El futuro digital es de todos | MinTIC

UNIVERSIDAD EL BOSQUE | MisiónTIC 2022

Ciclo 2

Semana 4

Data Access Object (DAO) y Data Transfer Object (DTO), manejo de excepciones y documentación de código

Lectura 2 - Manejo de las excepciones

Convenio de asociación no. 777 de 2021 suscrito entre el Fondo Único de Tecnologías de la Información y las Comunicaciones y la Universidad el Bosque.

Manejo de las excepciones

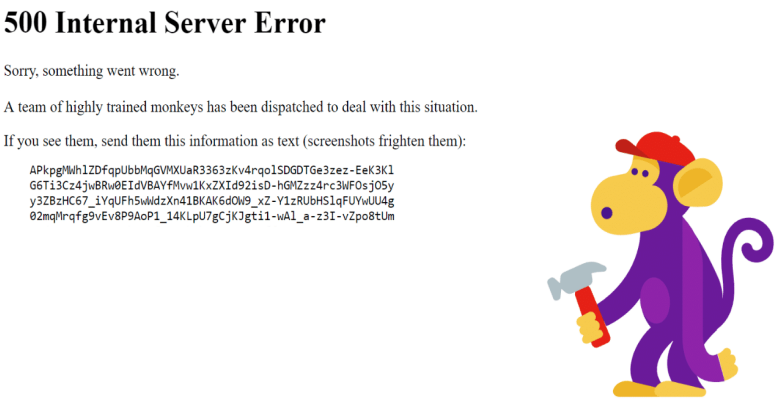
Se denomina excepción a aquella indicación de que se produjo un error en la aplicación y como su nombre lo señala se da cuando la ejecución de un método no finaliza de manera correcta y por el contrario termina de manera excepcional por consecuencia de una situación inesperada, en donde si no se le da el manejo correcto a este, el programa terminará abruptamente su ejecución.

Este término es una condición excepcional que cuando ocurre cambia el flujo normal del programa que está en ejecución. Estos errores pueden ser generados por la lógica, sintaxis o entes externos como vimos anteriormente e incluso generados por los objetos que manifiestan algún estado no predicho o una condición que no puede manejar. Si vemos las excepciones desde un punto de vista de cómo se tratará la excepción, se podrían dividir en dos grupos:

- Excepciones marcadas: Su captura es obligatoria.
- Excepciones no marcadas: Excepciones en tiempo de ejecución. No es obligatorio capturarlas.

El código que se encarga de ejecutar alguna acción cuando se aparece una excepción se llama manejador de excepciones y lo que hace es capturar la excepción lanzada.

1. Manejo de excepciones



En el momento en el que un programa deja de funcionar durante la ejecución de un método, se detecta un error, crea un objeto de una clase especial que lo representará y que debe incluir toda la información del problema, como el punto donde se produjo, la causa del error, etc. Luego, lanzará este objeto, para que sea tomado y se decida cómo solucionar el error, caso contrario el programa termina y en la consola de ejecución saldrá la información contenida en el objeto que representaba el error. Este objeto se conoce como una excepción.

Es un proceso que se repite hasta que alguien le dé un manejo a la excepción o esta se identifique, se tome antes que el programa deje de funcionar y se realice una acción para recuperar el error o por lo menos muestre un mensaje claro y comprensible al usuario, o hasta que el programa completo se detenga. Si dentro de una función aparece una excepción y la función no le da manejo, la excepción irá hacia la función que la invocó, y si esta tampoco le da manejo, la excepción continuará propagándose hasta llegar a la función inicial del programa, donde si finalmente tampoco se toma se interrumpirá la ejecución del programa.

Para el manejo de excepciones los lenguajes de programación suelen tener palabras reservadas, que permiten manejar las excepciones que puedan aparecer y así tomar acciones de recuperación para evitar que el programa finalice abruptamente, o ejecutar alguna acción adicional antes que esto suceda

En Phyton por ejemplo encontramos palabras como **try, except** y **finally** que son usadas para el manejo de excepciones con bloques que usan estas sentencias. En el bloque try se coloca el código que puede generar una excepción, el bloque except captura la excepción y permite procesarla visualizando por ejemplo los mensajes adecuados para el usuario. Como dentro de un bloque try se podrían producir excepciones de diferente tipo, se puede usar varios bloques except, así cada uno puede capturar un tipo distinto de excepción. Se debe colocar después de la sentencia except el nombre de la excepción que se quiere capturar. Un mismo bloque except puede capturar varias excepciones y aunque después de un bloque try pueden existir varios bloques except, se ejecutará a lo sumo, uno de ellos. Al final se ubica un bloque finally donde se escriben las sentencias de finalización o acciones de limpieza, esta se ejecuta siempre, haya o no una excepción. Si se coloca un bloque except, no se necesita el finally, y se puede tener un bloque try sólo con finally y sin except.

