**INTRODUCCIÓN: GESTIÓN EXCEPCIONES EN JAVA**

A continuación vamos a ver como el lenguaje Java implementa su propio sistema de gestión de excepciones, o como hemos mencionado anteriormente, también llamado sistema de tratamiento de errores. También veremos los primeros ejemplos sencillos sobre la gestión de excepciones.

Vamos a ver tres de las palabras reservadas para tratamiento de excepciones:

- Try.

- Catch.

- Finally.

**BLOQUE TRY**

**try {**

     System.out.println(“bloque de código donde pudiera saltar un error es este”);

**}**

**BLOQUE CATCH**

**catch (Exception e) {**

     System.out.println(“bloque de código donde se trata el problema”);

**}**

**BLOQUE FINALLY**

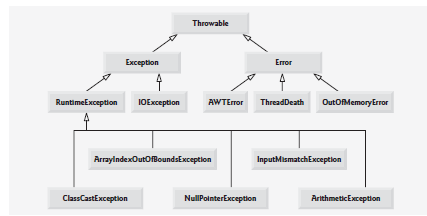
**finally {**

     System.out.println(“bloque de código ejecutado siempre”);

**}**

**EJEMPLO EXCEPCIONES**

Jerarquia de manejo de Excepciones, todos heredan de la clase Exception



try

{

*instrucciones*

*instrucciones para adquirir recursos*

} // fin del bloque try

catch ( *UnTipoDeExcepción excepción1* )

{

*instrucciones para manejar excepciones*

} // fin de bloque catch

**...**

catch ( *OtroTipoDeExcepción excepción2* )

{

*instrucciones para manejar excepciones*

} // fin de bloque catch

finally

{

*instrucciones*

*instrucciones para liberar recursos*

} // fin de bloque finally

EJEMPLO LANZA EXCEPCION UsoDeExcepciones

**Limpieza de la pila**

Cuando se lanza una excepción, pero no se atrapa en un alcance específi co, la pila de llamadas a métodos se

“limpia” y se hace un intento de atrapar (catch) la excepción en el siguiente bloque try exterior. A este proceso se le conoce como **limpieza de la pila**.

Ejmeplo LIMPIEZA DE PILA

**Los métodos getMessage, getStackTrace y printStackTrace de Throwable**

Ejmeplo METODOSEXCEPCIONES

**13.10 Excepciones encadenadas**

Las **excepciones encadenadas** permiten

que un objeto de excepción mantenga la información completa sobre el rastreo de la pila. En la fi gura 13.8 se

demuestran las excepciones encadenadas.

EJEMPLO ExcepcionesEncadenadas

DEBER

Escribir un programa que pida al usuario inicializar un arreglo de números enteros. El programa debe pedir el número de elementos que el usuario desea ingresar. Y luego debe pedirle que ingrese los elementos para inicializar el arreglo. Finalmente, el programa debe permitir al usuario indicar una posición en el arreglo y debe devolver el elemento en dicha posición.

El programa debe manejar todas las posibles excepciones que puedan ocurrir durante su ejecución y mostrar los mensajes apropiados, por ningún motivo el programa debe terminarse por algún error.

**EJERCICIO 2**

**Escribir un programa que le pida al usuario ingresar dos números para realizar una división. El programa debe manejar de manera correcta las posibles excepciones y mostrar los mensajes correspondientes. El programa debe manejar correctamente los siguientes casos:**

**Numerador: 100**

**Denominador: 0**

**Numerador: 100**

**Denominador: Hola**

Recibir numerador y denominador

Devolver el cociente

Scanner explorador = new Scanner (System.in);

Int numerador = explordor.nextInt();

Int resultado = cociente(numerador,denominador);

Imprimir reultado

Usar los siguientes valores:

100,7 100,0 100,hola