e2: Cotización de acciones en el mercado bursátil

Índice

1. Patrón Observador
2. Interfaz
3. Clientes

Dominio del problema

El dominio del problema gira en torno a la gestión y distribución de información bursátil en tiempo real.

Nos interesan los datos siguientes de cada acción:

- Código de acción
- Cierre
- Máximo
- Mínimo
- Volumen

El sistema debe atender a diversos clientes con requerimientos distintos y además permitir agregar nuevos clientes.

Y además, debe ser capaz de notificar automáticamente a todos los clientes interesados cada vez que los datos bursátiles cambien.

Patrón observador

El Patrón Observador es la solución ideal para este problema, ya que permite establecer una `relación de dependencia` entre un `sujeto` (las acciones) y `múltiples` observadores (los clientes). Cada vez que los datos de las acciones cambian, el sujeto notifica automáticamente a todos los observadores interesados.

Ventajas del Patrón Observador:

Modularidad:

Facilita la separación entre la lógica de gestión de datos bursátiles y las funcionalidades específicas de cada cliente.

Escalabilidad:

Permite añadir nuevos observadores fácilmente, sin necesidad de modificar el código existente.

Desacoplamiento:

Reduce las dependencias entre el sujeto y los observadores, simplificando el mantenimiento y la evolución del sistema. El observado no conoce a sus observadores.

Variación aplicada:

Se ha implementado una variante `push` del Patrón Observador, donde el sujeto envía directamente a los observadores la información relevante.

Esto evita que cada cliente tenga que solicitar (`pull`) los datos y averiguar que ha cambiando. Además nuestros clientes no saben exactamente como es el objeto.

Interfaz observer

Hemos decidido implementar nosotros la interfaz, dado que muy sencilla, en lugar de usar la que nos ofrece Java.

Además, vimos que no tenía mucho sentido sobrecargar a un cliente que solo le interesa el cierre con las notificaciones de los datos instantáneos (máximo, mínimo y volumen).

Entonces hemos implementado `2 tipos de observadores`:

- Uno notifica de cambios del precio de `cierre`.
- Otro de cambios de `valores instantáneos`.

Así si en un futuro se quisiera añadir un nuevo cliente, simplemente tendría que `subscribirse` al observador que les interese, sin tener que modificar nada.

Clientes

Hemos creado 2 clientes, uno simple y otro complejo.

Uno interesado solo en el precio de cierre.

Otro interesado en precio de cierre y datos instantáneos, para hacer gráficas en una app.

Los datos en ambos se almacenan en un Map, es decir de declara y se devuelven como Map, pero son un HashMap. (Principio de inversión de la dependencia). El map <código de la acción, datos>.

En el caso del cliente complejo, los datos instantáneos los guarda en un Map <código, registro>. El registro es un claro ejemplo de `patrón Inmutable`.

Decidimos guardarlo así porque los consideramos simplemente una "foto" del estado pasado. No tendría sentido poder modificarlos y podría provocar un comportamiento indeseado.

Estos datos se guardan de forma infinita. Seguramente algo a cambiar para una implementación en el mundo real.