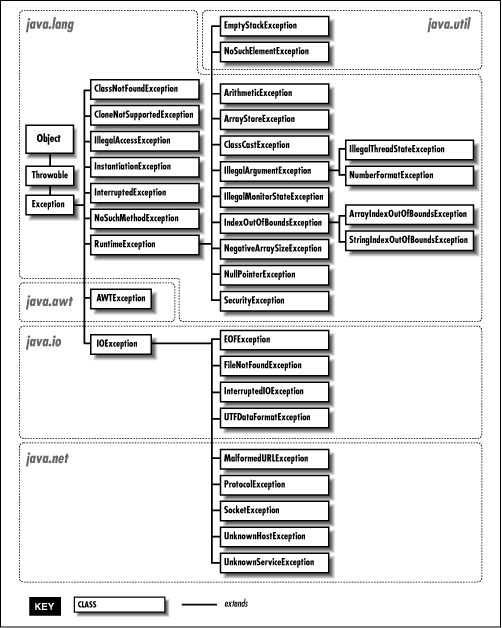
En java se pueden llegar a presenta los siguientes Errores

La clase padre de todos los errores es la clase **Throwable**, las clases hijas son las clases:

* **Error**: Causados por la máquina virtual de java (JVM).
* **Exception**: Causados por el desarrollador.
  + **UNCHECKED**: Errores que no conocemos inicialmente. Errores de lógica. Por ejemplo, los ariméticos o agregar a un índice de un arreglo que no existe.
    - **RuntimeException**: En el momento que la aplicación se está ejecutando.
  + **CHECKED**: Errores más esperados.
    - **SQLException**: Error en sintaxis de consultas de bases de datos.
    - **IOException**: Error al leer un archivo o entrada/salida de datos.
    - **FileNotFoundException**: Un archivo no fue encontrado.



<https://es.stackoverflow.com/questions/2306/listado-de-excepciones-en-java>

Para evitar que en el momento enque la aplicacion falle por algun motive se cierre o quede congelada , podemos intentar controlar su comportamiento con el bloquede codigo dentro de un

**Try Catch**

Como utilizarlo ?

try{

// Codigo que se ejecutara y que en

// el cual pueda llegar a presentarse algun error

}catch( *ExceptionType* *name*){ //<-- Prosible excepción que se presente en el codigo

// Codigo que se ejecutara si se presenta la excepción

}catch (*ExceptionType* *name*){

}finally{

/\* Codigo que se ejecutara sin importar

si se presenta o no una Excepción

como por ejemplo el cierre de la conexion a una base de datos

\*/

}

Existe una nueva funcionalidad en Java 9 Try-with-resources

Este lo que viene a solucionar es el tener que evitar el colocar otro try-catch dentro del finally en ciertos casos como el cierre de recursos como bases de datos o documentos.

*BufferedReader* reader = **new** BufferedReader(**new** InputStreamReader(System.in));

try(reader){

// Codigo...

}catch(*Exception* *ex*){

// Manejo de exception...

}

De este modo el recurso se cerrara de forma automática cuando termine.