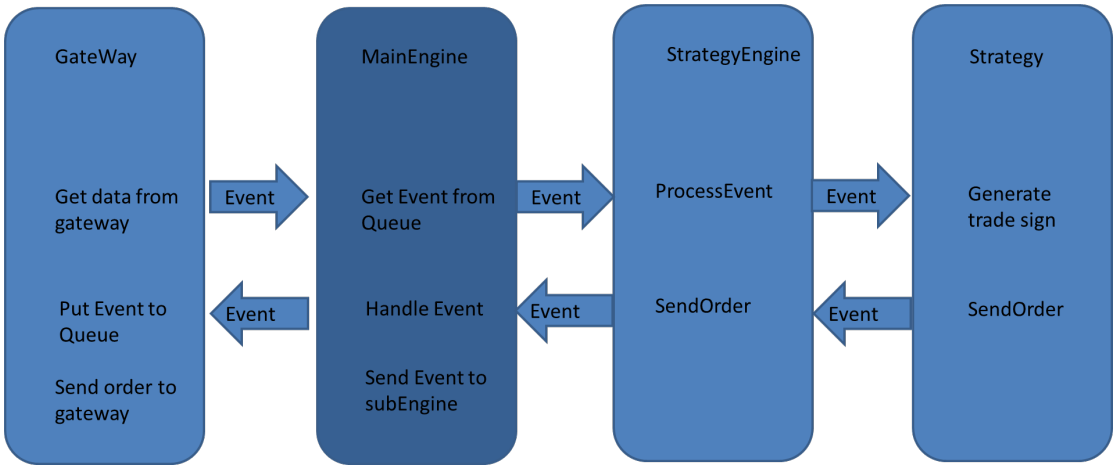
Trading System

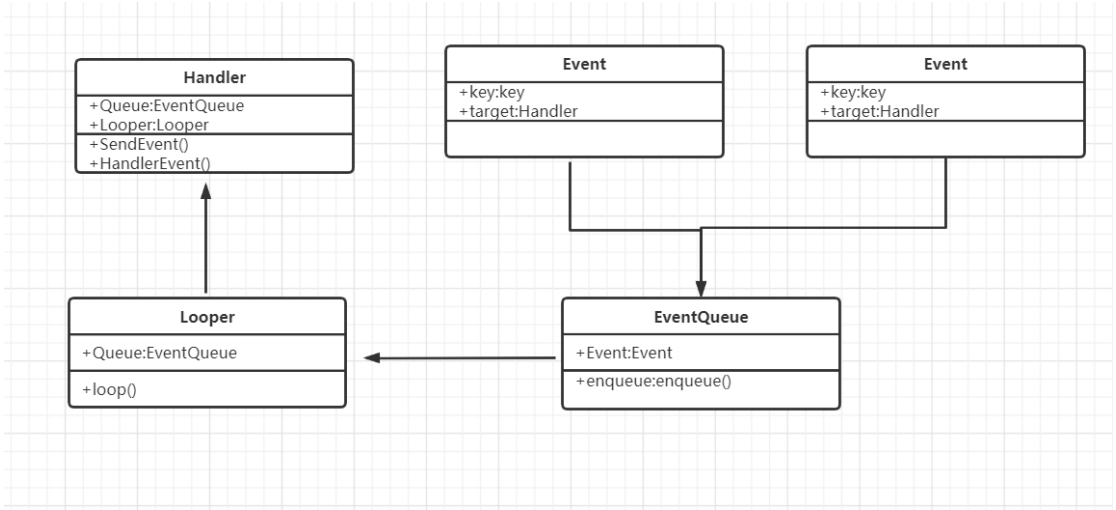
Oliver He

The code of the trading system this time basically realized the overall system framework. Including the entire event-driven system, the main engine, gateway, the definition of various data formats, and GUI, etc., it can realize the functions of subscription to stock data, and order submission, and basically completes the functions of the trading system. Compared with project4/5, the GUI and the entire data format were mainly improved this time, and due to the excessive time spent on the GUI, the Account module and Position module in the trading system were not yet perfect.

In the final version of the trading system, the main architecture of the trading system is completed, including the event engine, main engine, and GUI. The specific architecture diagram is as follows:



In this trading system, the most important and most important thing I use is the observer. I transformed the observer mode into event-driven. The specific structure diagram is as follows:



The biggest difference between this time and project4/5 is that the time type is determined. Different events use different time types, and different event types correspond to different processing methods. In this way, there is no need to traverse all methods for an event, which improves processing efficiency.

In this project, a large number of third-party libraries are used, such as Pyqt5 as the way to build the trading system GUI, pandas and numpy as the data processing method, ENUM as the parent class of the enumeration class, and so on. Using these third-party libraries has greatly improved the development efficiency

This project has the following three advantages and disadvantages: 1. The design was too large, which made me very painful in the early stage of the project. The huge project concept made the development progress extremely slow and I couldn't find the focus. 2. With the observer mode as the core, the entire trading system is designed, which allows the system development to find a direction. 3. Use classes to encapsulate data and methods.