CS 542 Design Patterns and Object-Oriented Analysis

Class Project

Due on Monday, December 6, 11:59PM.

Objectives

Practice object-oriented design and implementation with design patterns.

General Requirements

You may either work individually, in two-person teams, or in three-person teams.

You should submit both your code and a final project report.

Choose one of the three applications listed at the end of the document as the topic of your class project. Your goal is to develop the application using object-oriented analysis, design, and implementation techniques that we have learned in this class. The final project is an executable software system implemented in C++ with the described functions.

The final project report must include the detailed analysis and design, and demonstrate the appropriate use of object-oriented design techniques and design patterns, in particular

a. The design should identify all classes and objects for the software project, and their relationship to each other. The use of Unified Modeling Language (UML) Class Diagrams is required.

b. The design should demonstrate the use of **5** or more design patterns in different situations. Among the patterns, you must use at least one Structural pattern and one or more Behavioral patterns.

c. Each design decision (e.g., the use of a design pattern) should be explained. You should also include with a brief discussion of how these patterns participate in an intelligent solution to the original problem. For example, you should come up with a legitimate business concept that will benefit from your selection of patterns.

Implementation

Implement the main functions of the system in C++. Separate the class declarations in the header files and the implementation details in the source files. Include appropriate comments in your code.

Project Presentation

Each team will give an in-class presentation about their project in Week 15. A demonstration of your project application is required during the presentation. The presentation also counts towards your project grade. The detailed schedule and logistics of presentation will be announced later.

Submission

Submit two files to Cougar Course before the deadline:

1. A zip file that includes the source code of your project. The file should be named with the last names of your team members. E.g., Last1Last2.zip.
2. A PDF document for your project report named in the same way (e.g., Last1Last2.pdf).

Project List

1. **File Viewer**. Create a smart “tail” program for viewing live updates to log files and providing parsing tools. This tool would serve as a software development aid, allowing a developer to view real-time updates to log files produced by applications they are debugging. The tool would offer many useful features:

- File Selection: Allow the user to select a text file by browser file system

- Auto-Load: Allow user to specify a directory containing text files. The application will automatically load the most recently modified file and continuously check the directory for a newer file, removing the need for the user to load a new file when one is created in the target directory.

- Tailing: Continuously poll the target file for update and display to the user, automatically scrolling to the bottom upon update, and not locking the target file.

- Regular Expressions: Highlight or filter using a modifiable list of regular expressions. This helps highlight important messages or hide unimportant messages.

- Capture: capture some text and pop-out into another window at the click of a button. This allows a user to quickly separate some text for later analysis, which is especially useful with fast-growing log files.

2. **Stock Trading**. As you may or may not know, numerous companies are financed primarily by publicly trading shares of their ownership amongst the public in various stock exchanges. Purchasers buy shares of a company if they believe that the company will flourish over time and that those shares will be worth more money later on. Therefore, investors profit by choosing the right stock options, waiting until the price increases, and then selling those shares at a better price.

Build a tool that is quickly growing in popularity in the stock trading business: a trading bot. This bot would be an active participant in daily market trading on behalf of the users, and it would attempt to make stock trades on a daily basis in order to make a profit for the user, if possible. Investing strategy would be left up to the particular development team. The bot project should have the following features.

● A simulated stock market exchange must be made, with several different company stocks available, and whose prices change throughout the course of the program based on price changes that have occurred to the real-life shares throughout the course of a single day or several days in the real-world market. There should be a wide variety of stocks that end at a higher or lower price than they started, and a wide variety of price changes over time until the end of the program.

● Design and implement a simplified banking system with several planned cash deposits as well as an amount of starting money.

● Design and implement a market trading bot, which will:

○ Collect pricing information about all available shares on the market from the exchange simulation.

○ Rank all shares according to both current price and previous increases to determine the best possible candidates for investment.

○ Based on both available funding and available viable investment options, choose a number of shares from one or more companies to purchase as an investment.

○ Take home a profit of any kind at the end of the program.

● Display trading performance upon termination of the simulation, including but not limited to: the shares bought and which quantities of each, total profit, days elapsed, etc.

3. **CSUSM Parking Reservation Application**. The goal of this app is to help students find parking during peak hours when the main parking lots are generally full. It requires students to register with their unique ID. They will take note of where they parked and set the estimated time that they will be leaving. Any student registered with the app and is looking for parking later in the day may “reserve” the spot. (Since there is no way to really enforce reserved parking there isn’t a guarantee). Once the spot has been reserved, it will no longer be displayed. Features:

• Specific lot options (PS1, Lot B, C, F etc.) and whether they are full

• Search by time of departure

• GPS location within in the campus since the lots do not have numbered parking spaces

• “Parked” option with time stamp

• “Reserve” option. This will display the users who have logged in their estimated time of departure. (E.g., If I have class at 10:30 am and my last class ends at 2:15 pm, I log in “Parked” that I will be leaving at 2:15 pm. Anyone who may be looking for a spot ahead of time for their 2:30 pm class will be able to reserve the spot).

• Notifications

• Could possibly add ratings for consistency with departure time or delays