# [Exception Handling In Java](https://javaconceptoftheday.com/exception-handling-java/)

An exception is a problem that arises during the execution of a program. An exception can occur for many different reasons. Some of these exceptions are caused by user error, others by programmer error, and others by physical resources that have failed in some manner.

Exception Handling in java is implemented using five keywords.

1) try       2)catch       3)finally       4)throw         5)throws

****try block :****In try block, keep those statements which may throw exceptions during run time.

****catch block :**** This block handles the exceptions thrown by try block. It takes one argument of type java.lang.Exception.

****finally block :**** Whether exception is thrown or not and thrown exception is caught or not, this block will be always executed.

**public** **class** ExceptionHandling

{

**public** **static** **void** main(String[] args)

    {

        String[] s = {"abc", "123", "xyz", "456"};   //String Array containing valid and invalid numeric values

**for** (**int** i = 0; i < s.length; i++)

        {

**try**

            {

**int** intValue = Integer.parseInt(s[i]); //This statement may throw NumberFormatException

            }

**catch**(NumberFormatException ex)

            {

                System.out.println("The thrown NumberFormatException will be caught here");

            }

**finally**

            {

                System.out.println("This block is always executed");

            }

        }

    }

}

* When a statement throws an exception in the try block, the remaining part of the try block will not be executed. Program control comes out of the try block and enters directly into catch block.
* try, catch and finally blocks form one unit. i.e You can’t keep other statements in between try, catch and finally blocks.
* You can display the description of an exception thrown using Exception object in the catch block.

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19  20 | **public** **class** ExceptionHandling  {  **public** **static** **void** main(String[] args)      {  **try**          {  **int** i = 10/0;           //This statement throws ArithmeticException                System.out.println("This statement will not be executed");          }  //You can't keep statements here  **catch**(Exception ex)          {              System.out.println("This block is executed immediately after an exception is thrown");  ex.printStackTrace();     //This prints stack trace of exception          }  //You can't keep statements here  **finally**          {              System.out.println("This block is always executed");          }      }  } |
|  |  |

**Multiple Catch Blocks In Java:**

**public** **class** ExceptionHandling

{

**public** **static** **void** main(String[] args)

    {

        String[] s = {"abc", "123", **null**, "xyz"};   //String array containing one null object

**for** (**int** i = 0; i < 6; i++)

        {

**try**

            {

**int** a = s[i].length() + Integer.parseInt(s[i]);

                //This statement may throw NumberFormatException, NullPointerException and ArrayIndexOutOfBoundsException

            }

**catch**(NumberFormatException ex)

            {

                System.out.println("NumberFormatException will be caught here");

            }

**catch** (ArrayIndexOutOfBoundsException ex)

            {

                System.out.println("ArrayIndexOutOfBoundsException will be caught here");

            }

**catch** (NullPointerException ex)

            {

                System.out.println("NullPointerException will be caught here");

            }

            System.out.println("After executing respective catch block, this statement will be executed");

        }

    }

}

* From Java 7 onward, there is one more way for handling multiple exceptions. Multiple exceptions thrown by the try block can be handled by a single catch block using ****pipe (|) operator****. By using pipe operator, the above example can be written as,

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19  20  21  22 | **public** **class** ExceptionHandling  {  **public** **static** **void** main(String[] args)      {          String[] s = {"abc", "123", **null**, "xyz"};   //String array containing one null object    **for** (**int** i = 0; i < 6; i++)          {  **try**              {  **int** a = s[i].length() + Integer.parseInt(s[i]);                    //This statement may throw NumberFormatException, NullPointerException and ArrayIndexOutOfBoundsException              }    **catch**(NumberFormatException | NullPointerException | ArrayIndexOutOfBoundsException ex)              {                  System.out.println("Now, this block handles NumberFormatException, NullPointerException and ArrayIndexOutOfBoundsException");              }          }      }  } |

* java.lang.Exception is super class of all types of exception. (Types of exceptions will be discussed later). It handles all types of exceptions. In the above example, all catch blocks can be replaced by one catch block which handles all types of exceptions. This type of exception handling comes very handy when you are not sure about the types of exceptions your code may throw.

**try**

            {

**int** a = s[i].length() + Integer.parseInt(s[i]);

                //This statement may throw NumberFormatException, NullPointerException and ArrayIndexOutOfBoundsException

            }

**catch**(Exception ex)

            {

                System.out.println("This block handles all types of exceptions");

            }

* The order of catch blocks should be from most specific to most general ones. i.e Sub classes of Exception must come first and super classes later. If you keep the super classes first and sub classes later, you will get compile time error : ****Unreachable Catch Block****.

# [Nested try catch Blocks In Java](https://javaconceptoftheday.com/nested-try-catch-blocks-java/)

try-catch blocks can be nested. i.e one try block can contain another try-catch block. The syntax for nesting try blocks is,

**try** {    //Outer try block

    //Some Statement

**try** {   //Inner try block

        //Some Statements

    }**catch** (Exception ex)    //Inner catch block

    {

    }

}

**catch**(Exception ex)     //Outer catch block

{

}

* If the exception thrown by the inner try block can not be caught by it’s catch block, then this exception is propagated to outer try blocks. Any one of the outer catch block should handle this exception otherwise program will terminate abruptly.

public class ExceptionHandling

{

public static void main(String[] args)

{

String[] s = {"abc", "123", null, "xyz"}; //String array containing one null object

for (int i = 0; i < s.length; i++)

{

//First Level try-catch block

try

{

int a = s[i].length(); //This statement may throw NullPointerException

//second level try-catch block

try

{

System.out.println(s[i+1]); //This statement may throw ArrayIndexOutOfBoundsException

//third level try-catch block

try

{

a = Integer.parseInt(s[i]); //This statement may throw NumberFormatException

}

catch (NullPointerException e)

{

System.out.println("NumberFormatException will not be caught here");

}

}

catch (NumberFormatException ex)

{

System.out.println("NumberFormatException will be caught here");

}

}

catch(Exception ex)

{

System.out.println("This block catches all types of exceptions");

}

}

}

}

**public** **class** ExceptionPropogation {

/\*

\* In the following example also, try-catch blocks are nested. main() method

\* calls nestedTry() method. nestedTry() method has one try-catch block. First

\* statement in try block throws NumberFormatException which is not handled by

\* it’s catch block. So, It propagates to try-catch block of main method which

\* handles this exception.

\*/

**public** **static** **void** main(String[] args) {

**try** {

*nestedTry*();

} **catch** (Exception ex) {

System.***out***.println("NumberFormatException will be caught here in main method");

}

}

**static** **void** nestedTry() {

**try** {

**int** i = Integer.*parseInt*("abc"); // This statement throws NumberFormatException

} **catch** (NullPointerException ex) {

System.***out***.println("NumberFormatException will not be caught here");

}

}

}