

# ChatRoom

Wednesday, November 5, 2025 7:52 PM



## Example 4 — Stock Market Feed (Observer)

### Concept (business story)

A **MarketFeed** publishes price ticks for multiple symbols (e.g., "AAPL", "GOOG").

Multiple **Dashboards/Widgets** subscribe and react to price changes (show latest, compute stats, trigger alerts).

## Functional Requirements

### 1) Entities

- **Subject:** MarketFeed
- **Observers:**
  - TickerBoard (shows latest price per symbol)
  - MovingAverageWidget (running avg for a chosen symbol)
  - PriceAlert (fires when price crosses a threshold)

### 2) Core flows

- MarketFeed.publish(symbol, price) updates internal state and **notifies observers**.
- Observers receive (symbol, price) and decide whether to act (e.g., only for their tracked symbol).
- Observers can **attach/detach** at any time.
- Duplicate attach must be ignored (idempotent subscribe).

### 3) Data model

- Symbols are std::string (e.g., "AAPL").
- Prices are double (or float).

### 4) Notification semantics

- **Notify only when price actually changes for that symbol.**  
If the same price is re-published for the symbol, skip notifying.
- Notify **after** the feed updates its internal price map.

## Behavioral Rules

Rule	Description
R1	MarketFeed maintains a map: symbol -> lastPrice
R2	Publishing a new price for an unseen symbol creates it and notifies
R3	Observers receive (symbol, price) and may ignore if not relevant
R4	attach()/detach() are O(n) acceptable (beginner level)
R5	removeAll() clears all observers
R6	Use a <b>snapshot</b> of observers during notify to allow self-detach

## Non-Functional Constraints

- Beginner version: single-threaded, push model.
- Keep interfaces small and clear.
- Prefer const refs for strings in APIs.
- No I/O blocking in observers beyond std::cout.

## Interfaces (suggested shape, you can tweak)

### Observer side

- IMarketObserver::onTick(const std::string& symbol, double price)

### Subject side

- IMarketSubject::{attach, detach, removeAll, publish(symbol, price)}

### Concrete Subject

- MarketFeed with:
  - std::unordered\_map<std::string, double> last\_
  - std::vector<IMarketObserver\*> subs\_ (beginner)
  - publish(symbol, price) → update map → if changed → notify(symbol, price)

### Concrete Observers

- TickerBoard — prints "AAPL: 192.35" whenever it changes.
- MovingAverageWidget(symbol, windowN) — maintains running average (simple cumulative OK for beginner; windowed MA is stretch).
- PriceAlert(symbol, threshold, direction) — prints alert when price **crosses** threshold ( $\geq$  or  $\leq$  once per crossing).

## Test Scenario (Acceptance)

1. Create MarketFeed, attach 3 observers (board, MA(AAPL), alert(AAPL  $\geq$  200)).
2. Publish sequence:
  - o ("AAPL", 198.0)  $\rightarrow$  all relevant observers react
  - o ("GOOG", 101.0)  $\rightarrow$  TickerBoard reacts; others may ignore
  - o ("AAPL", 198.0)  $\rightarrow$  **no notify** (same price)
  - o ("AAPL", 201.0)  $\rightarrow$  Alert fires once; MA updates; Board updates
3. Detach MA, publish ("AAPL", 205.0)  $\rightarrow$  MA no longer prints
4. removeAll(); further publishes produce no output

Expected: outputs only on real changes; alert fires exactly when threshold is crossed.