[**Next →**](https://www.javatpoint.com/try-catch-block)[**← Prev**](https://www.javatpoint.com/java-regex)

*Exception Handling in Java*

An exception is an event that disrupts the normal flow of the program.

The **Exception Handling in Java** is one of the powerful *mechanism to handle the runtime errors* so that normal flow of the application can be maintained.

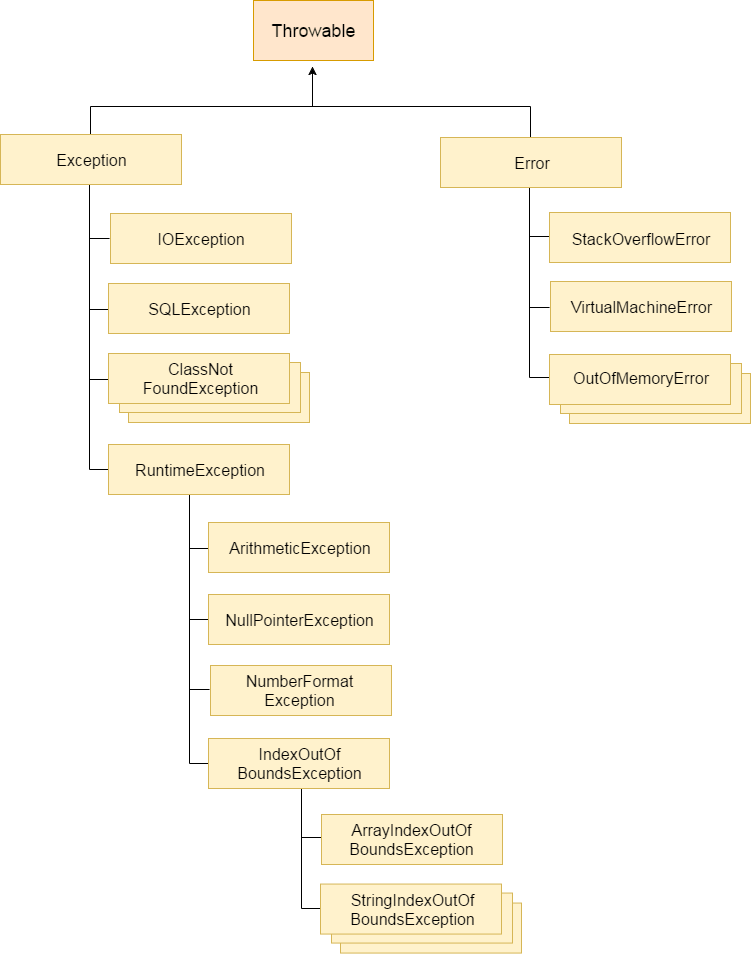
In other words, unwanted and unexpected behavior/event that interrupts the normal execution flow of the program is called exception in java

All the exceptions occur only at runtime.

A syntax error occurs at compile time.

### Advantage of Exception Handling

The core advantage of exception handling is **to maintain the normal flow of the application**. An exception normally disrupts the normal flow of the application that is why we use exception handling



## Java Exception Keywords

There are 5 keywords which are used in handling exceptions in Java.

## *try*

***catch***

***finally***

***throw***

***throws***

# Checked and Unchecked Exceptions in Java

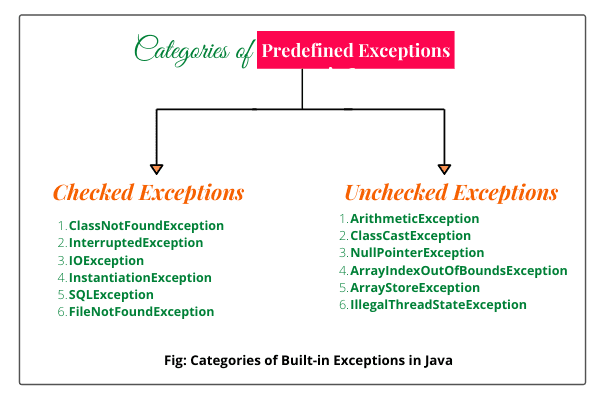
**Checked and Unchecked Exceptions in Java |**

The predefined exceptions are those exceptions that are already defined by the java system.

