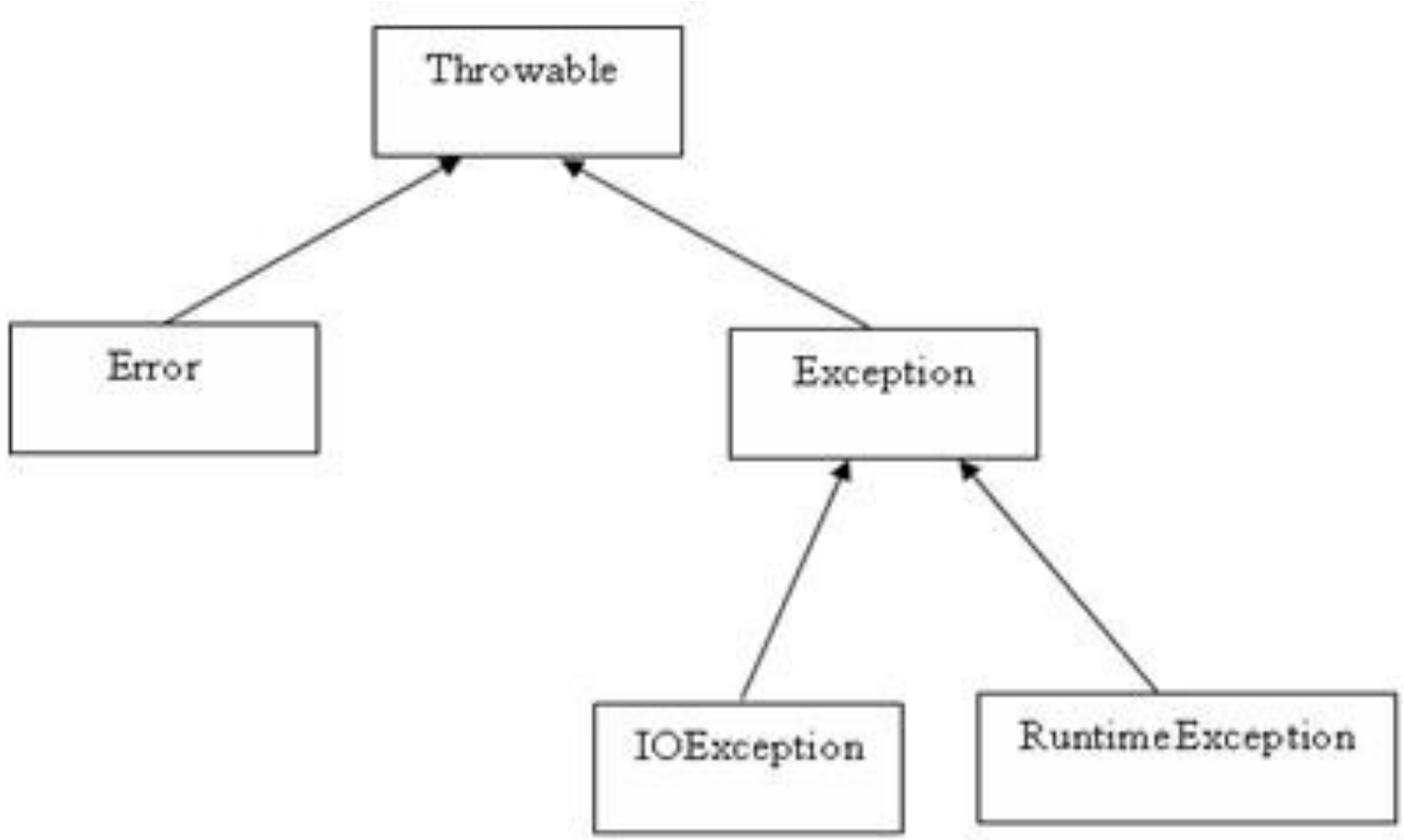
Exceptions:

An exception is a problem that arises during the execution of a program.

- **A user has entered invalid data.**
- **A file that needs to be opened cannot be found.**
- **A network connection has been lost in the middle of communications or the JVM has run out of memory**

Exception hierarchy



There are three categories:

- **Checked exceptions: A checked exception is an exception that is typically a user error or a problem that cannot be foreseen by the programmer. For example, if a file is to be opened, but the file cannot be found, an exception occurs. These exceptions cannot simply be ignored at the time of compilation.**

- **Runtime exceptions: A runtime exception is an exception that occurs that probably could have been avoided by the programmer. As opposed to checked exceptions, runtime exceptions are ignored at the time of compilation.**

- **Errors: These are not exceptions at all, but problems that arise beyond the control of the user or the programmer. Errors are typically ignored in your code because you can rarely do anything about an error. For example, if a stack overflow occurs, an error will arise. They are also ignored at the time of compilation.**

