ENTORNOS DE DESARROLLO

Testing con JUnit

CÓDIGO: Ronnald Sebastián Benítez Recalde

TESTEO: JACOBO LUIS DOMINGUEZ MORALES

ALUMNO:
JACOBO LUIS DOMINGUEZ MORALES

PROFESOR:
JAVIER MARTÍN RIVERO

Introducción

El código proporcionado por tu compañero implementa un programa en Java para controlar los ingresos y gastos de un usuario. A continuación, se analizan los aspectos más relevantes del código, se identifican áreas de mejora, y se evalúa si se cumple con los requisitos especificados en las instrucciones del proyecto. Además, se sugieren buenas prácticas y recomendaciones para mejorar la legibilidad, la robustez y la mantenibilidad del código.

Requisitos mínimos

1. Introducción del nombre del usuario antes de cualquier otra opción: El código cumple con este requisito, ya que en el menú inicial, la opción 1 permite al usuario introducir su nombre antes de realizar cualquier otra acción.

2. El saldo no puede ser negativo: Este requisito está bien implementado, ya que cuando el usuario intenta realizar un gasto mayor al saldo disponible, el sistema muestra un mensaje indicando que no hay suficiente saldo.

3. El saldo debe ser mayor que el gasto que se desee hacer: El código cumple con esta especificación al verificar que el gasto no exceda el saldo disponible antes de realizar la operación. Esto se valida correctamente en el método introducirGasto().

4. Mensajes personalizados para el usuario: El código implementa adecuadamente mensajes personalizados para cada acción realizada, mostrando el nombre del usuario en los mensajes.

Puntos a mejorar

1. Validación del ingreso: Aunque se valida el gasto, no se realiza una validación similar para los ingresos. Si el usuario introduce un valor negativo o cero para el ingreso, el sistema lo aceptará sin ninguna advertencia.

2. Mejor manejo de las excepciones: Aunque se utiliza un Scanner para la entrada del usuario, el código no está preparado para manejar posibles excepciones que puedan surgir al intentar leer datos erróneos. Usar un bloque try-catch para manejar estas excepciones mejoraría la robustez del programa.

3. El código tiene una buena estructura al separar la lógica en clases como Menu, Usuario y Principal. Sin embargo, la clase Usuario está tomando muchas responsabilidades (por ejemplo, manejar el saldo y mostrar los mensajes de error).

4. Aunque el código es comprensible, la falta de comentarios hace que sea más difícil de seguir, especialmente para aquellos que no están familiarizados con el código.

5. No se si se deba a los test que he hecho pero debido a que el saldo es una propiedad privada de Usuario no se puede acceder fuera de esta clase.