Refactoring Ejemplo 3

Por: Juan Sebastián Sánchez L.

Refactoring: Extracción de método updateProductQuantities

- Descripción: El código para actualizar las cantidades de productos y ventas se ha extraído a un nuevo método *updateProductQuantities* para mejorar la legibilidad y facilitar la reutilización.
- Razón de aplicación: Separar la lógica de actualización de cantidades en un método independiente hace que el código sea más claro y permite la reutilización de la funcionalidad si es necesario en el futuro.
- Consecuencia: El código es más legible y modular.

Refactoring: Cambio de nombres de variables

- Descripción: Se han cambiado los nombres de algunas variables, como *csvFileProducts*, *csvFileSales*, *csvFileOrders*, para que sean más descriptivos y sigan las convenciones de nomenclatura.
- Razón de aplicación: Nombres de variables descriptivos hacen que el código sea más comprensible y mantenible.
- Consecuencia: Las variables tienen nombres más descriptivos.

Refactoring: Encapsulamiento de propiedades en las clases Product, Order, y Sale.

- Descripción: Se han encapsulado las propiedades en las clases *Product*, *Order*, y *Sale* utilizando modificadores de acceso (private) y proporcionando métodos para acceder y modificar las propiedades.
- Razón de aplicación: El encapsulamiento de propiedades ayuda a ocultar la implementación interna de las clases y proporciona un mejor control sobre el acceso a las propiedades.
- Consecuencia: Las propiedades de las clases están encapsuladas y se acceden a través de métodos getter y setter.

Refactoring: Simplificación de la lógica en el método createItem en CSVReader.java.

- Descripción: Se ha simplificado la lógica en el método *createItem* utilizando un enfoque de switch para crear objetos de diferentes tipos.
- Razón de aplicación: La simplificación de la lógica hace que el código sea más claro y más fácil de entender.
- Consecuencia: El código es más conciso y legible.