ERRORES

- Un error es un fragmento de código que no sigue la sintaxis o semántica que el lenguaje de programación requiere
- Un error siempre es fatal
- Un error suele estar en el código, dicho fragmento de código no permitirá que se pueda compilar completo

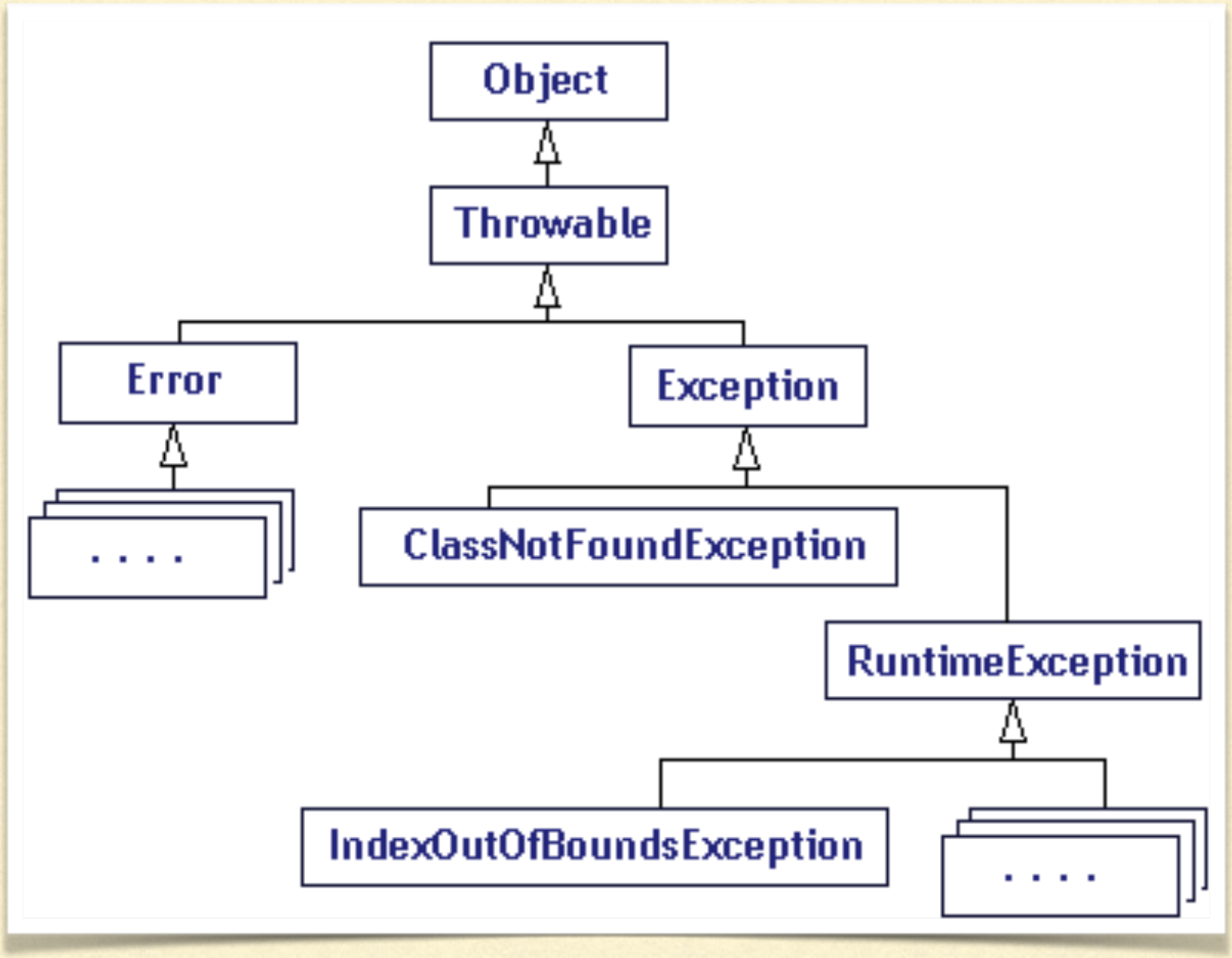
EXCEPCIONES

- Una excepción es una situación anómala que se presenta ya durante la ejecución de un programa

- Una excepción se puede manejar en tiempo de ejecución, por lo que la ejecución podría seguir a pesar de la excepción

- Las excepciones, en muchos casos, se presentan por motivos no directamente contenidos en el código

EXCEPCIONES



EXCEPCIONES

- ClassNotFoundException (lang)
- RuntimeException (lang)
- ClassCastException (lang)
- IndexOutOfBoundsException (lang)
- ArithmeticException (lang)
- NegativeArraySizeException (lang)
- InputMismatchException (util)
- NumberFormatException (lang)
- NullPointerException (lang)
- IOException (io)
- FileNotFoundException (io)

EXCEPCIONES

- **ClassNotFoundException**: se presenta cuando se intenta ejecutar un proyecto y no se puede encontrar una de las clases con las que éste trabaja. E. G.: cuando falta la clase main

- **RuntimeException**: representa a las excepciones que se pueden presentar durante la ejecución del programa sobre la JVM. Es la única clase de excepción que el usuario no tiene que capturar. E. G.: que falten recursos para ejecutar el programa

EXCEPCIONES

- **ClassCastExcpetion**: se presenta cuando se quiere hacer una conversión de clases de una superclase a una clase que no es su subclase.

- **IndexOutOfBoundsException**: se presenta al trabajar con listas, arreglos o cadenas y querer acceder a un índice que quede fuera de los límites definidos de dicho objeto

- **ArithmeticException**: se presenta cuando se da una condición extraordinaria aritmética. E. G.: hacer una división entre 0

EXCEPCIONES

- **NegativeArraySizeException**: se presenta cuando se quiere crear un arreglo y le damos un tamaño negativo.

- **InputMismatchException**: se presenta cuando se espera un tipo de dato de entrada pero se recibe otro tipo de dato. Éstas son comunes al trabajar con la clase Scanner

- **NullPointerException**: se presenta cuando intentamos acceder a un método o atributo de un objeto que tiene asignado un valor nulo. Ésta es común al trabajar con arreglos de objetos

EXCEPCIONES

- **NumberFormatException**: se presenta cuando se quiere hacer un parseo de una cadena a un entero pero la cadena no tiene formato de números

- **IOException**: se presenta cuando hay errores en alguna lectura o salida de datos a algún medio externo. Éstas se pueden presentar frecuentemente al trabajar con archivos

- **FileNotFoundException**: se presenta al querer abrir un archivo al cual no se puede acceder, ya sea por permisos o porque no existe

MANEJO DE EXCEPCIONES

- Por heredar de las clases Object y Throwable las excepciones tienen algunos métodos de utilidad:

- toString(): regresa una cadena con el nombre de la clase, se tiene que redefinir

- getMessage(): regresa una cadena con un mensaje de error original que ya incluye cada excepción

MANEJO DE EXCEPCIONES

- Para el manejo de excepciones existen tres palabras reservadas:
- **try{...}**: encierra al fragmento de código en el que se puede presentar la excepción
- **catch(Exception){...}**: contiene código que se ejecutará para manejar la excepción en caso de que se presente una. Como argumento recibe un objeto de tipo Exception o una subclase para especificar que clase de excepción se va a manejar con ese código
- **finally{...}**: contiene un fragmento de código que siempre se ejecutará sin importar si se presentó o no una excepción