Class Diagram for the Online Stock Brokerage System

Learn to create a class diagram for a stock brokerage system using the bottom-up approach.

We'll cover the following

In this lesson, we'll identify and design the classes, abstract classes, and interfaces based on the requirements that we have previously gathered from the interviewer in an online stock brokerage system.

Components of a stock brokerage system

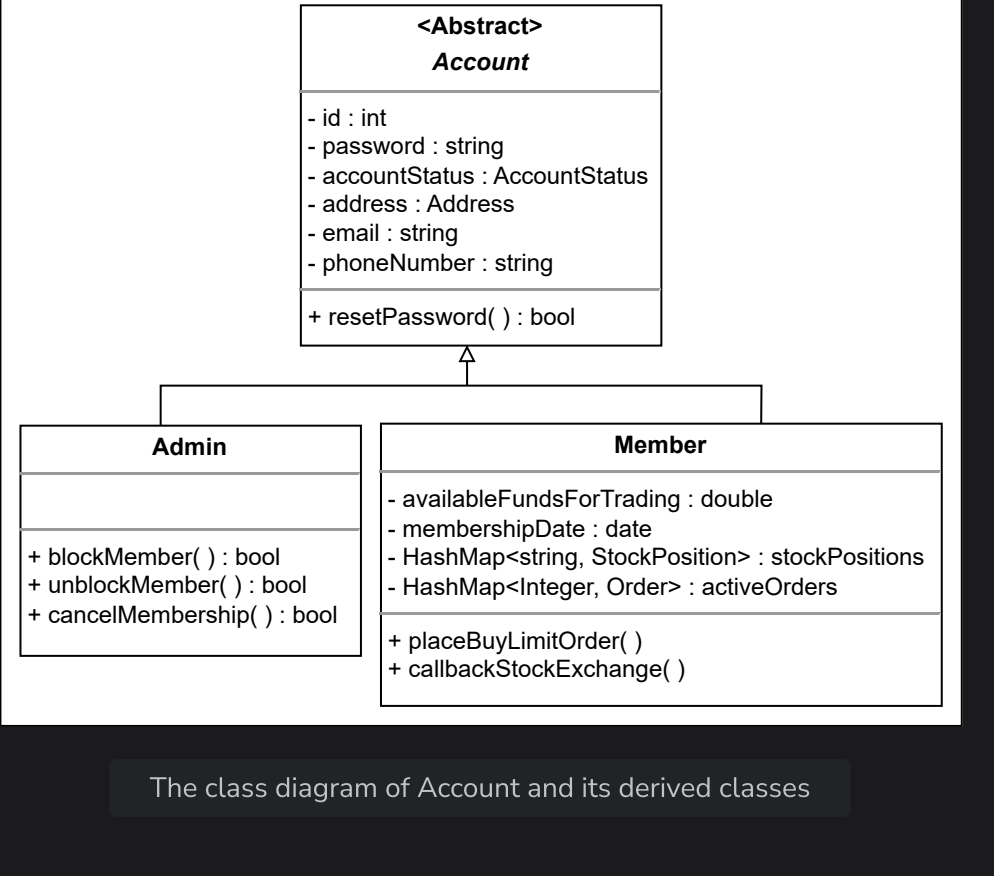
As mentioned earlier, we will design the online stock brokerage system using a bottom-up approach.

Account

`Account` is an abstract class that is used to store the account information of a person. This class has members like account ID, password, account status, address, email, and phone number. There can be two types of accounts, i.e., `Admin` and `Member`.

`Member`: They can search the stock, place an order to buy or sell stocks, create an account, start a membership, add stocks to the wishlist, add buying and selling limits, as well as perform transactions in many ways.

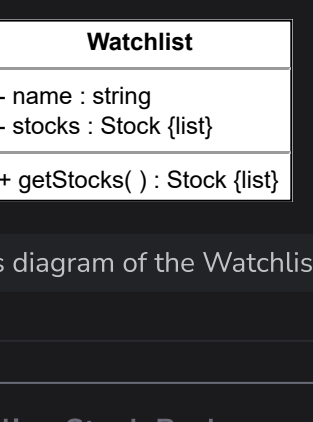
`Admin`: They can block or unblock members, and cancel their membership.



The class diagram of Account and its derived classes

Watchlist

A list of stocks that an investor keeps an eye on to profit from price drops is called a **watchlist**. The visual representation of the `Watchlist` class is provided below.



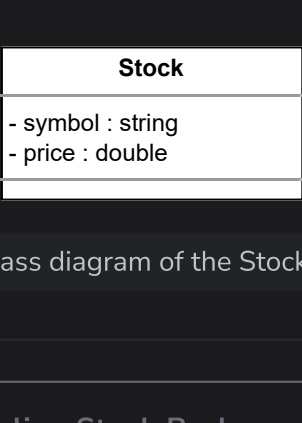
The class diagram of the Watchlist class

R2: Online Stock Brokerage System

R2: Users are allowed to have numerous watchlists consisting of different stock quotes.

Stock

A **stock** is also known as equity. It is a security that represents a portion of the issuing company's ownership. The `Stock` class will have a symbol, price, etc.



