Java - Nested Try Block

Nested Try Block

A try block can be nested within another try block. This structure is termed as Nested try block. Whenever an exception is raised within a nested try block, its exception is pushed to Stack. The exception propagates from child to parent try block and so on.

Syntax

The syntax for nested catch blocks looks like the following –

```
try { // parent try block
   try {  // child try block

   }
   catch(ExceptionType1 e1){  // child catch block

   }
} catch (ExceptionType2 e1) { // parent catch block

}
```

The previous statements demonstrate two try/catch blocks, but you can have any number of them. If an exception occurs in the protected child code, the exception is thrown to the catch block of the child list. If the data type of the exception thrown matches ExceptionType1, it gets caught there. If not, the exception passes up to the parent 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.

Pointer To Remember While Using Nested Try Block

- Child catch block should have specific exception for better code clarity. Parent catch block can have more generic exception handled so that if child catch block is not able to handle the exception then parent catch block can handle it.
- There in no restriction on exception hiearchy to be used in child vs parent catch block.
- If a exception is handled correctly in child catch block, then in parent, another exception can be raised and handled.

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

Java Nested Try Block Example

Here is code segment showing how to use nested try/catch statements. In this example, we're creating an error by dividing a value by 0 in a nested try block. The child catch block is handling the exception and printing the same. Now in parent try block, we're again creating an error by using an invalid array index while accessing array elements and exception is raised.

Open Compiler

```java
package com.tutorialspoint;

public class ExcepTest {

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

More Examples

Example 1

In this code segment, we're showing how to use another example of nested try/catch statements. In this example, we're creating an error by dividing a value by 0 in nested try block but we're not handling in corresponding catch block. As parent try block is handling the exception raised as generic exception, it captures the exception raised by child catch block and prints the same.

Open Compiler

```
package com.tutorialspoint;

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

Example 2

In this code segment, we're showing the case of nested try/catch statements where exceptions are not handled by any of the catch block. In this example, we're creating an error by dividing a value by 0 in nested try block but we're not handling this kind of exception in any catch block. Now JVM will capture the exception and terminate the program without printing the last statement.

</> Open Compiler

```
package com.tutorialspoint;

public class ExcepTest {
   public static void main(String args[]) {
      try {
         int a[] = new int[2];
         try {
            int b = 0;
            int c = 1/b;
         }catch(ArrayIndexOutOfBoundsException e) {
            System.out.println("Exception thrown: " + e);
         }
         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 in thread "main" java.lang.ArithmeticException: / by zero
	at com.tutorialspoint.ExcepTest.main(ExcepTest.java:10)
```