All the predefined exceptions are further divided into two groups:

1. **Checked Exceptions**

2. **Unchecked Exceptions**

****

## Checked Exceptions in Java

Checked exceptions are those exceptions that are checked by the java compiler itself at compilation time

Checked exceptions must be handled either by using try and catch block or by using throws clause in the method declaration.

If not handles properly, it will give a compile-time error.

Most people have confusion and say that checked exceptions occur at compile-time, that is wrong.

All exceptions always occur at runtime only but some exceptions are detected at compile-time and some other at runtime.

The exceptions that are checked by Java compiler at compilation time is called checked exception in Java.

All the exceptions except RuntimeException, Error, and their subclasses are checked exceptions.

**Note:** Compile-time errors are not exceptions. They come under errors. In Java, only runtime errors come under exceptions.

## List of Checked Exceptions in Java

A list of some important checked exceptions are given below:

* ClassNotFoundException
* InterruptedException
* InstantiationException
* IOException
* SQLException
* IllegalAccessException
* FileNotFoundException, etc

## Unchecked Exceptions (Runtime Exceptions) in Java

Unchecked exceptions in Java are those exceptions that are checked by JVM, not by java compiler.

They occur during the runtime of a program.

All exceptions under runtime exception class are called unchecked exceptions or runtime exceptions in Java.

We can write a Java program and compile it. But we cannot see the effect of unchecked exceptions and errors until we run the program. This is because Java compiler allows us to write a Java program without handling unchecked exceptions and errors.

Java compiler does not check runtime exception at compile time whether programmer handles them or not.

## 

## 

## List of Unchecked Exceptions in Java

Some important examples of runtime exceptions are given below:

* ArithmeticException
* ClassCastException
* NullPointerException
* ArrayIndexOutOfBoundsException
* NegativeArraySizeException
* ArrayStoreException
* IllegalThreadStateException
* SecurityException, etc.

Example 1:public class Unchecked {

public static void main(String[] args) {

try {

int a =10;

int b =0;

int c = a/b;

System.out.println(c);

}

catch (ArithmeticException e){

System.out.println(" ok let it go ");

}

System.out.println(" This is my rest of program ...... continues ");

}

# Multiple Catch Block in Java with Example

When statements in a single try block generate multiple exceptions, we require multiple catch blocks to handle different types of exceptions. This mechanism is called **multi-catch block in java**.

Each catch block is capable of catching a different exception.

That is each catch block must contain a different exception handler.

The syntax for using a single try with more than one catch block in java is as follows:

**Syntax:**

try

{

statements;

}

catch(ExceptionType1 e1)

{

statements;

}

catch(ExceptionType2 e2)

{

statements;

}

catch(ExceptionType3 e3)

{

statements;

}

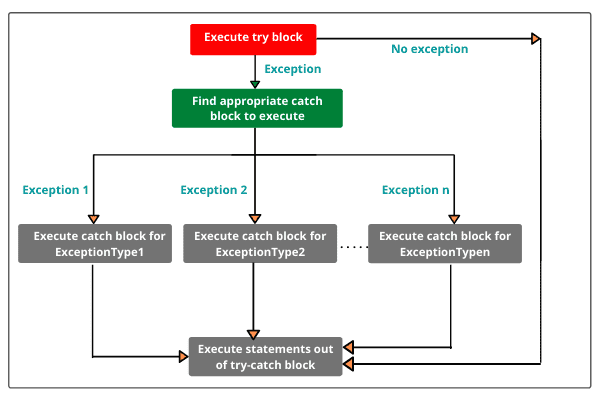
. . .

