Informe para la Práctica 2: Cotización de Acciones en el Mercado Bursátil

Introducción

En esta práctica se diseña e implementa un sistema para la gestión de información bursátil de distintas acciones. Este sistema permite actualizar los datos de una acción (precio de cierre, máximo, mínimo, volumen) y notificar automáticamente a múltiples tipos de clientes interesados.

Los clientes pueden tener diferentes requerimientos de información, como:
- Clientes sencillos, que solo desean conocer el precio de cierre.
- Clientes detallados, que necesitan información completa de una acción.

El sistema debe ser extensible para soportar nuevos tipos de clientes en el futuro, sin requerir cambios en el código existente. Para ello, se emplean principios SOLID y patrones de diseño adecuados.

Principios SOLID Aplicados

Responsabilidad Única (SRP):
- Cada clase tiene una única responsabilidad:
  - Accion gestiona los datos de una acción y Mercado la notificación a los observadores.
  - ClienteSencillo y ClienteDetallado son responsables de procesar y mostrar los datos según su propio formato.
  - DatosAccion encapsula la información bursátil de una acción.

Esto garantiza que los cambios en una parte del sistema no afecten a otras partes.

Abierto/Cerrado (OCP):
- El sistema está diseñado para ser extensible. Si se necesita añadir un nuevo tipo de cliente (por ejemplo, un cliente que solo desea conocer el volumen negociado), basta con implementar la interfaz Observador en una nueva clase, sin modificar el código de las clases existentes.

Sustitución de Liskov (LSP):
- Todas las implementaciones de Observador (ClienteSencillo y ClienteDetallado) pueden sustituir a la interfaz sin alterar el comportamiento esperado del sistema. Esto asegura que cualquier observador registrado puede recibir y procesar actualizaciones de manera uniforme.

Segregación de Interfaces (ISP):
- La interfaz Observador es específica y contiene un único método, actualizar. Esto evita que las clases cliente tengan que implementar métodos innecesarios, cumpliendo con este

principio.

Inversión de Dependencias (DIP):
- La clase Accion no depende de implementaciones concretas de los clientes, sino de la interfaz Observador. Esto permite desacoplar el modelo de datos bursátiles de la lógica específica de los clientes.

Patrón de Diseño Utilizado

Patrón Observer:
Se utiliza el patrón Observer para permitir que múltiples clientes (observadores) se registren y reciban notificaciones automáticas cuando se actualicen los datos de una acción. Este patrón es ideal para este sistema porque:

- Extensibilidad: Se pueden añadir nuevos tipos de clientes sin modificar el código existente. Solo se necesita implementar la interfaz Observador.
- Desacoplamiento: La clase Accion no necesita conocer los detalles específicos de los clientes, lo que facilita su mantenimiento.
- Actualización automática: Cuando los datos de una acción cambian, todos los clientes interesados reciben las actualizaciones de manera sincronizada.

Justificación de la Variación del Patrón:
En este diseño, el patrón Observer se implementa de forma simple:
- Accion actúa como el sujeto.
- Las clases que implementan Observador (como ClienteSencillo y ClienteDetallado) son los observadores.
- Mercado se encarga de centralizar la gestión de los observadores. Cuando se actualizan los datos de una acción mediante el método actualizarDatos en Accion, se notifica a Mercado, que a su vez llama al método actualizar en cada observador registrado con esa acción.

Diagrama de Clases

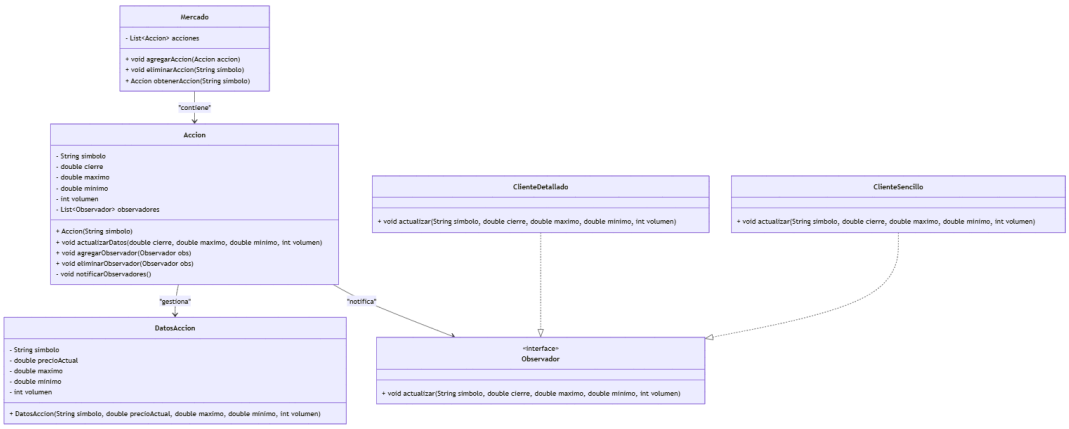

Diagrama de Secuencial