Vietnam National University, Ho Chi Minh City
Ho Chi Minh City University of Technology
268 Ly Thuong Kiet Street, District 10, Ho Chi Minh City, Viet Nam

Subject: Advanced Programming

Assignment 2
Real-time Stock Monitoring System

Objective

Design and implement a system that simulates real-time stock monitoring, providing users with up-to-date stock information from multiple sources in a customizable display. The system should demonstrate the use of the Observer, Adapter, and Singleton design patterns.

Description

The Real-time Stock Monitoring System enables users to track current stock prices, ensuring they have the latest market data for informed decision-making. By aggregating data from multiple sources, such as the HOSE and HNX exchanges, the system provides a comprehensive view of the market.

Core Components

You are given a class diagram as presented in the Appendix.

The main components of the system are:

- `Stock`: Represents a stock, including stock `code` and `name`.
- `StockPrice`: Represents a stock price update, including attributes such as stock, average price, timestamp, and volume, which store the average price and total matched volume within one minute.
- `StockFeeder`: Manages stock prices and notifies registered observers when prices change.
    - `addStock(Stock stock)`: Adds a stock to the monitoring list.
    - `registerViewer(String code, StockViewer stockViewer)`: Registers a new observer.
    - `unregisterViewer(String code, StockViewer stockViewer)`: Unregisters an observer.
    - `notify(StockPrice stockPrice)`: Notifies all registered observers about stock price changes.

- `StockViewer (Observer interface)`: Observers implement this interface to receive stock updates.
- `PriceFetchManager (Adapter pattern)`: Fetches stock prices from external sources and updates the `StockFeeder`.
    - `run()`: Executes every 1 minute to fetch stock prices from the market and update the `StockFeeder`.
- `PriceFetcher`: An interface that follows the Adapter pattern.

Utility Classes

- `HosePriceFetchLib` and `HnxPriceFetchLib`: Libraries that fetch real stock prices from the respective stock exchanges.
- `Logger`: Logs messages to the console.

Business Rules

- When register/unregister viewer, if the `stockCode` is not in the monitoring list nor the viewer in/not in the tracking list, it logs to console with message "Error when [register/unregister] with <stockCode>". Otherwise, nothing happen.
- The `StockRealtimePriceView` must log stock price updates to the console whenever it receives an update.
- The `StockTickerView` must log ticker information to the console every 10 seconds, including the following details:
    - Stock code
    - Highest price
    - Lowest price
    - Opening price
    - Closing price
    - Total volume
    - Average price
    - Timestamp
- The `StockAlertView` must log an alert to the console when the stock price meets predefined conditions.

- ***Note**: System must use class Logger to print messages to the console.*

Student Tasks

Your task is to implement the system and ensure all required components function correctly. You must implement all **yellow-colored classes** in the class diagram.

***Note**: The class diagram does not show basic methods such as getters, setters, and overridden functions. Additionally, to simplify the diagram, some methods have been omitted, but you can find more details in the provided code.*

Vietnam National University, Ho Chi Minh City
Ho Chi Minh City University of Technology
268 Ly Thuong Kiet Street, District 10, Ho Chi Minh City, Viet Nam

