**Exception Handling – Part-04 – Customized exception handling by try catch**

* **Customized exception handling by using try/catch:**

It is highly recommended to handle exceptions.

The code which may raise an exception is called risky code and we have to define that code inside ***try*** block and corresponding handling code we have to define inside ***catch*** block.

try{

Risky code;

}

catch(Exception e){

Handling code

}

Example:

Without try-catch:

class Test{

public static void main(String[] args){

System.out.println(“Statement1”);

System.out.println(10/0);

System.out.println(“Statement3”);

}

}

Output:

Statement1

RE: AE: / by zero

Abnormal termination

With try-catch:

class Test{

public static void main(String[] args){

System.out.println(“Statement1”);

try{

System.out.println(10/0);

} catch(Exception e){

System.out.println(10/2);

}

System.out.println(“Statement3”);

}

}

Output:

Statement1

5

Statement2

Normal termination

* **Control flow in try-catch:**

try{

Statement1;

Statement2;

Statement3;

}

catch(Exception e){

Statement4;

}

Statement5;

Case\_01: If there is no exception

1, 2, 3, 5 – Normal termination

Case\_02: If an exception raised at statement2 and corresponding catch block matched.

1, 4, 5 – Normal termination

Case\_03: If an exception raised at statement2 and corresponding catch block not matched.

1, Abnormal termination

Case\_04: If an exception raised at statement4 or statement5 then it is always abnormal termination.

Thumb rule: If an exception is raised outside the try block, then that will be always an abnormal termination.

Note:

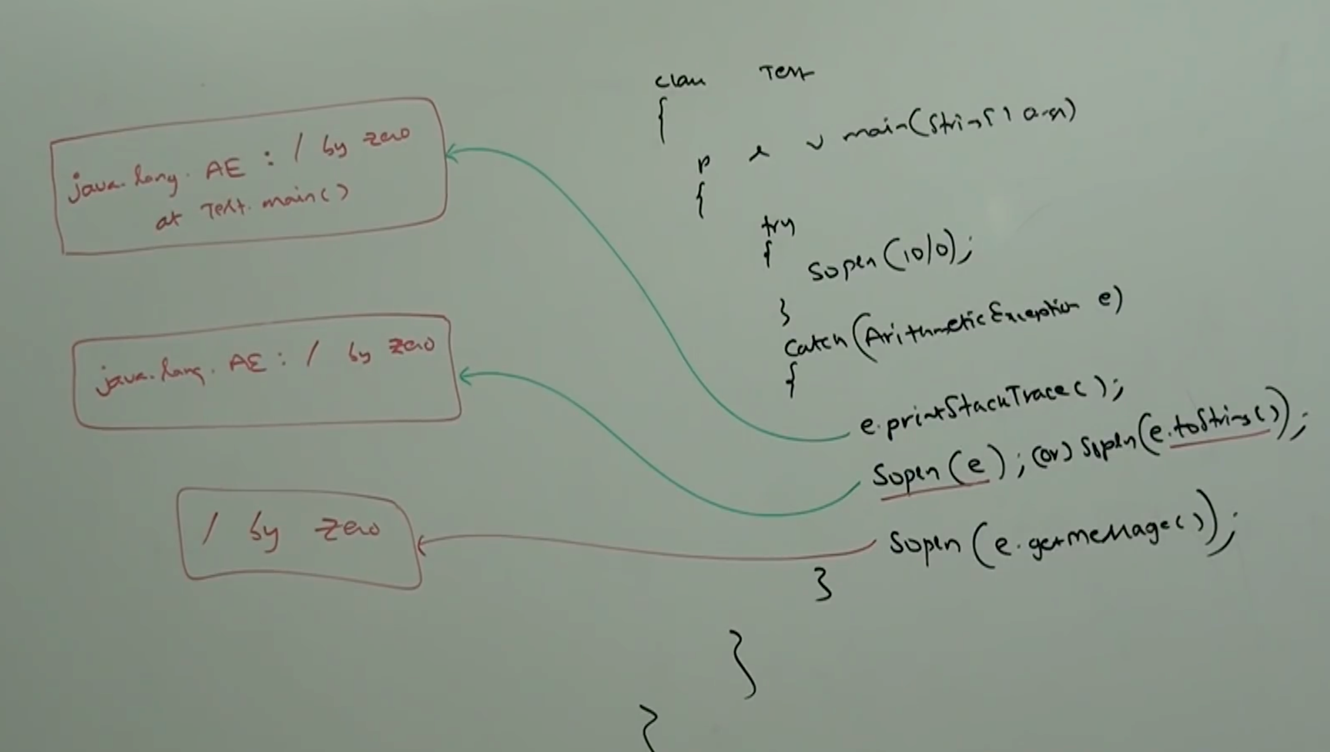
1. Within the try block if anywhere an exception raised then rest of the try block won’t be executed even though we handled that exception. Hence, with in the try block we have to take only risky code and length of try block should be as less as possible.
2. In addition to try block there may a chance of raising an exception inside catch and finally blocks.
3. If any statement which is not part of try block and raises an exception then it is always abnormal termination.

* **Methods to print exception information:**

Throwable class defines the following methods to print exception information.

|  |  |  |
| --- | --- | --- |
| S.No | Method | Printable Format |
| 1 | printStackTrace() | Name of exception: Description  Stack Trace |
| 2 | toString() | Name of exception: Descripton |
| 3 | getMessage() | Description |

Example:



Note:

Internally default exception handler will use

printStackTrace() to print exception information to the

console.

* **try with multiple catch blocks:**

The way of handling an exception is varied from exception to exception, hence for every exception type it is highly recommended to take separate catch block that is try with multiple catch blocks is always possible and recommended to use.

Analogy:

We shouldn’t tell the same answer for all the questions, the answer must be based on the question. Apply the same analogy here, use different catch block for different exception instead of having single catch block for all the exception.

Example:

try{

Risky code;

} catch(Exception e){

}

Worst programming practice.

try{

Risky code;

} catch(ArithmeticException e){

Perform alternative arithmetic operations;

} catch(SQLException e){

Use MySQL DB instead of Oracle DB.

} catch(FileNotFoundException e){

Use local file instead of remote file;

} catch(Exception e){

Default exception handling;

}

Note:

If try with multiple catch blocks present then order of catch blocks is very important. We have to take child first and then parent, otherwise we will get compile time error saying.

Exception xxx has already been caught

Example:

try{

Risky code;

} catch(Exception e){

} catch(ArithmeticException e){

}

CE: exception java.lang.ArithmeticException has already been caught.

try{

Risky code;

} catch(ArithmeticException e){

} catch(Exception e){

}

We can’t declare two catch blocks for the same exception, otherwise we will get compile time error.

try{

Risky code;

} catch(ArithmeticException e){

} catch(ArithmeticException e){

}

CE: exception java.lang.ArithmeticException has already been caught.