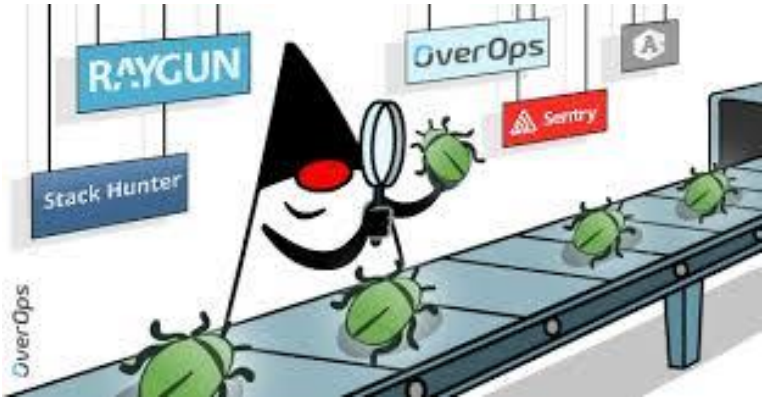
En java, la herramienta que permite capturar una excepción es el bloque try – catch – finally, donde try al igual que en el lenguaje anterior se usa para el código que se va a ejecutar dentro del bloque y que se prevé genere una excepción, catch es ahora el que captura y maneja las excepciones o sea el código que se ejecutará cuando se produzca la excepción, y finalmente el bloque finally siempre será ejecutado haya o no excepciones, y puede usarse por ejemplo para la creación de logs o seguimientos de lo que se produce. Esta palabra reservada finally define un tercer bloque de código en el manejador de excepciones que le indica al programa que

instrucciones ejecutar, esto de forma independiente a los bloques try y catch, por tanto, si el bloque try se ejecuta sin errores, pasa al bloque finally y si se genera un error, se ejecutar el bloque catch y el código del bloque finally.

También existen unas excepciones ya predefinidas en algunos programas como Java, cuyo nombre nos indican la condición de error, por ejemplo:

- ArithmeticException: Excepciones aritméticas casi siempre por resultados de una división por 0.
- NullPointerException: Se produce al acceder a una variable o método antes de definirse.
- IncompatibleClassChangeException: al cambiar una clase que tiene referencias en otros objetos, específicamente cuando los objetos no han sido recompilados.
- ClassCastException al intentar convertir un objeto a otra clase no válida.
- OutOfMemoryException: al intentar crear un objeto con new falla, pero por memoria.
- NoClassDefFoundException: al hacer referencia a una clase que el sistema no encuentra.
- ArrayIndexOutOfBoundsException:cuando se trata de acceder a un elemento de un arreglo más allá de los límites definidos para el mismo.
- UnsatisfiedLinkException al acceder a un método nativo inexistente.

2. Procesamiento y propagación de excepciones

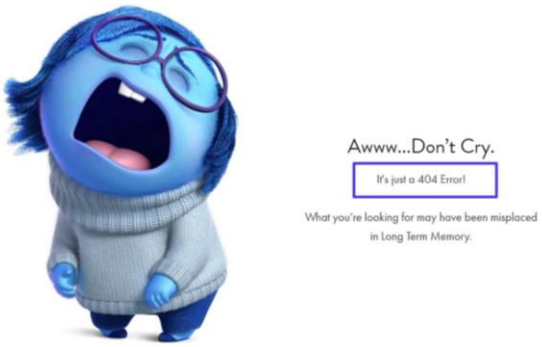


Una parte importante es la captura de una excepción. En esta parte se debe tener en cuenta dejar un registro o constancia de la ocurrencia de la excepción como por ejemplo en archivos logs o a través de mensajes de pantalla, lo importante es dar alguna información del contexto en que se generó la excepción; información como el tipo de excepción ocurrida, momento en que se generó, llamadas previas a la excepción, etc. para con eso facilitar el diagnóstico en caso de corrección de la aplicación y evitar que la excepción vuelva a aparecer.

Otra parte es la propagación de la excepción. Las excepciones se pueden tratar dentro de un bloque manejador de excepciones, o no, en cuyo caso se tiene que indicar explícitamente a través del método. Aquí el método superior debe incluir los bloques try y catch o pasar de nuevo la excepción, propagando así la excepción de un método a otro hasta que llegue al último método del programa o sea el método main.

También se puede dar que no se desee propagar la excepción tal cual se capturó, sino se envíe una excepción diferente, más significativa para el que llamó la función actual y que contiene cierta información de contexto.

3. Diseño de excepciones



Al desarrollar programas nos podemos encontrar con el caso en el que las excepciones que se encuentren definidas en el API de la aplicación, las predefinidas, no sean las que se requieran en ese momento o entidad particular, por tanto, se necesitaran excepciones propias del negocio, por ejemplo, en el caso de una aplicación para una entidad bancaria, en donde se requiere una excepción para cuando el saldo de la cuenta sea insuficiente para realizar una operación como retiro o transferencia.

En este caso se pueden crear clases que tengan un comportamiento como excepciones, en cuyo caso solo se necesita heredar de Exception. Estas se invocan y se pueden lanzan igual que las excepciones del API, pero con información más precisa del problema pues se está programando a la medida.

En el caso de la entidad bancaria, se podría crear una clase de excepción propia como SaldoInsuficienteException, donde se tendría la clase Cuenta, que podría lanzar esta excepción creada en el momento que el cliente intenta retirar un monto mayor a su saldo en la cuenta.

Las excepciones se pueden crear cuando el programa hace algo prohibido o ilegal que es lo que suele pasar, o el programa explícitamente podría generar una excepción al ejecutar una sentencia y extendiendo la clase Exception para dar la funcionalidad extra que se necesita. A diferencia de las excepciones predefinidas que se ejecutan en tiempo de ejecución y por tanto son excepciones irrecuperables, las excepciones que se generan explícitamente, pueden ser más recuperables, por ejemplo, si un archivo no se puede abrir se le podría solicitar al usuario indicar otro archivo.

Los métodos que lanzan excepción deben también capturarlas y declararlas como parte de la interface del método. Tengamos en cuenta que las excepciones no disminuyen el trabajo de control de errores sino una ventaja importante es que se puede tener localizado este control de errores sin tener que controlar los miles posibles valores de retorno.

Semana 4 | Data Access Object (DAO) y Data Transfer Object (DTO), manejo de excepciones y documentación de código.