Tarea Extraclase 2

Isaac Barrios Campos, Luis Diego García Rojas
Isriom@estudiantec.cr guisogarcia1010@estudiantec.cr
Área académica de Ingeniería en Computadores
Instituto Tecnológico de Costa Rica

ÍNDICE

I. EXCEPCIONES

I-A. ¿Que son las excepciones?

Una excepción es una alerta que se da durante la ejecución de un programa para indicar que hubo problema. Las excepciones, como su nombre lo indica, se producen cuando la ejecución de un método no termina correctamente, sino que termina de manera excepcional como consecuencia de una situación no esperada. [1]

I-B. ¿Para que sirven?

- Permite mantener limpio y ordenado el código
- Impide la pérdida de tiempo y de recursos a problemas que son fáciles de evitar con el manejo de excepciones.
- Se usan para visualizar situaciones inusuales durante la ejecución del código.
- Admite la validación de lo que el usuario ingresa sin que se vaya a caer nuestro programa.
- Permite controlar todos los posibles errores para evitar la elegibilidad de código.

I-C. ¿Cómo se hace el manejo de errores en lenguajes sin excepciones?

El manejo de errores en lenguajes sin excepciones es un tanto complicado ya que hay que asegurarse de enviar la información de la forma correcta y en el entorno correcto. Esto se puede realizar a través de apuntadores que nos ayudan a observar donde están los errores específicos. Para ello se utilizan librerías de sistema que tienen funciones disponibles para tratar dichos errores. Para poder verificar las excepciones se utilizan condicionales que permitan identificar si existe o no el objeto que causa error, e intente a través de funciones resolver el problema y mandar el mensaje al usuario.

I-D. Buenas prácticas a la hora de utilizar excepciones

Según el sitio web Club de la Tecnología [2], se deben de tomar en cuenta los siguientes aspectos:

- Limpia los recursos en el bloque Finally o utiliza la sentencia de Try con recursos: Sucede con recurso que utiliza un recurso en el bloque try, como un InputStream, que necesita cerrar después. Un error común en estas situaciones es cerrar el recurso al final del bloque try.
- Utilizar Excepciones específicas: siempre intente encontrar la clase que mejor se adapte a su evento de excepción.
- Documente las excepciones que especifique: asegúrese de agregar una declaración @throws a su Javadoc y describir las situaciones que puedan causar la excepción además de tomar en cuneta al autor.
- Lance excepciones con mensajes descriptivos: debe describir el problema con la mayor precisión posible y proporcionar la información más relevante para comprender el evento que causo la excepción.
- Captura la excepción más específica primero: Siempre capture primero la clase de excepción más específica y agregue los bloques de captura menos específicos al final de su lista.
- No capture Throwable: Si usa Throwable en una cláusula cath, no solo capturará todas las excepciones; también detectará todos los errores.
- No ignore excepciones
- No registre una excepción para luego lanzarla: Puede ser intuitivo registrar una excepción cuando ocurrió y luego volver a lanzarla para que la persona que la llame pueda manejarla adecuadamente. Pero escribirá múltiples mensajes de error para la misma excepción.
- Envuelva la excepción sin consumirla: A veces es mejor capturar una excepción estándar y envolverla en una personalizada.

I-E. ¿Cómo puede crear sus propias excepciones?

Muchas veces durante la elaboración de programas se requiere la creación de nuevas excepciones personalizadas. La implementación de una excepción personalizada es un

tanto sencilla solo necesita extender Exception de la sigueiente manera:

- Se coloca la llamada a la función susceptible de producir una excepción en un bloque try...catch
- En dicha función se crea mediante new un objeto de la clase Exception o derivada de ésta
- Se lanza mediante throw el objeto recién creado
- Se captura en el correspondiente bloque catch
- En este bloque se notifica al usuario esta eventualidad imprimiendo el mensaje asociado a dicha excepción, o realizando una tarea específica.

Además de eso, debes seguir algunas buenas prácticas, que hacen que su código sea más fácil de leer y su API más fácil de utilizar, como las que se detallaron anteriormente.

II. Frameworks para logs

II-A. ¿Que son?

Los frameworks de loggin permiten estandarizar el proceso de registro de una aplicación, es decir invoca una API del sistema de archivos para escribir un archivo .txt de forma predeterminada.

II-B. ¿Cuáles son los más populares?

- Log4J
- Java Logging API
- tinylog
- Logback
- Apache Commons Logging
- SLF4J

El framework para log escogido para esta tarea fue el de Log4J

II-C. ¿Porqué Log4J?

Este es el framework de log más utilizado ya que fue uno de los primeros que existieron para este fin, además cuenta una versión específica para cada tipo de proyecto.

II-D. Anexos

https://github.com/isriom/Tarea-chat.git

Referencias

[1] Universidad de Los Andes,«Manejo de las Excepciones» [En línea]. Available:https://universidad-de-los-andes.gitbooks.io/fundamentos-de-programacion/content/Nivel4/5ManejoDeLasExcepciones.html [Último acceso: 19 10 2020].

[2] G. Briceño, «Club de Tecnología,» 13 11 2017. [En línea]. Available: https://www.clubdetecnologia.net/blog/2017/java-buenas-practicas-para-el-manejo-de-excepciones/. [Último acceso: 20 10 2020].