# Try Catch in Java – Exception handling

BY CHAITANYA SINGH | FILED UNDER: EXCEPTION HANDLING

In the previous tutorial we discussed what is exception handling and why we do it. In this tutorial we will see try-catch block which is used for exception handling.

## Try block

The try block contains set of statements where an exception can occur. A try block is always followed by a catch block, which handles the exception that occurs in associated try block. A try block must be followed by catch blocks or finally block or both.

### Syntax of try block

```
try{
   //statements that may cause an exception
}
```

While writing a program, if you think that certain statements in a program can throw a exception, enclosed them in try block and handle that exception

## Catch block

A catch block is where you handle the exceptions, this block must follow the try block. A single try block can have several catch blocks associated with it. You can catch different exceptions in different catch blocks. When an exception occurs in try block, the corresponding catch block that handles that particular exception executes. For example if an arithmetic exception occurs in try block then the statements enclosed in catch block for arithmetic exception executes.

### Syntax of try catch in java

```
try
{
     //statements that may cause an exception
}
catch (exception(type) e(object))
{
     //error handling code
}
```

## Example: try catch block

If an exception occurs in try block then the control of execution is passed to the corresponding catch block. A single try block can have multiple catch blocks associated with it, you should place the catch blocks in such a way that the generic exception handler catch block is at the last(see in the example below).
The generic exception handler can handle all the exceptions but you should place is at the end, if you place it at the before all the catch blocks then it will display the generic message. You always want to give the user a meaningful message for each type of exception rather then a generic message.

```
class Example1 {
   public static void main(String args[]) {
      int num1, num2;
      try {
         /* We suspect that this block of statement can throw
          * exception so we handled it by placing these statements
          * inside try and handled the exception in catch block
          */
         num1 = 0;
         num2 = 62 / num1;
         System.out.println(num2);
         System.out.println("Hey I'm at the end of try block");
      }
      catch (ArithmeticException e) {
         /* This block will only execute if any Arithmetic exception
          * occurs in try block
          */
         System.out.println("You should not divide a number by zero");
      }
      catch (Exception e) {
         /* This is a generic Exception handler which means it can handle
          * all the exceptions. This will execute if the exception is not
          * handled by previous catch blocks.
          */
         System.out.println("Exception occurred");
      }
      System.out.println("I'm out of try-catch block in Java.");
   }
}
```

Output:

```
You should not divide a number by zero
I'm out of try-catch block in Java.
```

# Multiple catch blocks in Java

The example we seen above is having multiple catch blocks, lets see few rules about multiple catch blocks with the help of examples. To read this in detail, see catching multiple exceptions in java.
1. As I mentioned above, a single try block can have any number of catch blocks.
2. A generic catch block can handle all the exceptions. Whether it is ArrayIndexOutOfBoundsException or ArithmeticException or

NullPointerException or any other type of exception, this handles all of them. To see the examples of NullPointerException and ArrayIndexOutOfBoundsException, refer this article: Exception Handling example programs.

```
catch(Exception e){
  //This catch block catches all the exceptions
}
```

If you are wondering why we need other catch handlers when we have a generic that can handle all. This is because in generic exception handler you can display a message but you are not sure for which type of exception it may trigger so it will display the same message for all the exceptions and user may not be able to understand which exception occurred. Thats the reason you should place is at the end of all the specific exception catch blocks

3. If no exception occurs in try block then the catch blocks are completely ignored.
4. Corresponding catch blocks execute for that specific type of exception: catch(ArithmeticException e) is a catch block that can hanlde ArithmeticException
catch(NullPointerException e) is a catch block that can handle NullPointerException
5. You can also throw exception, which is an advanced topic and I have covered it in separate tutorials: user defined exception, throws keyword, throw vs throws.

## Example of Multiple catch blocks

```
class Example2{
   public static void main(String args[]){
     try{
         int a[]=new int[7];
         a[4]=30/0;
         System.out.println("First print statement in try block");
     }
     catch(ArithmeticException e){
        System.out.println("Warning: ArithmeticException");
     }
     catch(ArrayIndexOutOfBoundsException e){
        System.out.println("Warning: ArrayIndexOutOfBoundsException");
     }
     catch(Exception e){
        System.out.println("Warning: Some Other exception");
     }
    System.out.println("Out of try-catch block...");
   }
}
```

Output:

```
Warning: ArithmeticException
```

```
Out of try-catch block...
```

In the above example there are multiple catch blocks and these catch blocks executes sequentially when an exception occurs in try block. Which means if you put the last catch block ( catch(Exception e)) at the first place, just after try block then in case of any exception this block will execute as it can handle all exceptions. This catch block should be placed at the last to avoid such situations.

## Finally block

I have covered this in a separate tutorial here: java finally block. For now you just need to know that this block executes whether an exception occurs or not. You should place those statements in finally blocks, that must execute whether exception occurs or not.