catch(ExceptionTypen en)

{

statements;

}



try

{

statements;

}

catch(NullPointerException np)

{

statements;

}

catch(RuntimeException re)

{

statements;

}

catch(IOException ieo)

{

statements;

}

catch(Exception e)

{

statements;

}

# Finally Block in Java |

**Finally Block in Java | A “finally” is a keyword used to create a block of code that follows a try or catch block.**

**A finally block contains all the crucial codes such as closing connections, stream, etc**

**‘finally’ - is always executed whether an exception occurs within a try block or not.**

**When finally block is attached with a** [***try-catch block***](https://www.scientecheasy.com/2020/09/java-try-catch-block.html/)**, it is always executed whether the catch block has handled the exception thrown by try block or not.**

**The syntax for try-finally and try-catch-finally is as follows:**

**Syntax for try-finally block:**

**try**

**{**

**statement1;**

**statement2;**

**}**

**finally // finally block**

**{**

**statement3;**

**}**

**Syntax for try-catch-finally block:**

**try**

**{**

**statement1;**

**statement2;**

**}**

**catch(Exceptiontype e1)**

**{**

**statement3;**

**}**

**statement4;**

**finally**

**{**

**statement5;**

**}**

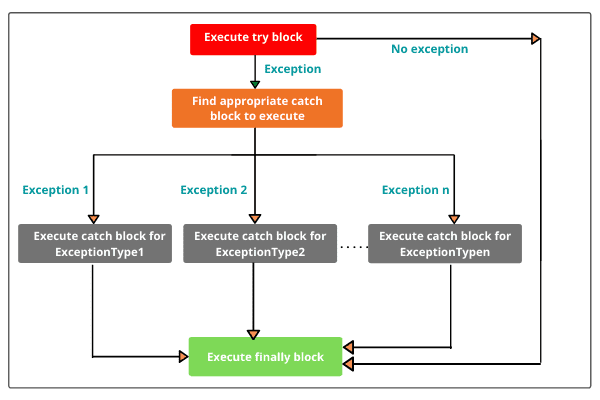
**Some important rules of using finally block or clause are:**

**1. A finally block is optional but at least one of the catch or finally block must exist with a try.**

**2. It must be defined at the end of last catch block. If finally block is defined before a catch block, the program will not compile successfully.**

**3. Unlike catch, multiple finally blocks cannot be declared with a single try block.**

Control flow of try-catch-finally block in Java

****

## Use of finally block in Java

## Java try-catch-finally block Example:

# 

# 

# Throw Keyword in Java | Java Throw Exception

# A**ll the programs of** [**exception handling**](https://www.scientecheasy.com/2020/02/exception-handling-in-java-realtime-example.html/) **that Java runtime system (JVM) was responsible for identifying exception class,**

# **creating its object,**

# **and throwing that object.**

**If we want to throw an exception manually or explicitly, for this, Java provides a keyword throw.**

## Throw Keyword in Java

## Throws Keyword in Java

**Throws keyword in Java is used in the method declaration.**

**It provides information to the caller method about exceptions being thrown .**

**Throws keyword is used in case of checked exception only because if we are not handling runtime exceptions (unchecked exceptions).**

**The general syntax of using throws statement with a method declaration is as follows:**

**Syntax:**

**access\_specifier return\_type method\_name(parameter list) throws exception**

**{**

**// body of the method.**

**}**

**syntax:**

**public void method1( int a, int b) throws expetion2, expection1, e3,e4 {**

**}**

**Java throws keyword can be used to throw multiple exceptions thrown by a method at a time. Multiple exceptions thrown by a method can be declared by separating them in comma with the help of throws keyword.**

**The general syntax is as follows:**

**Syntax:**

**access\_specifier return\_type method\_name(parameter\_list) throws exception1, exception2, . . . . exceptionN**

**{**

**// body of the method.**

**}**