Topics

- Catching Exceptions
- try { .. } catch () { .. }

Exception Handling

innovate achieve lead

- Exception Handling Requires Four Steps

1. Finding the Problem→ Identify the Statements Which May Result in Exception. Put all those statements in a *try {..}* block.
2. Inform that an Exception is thrown (Throw the Exception) *<< Note Down the difference between **throw** vs **throws** clauses>>*
3. Catch the Exception Using catch(..) statements
4. Provide the Exception Handling Code in catch(..) { .. } blocks

Exception Handling Syntax


```
try
{
    <statements that can throw exceptions>
}
catch(Exception-Type-1 e1)          {….}
catch(Exception-Type-2 e2)          {….}
catch(Exception-Type-3 e3)          {….}
……..
catch(Exception-Type-N e4)          {….}
```

Important Points :

1. try { .. } block may have one or multiple statements.
2. try { .. } block may be capable of either throwing either a single type or multiple types of exceptions.
3. There can be multiple catch() { .. } blocks associated with single try { .. } block.
4. If try{} block can throw multiple exceptions then user should catch all exceptions. (one catch block for each type of exception)

Catching an Exception : Example 1

innovate achieve lead

```
class ExceptionDemoTest
{
        public static void main(String args[])
        {
                System.out.println(" Hello Exceptions");
                int d = 0;
                int x = 10;
                try
                {
                        x = 42 / d;
                }
                catch(ArithmeticException e) { System.out.println(e); }
                System.out.println("x=" +x);
                System.out.println(" Exception Demo Ends");
        }// End of Method
}// End of class
```

OUTPUT

```
Hello Exceptions
java.lang.ArithmeticException: / by zero
x=10
 Exception Demo Ends
```

Catching Multiple Exceptions Example


```
class ExceptionDemoTest
{
  public static void main(String args[])
  {
    int a[]= {5,10};
          try
          {
                int b= Integer.parseInt(args[0]);
                int x = a[b]/(b-a[1]);
                System.out.println("x="+x);
          }
          catch(ArithmeticException e)          { System.out.println(e); }
          catch(NumberFormatException e)  { System.out.println(e); }
          catch(ArrayIndexOutOfBoundsException e) { System.out.println(e);}
     System.out.println("Exception Demo Ends");
   }// End of Method
}// End of class
```

Catching Multiple Exceptions Example …

java ExceptionDemoTest

```
java.lang.ArrayIndexOutOfBoundsException: 0
Exception Demo Ends
```

java ExceptionDemoTest 0

```
x=0
Exception Demo Ends
```

java ExceptionDemoTest 1

```
x=-1
Exception Demo Ends
```

java ExceptionDemoTest a

```
java.lang.NumberFormatException: For input string: "a"
Exception Demo Ends
```

Nested Try Statements

innovate achieve lead

- Try{ } statements can be nested. One try block may contain another try block
- In case of nested try blocks, context of that exception is pushed onto stack.
- Inner try block may/or may not have catch statements associated with it.
- If an exception is thrown from inner try block then first inner catch statements are matched (if present) . If no match is found then outer try block are matched. If there also no match found then default handler will be invoked.
- However, if outer try block throws the exception then only outer try blocks are matched.

Nested try statements : A Typical Syntax

innovate achieve lead

```
try
{
Statement- A;
Statement-B;
      try
      {
       Statement-C;
       Statement-D;
      }
     catch(CException e) {  ....  }
     catch(DException e) {  ....  }
}
catch(AException e) {  ....  }
catch(BException e) {  ....  }
```

```
try
{
Statement-A;
Statement-B;
      try
      {
       Statement-C;
       Statement-D;
      }
}
catch(AException e) {  ....  }
catch(BException e) {  ....  }
catch(CException e) {  ....  }
catch(DException e) {  ....  }
```

Nested try statements : A Typical Syntax …

```
try
{
Statement-A;
Statement-B;
      try
      {
       Statement-C;
       Statement-D;
      }
     catch(CException e) {  ….  }
     catch(DException e) {  ….  }
}
catch(AException e) {  ….  }
catch(BException e) {  ….  }
catch(CException e) {  ….  }
catch(DException e) {  ….  }
```

Nested try { .. } : Example

```
class nestedtry
{
public static void main(String args[])
{
        int a[] = { 2,5,6};             // { a[0] = 2, a[1] = 5, a[2] = 6}
        try         // outer try
        {
                int b = Integer.parseInt(args[0]);
                 try        // inner try
                {
                        int c[] = { 4,5,6};   // { c[0] = 4, c[1] = 5, c[2] = 6}
                        int d  = c[b]/(c[b]-4);
                } // End of inner try
                // catch Blocks Associated With Inner Try Block
                catch(ArrayIndexOutOfBoundsException e)
                {
                        System.out.println("Exception : "+ e.toString());
                        System.out.println("By Inner try");
                }
```

Nested try { .. } : Example …

```
    catch(ArithmeticException e)
    {
    System.out.println("Exception : "+ e.toString());
    System.out.println("By Inner try");
    }
} // End of outer try
// Catch Blocks Asscoiated With Outer try Block
catch(ArrayIndexOutOfBoundsException e)
     {
     System.out.println("Exception : "+ e.toString());
     System.out.println("By Outr try");
     }
catch(NumberFormatException e)
     {
     System.out.println("Exception : "+ e.toString());
     System.out.println("By Outer try");
     }
} // End of main
} // End of class
```

Nested try { .. } : Example …

java nestedtry

```
Exception : java.lang.ArrayIndexOutOfBoundsException: 0
By Outer try
```

java nestedtry 4

```
Exception : java.lang.ArrayIndexOutOfBoundsException: 4
By Inner try
```

java nestedtry 0

```
Exception : java.lang.ArithmeticException: / by zero
By Inner try
```

Thank You