<codoa codo/>

Codo a Codo inicial
Clase

BA

Les damos la bienvenida

Vamos a comenzar a grabar la clase

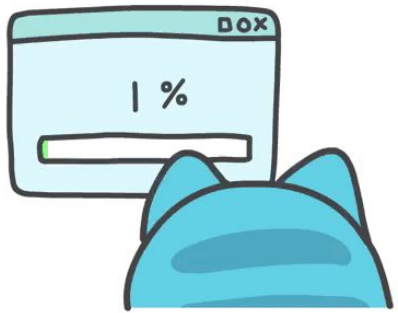
Write Once, Run Anywhere

(Escríbelo una vez, ejecútalo en cualquier lugar)

Introducción a las Excepciones

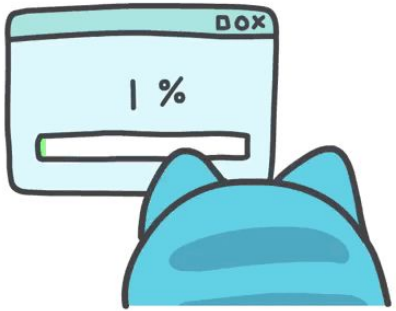
Introducción a las excepciones

- Un problema importante en el desarrollo de software es la **gestión de condiciones de error**; no importa lo eficiente que sea el software ni su calidad, **siempre aparecerán errores por múltiples razones**; surgirán, por ejemplo, imprevistos de programación de los sistemas operativos, agotamiento de recursos, etcétera.



Introducción a las excepciones

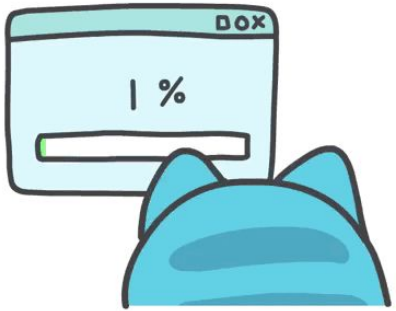
- **Las excepciones** son el **medio que ofrecen algunos lenguajes de programación para tratar situaciones anómalas** que pueden suceder cuando ejecutamos un programa.



Introducción a las excepciones

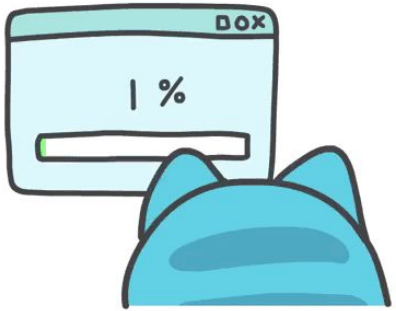
Algunos casos de **situaciones anómalas** que se pueden citar, son, por ejemplo:

- Invocar a un método sobre un objeto “null”,
- Dividir un número por “0”,
- Intentar abrir un fichero que no existe para leerlo, quedarnos sin memoria en la JVM al ejecutar,
- Sufrir un apagón eléctrico al ejecutar nuestro código.



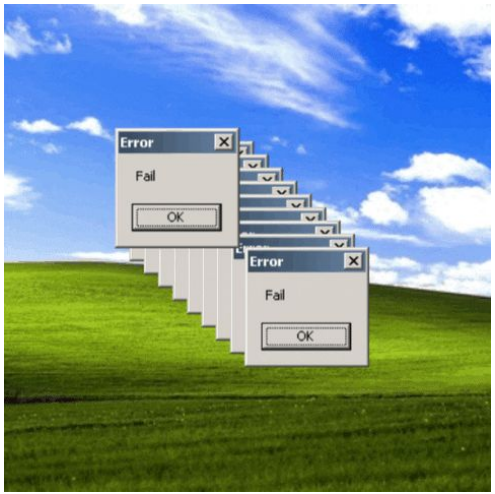
Introducción a las excepciones

- **Java adopta el manejo de excepciones** como mecanismo para el tratamiento de **situaciones anómalas**;
- Normalmente **el sistema interrumpe el flujo normal de ejecución del programa cuando se produce una excepción**;
- Java permite al programador **intentar la recuperación** de estas condiciones y **decidir si continuar o no la ejecución del programa.**



Concepto de excepciones

- Se denomina **excepción al evento desencadenado** a partir de **una situación que no se puede resolver e interrumpe el flujo normal de un programa**.
- **La forma en que el programador trate la misma** es lo que se conoce generalmente como **manejo o gestión de la excepción**.
- Es una condición de **error en tiempo de ejecución** (es decir **cuando el programa ya ha sido compilado y se está ejecutando**).



