UNIVERSIDAD DE LA FRONTERA

Evidencia de la semana - Caso de los Números adyacentes

Alumnos:

1. Antonio Acuña Dassé
2. Antonia Córdova Reyes
3. Alex Rojas Inostroza

Profesor: Samuel Eduardo Sepúlveda Cuevas

Fecha: 30/09/2024

Fecha de entrega: 02/10/2024

Institución: Universidad de la Frontera, Temuco, Chile

Introducción: El presente reporte documenta la solución al caso de encontrar el mayor producto de números adyacentes en un arreglo de enteros. La implementación de la solución se realizó siguiendo buenas prácticas de desarrollo, gestionando los errores mediante el uso de excepciones y asegurando la calidad del código a través de pruebas unitarias implementadas con JUnit 5.

Descripción del Caso: El objetivo del caso es desarrollar un método llamado productoAdyacentes que reciba como parámetro un arreglo de enteros y devuelva el mayor producto de números adyacentes que se pueda obtener en dicho arreglo. Por ejemplo, dado el arreglo {1, -4, 2, 2, 5, -1}, el mayor producto de números adyacentes es el de los números 2 y 5, lo que da como resultado 10.

Análisis del Caso: Para resolver el problema, se consideraron los siguientes aspectos: 1. Parámetros de entrada: El método recibe un arreglo de enteros (int[]). 2. Valor de retorno: El método retorna un entero que representa el mayor producto de números adyacentes. 3. Gestión de errores: Se validaron casos como arreglo vacío, arreglo con un solo elemento y arreglo con ceros. 4. Implementación de pruebas unitarias: Se definieron casos de prueba con diferentes configuraciones de arreglos para garantizar el correcto funcionamiento del método.

Implementación de la Solución: La solución se implementó en la clase `ProductosAdyacentes` con el método `encontrarMayorProductoAdyacente`. El método realiza la validación del arreglo y recorre los elementos para calcular el producto de cada par de números adyacentes. Se consideraron los siguientes puntos: - Validación de la entrada para asegurar que el arreglo tenga al menos dos elementos. - Manejo de excepciones para gestionar errores en la entrada del usuario. - Cálculo del mayor producto adyacente a través de un recorrido del arreglo.

Diseño e Implementación de las Pruebas Unitarias: Se diseñaron e implementaron pruebas unitarias usando JUnit 5 para validar el comportamiento del método `encontrarMayorProductoAdyacente`. Las pruebas abarcan los siguientes escenarios: 1. Arreglos con números positivos. 2. Arreglos con números negativos y positivos combinados. 3. Arreglos con números negativos. 4. Arreglos con ceros. 5. Arreglos con dos elementos. 6. Arreglos con elementos iguales. Cada prueba se ejecutó exitosamente, validando que el método retorne el valor esperado para cada caso.

Diseño e Implementación de la Gestión de Excepciones

Para gestionar los errores en tiempo de ejecución, se implementaron las siguientes validaciones y excepciones:

1. **IllegalArgumentException**: Se lanza cuando el arreglo tiene menos de dos elementos, lo cual impide calcular el producto adyacente.
2. **InputMismatchException**: Se lanza cuando el usuario ingresa un valor no numérico al momento de introducir los elementos del arreglo.
3. **Validación de arreglos vacíos o nulos**: Se verifica que el arreglo no esté vacío o que no contenga solo ceros, ya que esto también podría generar errores en el cálculo.

Estas excepciones permiten manejar los errores de manera adecuada y evitan que el programa falle inesperadamente. Además, se asegura de que el usuario reciba mensajes claros para corregir su entrada en caso de error.

Resultados: El desarrollo del método `encontrarMayorProductoAdyacente` se completó con éxito. Se validaron todos los casos de prueba definidos y se gestionaron adecuadamente los errores potenciales. El uso de JUnit 5 facilitó la ejecución de pruebas y la identificación de posibles fallos en la lógica del programa.

Conclusiones: El método `encontrarMayorProductoAdyacente` fue desarrollado siguiendo las buenas prácticas de programación y aplicando las principales técnicas de gestión de errores y pruebas unitarias. Se lograron resultados satisfactorios que garantizan la correcta funcionalidad del método para diferentes configuraciones de arreglos de números enteros.

Anexo: Link al Repositorio Repositorio GitHub:
https://github.com/Ant0ni023/Evidencia05NumerosAdyacentes.git