Computación Tolerante a Fallas

Sección D06

Otras herramientas para el manejo de errores

Fernández Venegas David Guadalupe

Código: 216437352

Profesor: López Franco Michel Emanuel

Objetivo:

Conocer sobre las distintas herramientas utilizadas para el manejo de errores en la programación.

Introducción:

En el manejo de errores se busca actuar de manera anticipada para poder controlar situaciones inesperadas en donde se interrumpe el flujo normal de una aplicación o programa al ser ejecutados. El objetivo es poder identificar, capturar y gestionar estos errores de manera que el programa pueda recuperarse o finalizar de manera controlada, brindando a los usuarios una experiencia más satisfactoria y evitando fallos catastróficos y evitando que los errores detengan la ejecución del programa por completo.

Desarrollo:

A continuación se presentará diferentes técnicas usadas en el desarrollo de programas, utilizadas para detectar y abordar errores.

- Asserts

Las afirmaciones (asserts) son declaraciones que verifican si una determinada condición es verdadera o falsa. Se utilizan principalmente para realizar verificaciones de control durante el desarrollo y detener la ejecución si una condición no se cumple como se esperaba. Los asserts son valores booleanos colocados en punto específico de un programa, los cuales serán verdaderas hasta que se demuestre lo contrario. Este tipo de sentencias se utilizan como ayuda en las correcciones de un programa. Si la expresión contenida dentro del mismo es False, se lanzará una excepción (AssertionError).

Formas de utilizar los asserts:

- Precondicion: Colocada al inicio de una sección de código, determinando el conjunto de sentencias bajo las cuáles se espera que el código sea ejecutado.
- Postcondicion: Colocada al final, describiendo la sentencia esperada al final de la ejecución.
- Class invariants: para validar el estado de una clase según está definido en su contrato, siempre se debe cumplir independientemente de las operaciones que se realicen.
- Código no alcanzable en tiempo de ejecución: partes del programa que se espera que no sea alcanzable, como cláusulas else o default en sentencias switch.

En donde no utilizar asserts:

- No se deben usar para comprobar argumentos en métodos públicos: los asserts pueden habilitarse o deshabilitarse, comprobar los argumentos se considera parte de las responsabilidades del método y su especificación.
- No se deben usar para realizar tareas: ya que los asserts pueden deshabilitarse, las tareas dejarían de ejecutarse y de proporcionar la funcionalidad del programa.

- Try-Exception

Una excepción es la indicación de que se produjo un error en el programa. Las excepciones, se producen cuando la ejecución de un método no termina correctamente, sino que termina de manera excepcional como consecuencia de una situación no esperada.

Cuando se produce una situación anormal durante la ejecución de un programa (por ejemplo se accede a un objeto que no ha sido inicializado o tratamos de acceder a una posición inválida en un vector), si no manejamos de manera adecuada el error que se produce, el programa va a terminar abruptamente su ejecución. Decimos que el programa deja de funcionar y es muy probable que el usuario que lo estaba utilizando ni siquiera sepa qué fue lo que pasó.

Cuando durante la ejecución de un método la computadora detecta un error, crea un objeto de una clase especial para representarlo, el cual incluye toda la información del problema, tal como el punto del programa donde se produjo, la causa del error, etc. Luego, "dispara" o "lanza" dicho objeto, con la esperanza de que alguien lo atrape y decida como recuperarse del error. Si nadie lo atrapa, el programa termina, y en la consola de ejecución aparecerá toda la información contenida en el objeto que representaba el error.

La sentencia try funciona de la siguiente manera:

1. Primero, se ejecuta la cláusula try (la(s) línea(s) entre las palabras reservadas try y la except).
2. Si no ocurre ninguna excepción, la cláusula except se omite y la ejecución de la cláusula try finaliza.
3. Si ocurre una excepción durante la ejecución de la cláusula try, se omite el resto de la cláusula. Luego, si su tipo coincide con la excepción nombrada después de

la palabra clave except, se ejecuta la cláusula except, y luego la ejecución continúa después del bloque try/except.

4. Si ocurre una excepción que no coincide con la indicada en la cláusula except se pasa a los try más externos; si no se encuentra un gestor, se genera una unhandled exception (excepción no gestionada) y la ejecución se interrumpe con un mensaje como el que se muestra arriba.
5. Si no sabemos que excepción hay que saltar, podemos utilizar la cláusula Exception, que controla cualquier tipo de excepcion.
6. Además, podemos utilizar el bloque de else, que va después de try y except, para ejecutar si no ha ocurrido ninguna excepción.
7. Además de los bloques de try, except y else se puede añadir el bloque finally. En donde se ejecutar siempre, sin importar si hubo una excepción. Utilizado normalmente como acción de limpieza.

- Diferencia entrer Assert y Try-Exception

Las excepciones se encargan de hacer que el programa sea robusto controlando las situaciones inesperadas pero posibles, los assert se encargan de que el programa sea correcto. Los assert deberían ser usados para asegurar algo, mientras que las excepciones deberían usarse para comprobar algo que podría ocurrir. Un assert termina la ejecución (ya que no se suele capturar la excepción que se produce) mientras que una excepción permite al programa continuar con la ejecución. Los asserts no deben ser sustitutos de condiciones de validación que debería hacer el programa en métodos públicos de una clase. Los assert son una herramienta en tiempo de desarrollo, las excepciones además son una herramienta para la ejecución en producción.

Conclusión:

El manejo de errores en la programación es crucial para garantizar el correcto funcionamiento de las aplicaciones y mejorar la experiencia del usuario. A lo largo de este trabajo, se han explorado algunas herramientas utilizadas para abordar los errores de manera efectiva. El comprender y hacer uso de estas es de gran importancia en el desarrollo de programas, por lo que es de vital importancia tener conocimiento de estas para ponerlas en práctica de manera efectiva.

Bibliografía:

- *Assertions. (Afirmaciones) un primer acercamiento*. (s. f.). SG Buzz. https://sg.com.mx/content/view/562
- Picodotdev. (2015, 14 febrero). La palabra clave assert de java Y un ejemplo. *Blog Bitix*. https://picodotdev.github.io/blog-bitix/2015/02/la-palabra-clave-assert-de-java-y-un-ejemplo/#goog_rewarded
- *Tratamiento de errores. sentencia Try-Except — Documentación de curso de Python para Astronomia - 20191128*. (s. f.). https://research.iac.es/sieinvens/python-course/errores_depuracion.html
- De Los Andes, U. (s. f.). *Manejo de las excepciones · Fundamentos de programación*. https://universidad-de-los-andes.gitbooks.io/fundamentos-de-programacion/content/Nivel4/5_ManejoDeLasExcepciones.html
- *Excepciones en Python*. (s. f.). El Libro De Python. https://ellibrodepython.com/excepciones-try-except-finally#:~:text=Las%20excepciones%20en%20Python%20son,cuando%20se%20produce%20un%20error.