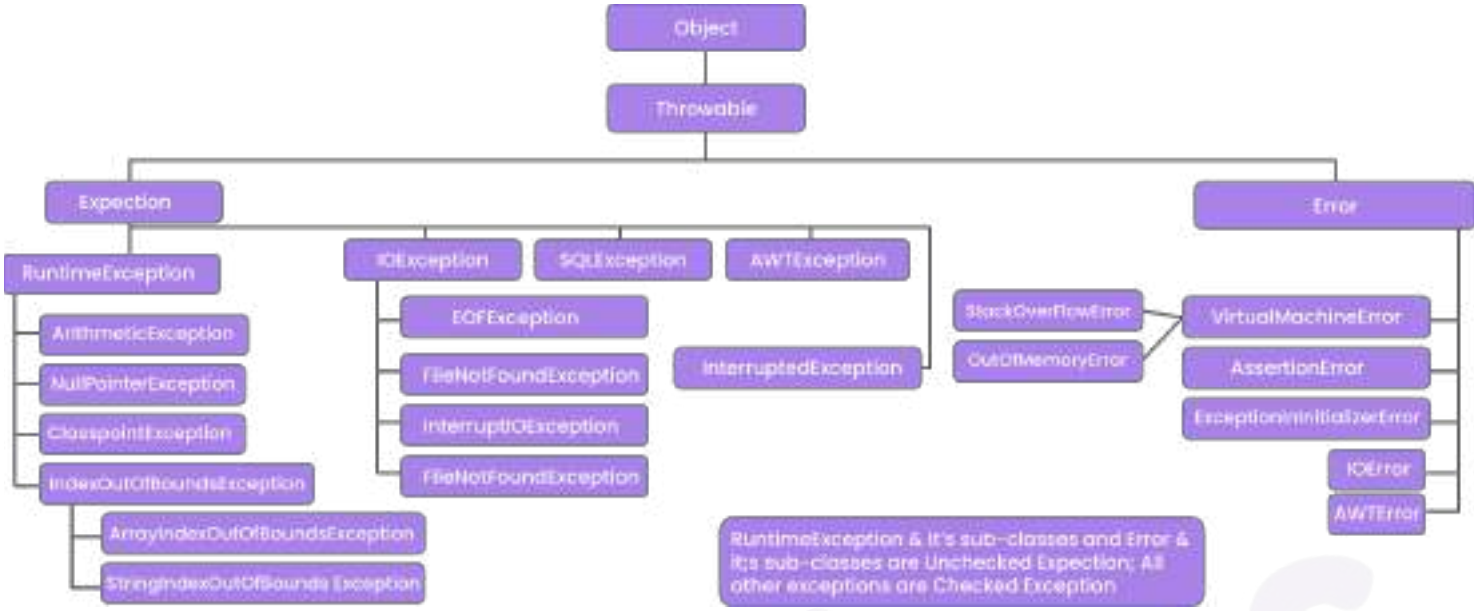
Exception hierarchy



Exception

- Most of the cases exceptions are caused by our program and these are recoverable

Example::

If FileNotFoundException occurs then we can use a local file and we can continue the rest of the program execution normally.

Error

- Most of the cases errors are not caused by our program these are due to lack of system resources and these are non-recoverable.

Example::

If OutOfMemoryError occurs being a programmer we can't do anything the program will be terminated abnormally.

Checked vs UnCheckedException

- The exceptions which are checked by the compiler whether programmer handling or not, for smooth execution of the program at the runtime are called “CheckedException”.

Example::

FileNotFoundException,IOException,SQLException...

- The exceptions which are not checked by the compiler whether the programmer is handling or not such type of exceptions are called "UnCheckedExceptions".

Example::

NullPointerException,ArithmeticException

Note

- RunTimeException and its child classes,Error and its child classes are called as "UncheckedException", remaining all exceptions are considered as "CheckedExceptions".
- Whether the exception is checked or unchecked compulsorily it should occur at runtime only and there is no chance of Occuring any exception at compile time.

Fully Checked Exception

A checked exception is said to be a fully checked exception if and only if all its child classes are also checked.

Example:

IOException, InterruptedException

Partially Checked Exception

A checked exception is said to be partially checked if and only if some of its child classes are unchecked.

Example:

Throwable, Exception

Describe the behaviour of following exceptions?

1. RunTimeException **Ans.UncheckedException**
2. Error **Ans.UncheckedException**
3. IOException **Ans.fullychecked**
4. Exception **Ans.partiallychecked**
5. InterruptedException **Ans.fullychecked**
6. Throwable **Ans.partially checked**
7. ArithmeticException **Ans.unchecked**
8. NullPointerException **Ans.unchecked**
9. FileNotFoundException **Ans.fullychecked**

Control flow in try catch

```
try{
 Statement-1;
 Statement-2;
 Statement-3;
}catch( X e){
 Statement-4;
}
Statement5;
```

Case1

If there is no exception
1,2,3,5 normal termination

Case2

if an exception raised at statement2 and corresponding catch block matched
1,4,5 normal termination

Case 3
if any exception raised at statement2 but the corresponding catch block not matched, followed by abnormal termination

Case 4
if an exception raised at statement 4 or statement 5 then its always abnormal termination of the program.

Note

1. Within the try block if anywhere an exception is raised then the rest of the try block won't be executed even though we handled that exception.Hence we have to place/take only risk code inside try block and length of the try block should be as less as possible.
2. If any statement which raises an exception and it is not part of any try block then it is always an abnormal termination of the program.

Various methods to print exception information
Throwable class defines the following methods to print exception information to the console

printStackTrace()

- This method prints exception information in the following format.
- Name of the exception:description of exception stack trace

toString()

- This method prints exception information in the following format
- Name of the exception : description of exception

getMessage()

- This method returns only the description of the exception Description.

Example:

```
public class Test{
       public static void main(String[] args){
  try{
   System.out.println(10/0);
  }catch(ArithmeticException e){
   e.printStackTrace();
   System.out.println(e);
   System.out.println(e.getMessage());
  }
 }
}
```

Note:
Default exception handler internally uses **printStackTrace()** method to print exception information to the console.