Java Multiple Catch Blocks

Multiple Catch Blocks in Java

Multiple catch blocks in Java are used to catch/handle multiple exceptions that may be thrown from a particular code section. A try block can have multiple catch blocks to handle multiple exceptions.

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.

Points to Remember while using Multiple Catch Blocks

- Only one type of exception can be handled at a time. In protected, only one type of exception will be raised, so it will be handled in relevant catch block only.
- Order of catch block is very important. Order should be from specific exception to generic one. In case of parent exception block comes before the child exception block, compiler will complain and will throw compile time error.

Example of Java 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
```

Handling Multiple Exceptions Using Multiple Catch Blocks

In this code segment, we're showing how to use another example of multiple try/catch statements. In this example, we're creating an error by dividing a value by 0 and handling it using ArithmeticException.

Example

</> 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 (ArithmeticException e) {
         System.out.println("ArithmeticException thrown  :" + e);
      }
      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

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

Handling Multiple Exceptions Within A Single 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

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

```
    throw ex;
```

Example

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
```