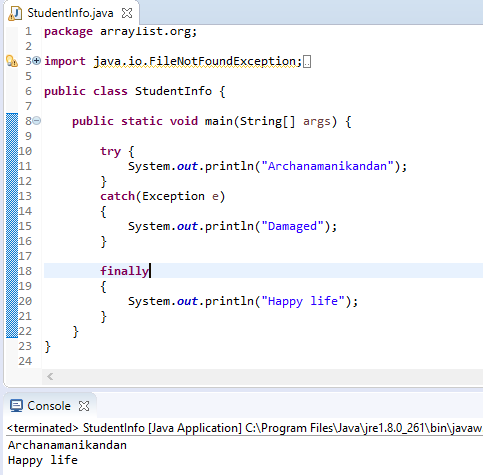
Here, we are having try block to read the data from the file and if file is not found mean we mentioned the corresponding catch block. But what is the issue mean, if 2nd line read data issue mean java handling the exception code from the catch block then who is the responsible person to close the db connection?

Here, finally block is recommended.

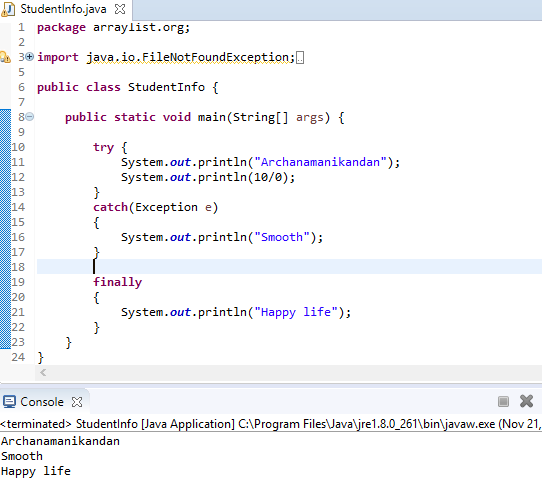
\*\***\*\*Finally block is always executed whether exception raised or not.**

**Multiple cases:**

**Case 1: If there is no exception raised in try block then catch block won’t be executed. Try block and finally block only will be executed.**



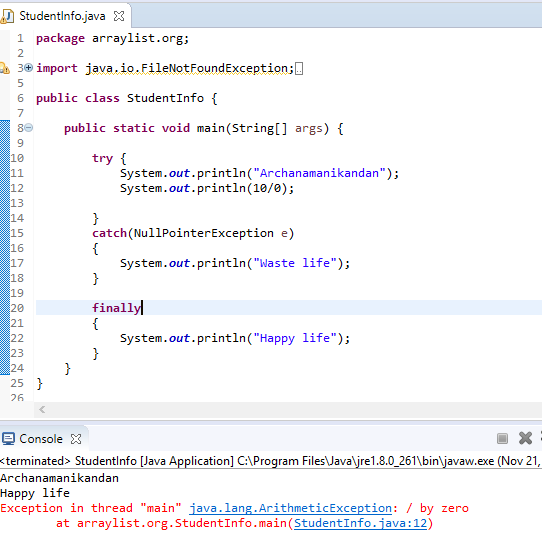
Case 2: If try block has exception and we have corresponding catch block mean there is no issue to print the catch block.



Case 3: If we have exception in try block but there is no corresponding catch block mean what?

