Luisa Gómez Orrego

202222249

Repositorio Analizado:

<https://github.com/WildCodeSchool/quest-java-singleton-demo.git>

• Información general del proyecto: para qué sirve, cuál es la estructura general del diseño, qué grandes retos de diseño enfrenta (i.e. ¿qué es lo difícil?). Deben incluir la URL para consultar el proyecto.

Este proyecto modela de manera básica una interfaz para la compra de muebles. Esta plataforma debe dar acceso a los clientes para acceder a la oferta de muebles. El mayor reto se presenta al asegurarse de hacer una única conexión entre la cuenta de usuario y sus compras.

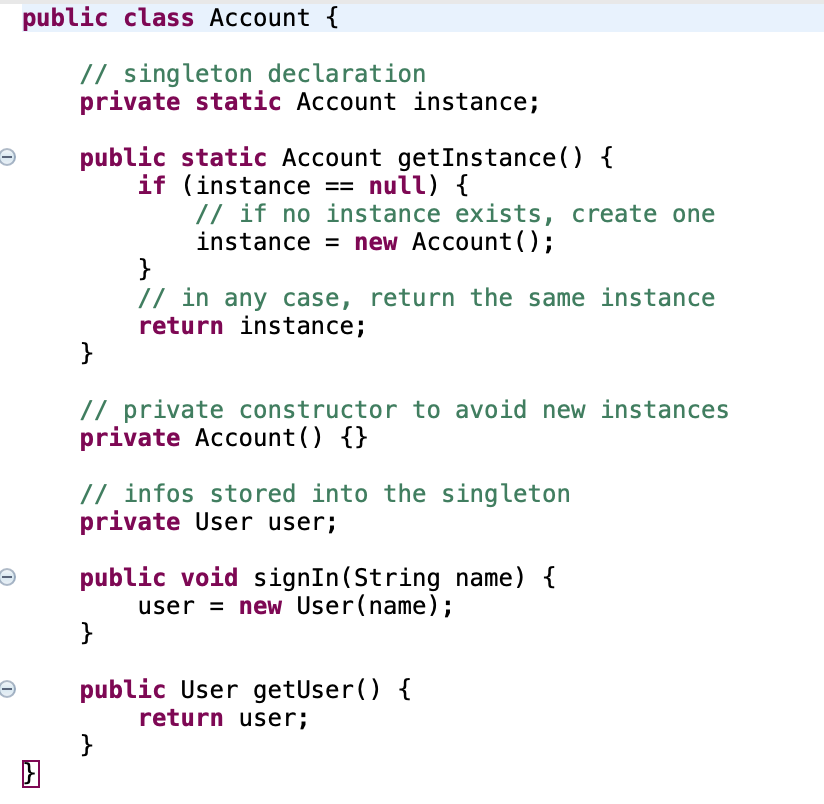
• Información y estructura del fragmento del proyecto donde aparece el patrón. No se limite únicamente a los elementos que hacen parte del patrón: para que tenga sentido su uso, probablemente va a tener que incluir elementos cercanos que sirvan para contextualizar.

Este proyecto es un ejemplo de una interfaz con el usuario para la compra y venta de muebles, se usa para distinguir entre la creación de una cuenta en pla página web, el catálogo, el login y demás opciones relacionadas. Este paquete pretende buscar una cuenta de usuario, para eso usa el método singleton para cerciorarse de que no está recibiendo más de una instancia.

• Información general sobre el patrón: qué patrón es y para qué se usa usualmente.

El patrón singleton es un patrón de creación, es decir “factory method”, se usa principalmente para asegurarse de que un método tenga una sola instancia, es decir no reciba más parámetros. También ayuda a que el método tenga acceso global, pues si tiene dentro del mismo información relevante para el resto del proyecto, permite que se acceda desde cualquier punto.

• Información del patrón aplicado al proyecto: explicar cómo se está utilizando el patrón dentro del proyecto.



1. Se declara la clase “Account”
2. Se declara la única instancia de “Account”, implementando así el patrón singleton. Es privado para asegurarse de que no tiene más accesos.
3. El método get instance es la primera implementación de el patrón singleton, pues busca una “Account” existente y en caso de que el resultado sea nulo, la cuenta de usuario no exista, se crea una nueva. De lo contrario, se redirige a dicha cuenta.
4. Se crea el constructor para account, este es privado para asegurarse de que solo pueda ser utilizado dentro de su propia clase, así se genera mayor acoplamiento pero de la misma manera se asegura de que no se pueda acceder a el constructor desde ningún otro punto del programa.
5. Se almacena la información del usuario dentro del singleton.
6. Se utiliza el método “sign in “ para crear un nuevo usuario.
7. Se utiliza el m-etodo getUser para acceder a la información de usuario.
8. Se retorna el usuario en cuestión.

• ¿Por qué tiene sentido haber utilizado el patrón en ese punto del proyecto? ¿Qué ventajas tiene?

Teniendo en cuenta de que esta interfaz se radica en las cuentas de los usuarios que pretenden comprar dentro de esta tienda de muebles. Se debe tener en cuenta diferentes aspectos, por ejemplo, la búsqueda de un usuario debe tener únicamente una entrada o instancia. Algo similar a porqué se restringe un teclado alfabético en una interfaz de calculadora. El método singleton se asegura de que solo se reciba una instancia. Adicionalmente proporciona fácil acceso de manera global, es decir otros paquetes. Además tiene la facilidad de contar con inicialización bajo demanda.

• ¿Qué desventajas tiene haber utilizado el patrón en ese punto del proyecto?

El método singleton genera acoplamiento en el código por lo que podría dificultar las pruebas del programa y limita así la capacidad de arreglarlo.

• ¿De qué otras formas se le ocurre que se podrían haber solucionado, en este caso particular, los problemas que resuelve el patrón?+

Creo que el patrón singleton es el que más sentido tiene en términos generales periodo todas maneras tras investigar, sería útil el diseño basado en interfaces para así asegurarse de poder hacer implementaciones para no afectar el código que las utiliza.