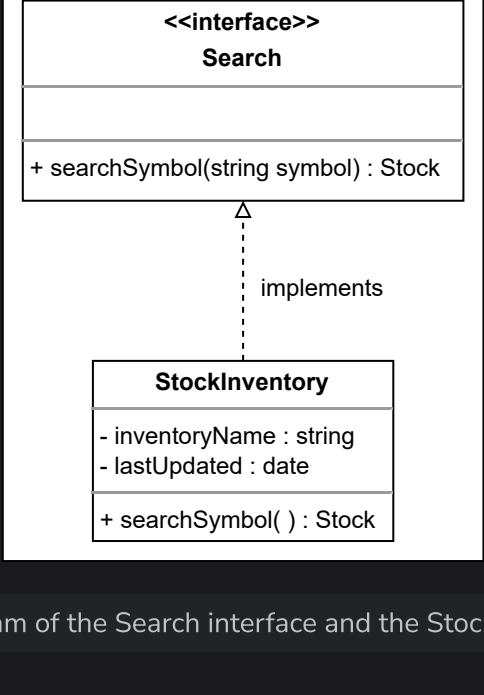
The class diagram of the Stock class

R1: Online Stock Brokerage System

R1: The system should allow the user to easily trade in stocks (buy or sell them).

Search and stock inventory

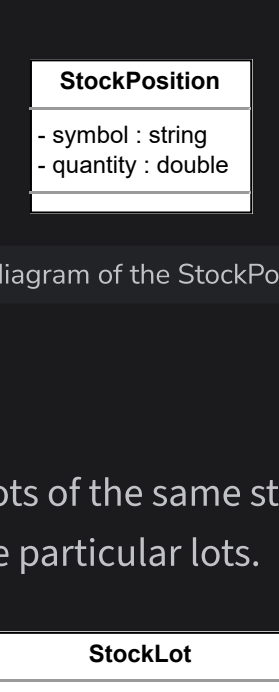
The `StockInventory` class will retrieve and keep up with the most recent stock values from the `StockExchange` class (defined later). The `StockInventory` class implements the `Search` interface.



The class diagram of the Search interface and the StockInventory class

Stock position

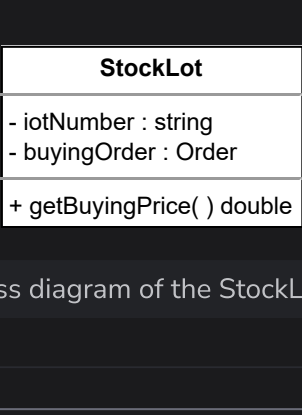
All the stocks that the user owns will be included in the `StockPosition` class.



The class diagram of the StockPosition class

Stock lot

A member may purchase various lots of the same stock at various dates. The `StockLot` class will represent these particular lots.



The class diagram of the StockLot class

R3: Online Stock Brokerage System

R3: Users may own different lots of the same stock. This implies that the system should be able to distinguish between several lots of the same stock if a user has purchased the same stock more than once.

Order

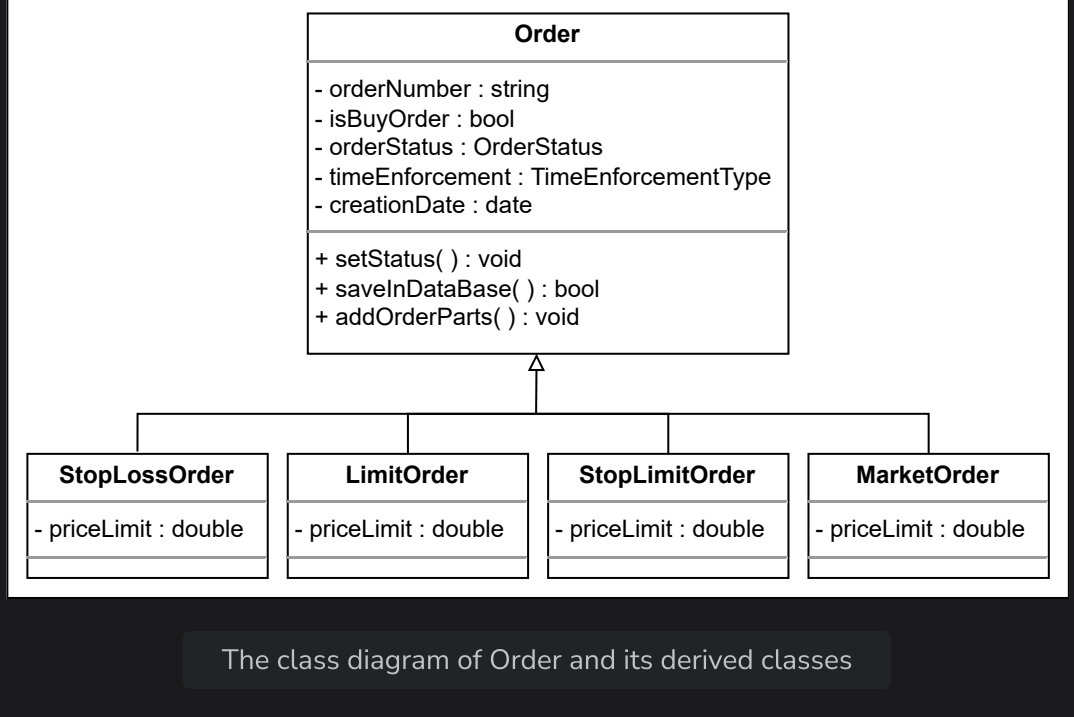
Members can place stock trading orders when they want to sell or acquire stock positions. There are four types of orders supported by the system:

`MarketOrder`: This allows customers to purchase or sell equities immediately at the market's going rate (current market price).