Built-in Exceptions: UnChecked Exceptions

Exception	Description
ArithmeticException	**Arithmetic error, such as divide-by-zero.**
ArrayIndexOutOfBoundsException	**Array index is out-of-bounds.**
ArrayStoreException	**Assignment to an array element of an incompatible type.**
ClassCastException	**Invalid cast.**
IllegalArgumentException	**Illegal argument used to invoke a method.**
IllegalMonitorStateException	**Illegal monitor operation, such as waiting on an unlocked thread.**
IllegalStateException	**Environment or application is in incorrect state.**
IllegalThreadStateException	**Requested operation not compatible with current thread state.**
IndexOutOfBoundsException	**Some type of index is out-of-bounds.**
NegativeArraySizeException	**Array created with a negative size.**
NullPointerException	**Invalid use of a null reference.**
NumberFormatException	**Invalid conversion of a string to a numeric format.**
SecurityException	**Attempt to violate security.**
StringIndexOutOfBounds	**Attempt to index outside the bounds of a string.**

Java Checked Exception:

Exception	Description
ClassNotFoundException	**Class not found.**
CloneNotSupportedException	**Attempt to clone an object that does not implement the Cloneable interface.**
IllegalAccessException	**Access to a class is denied.**
InstantiationException	**Attempt to create an object of an abstract class or interface.**
InterruptedException	**One thread has been interrupted by another thread.**
NoSuchFieldException	**A requested field does not exist.**
NoSuchMethodException	**A requested method does not exist.**

SN	Methods with Description
1	**public String getMessage()** Returns a detailed message about the exception that has occurred. This message is initialized in the Throwable constructor.
2	**public Throwable getCause()** Returns the cause of the exception as represented by a Throwable object.
3	**public String toString()** Returns the name of the class concatenated with the result of getMessage()
4	**public void printStackTrace()** Prints the result of toString() along with the stack trace to System.err, the error output stream.
5	**public StackTraceElement [] getStackTrace()** Returns an array containing each element on the stack trace. The element at index 0 represents the top of the call stack, and the last element in the array represents the method at the bottom of the call stack.
6	**public Throwable fillInStackTrace()** Fills the stack trace of this Throwable object with the current stack trace, adding to any previous information in the stack trace.

Catch: A method catches an exception using a combination of the try and catch keywords. A try/ catch block is placed around the code that might generate an exception.

```
// File Name : ExcepTest.java
import java.io.*;
public class ExcepTest{

public static void main(String args[]){
try{
int a[]=new int[2];
System.out.println("Access element three :"+ a[3]);
}
catch(ArrayIndexOutOfBoundsException e)
{ System.out.println("Exception thrown:"+ e);
}
System.out.println("Out of the block");
}
}
```

The throw/throws keyword:

If a method does not handle a checked exception, the method must declare it using the throwskeyword. The throws keyword appears at the end of a method's signature.

The finally Keyword

The finally keyword is used to create a block of code that follows a try block. A finally block of code always executes, whether or not an exception has occurred.

```
public class ExcepTest{

public static void main(String args[]){
int a[]=new int[2];
try{
System.out.println("Access element three :"+ a[3]);
}
catch(ArrayIndexOutOfBoundsException e)
{
System.out.println("Exception thrown  :"+ e);
}
finally{
  a[0]=6;
System.out.println("First element value: "+a[0]);
System.out.println("The finally statement is executed");
}
}
}
```

- A catch clause cannot exist without a try statement.
- It is not compulsory to have finally clauses whenever a try/catch block is present.
- The try block cannot be present without either catch clause or finally clause.
- Any code cannot be present in between the try, catch, finally blocks

Declaring your own Exceptions:

- **All exceptions must be a child of Throwable.**
- **If you want to write a checked exception that is automatically enforced by the Handle or Declare Rule, you need to extend the Exception class.**
- **If you want to write a runtime exception, you need to extend the RuntimeException class.**

Common Exceptions:

In Java, it is possible to define two categories of Exceptions and Errors

- **JVM Exceptions: - These are exceptions/errors that are exclusively or logically thrown by the JVM. Examples: NullPointerException, ArrayIndexOutOfBoundsException, ClassCastException**
- **Programmatic exceptions:- These exceptions are thrown explicitly by the application or the API programmers. Examples: IllegalArgumentException, IllegalStateException.**

How to Rethrow an Exception in Java

When an exception is cached in a catch block, you can re-throw it using the throw keyword (which is used to throw the exception objects).

```
try {
   int result = (arr[a])/(arr[b]);
   System.out.println("Result of "+arr[a]
+"/"+arr[b]+": "+result);
}
catch(ArithmeticException e) {
   throw e;
}
```