# Chapter 9: Java Exceptions

- **Exception** is an abnormal condition that arises in a code sequence at run time i.e. exception is a run-time error.
- **Exception Handling:-**
  Exception Handling is a process of handling the exception object by using try-catch-finally block to prevent the program from abnormal termination and continue with the remaining code.

**Java Exceptions:**

- Exception is a runtime error.
- All exceptions occur only at runtime but some exceptions are detected at compile time and other at runtime.

**Checked Exceptions:**

- The exceptions that are checked at compilation time by Java compiler are called Checked Exceptions.
- In case of Checked Exceptions, the programmer should either handle them or throw them without handling them.
- Programmer cannot ignore Checked Exceptions as Java Compiler will remind him of them.

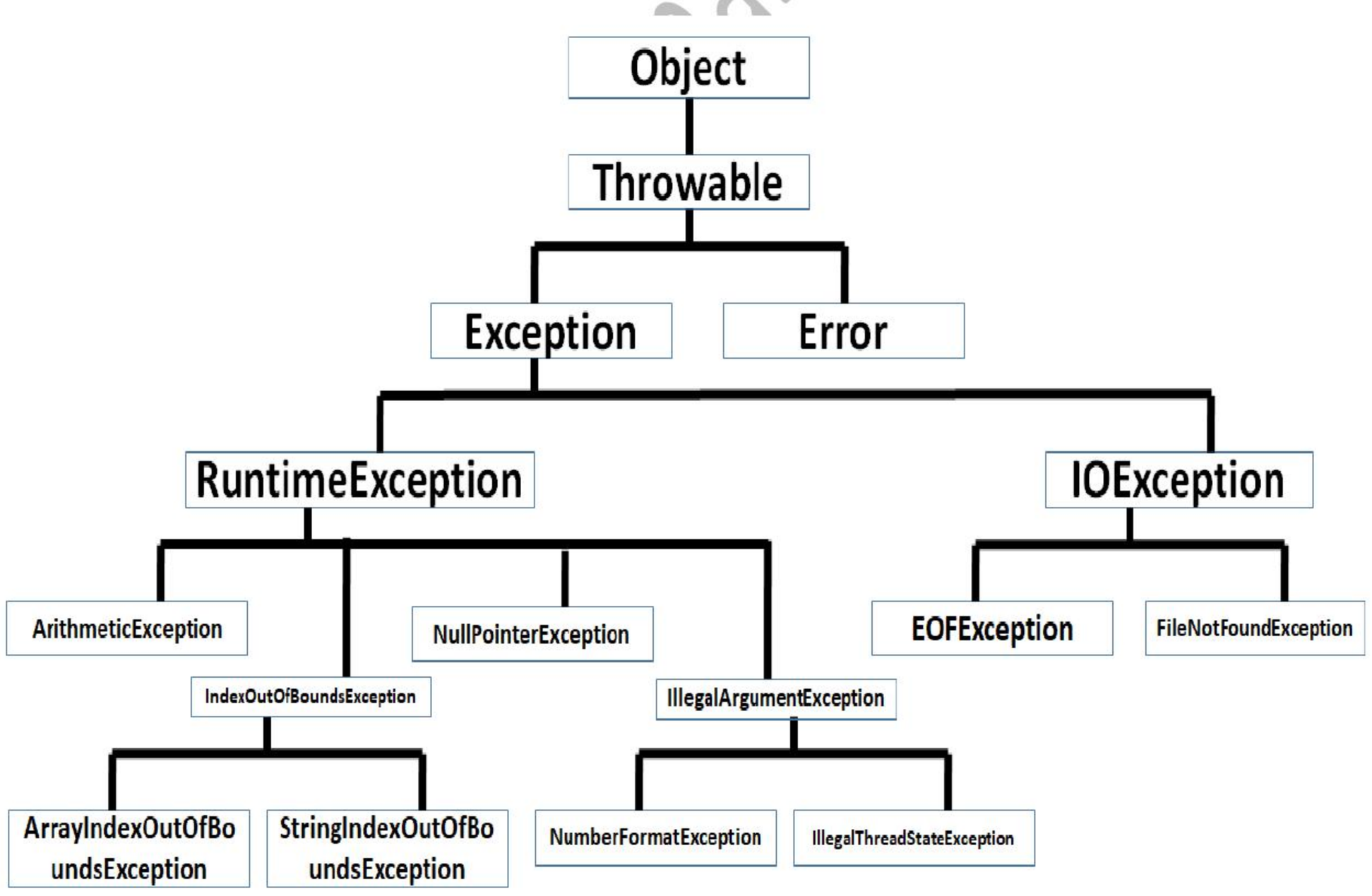
| Exception | Description |
|---|---|
| IOException | IO activity could not be performed. |
| ClassNotFoundException | Class not found |
| IllegalAccessException | Access to a class is denied |

**Unchecked Exceptions:**

- The exceptions that are checked by the JVM are called Unchecked Exceptions.
- Programmer can write a Java program with unchecked exceptions and errors and can compile the program.
- Programmer can see their effect only when he runs the program.

| Exception | Description |
|---|---|
| ArithmeticException | Arithmetic Error, such as divide by zero |
| ArrayIndexOutOfBoundsException | Array index is out-of-bounds |
| NegativeArraySizeException | Array created with a negative size |
| NullPointerException | invalid use of Null reference |
| NumberFormatException | invalid conversion of a string to a numeric format |

**Exception Hierarchy:**



**Exception Handling:**

- try – code that might throw an excpetion.
- catch – code to handle exception
- finally – code that will always get executed.
- throw – to manually throw exception.
- throws – an exception that is throw out of method.

**try with single catch:**

- No statement is allowed  between try and catch block.
- catch blocks argument is always of type Throwable.

**try with multiple catch:**

- Super class exception should appear after all of its subclasses.
- We cannot have multiple catch blocks for the same exception.
- Multiple catch blocks should appear back to back after try block.

**try with multi catch:**

- multi-catch block Exceptions should have only 1 reference variable.
- exceptions should not be related by subclassing.
- Unrelated Exceptions can be written in any order.

**finally keyword:**

- When exceptions are thrown, execution in a method takes a rather abrupt non-linear path that alters the normal flow through the method.
- The finally block will execute whether or not an exception is thrown.
- The finally clause can be useful for closing file handles and freeing up any other resources that might have been allocated at the beginning of a method with the intent of disposing them before returning.

**finally keyword : Important Points to Remember:**

1) There can be no statement between catch and finally block

2) finally block can exists without catch block but in such case exception would get propogated.

3) try can have multiple catch by only single finally block.

4) finally block would not get executed or get executed completely under following conditions:

1) When the System.exit(1) method is called before executing the finally block.

2) When the return statement is used in the finally block.

3) When an exception arises in the code written in the finally block.

**throw Keyword:**

- It is possible for the program to throw an exception explicitly, using throw statement.
- The flow of execution stops immediately after the throw statement.
  - The **nearest enclosing try block** is inspected to see if it has a catch statement that matches the type of the exception.
  - If it does **find a match**, **control is transferred** to that statement.
  - If **not**, then the **next enclosing try statement** is inspected and so on.
  - If **no matching catch** is found, then the **default exception handler** halts the program.

**throws keyword:**

- If a method is capable of causing an exception that it does not handle, it must specify this behavior so that callers of the method can guard themselves against that exception.
- This can be done by including a throws clause in the methods declaration.
- The throws clause lists the type of exceptions that a method might throw. If they are not, a compile-time error will result.