Errores de sintaxis y excepciones, a no confundir...

- Los **errores de sintaxis** son detectados durante la **compilación, estos errores son salvados antes de poner en ejecución un programa, es decir un error de estos tiene vuelta atrás**.
- Pero **las excepciones pueden provocar situaciones irreversibles**, su control debe hacerse **en tiempo de ejecución** y eso presenta un gran problema. **Una vez que se lanzó una excepción en tiempo de ejecución no hay vuelta atrás.**

Las excepciones se diseñan y se lanzan

- Las excepciones no son automáticas.
- Es el programador quién se anticipa a posibles errores dentro de un método y ante ello decide cuando lanzar una excepción.
- Es trabajo de equipo y gabinete. Las excepciones son planificadas dentro del proyecto y como parte del mismo.

Hay que saber lanzar...

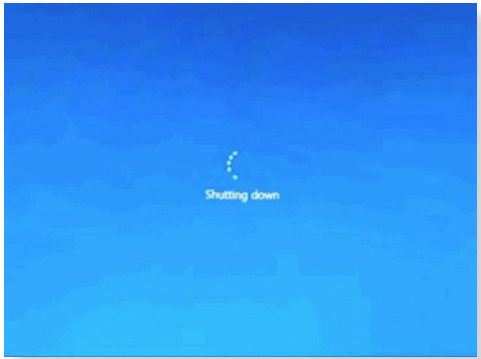
- En Java **se puede preparar previamente el código susceptible a provocar errores de ejecución**
- De modo que **si ocurre una excepción, el código es lanzado** *(throw-throws)* a una **determinada rutina** previamente **preparada por el programador**, que permite **manipular esa excepción**.
- **Si la excepción no fuera capturada**, la ejecución del **programa se detendría** irremediablemente 😢
- Se trata de lanzar excepciones y lanzarlas bien.

Paquete de excepciones en Java

- En Java hay **muchos tipos de excepciones** (de operaciones de entrada y salida, de operaciones irreales, matemáticas, etc).
- El paquete **java.lang.Exception y sus subpaquetes contienen todos los tipos de excepciones.**
- Recordemos que la biblioteca **java.lang se importa automáticamente.**

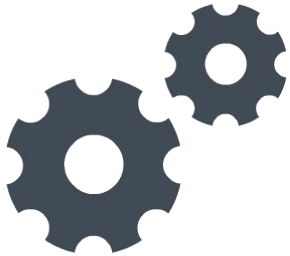
Los errores y las excepciones

- **Los errores los produce el sistema y son incontrolables para el programador** mientras que las **excepciones son fallos más leves, y más manipulables.**
- Hay una clase, la **java.lang.Error** y sus subclases que sirven para definir los **errores irrecuperables más serios**.
- **Esos errores causan parada en el programa**, por lo que el programador no hace falta que los manipule.



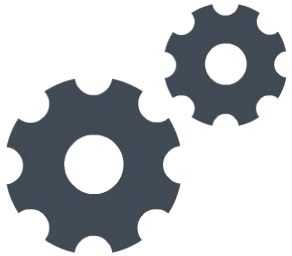
Manejo de excepciones

- El manejo de errores usando excepciones **no evita errores**, **solo permite su detección y su posible reparación.**
- **Una vez que se levanta una excepción, esta no desaparece aunque el programador lo ignore,** una condición de excepción se debe reconocer y manejar porque, en caso contrario, **se propagará dinámicamente hasta alcanzar el nivel más alto de la función**; si también falla el nivel de función más alto, **la aplicación termina sin opción posible.**



¿Cómo trabaja una excepción?

- Cuando se produce un error **se genera un objeto asociado a esa excepción.**
- Este objeto es de la clase Exception o de alguna de sus herederas por ejemplo *AritmethicException*.
- Este **objeto se pasa al código** que se ha definido para manejar la excepción.
- **Dicho código puede manipular las propiedades del objeto Exception**.



Qué permite el manejo de excepciones

En general, el mecanismo de excepciones en Java permite:

a) detectar errores con posibilidad amplia de recuperación;

b) limpiar errores no manejados, y

c) evitar la propagación sistemática de errores en una cadena de llamadas dinámicas.

