Caso de los números adyacentes

UNIVERSIDAD
DE LA FRONTERA

Estudiantes:

- **Benjamín Fernández**
- **Jesús Tapia**
- **Yoandri Villarroel**

https://github.com/JesusTapiaMartin/NumerosAdyacentes.git

Introducción

Como grupo crearemos un programa para resolver un caso llamado “Caso de los números adyacentes” como actividad de clase semanal donde el desarrollo de la solución está basada en métodos y aplicar las buenas prácticas que se ha indicado en varias clases de Programación Orientada a Objetos.

Además de los métodos y las buenas prácticas hay que poner en práctica la utilización de pruebas unitarias utilizando el Junit y la gestión de errores utilizando excepciones.

Las herramientas que utilizaremos para esta actividad son las siguientes:

- IntelliJ IDEA.
- Junit.
- Apuntes de clases.
- Apuntes externos (videos o páginas).
- Repositorio Github.
- API de Java.

Descripción del caso

En el caso presente se entrega un arreglo de enteros a un método llamado “*productoAdyacentes”*. Dicho método debe retornar el mayor producto de números adyacentes que encuentre.
Tenemos el siguiente ejemplo para entender y/o guiarse:

Dado el arreglo: arreglo = {1, -4, 2, 2, 5, -1}, el mayor producto de números adyacentes es el de los números 2 y 5, en cuyo caso el método retorna el valor 10.

Luego, si se hace algo como System.out.print(“El producto adyacente es: “ + productoAdyacente(arreglo)), en pantalla tendría que salir 10.

Tiempo estimado y tiempo real de cada actividad

Actividad	Tiempo Estimado	Tiempo Real
0	30 min	15 min
1	1 hora	50 min
2	10 min	11 min
3	20 min	23 min

Los comentarios o razones de las diferencias o igualdades del tiempo estimado y tiempo real están escritas en la conclusión.

Actividad 0: Análisis del caso

Parámetros de entrada: int[] (arreglo) y int (entero)

Valor de retorno: int

Instrucciones: Utilización de diferentes métodos para luego crear el método definitivo que utilizara los métodos anteriormente creados.

El tiempo estimado y el tiempo real de esta actividad está en la tabla de arriba.

Actividad 1: Implementando el método

Implementando la solución (Diseño - Implementación):

Creamos una clase llamada “Métodos” lo cual tendrá los siguientes métodos:”crearArreglo”, “llenarArreglo”, “mostrar”, “primerMayor”, “segundoMayor” y “producto.

Por lo tanto, se creó una clase llamada “ProductoAdyacente”, la cual tendrá un método llamado “productoAdyacente”, donde el método utiliza los métodos de la clase “Métodos” para conseguir el producto adyacente de un arreglo dado.

Se nos pide que verifiquemos el método con un arreglo dado el cual es el siguiente: **arreglo = {1, -4, 2, 2, 5, -1},** por lo cual creamos otro método llamado “pruebaMetodo” el cual tiene las mismas instrucciones que el método “productoAdyacente” pero con el arreglo dado y un resultado esperado que en este caso es 10.

Al momento de probar el arreglo nos dio como resultado 10, por lo tanto, verificamos que el método “funciona”.

El tiempo estimado y el tiempo real de esta actividad está en la tabla de arriba.

Actividad 2: Probando el codigo

Definimos 4 casos de prueba (sin programar) que son relevantes para nuestro método “productoAdyacente”:

1. Casos de pruebas SOLO con valores positivos
2. Casos de pruebas SOLO con valores negativos
3. Casos de pruebas con valores mixtos
4. Casos de pruebas con los valores mayores iguales

1.- Se quiere probar el método con sólo valores positivos y esto nos debería dar un producto adyacente positivo

2.- Se quiere probar el método con sólo valores negativos y también nos debería dar un producto adyacente positivo por multiplicación de 2 negativos.

3.- Se quiere probar el método con valores mixtos (positivos y negativos) lo cual la salida depende si los números adyacentes tendrán diferente signo o igual signo.

4.- Se quiere probar el método con valores mayores iguales, lo cual la salida debería salir el mismo resultado del producto ya que todos los números son iguales.

El tiempo estimado y el tiempo real de esta actividad está en la tabla de arriba.

Actividad 3: Garantizando el éxito

Errores:

a) Un método ha recibido un argumento ilegal o inapropiado a la lógica del método crearArreglo.
b) Cuando intentamos acceder a un objeto con un valor null
c) Se produce un error aritmético durante la ejecución de un programa.
d) Un método ha recibido un argumento ilegal o inapropiado a la lógica del método productoAdyacentes.

Gestion error:

a) Creamos un try-catch para manejar cualquier excepción de tipo IllegalArgumentException que pueda ocurrir al intentar crear el arreglo. Si se produce esta excepción, se imprime el mensaje “Hubo un error” y se devuelve null Si no se produce ninguna excepción, se devuelve el arreglo creado.

   IllegalArgumentException es una clase de excepción en Java que se lanza cuando un método recibe un valor que no cumple con las condiciones o restricciones que se esperan.
b) Creamos un try-catch para manejar cualquier excepción de tipo null que pueda ocurrir al intentar acceder al objeto con valor null.
c) Este try-catch lo creamos para manejar situaciones en las que se intenta realizar una operación aritmética y existe la posibilidad de que se produzca una excepción de ArithmeticException. Si se lanza una excepción de este tipo, se imprimirá un mensaje “Error al dividir por cero”
d) Creamos un try-catch para manejar cualquier excepción de tipo IllegalArgumentException que pueda ocurrir al intentar crear el arreglo. Si se produce esta excepción, se imprime el mensaje “Hubo un error” y se devuelve null Si no se produce ninguna excepción, se devuelve el arreglo creado.

Conclusión

Como resultado de todo lo anterior, se pudo diseñar e implementar una solución al problema de la actividad semanal “caso adyacentes”, donde se discutió entre el grupo el código con los métodos necesarios para encontrar el mayor producto de números adyacentes de un arreglo.

Tiempo estimados y tiempo reales:

- Actividad 0: Estimamos un tiempo de 30 minutos para completarlo, el cual tuvo como tiempo real 15 minutos, la razón de este tiempo es que se pudo avanzar más rápido de lo esperado debido a que creímos que nos podíamos demorar más analizando el problema y pensando en la posible solución.

- Actividad 1: Estimamos un tiempo de 1 hora y lo logramos en 50 minutos, con tan solo 10 minutos de diferencia debido a que al crear el código tuvimos que recurrir a buscar información ya que hay 2 integrantes que deben acostumbrarse al cambio de lenguaje de programación.

- Actividad 2: Estimamos un tiempo de 10 mins y se logró en 11 minutos, con 1 minuto de diferencia debido que solo teníamos que avanzar en el documento en esta actividad y revisamos la ortografía del documento y la completación de la actividad 2, sin embargo, se podría lograr terminarlo en menos de 10 minutos.

- Actividad 3:Estimamos un tiempo de 20 mins y se logró en 23 minutos con 3 minutos de diferencia debido a que tuvimos que analizar bien el código para encontrar las excepciones para añadirlo al programa y subir la nueva versión del código.

Por ende, en esta actividad logramos como grupo desarrollar una mejor versión del primer código.