Discussion 12

Why use Exceptions rather than simply Error Handling

<https://www.tutorialspoint.com/java/java_exceptions.htm>

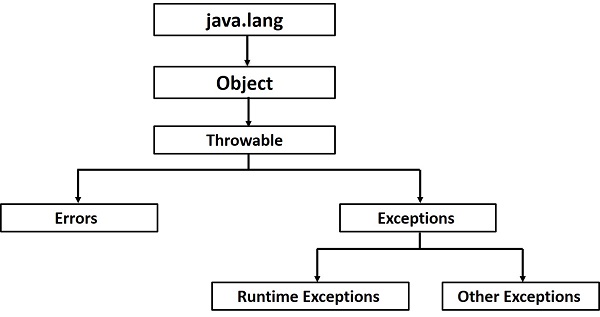
Using exceptions rather than just traditional error handling allows you to remain control in your program. When an undesired or unexpected result occurs, exceptions give you the ability to handle the error instead of having the program crash or terminate prematurely. This also gives you the ability to give the user feedback on what went wrong, and if they entered invalid data. The logic of the try.. catch blocks are much easier to understand and follow because everything is grouped together. Object oriented exception handling gives programmers the flexibility to appropriately deal with exceptions as they decide how to handle them.

Notes:

**Checked exceptions** − A checked exception is an exception that occurs at the compile time, these are also called as compile time exceptions. These exceptions cannot simply be ignored at the time of compilation, the programmer should take care of (handle) these exceptions.

**Unchecked exceptions** − An unchecked exception is an exception that occurs at the time of execution. These are also called as **Runtime Exceptions**. These include programming bugs, such as logic errors or improper use of an API. 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.



Throw: You can throw an exception, either a newly instantiated one or an exception that you just caught, by using the **throw** keyword.

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

Try to understand the difference between throws and throw keywords, *throws* is used to postpone the handling of a checked exception and *throw* is used to invoke an exception explicitly.

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.