**Exceptional Handling**

An Exception is an abnormal condition in which normal execution of code gets hampered.

Exceptions occur only at runtime during program execution.

**Throwable** is the Super class for all the Exceptions in Java.

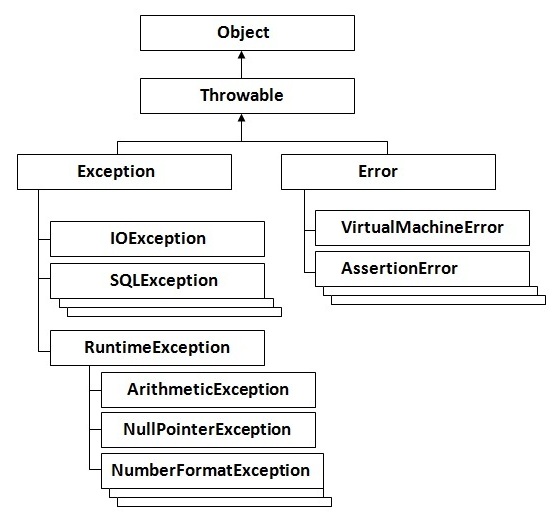
There are 2 categories under Throwable are:

1. Error:
   * + Errors are thrown by the Java run time system and indicate some irrecoverable conditions that occur during the Program Execution.
     + Once thrown by the system, the application cannot recover for it and will come to halt.
     + Errors need not be handled(Catched)
     + E.g.: java.lang.OutOfMemoryError, java.lang.StackOverflowError
2. Exception:
   * + Unlike Errors, Exceptions can be handled to prevent the program to automatically terminate.
     + We will be able to recover from Application and we can go forward.
     + Exceptions can be handled using try and catch blocks
   * Runtime Exception (Unchecked Exceptions) Ex: ArithmeticException Exceptions
   * Exceptions that extend Exception is called checked Exceptions. Ex: IOException, SQLException.

**Exceptional Handling** is a mechanism to handle the runtime errors such as ClassNotFound, IO, SQL, Remote etc.

**Advantage of Exceptional Handling:**

The core advantage of exception handling is **to maintain the normal flow of the application.** Exception normally disrupts the normal flow of the application that is why we use exception handling.



Types of Exceptions:

* **Checked Exceptions:**

The classes that extend Throwable class except RuntimeException and Error are known as checked Exceptions e.g. IOException, SQLException etc. Checked exceptions are checked at compile-time.

* **Unchecked Exceptions:**

The classes that extend RuntimeException are known as unchecked exceptions e.g., ArithemeticException, NullPointerException, ArrayIndexOutOfBoundsException etc. Unchecked exceptions are not checked at compile-time rather they are checked at runtime.

* **Errors:**

Error is irrecoverable e.g. OutOfMemoryError, VirtualMachineError, AssertionError etc.

­­­­­­­­­­­­­­

**Try, catch and finally:**

* + - The code which might throw some exceptions should be kept in try block.
    - The catch block can have the code to handle the exception or log the same.
    - finally block can be used to clean up code or release some resources that are utilized in the program.

**Rules:**

* When there is a try block, catch or finally block should be there.
* There should not be any statements between try and catch and finally.
* finally block is always optional.
* There can be multiple catch blocks.
* There should be only one finally for a try block.
* Catch block is executed only when an exception is thrown.
* The narrow exceptions should be declared first and broader exceptions last.
* When an exception is thrown, the rest of the lines of code in try block will not get executed and the control directly goes for catch or finally.

**Exception Propagation:**

**Throws:**

Not all the times we will be willing to handle the exception. In that case, you can throw the exception, so that one who is calling the method should ensure handle the exceptions.

Either you should handle the exceptions using try-catch block or throws clause in your method.

When marked as throws, the calling methods have to handle the exception or declare to throw the same.

**Re Throw the Exceptions:**

You can also re throw the exceptions that you are catching. This is for some kind of logging purposes.

When throwing the exception back to the calling method, it is mandatory to throws the exception.

Ex: public void readFile() throws **FileNotFoundException** {

try{

FileReader fr = new FileReader();

catch(FileNotFoundException e){

log.info(e.getMessage);

**throw e**;}

When you throw an exception from catch block, you should declare that in methods throws clause.

**To handle or throws exceptions?**

Depends on the situation here:

When your application has to send information to 3rd party, then you should throws the exception, so that 3rd party will know, what kind of Error is thrown.