Java Try Catch Block

An exception (or exceptional event) is a problem that arises during the execution of a program. When an **Exception** occurs the normal flow of the program is disrupted and the program/Application terminates abnormally, which is not recommended, therefore, these exceptions are to be handled.

Java `try` and `catch`

A method catches an exception using a combination of the **`try`** and **`catch`** keywords. A **`try`** and **`catch`** block is placed around the code that might generate an exception. Code within a **`try`** and **`catch`** block is referred to as protected code, and the syntax for using try/catch looks like the following –

The `try` Block

The code which is prone to exceptions is placed in the **`try`** block. When an exception occurs, that exception occurred is handled by **`catch`** block associated with it. Every try block should be immediately followed either by a **`catch`** block or **`finally`** block.

The `catch` Block

A **`catch`** statement involves declaring the type of exception you are trying to catch. If an exception occurs in protected code, the **`catch`** block (or blocks) that follows the try is checked. If the type of exception that occurred is listed in a **`catch`** block, the exception is passed to the **`catch`** block much as an argument is passed into a method parameter.

Syntax of Java try and catch Block

```
try {
   // Protected code
} catch (ExceptionName e1) {
   // Catch block
}
```

Example of Java try and catch Block

In following example, an array is declared with 2 elements. Then the code tries to access the 3rd element of the array which throws an exception. As we've enclosed the code with a try block, this exception can be handled within next catch block which we've declared

to catch ArrayIndexOutOfBoundsException. After catching the exception, we can take the corresponding action. Here we're printing the details of the exception thrown.

</> Open Compiler

```
package com.tutorialspoint;

public class ExcepTest {

   public static void main(String args[]) {
      try {
         int a[] = new int[2];
         System.out.println("Access element three :" + a[3]);
      } catch (ArrayIndexOutOfBoundsException e) {
         System.out.println("Exception thrown  :" + e);
      }
      System.out.println("Out of the block");
   }
}
```

Output

```
Exception thrown  :java.lang.ArrayIndexOutOfBoundsException: 3
Out of the block
```

Multiple Catch Blocks With Java Try

A try block can be followed by multiple catch blocks. The syntax for multiple catch blocks looks like the following –

Syntax: Multiple Catch Blocks

```
try {
   // Protected code
} catch (ExceptionType1 e1) {
   // Catch block
} catch (ExceptionType2 e2) {
   // Catch block
} catch (ExceptionType3 e3) {
```

```
   // Catch block
}
```

The previous statements demonstrate three catch blocks, but you can have any number of them after a single try. If an exception occurs in the protected code, the exception is thrown to the first catch block in the list. If the data type of the exception thrown matches ExceptionType1, it gets caught there. If not, the exception passes down to the second catch statement. This continues until the exception either is caught or falls through all catches, in which case the current method stops execution and the exception is thrown down to the previous method on the call stack.

Example: Multiple Catch Blocks

Here is code segment showing how to use multiple try/catch statements. In this example, we're creating an error by dividing a value by 0. As it is not an ArrayIndexOutOfBoundsException, next catch block handles the exception and prints the details.

</> Open Compiler

```
package com.tutorialspoint;

public class ExcepTest {

   public static void main(String args[]) {
      try {
         int a[] = new int[2];
         int b = 0;
         int c = 1/b;
         System.out.println("Access element three :" + a[3]);
      }
      catch (ArrayIndexOutOfBoundsException e) {
         System.out.println("ArrayIndexOutOfBoundsException thrown  :" + e);
      }catch (Exception e) {
          System.out.println("Exception thrown  :" + e);
      }
      System.out.println("Out of the block");
   }
}
```

Output

```
Exception thrown  :java.lang.ArithmeticException: / by zero
Out of the block
```

Learn **Java** in-depth with real-world projects through our **Java certification course**. Enroll and become a certified expert to boost your career.

Catching Multiple Exceptions with Java Try and Catch Block

Since Java 7, you can handle more than one exception using a single catch block, this feature simplifies the code. Here is how you would do it –

Syntax: Catching Multiple Exceptions

```
catch (IOException|FileNotFoundException ex) {
   logger.log(ex);
   throw ex;
```

Example: Catching Multiple Exceptions

Here is code segment showing how to use multiple catch in a single statement. In this example, we're creating an error by dividing a value by 0. In single statement, we're handling ArrayIndexOutOfBoundsException and ArithmeticException.

</> Open Compiler

```
package com.tutorialspoint;

public class ExcepTest {

   public static void main(String args[]) {
      try {
         int a[] = new int[2];
         int b = 0;
         int c = 1/b;
         System.out.println("Access element three :" + a[3]);
      }
      catch (ArrayIndexOutOfBoundsException | ArithmeticException e) {
         System.out.println("Exception thrown  :" + e);
      }
      System.out.println("Out of the block");
```

```
    }
}
```

Output

```
Exception thrown  :java.lang.ArithmeticException: / by zero
Out of the block
```