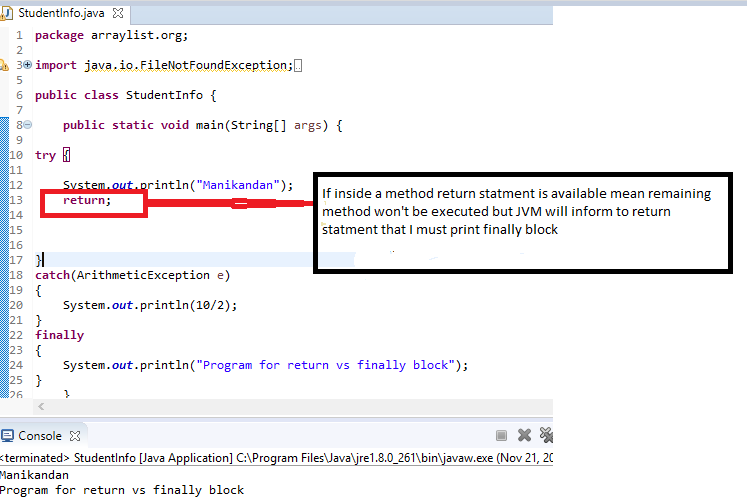
Solution: Abnormal termination If corresponding catch block is not available mean JVM is print like given below.

Try block instruction, finally block instruction then only catch block instruction



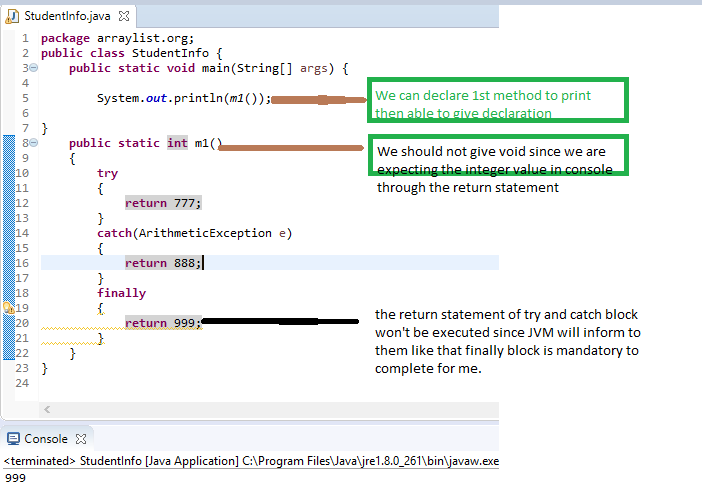
**Finally block vs return statement:**

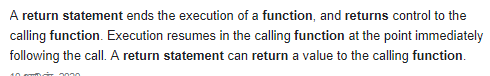
**Return statement:** If inside a method return statement is available mean rest of the method won’t be executed.



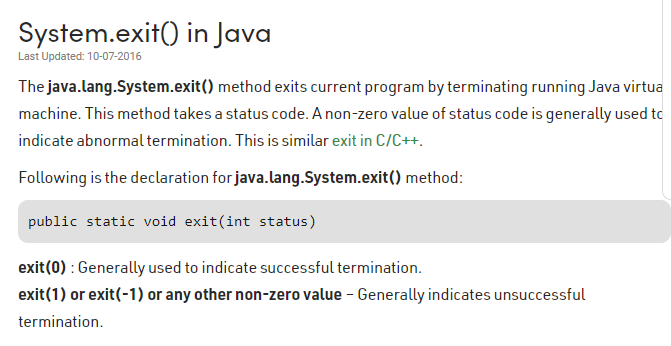
Here, finally blocks dominate the return statement.

