**Exception handling in java**

## What is an exception?

An Exception can be anything which interrupts the normal flow of the program. When an exception occurs program processing gets terminated and doesn’t continue further. In such cases we get a system generated error message. The good thing about exceptions is that they can be handled.

**Reasons for Exceptions**There can be several reasons for an exception. For example, following situations can cause an exception – Opening a non-existing file, Network connection problem, Operands being manipulated are out of prescribed ranges, class file missing which was supposed to be loaded and so on.

## Difference between error and exception

**Errors** indicate serious problems and abnormal conditions that most applications should not try to handle. Error defines problems that are not expected to be caught under normal circumstances by our program. For example memory error, hardware error, JVM error etc.  
**Exceptions** are conditions within the code. A developer can handle such conditions and take necessary corrective actions.

* DivideByZero exception
* NullPointerException
* ArithmeticException
* ArrayIndexOutOfBoundsException

**Advantages of Exception Handling**

* Exception handling allows us to control the normal flow of the program by using exception handling in program.
* It throws an exception whenever a calling method encounters an error providing that the calling method takes care of that error.
* It also gives us the scope of organizing and differentiating between different error types using a separate block of codes.

## Types of exceptions

* There are two types of exceptions
* 1)Checked exceptions  
  2)Unchecked exceptions

**Checked exceptions**  
All exceptions other than Runtime Exceptions are known as Checked exceptions as the compiler checks them during compilation to see whether the programmer has handled them or not. If these exceptions are not handled/declared in the program, it will give compilation error.

**Examples of Checked Exceptions**   
ClassNotFoundException  
IllegalAccessException  
NoSuchFieldException  
EOFException etc.

**Unchecked Exceptions**  
Runtime Exceptions are also known as Unchecked Exceptions as the compiler do not check whether the programmer has handled them or not but it’s the duty of the programmer to handle these exceptions and provide a safe exit.  
These exceptions need not be included in any method’s throws list because compiler does not check to see if a method handles or throws these exceptions.

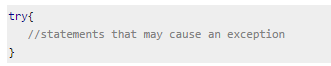
**Examples of Unchecked Exceptions:-**  
ArithmeticException  
ArrayIndexOutOfBoundsException  
NullPointerException  
NegativeArraySizeException etc.

# Try Catch in Java – Exception handling

### What is Try Block?

The try block contains a block of program statements within which an exception might occur. A try block is always followed by a catch block, which handles the exception that occurs in associated try block. A try block must followed by a Catch block or Finally block or both.

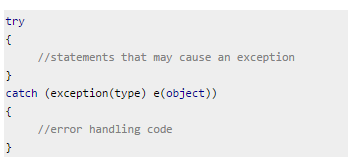
#### Syntax of try block



### What is Catch Block?

A catch block must be associated with a try block. The corresponding catch block executes if an exception of a particular type occurs within the try block. For example if an **arithmetic exception** occurs in try block then the statements enclosed in catch block for arithmetic exception executes

#### Syntax of try catch in java



1. If an exception occurs in try block then the control of execution is passed to the catch block from try block. The exception is caught up by the corresponding catch block. A single try block can have multiple catch statements associated with it, but each catch block can be defined for only one exception class. The program can also contain [**nested**](http://beginnersbook.com/2013/04/nested-try-catch/) **try-catch-finally blocks**.
2. After the execution of all the try blocks, the code inside the finally block executes. It is not mandatory to include a finally [**block**](http://beginnersbook.com/2013/04/java-finally-block/) at all, but if you do, it will run regardless of whether an exception was thrown and handled by the try and catch blocks

#### Multiple catch blocks in Java