Appendix

Initial Class Diagram

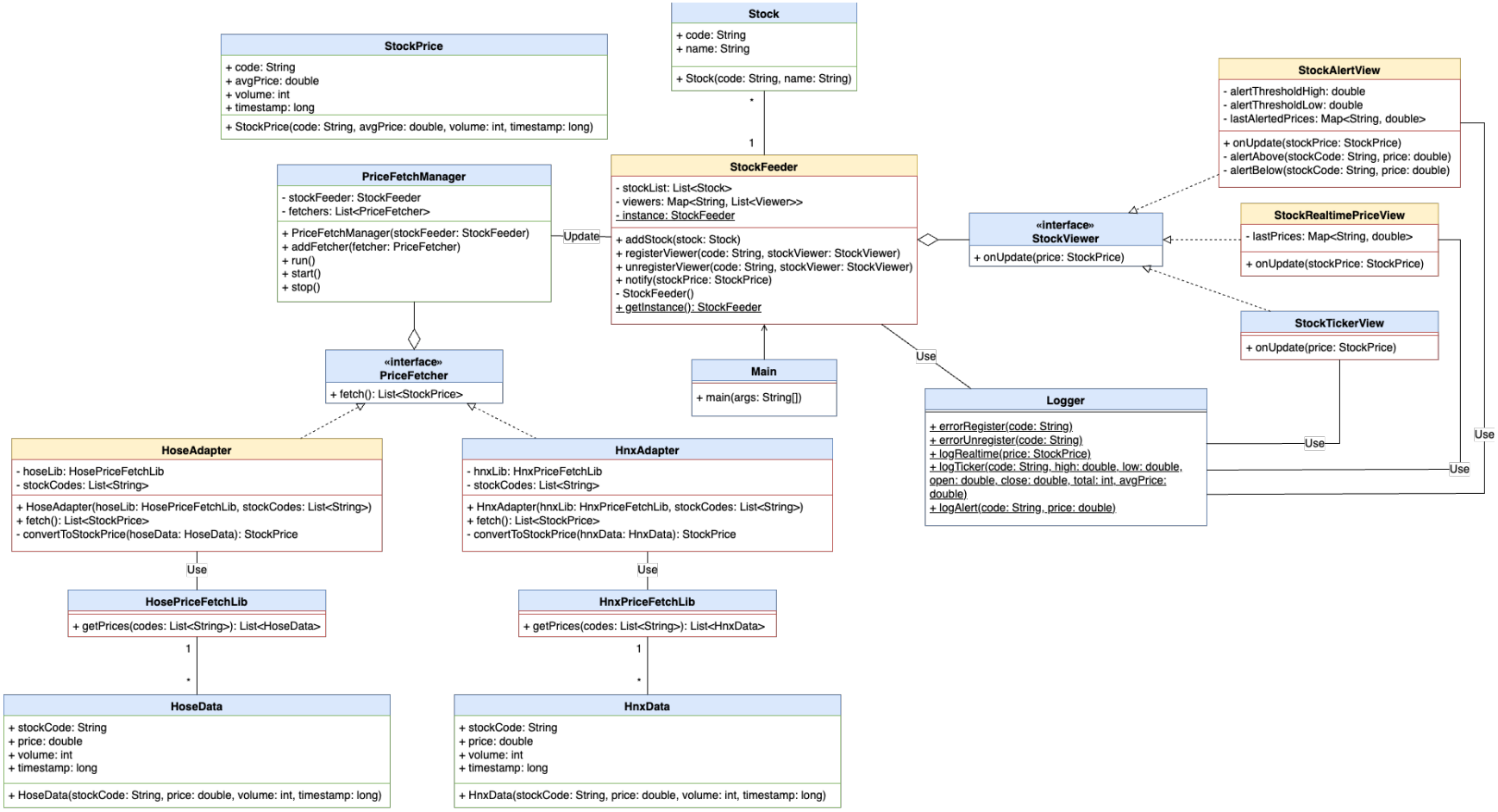

Figure 1. Initial class diagram

Blue parts are classes that are provided in code.

Yellow parts are classes that you must implement. These classes are fixed as the diagram in Figure 1.

***Note**: Students may add fields/methods to the required classes if necessary.*

Guidance: In order to run the project, please follow `README.md` in the folder Initial Code.

Submission Instructions:

- You will complete the assignment **individually** and must submit the code as a **ZIP file** via **LMS**.
- Before submitting, rename the folder **Initial Code** using the following format:
  `StudentName_StudentID`
  - Example: `NguyenVanA_1952968`
- **Compatibility Test**: You will be provided with some simple test cases to check the compatibility of your submission on **April 1, 2025**.
- **Final Submission Deadline**: **23:55 - April 8, 2025**

Important Notes:

- Ensure that your submission folder follows **Maven's standard structure**[1].
- All assignments will be reflected in the **Final Exam** as **harmony questions**, and your answers to those questions will be used to **re-evaluate your assignment grade** using the following formula (on a scale of 10):

  Final Assignment Score = {2 × (A1/100) × B1 / [(A1/100) + B1]} × 10

  where:

  - A1 = Assignment Score (out of 100)

[1] Apache Software Foundation (2025). *Introduction to the Standard Directory Layout.* https://maven.apache.org/guides/introduction/introduction-to-the-standard-directory-layout.html. Last accessed on 08:00, March 18, 2025.

- B1 = Final Exam Performance (Correct Answers / Total Related Questions)

Example:

- A student passes **95/100** test cases → **A1/100 = 0.95**
- In the final exam, they correctly answer **5/10** questions related to the assignment 2 → **B1 = 0.5**
- Their final assignment 2 score = **[2 × 0.95 × 0.5 / (0.95 + 0.5)] × 10 = 6.5**