**Case 2: try ,catch and finally block has return statement.**

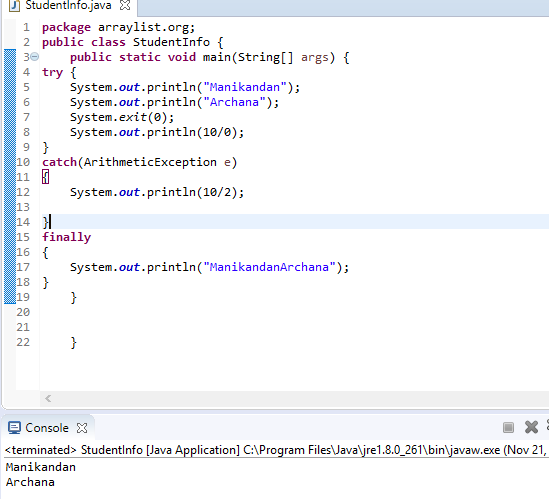




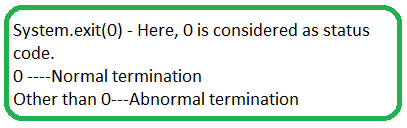
**Finally block vs system.exit(0):**



**System.exit() is used tto terminates the JVM explicity. If we mentioned this mean remaining code won’t be execuated.**

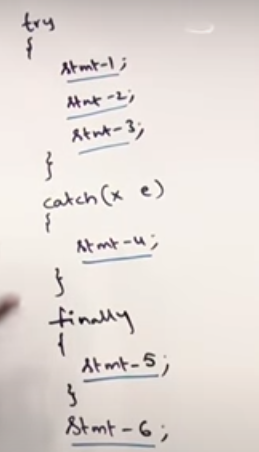


In above program, after giving the system.exit(0) JVM will get shut down and must not execute remaining code including finally block.

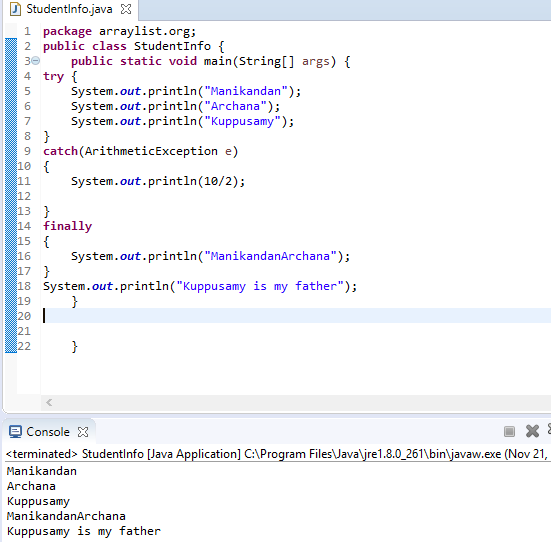
****

**Control flow in try catch finally:**

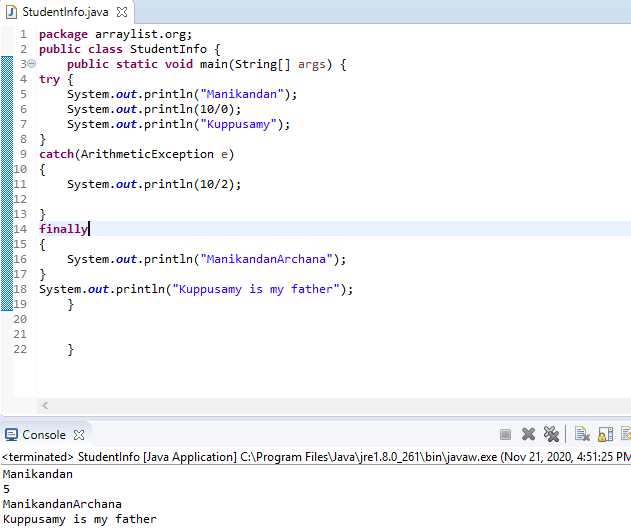
Program:



Case 1: If there is no exception in try block mean catch block won’t be executed and remaining statements will be executed and the program terminates normally.



Case 2: If an exception raised at 2 and corresponding catch block matched mean Statement 1,4,5 and 6 will be executed and statement 2 and 3 won’t be executed and the program gets normal termination only ( exception raised and handled)