En que deriva el manejo de excepciones

En resumen, las alternativas para el manejo de errores en programas son:

1. **Terminar** el programa.
2. Devolver **un valor** que represente el error.
3. **Llamar a una rutina de error** proporcionada por el usuario.

Precaución

El manejo de errores usando excepciones **no evita errores**, solo permite su detección y la posible recuperación del flujo normal de ejecución.

Mecanismo del manejo de excepciones en Java

El modelo de un mecanismo de excepciones consta fundamentalmente de **cinco palabras reservadas**: ***try, throw, throws, catch y finally.***

- **try** es un bloque para detectar excepciones,
- **catch** es un manejador para capturar excepciones de los bloques try,
- **throw** es una expresión para levantar (raise) excepciones explícitamente **dentro de un método** o bloque de código.
- **throws** indica las excepciones que puede elevar un método, **se utiliza en la cabecera o firma del método** para declarar **todas las excepciones que podría lanzar** el método durante su ejecución.
- **finally** es un bloque opcional situado después de los catch de un try.

Pasos del modelo try-catch-finally

- Los pasos del modelo son:

1. **Establecer un conjunto de operaciones** para **anticipar errores;** esto se realiza en un **bloque try**.
   Cuando una rutina encuentra un error, **lanza una excepción**; el lanzamiento **(throwing) es el acto de levantar una excepción.**
2. Se activa el **bloque catch anticipando** el error y **capturando** la excepción lanzada, para propósitos de **limpieza o recuperación**.
3. El mecanismo de excepciones se completa con:
   • Un **bloque finally** que, si se especifica, **siempre se ejecuta al final** de un try-catch.

Captura de las excepciones bloque catch

Cuando **una excepción se lanza desde un bloque try**, se pone **en marcha** el **mecanismo de captura**; la **excepción será capturada por un catch** que sigue al bloque try.

Una cláusula catch consta de tres elementos:

- La palabra reservada catch,
- La declaración de un único argumento que será un objeto para manejo de excepciones (declaración de excepciones)
- Un bloque de sentencias.

Si la cláusula catch se selecciona para manejar una excepción, entonces se ejecutará su bloque de sentencias.

La **clase base** de la cual derivan las excepciones **es Exception**; por ello el manejador `catch(Exception e)` **captura cualquier excepción lanzada**.

Cláusula finally

Java proporciona la posibilidad de definir **un bloque de sentencias que se ejecutarán siempre**, ya sea que **termine el bloque try** normalmente **o se produzca una excepción.**
Esta cláusula **es opcional**, si está presente debe situarse **después del último catch** del bloque try; incluso, se permite que un bloque try **no tenga catch asociados si tiene el bloque definido por la cláusula finally.**

Bloques try-catch-finally

- El esquema siguiente indica cómo se especifica un bloque try-catch-finally:

```
try
{
// sentencias
// acceso a archivos(uso exclusivo ...)
// peticiones de recursos
}
catch(Excepcion1 e1) {...}
catch(Excepcion2 e2) {...}
finally
{
// sentencias
// desbloqueo de archivos
// liberación de recursos
}
```

Desafío de Clase I

Diseño, gestión y lanzamiento de una excepción

- Realizaremos una función división, la dividiremos por 0 y observaremos el resultado.
- Agregaremos lanzamientos de excepciones y bloques try catch que gestionen los errores y permitan una salida más correcta del programa.
- *Clase Division y ExDivision del repo.*

Desafío de Clase II

- Mediante una función accederemos a un array.
- Esta función arrojará una excepción del tipo ArrayIndexOutOfBoundsException en caso de querer acceder a un index no existente.
- El lanzamiento throws se realizará desde el método main.
- El error será gestionado con el bloque try-catch-finally en el método main.
- *Clase OutBoundIndex del repo*

Desafío para la casa

- Realizaremos dos ingresos numérico por teclado, y calcularemos una multiplicación
- Gestionaremos la excepción de tipo del objeto Scanner desde el método main.
- Gestionaremos la excepción mediante bloques try catch para que nos que permitan una salida más correcta del programa.

No te olvides de dar el presente

Recordá:

- Revisar la Cartelera de Novedades.
- Hacer tus consultas en el Foro.
- Realizá los ejercicios obligatorios.

Todo en el Aula Virtual.