Exception Handling

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

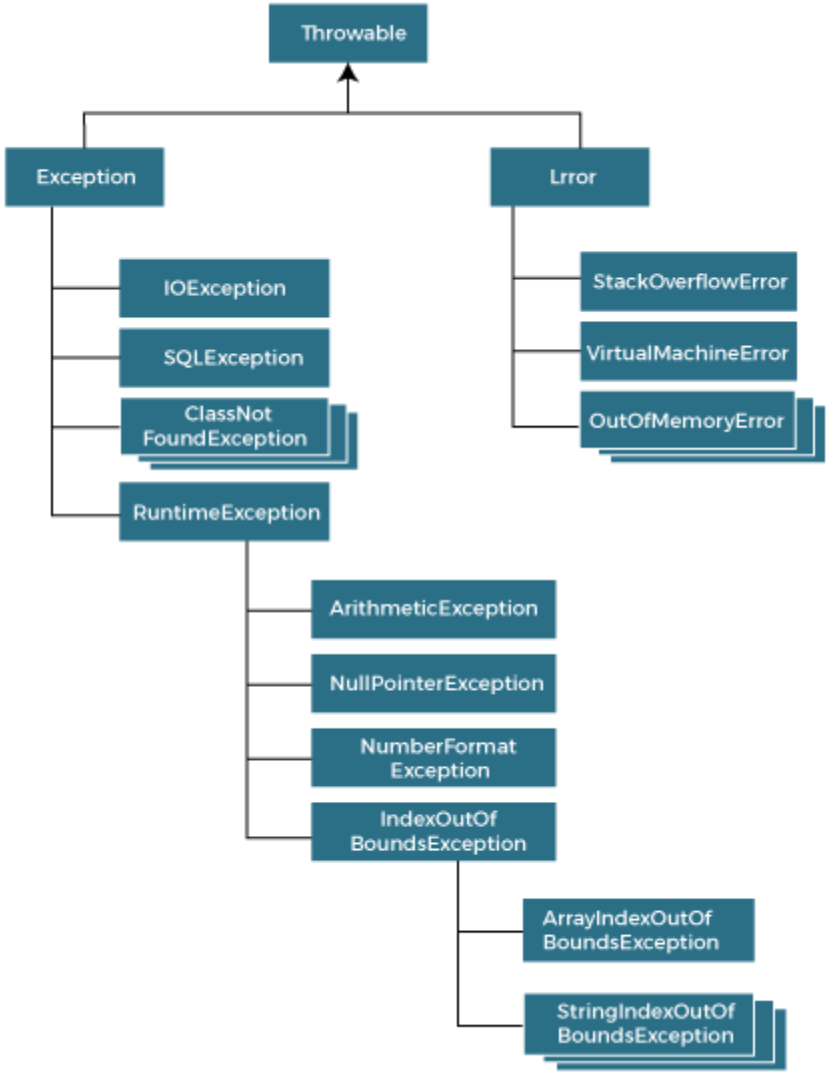
In this tutorial, we will learn about Java exceptions, it's types, and the difference between checked and unchecked exceptions.

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 need to handle exceptions. Let's consider a scenario:

```
1.statement 1;
2.statement 2;
3.statement 3;
4.statement 4;
5.statement 5;//exception occurs
6.statement 6;
7.statement 7;
8.statement 8;
9.statement 9;
10.statement 10;
```

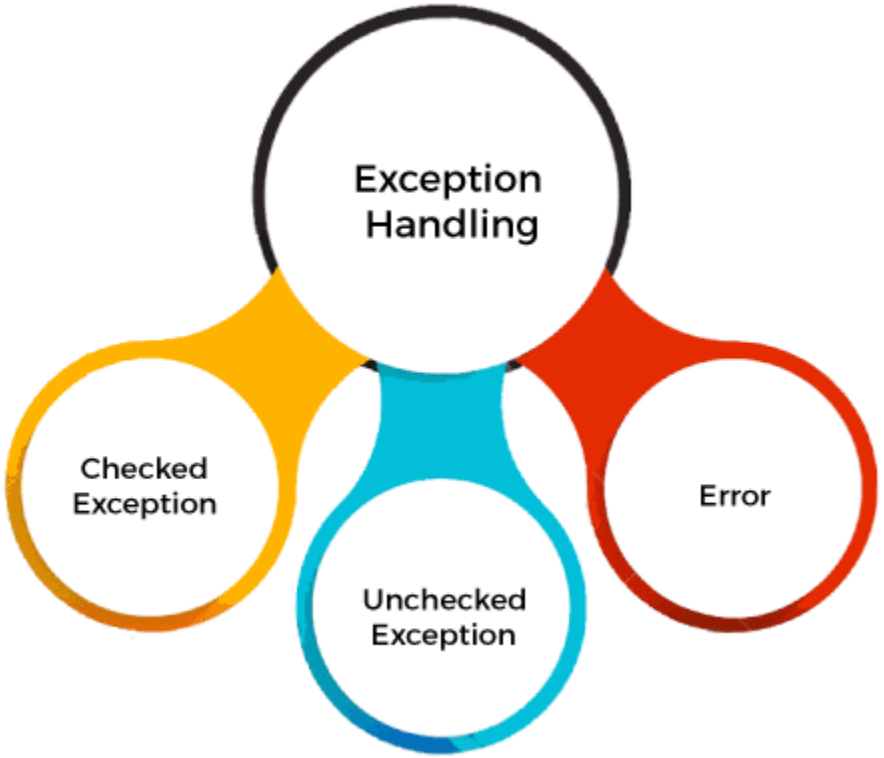
Suppose there are 10 statements in a Java program and an exception occurs at statement 5; the rest of the code will not be executed, i.e., statements 6 to 10 will not be executed. However, when we perform exception handling, the rest of the statements will be executed. That is why we use exception handling


Throwable
Exception
Lrror
IOException
SQLException
ClassNot FoundException
RuntimeException
StackOverflowError
VirtualMachineError
OutOfMemoryError
ArithmeticException
NullPointerException
NumberFormat Exception
IndexOutOf BoundsException
ArrayIndexOutOf BoundsException
StringIndexOutOf BoundsException

Types of Exceptions

There are mainly two types of exceptions: checked and unchecked. An error is considered as the unchecked exception. However, according to Oracle, there are three types of exceptions namely:

1.Checked Exception
2.Unchecked Exception
3.Error



Exception Handling Example

Let's see an example of Java Exception Handling in which we are using a try-catch statement to handle the exception.

JavaExceptionExample.java

```
1.public class JavaExceptionExample{
2.  public static void main(String args[]){
3.   try{
4.     //code that may raise exception
5.     int data=100/0;
6.   }catch(ArithmeticException e){System.out.println(e);}
7.   //rest code of the program
8.   System.out.println("rest of the code...");
9.  }
10.}
```

in thread main java.lang.ArithmeticException:/ by zero rest of the code...