# Errors and Exceptions

## Exceptions

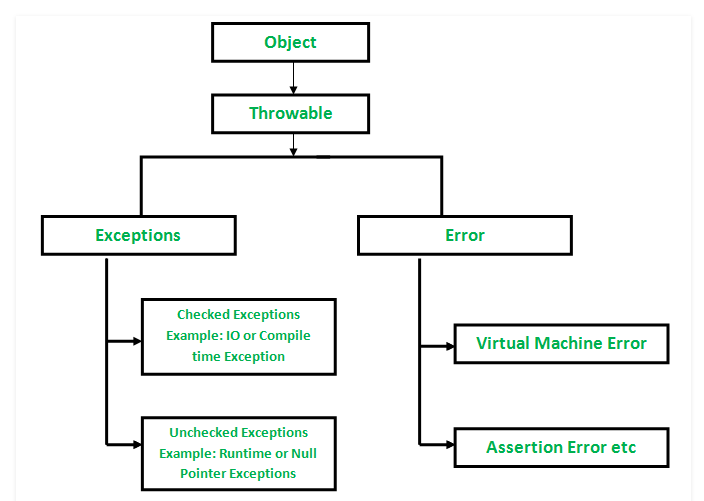
An unwanted event that occurs during the execution of a program i.e. at run time that disrupts the normal flow of the program’s instructions. It can be handled using code. An Exception “indicates conditions that a reasonable application might want to catch.” Exceptions are the conditions that occur at runtime and may cause the termination of program. But they are recoverable using try, catch and throw keywords. Exceptions are divided into two categories: [checked and unchecked exceptions](https://www.geeksforgeeks.org/checked-vs-unchecked-exceptions-in-java/).

## Error

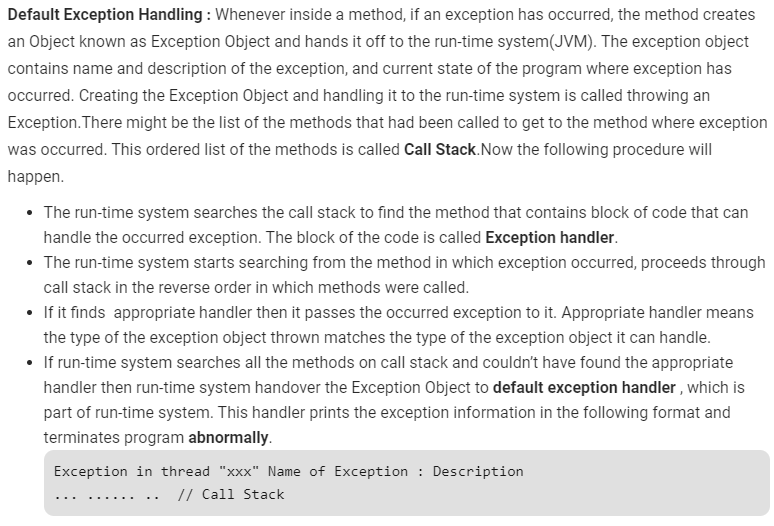
An Error indicates serious problem that a reasonable application should not try to catch.

Both Errors and Exceptions are the subclasses of java.lang.Throwable class. Errors are the conditions which cannot get recovered by any handling techniques. It surely cause termination of the program abnormally. Errors belong to unchecked type and mostly occur at runtime. Some of the examples of errors are Out of memory error or a System crash error. Like indefinite loop can cause stack overflow error.

## **Exception Hierarchy**



## **How JVM handle an Exception?**



Exception handler should be appropriate. Suppose we have an exception called Null pointer exception and in try catch we have handled arithmetic exception, then it will still give the exception because we have handled arithmetic exception not null pointer exception. Similarly if we handle ‘Exception’ in catch block then all type of exceptions will be handled here because Exception is parent class of all exceptions.

## **How Programmer handles an exception?**

Java exception handling is managed via five keywords: try, catch, [throw](https://www.geeksforgeeks.org/throw-throws-java/), [throws](https://www.geeksforgeeks.org/throw-throws-java/), and finally.

Try: Program statements that you think can raise exceptions are contained within a try block. If an exception occurs within the try block, it is thrown.

Catch: Your code can catch this exception (using catch block) and handle it in some rational manner. System-generated exceptions are automatically thrown by the Java run-time system.

Throw: To manually throw an exception, use the keyword [throw](https://www.geeksforgeeks.org/throw-throws-java/).

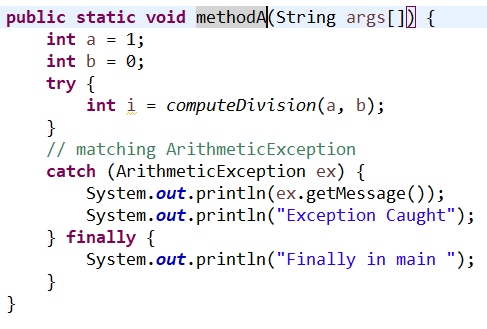
Throws: Any exception that is thrown out of a method must be specified as such by a [throws](https://www.geeksforgeeks.org/throw-throws-java/) clause.

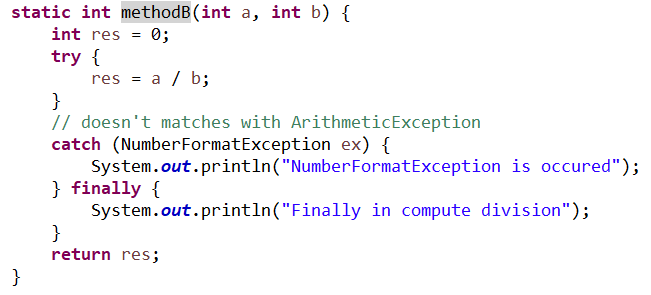
Finally: Any code that absolutely must be executed after a try block completes is put in a finally block.

