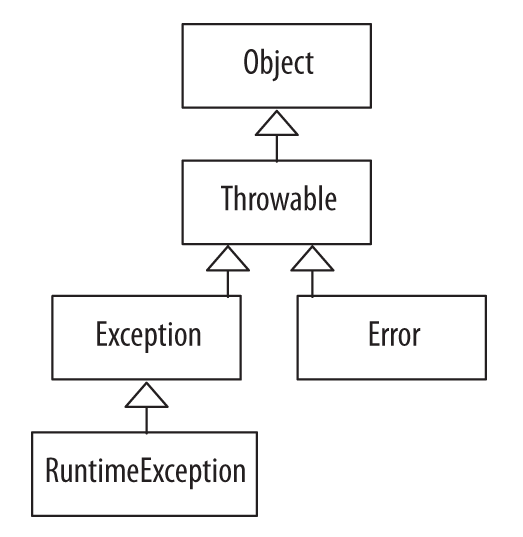
# The Exception Hierarchy



# Checked/Unchecked Exceptions and Errors

## CHECKED EXCEPTIONS

Checked exceptions are checked by the compiler at compile time.

Methods that throw a checked exception must indicate so in the method declaration using the throws clause. This must continue all the way up the calling stack until the exception is handled.

All checked exceptions must be explicitly caught with a catch block.

Checked exceptions include exceptions of the type Exception, and all classes that are subtypes of Exception, except for RuntimeException and the subtypes of RuntimeException.

## UNCHECKED EXCEPTIONS

The compiler does not check unchecked exceptions at compile time.

Unchecked exceptions occur during runtime due to programmer error (out-of-bounds index, divide by zero, and null pointer exception) or system resource exhaustion.

Unchecked exceptions do not have to be caught.

Methods that may throw an unchecked exception do not have to (but can) indicate this in the method declaration.

Unchecked exceptions include exceptions of the type RuntimeException and all subtypes of RuntimeException.

## ERRORS

Errors are typically unrecoverable and present serious conditions.

Errors are not checked at compile time and do not have to be (but can be) caught/handled.

## COMMON CHECKED EXCEPTIONS

**ClassNotFoundException**

Thrown when a class cannot be loaded because its definition cannot be found.

**IOException**

Thrown when a failed or interrupted operation occurs. Two common subtypes of IOException are EOFException and FileNotFoundException.

**FileNotFoundException**

Thrown when an attempt is made to open a file that cannot be found.

**SQLException**

Thrown when there is a database error.

**InterruptedException**

Thrown when a thread is interrupted.

**NoSuchMethodException**

Thrown when a called method cannot be found.

**CloneNotSupportedException**

Thrown when clone() is called by an object that is not cloneable.

## COMMON UNCHECKED EXCEPTIONS

**ArithmeticException**

Thrown to indicate that an exceptional arithmetic condition has occurred.

**ArrayIndexOutOfBoundsException**

Thrown to indicate index out of range.

**ClassCastException**

Thrown to indicate an attempt to cast an object to a subclass of which it is not an instance.

**DateTimeException**

Thrown to indicate indicate problems with creating, querying and manipulating date-time objects.

**IllegalArgumentException**

Thrown to indicate that an invalid argument has been passed to a method.

**IllegalStateException**

Thrown to indicate that a method has been called at an inappropriate time.

**IndexOutOfBoundsException**

Thrown to indicate that an index is out of range.

**NullPointerException**

Thrown when code references a null object but a nonnull object is required.

**NumberFormatException**

Thrown to indicate an invalid attempt to convert a string to a numeric type.

**UncheckedIOException**

Wraps an IOException with an unchecked exception.

## COMMON ERRORS

**AssertionError**

Thrown to indicate that an assertion failed.

**ExceptionInInitializeError**

Thrown to indicate an unexpected exception in a static initializer.

**VirtualMachineError**

Thrown to indicate a problem with the JVM.

**OutOfMemoryError**

Thrown when there is no more memory available to allocate an object or perform garbage collection.

**NoClassDefFoundError**

Thrown when the JVM cannot find a class definition that was found at compile time.

**StackOverflowError**

Thrown to indicate that a stack overflow occurs.

# Exception Handling Keywords

## THE THROW KEYWORD

## THE TRY-CATCH-FINALLY STATEMENT

The try-catch statement includes one try and one or more catch blocks.

The try block contains code that may throw exceptions. All checked exceptions that may be thrown must have a catch block to handle the exception. If no exceptions are thrown, the try block terminates normally. A try block may have zero or more catch clauses to handle the exceptions.

The order of the catch clauses in a try/catch block defines the precedence for catching exceptions. Always begin with the most specific exception that may be thrown and end with the most general. ->

* The catch block with the Exception parameter should always be last in the ordered list.

This finally block is optional and is only used where needed. When used, it is executed last in a try-finally block and will always be executed, whether or not the try block terminates normally. If the finally block throws an exception, it must be handled.

## THE TRY-WITH-RESOURCES STATEMENT

The try-with-resources statement is used for declaring resources that must be closed when they are no longer needed.

Any resource that implements the **AutoClosable** interface may be used with the try-with-resources statement.

## THE MULTI-CATCH CLAUSE

The multi-catch clause is used to allow for multiple exception arguments in one catch clause.

# The Exception Handling Process

Here are the steps to the exception handling process:

* An exception is encountered resulting in an exception object being created.
* A new exception object is thrown.
* The runtime system looks for code to handle the exception beginning with the method in which the exception object was created. If no handler is found, the runtime environment traverses the call stack (the ordered list of methods) in reverse looking for an exception handler. If the exception is not handled, the program exits and a stack trace is automatically output.
* The runtime system hands the exception object off to an exception handler to handle (catch) the exception.

# Defining Your Own Exception Class

Programmer-defined exceptions should be created when those other than the existing Java exceptions are necessary. In general, the Java exceptions should be reused wherever possible:

* To define a checked exception, the new exception class must extend the Exception class, directly or indirectly.
* To define an unchecked exception, the new exception class must extend the RuntimeException class, directly or indirectly.
* To define an unchecked error, the new error class must extend the Error class.

User-defined exceptions should have at least two constructors—a constructor that does not accept any arguments and a constructor that does