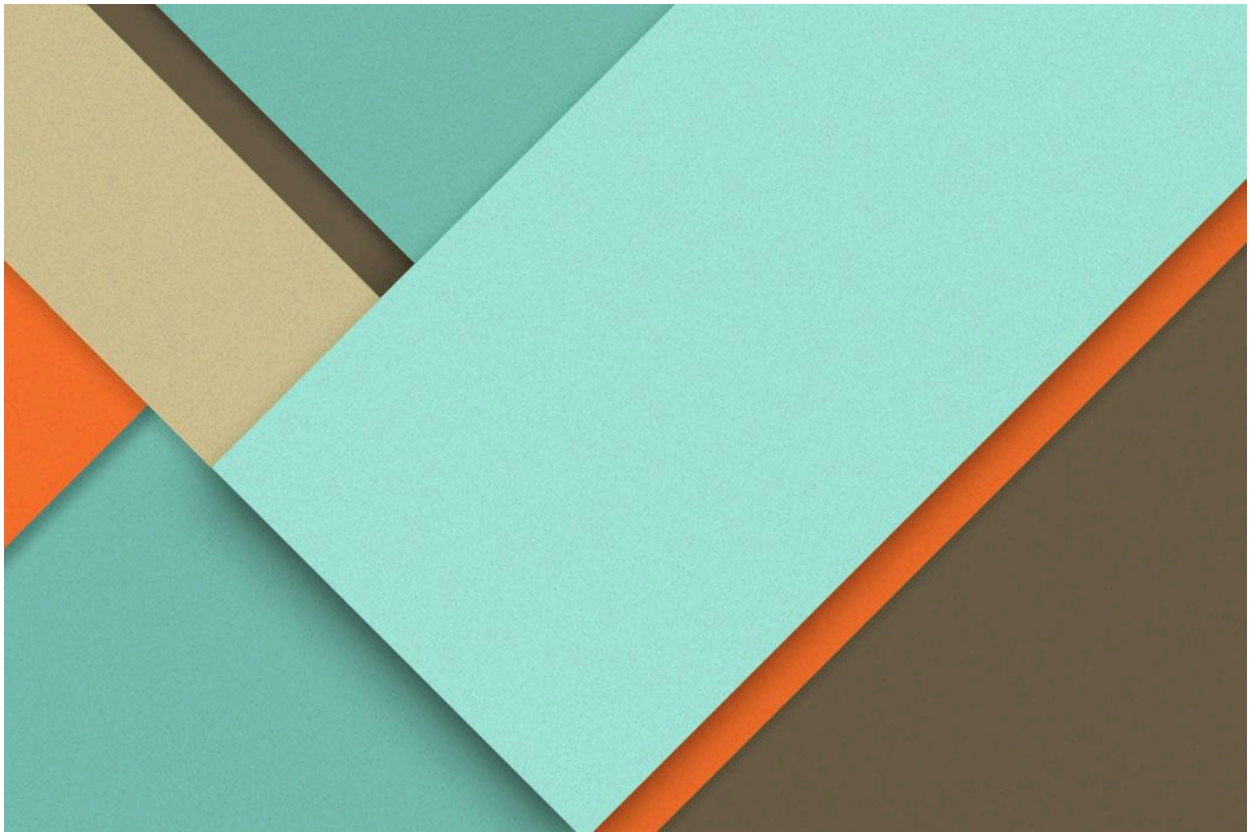

JUNIT

25/04/2025

RONNALD SEBASTIAN BENITEZ CANO

1-Debería de Evitar el uso de variables static globales por que el Problema que causa esto es el uso de variables static como usuario, saldo y sc acopla mucho la lógica.
La mejora que pondría es encapsular estos datos en una clase CuentaFinanciera y trabajar con instancias.
Por qué esto facilita mucho más la pruebas unitarias y manejo de múltiples cuentas y la libre reutilización del código en cuestión

2- Deberías de separar la lógica de entrada y salida de la lógica del negocio por que esto crea un problema en la lógica de negocio como modificar el saldo por que está mezclada con System.out y Scanner.
La mejora que pondría es crear métodos que se manejan solos los datos y otros que se encarguen de la interacción con el usuario por qué esto permitirá reutilizar el código (por ejemplo, en una interfaz gráfica) y hacer las pruebas más fáciles.

3- El uso de constantes para los menús

El problema que veo son los textos de menú están muy quemados en el código.
La mejora que pondría es el uso de constantes estáticas o incluso un enum para los tipos de gasto por qué esto mejora la mantenibilidad y la claridad del código.

4- Un control de errores mucho más sólido

El problema en sí es el usuario la cual si no introduce valores válidos, solo se imprime un mensaje la mejora que pondría es agregar la validación o reintento automático al capturar valores inválidos por qué esto mejora mucho la experiencia del usuario.

5. Opción 5 en el menú también imprime "Opción inválida"

El problema es que el default del switch se ejecuta siempre, incluso si se elige la opción 5.para mejorarlo debería de Añadir un break después de System.out.println("Saliendo...");.

para evitar mensajes confusos.

6. Modularidad y pruebas unitarias

Problema: Las funciones dependen de la entrada del usuario (Scanner), lo que dificulta las pruebas.
Mejora: Refactoriza para permitir inyectar datos desde pruebas (por ejemplo, con parámetros).
Por qué: Hace posible escribir pruebas automatizadas efectivas con JUnit.