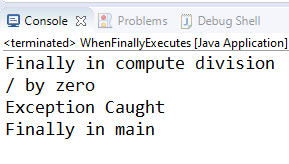
## When Finally Block is executed

Finally block is executed at the end after catch block.

Suppose there is a case in methodA, methodA calls methodB, in methodB I got an Arithmetic exception. But methodB has catch block handling only Number Format exception and a finally block. Then in the call stack control will come back to the methodA and catch the Arithmetic Exception. Now question is when the finally block of methodB will be executed. It will executed even before the execution of catch block of Arithmetic Exception. Check the output below. Correction in below screenshot is int i = methodB (a, b);







<http://www.kscodes.com/java/return-statement-in-try-catch-and-finally-block/>



/General/src/com/gurman/exceptions/WhenFinallyExecutes.java

## Checked Exceptions

Exceptions which are known at compile time only are checked exceptions. If some code within a method throws a checked exception, then the method must either handle the exception or it must specify the exception using *throws*keyword.

E.g. FileNotFoundException, IOException

## Unchecked Exceptions

Exceptions which we get to know at run time only. Either JVM will handle the exception or in more civilized way programmer need to write the code to handle the exception using catch, throw, throws keywords.

E.g. ArithmeticException or NullPointerException

## Throw and throws in Java

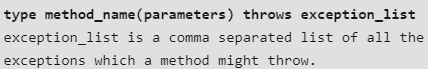
### **Throw**

The throw keyword in Java is used to explicitly throw an exception from a method or any block of code. We can throw either [checked or unchecked exception](https://www.geeksforgeeks.org/checked-vs-unchecked-exceptions-in-java/). The throw keyword is mainly used to throw custom exceptions.



### **Throws**

Throws is a keyword in Java which is used in the signature of method to indicate that this method might throw one of the listed type exceptions. The caller to these methods has to handle the exception using a try-catch block.

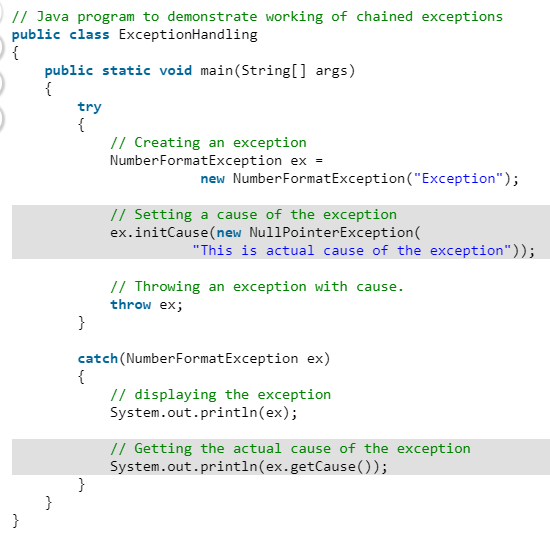


<https://www.geeksforgeeks.org/throw-throws-java/>

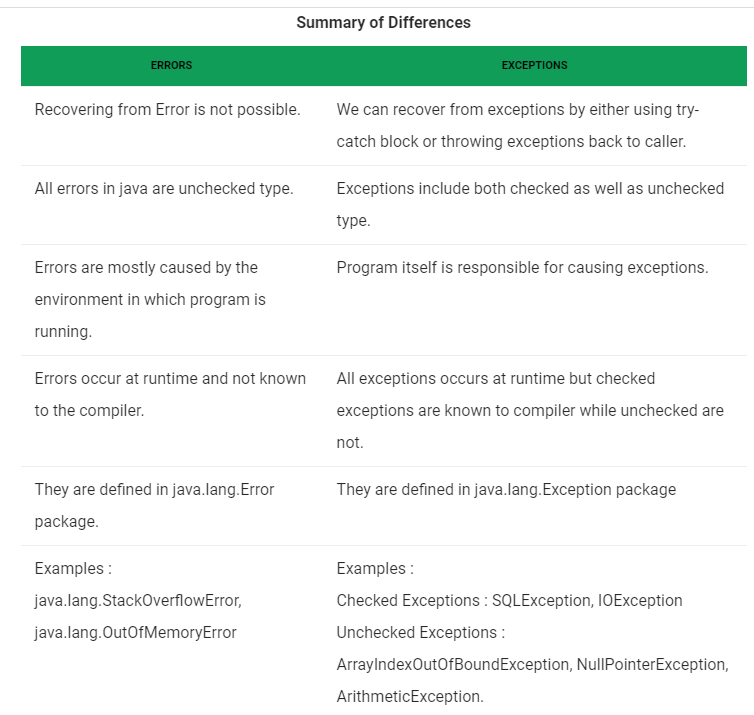
## Chained Exceptions in Java

Using this we can hide the actual cause of exception and wrap it with a different exception. It is just like exception wrapping mainly used to hide the actual exception. It is mainly used to in real time projects to send always the custom exception instead of actual exception. We can also add the cause of exceptions using init method of throwable class.

Chained Exceptions allows to relate one exception with another exception, i.e. one exception describes cause of another exception. For example, consider a situation in which a method throws an ArithmeticException because of an attempt to divide by zero but the actual cause of exception was an I/O error which caused the divisor to be zero. The method will throw only ArithmeticException to the caller. So the caller would not come to know about the actual cause of exception. Chained Exception is used in such type of situations.



## Difference between Error and Exception



## List of Built-in exceptions

<https://www.geeksforgeeks.org/built-exceptions-java-examples/?ref=rp>

## Return Statement in try catch and finally block

<http://www.kscodes.com/java/return-statement-in-try-catch-and-finally-block/>