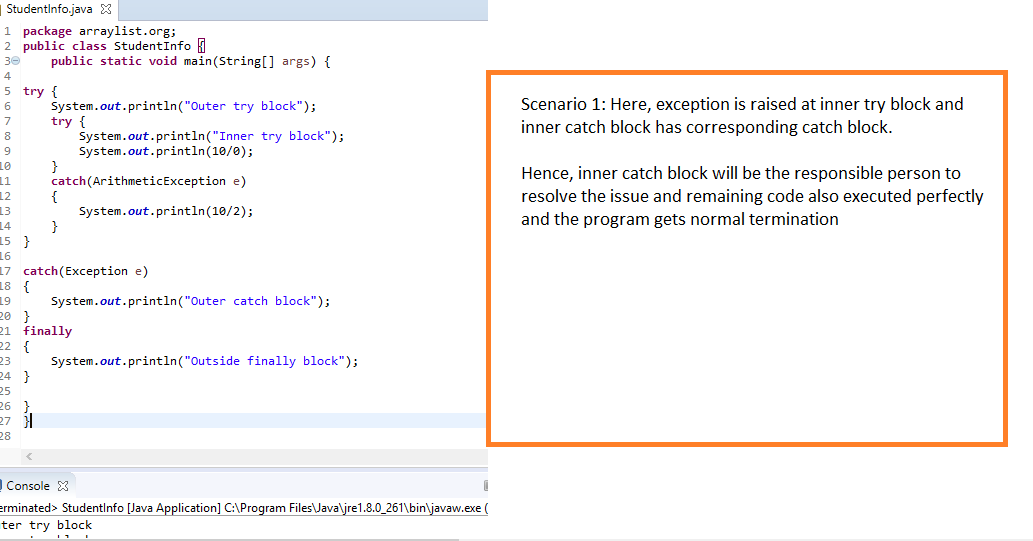


Case 3:If an exception raised at statement 2 but there is no corresponding catch block mean statement 1 (try) and 5 (Finally) will be executed. Statement 6 also won’t be executed and the program gets abnormal termination since exception is raised but not handled.

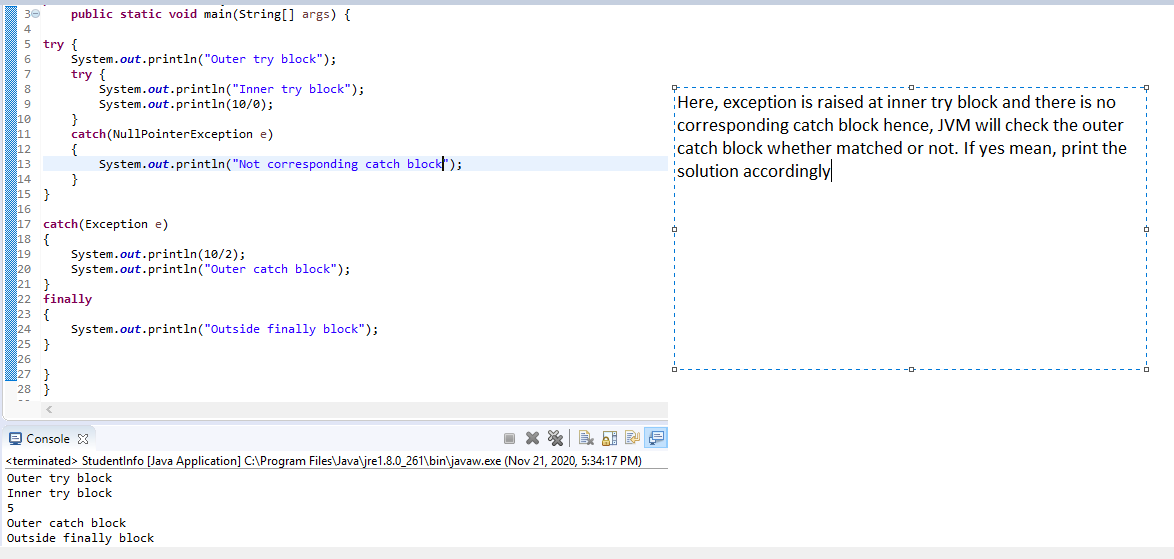
Nested Try catch finally:

Advantages of inner try catch finally block is, if really exception is raised mean inner catch block will be resolve it.

If we didn’t mention inner try mean remining code won’t be executed. To avoid this issue nested try catch finally concept is using by the programmer.



Scenario 2:



Scenario 3:

