Design Pattern Explanations

Observer Pattern – Stock Alert System

This design applies the Observer Design Pattern to manage low stock alerts in a system. The Stock class acts as the Subject, maintaining a list of observers (StockAlertReceiver) that should be notified when stock levels drop.

Key Components:
- StockAlertReceiver: An interface that defines the notifyLowStock() method for alert receivers.
- Stock: The observable class that holds a list of alert receivers and notifies them when quantity falls below a threshold.
- CFO: A concrete observer that implements StockAlertReceiver, receiving stock alerts and providing methods to monitor stock and manage financial tasks.

Benefits:
- Promotes loose coupling between the Stock class and alert handlers.
- Allows dynamic registration/removal of alert receivers.
- Supports Open/Closed Principle – new alert types can be added without modifying the Stock class.

Observer Pattern – Invoice Alert System

This design also uses the Observer Design Pattern, this time to manage pending invoice notifications.
The Invoice class serves as the Subject, notifying registered observers about overdue or unpaid invoices.

Key Components:
- InvoiceAlertReceiver: An interface for observers to implement the notifyPendingInvoice() method.
- Invoice: Maintains a list of observers and notifies them based on invoice due dates.
- ReminderService: Implements the InvoiceAlertReceiver interface to receive alerts and send reminders for pending invoices.

Benefits:
- Decouples invoice logic from notification services.
- Makes the system more maintainable and scalable.
- New alert mechanisms can be integrated without changing the existing core logic.

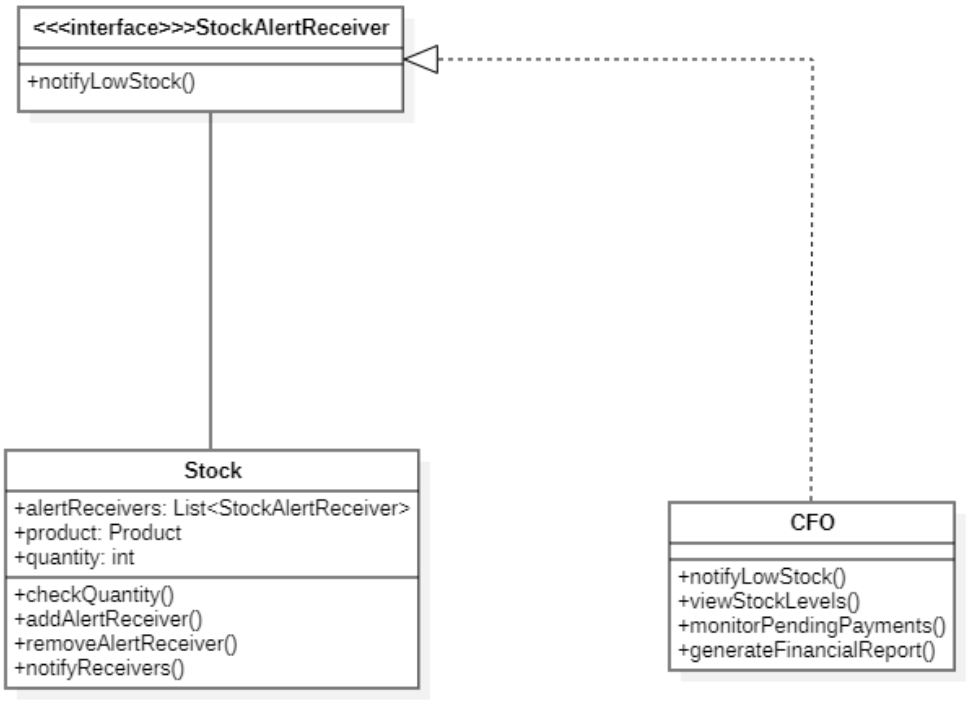

Figure 1: Stock Alert System using Observer Pattern

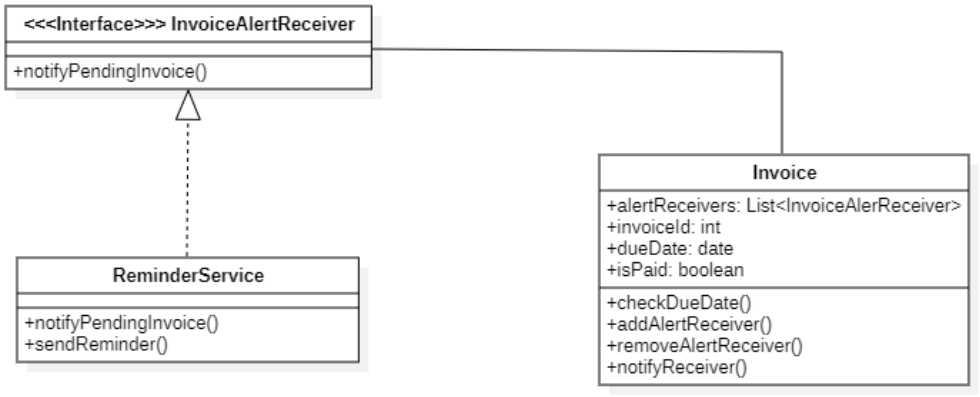

Figure 2: Invoice Alert System using Observer Pattern