`LimitOrder`: A user may specify a price at which they wish to purchase or sell a stock.

`StopLossOrder`: An order to purchase or sell whenever the stock hits a specific price.

`StopLimitOrder`**:** The `StopLimitOrder` becomes a limit order to purchase or sell at the limit price, or better, if the stop price is reached.



The class diagram of Order and its derived classes

R5: Online Stock Brokerage System

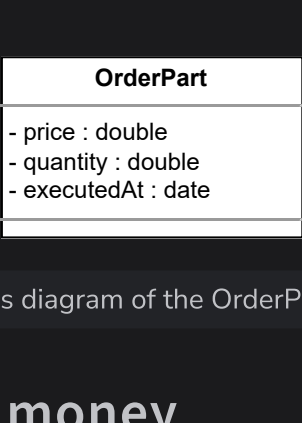
R5: The system should allow the user to order the stock trade of the types given below:

- **Market order:** Buy or sell stocks at the current market price.
- **Limit order:** Buy or sell stocks at the price set by the user.

- **Stop-loss order:** Buy or sell stocks when they reach a certain price.
- **Stop-limit order:** Buy or sell stocks with a restriction on the limit price (maximum price to be paid, minimum price to be received, etc).

Order part

Multiple order parts might be used to complete an order. The `OrderPart` class contains the price, quantity, and execution date.

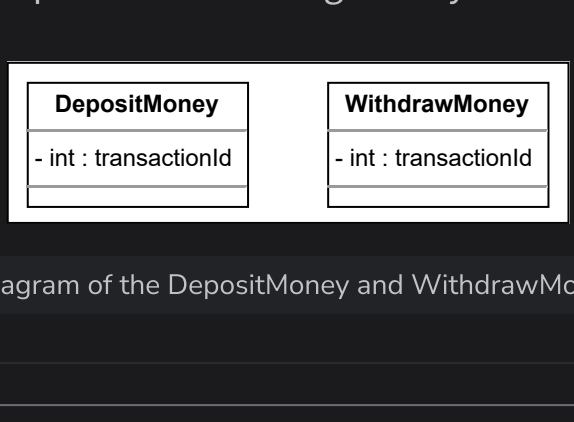


The class diagram of the OrderPart class

Deposit and withdraw money

The `DepositMoney` class represents a transfer of money from one party to another.

The `WithdrawMoney` class represents removing money from an account.



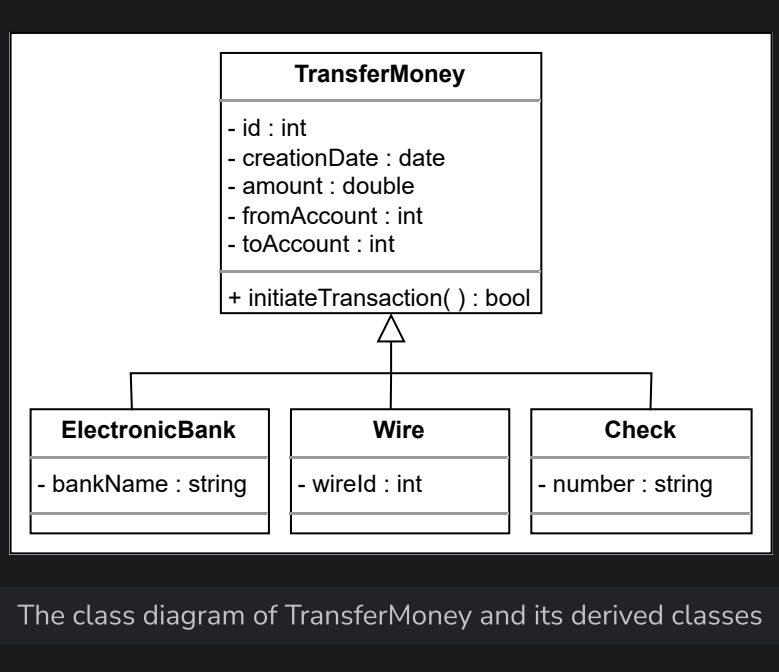
The class diagram of the DepositMoney and WithdrawMoney classes

R6: Online Stock Brokerage System

R6: The system should allow the user to make deposits and withdrawals using checks, wire transfers, or electronic bank transfers.

Transfer money

Users should be able to deposit and withdraw money either via check, wire, or electronic bank transfer.



The class diagram of TransferMoney and its derived classes

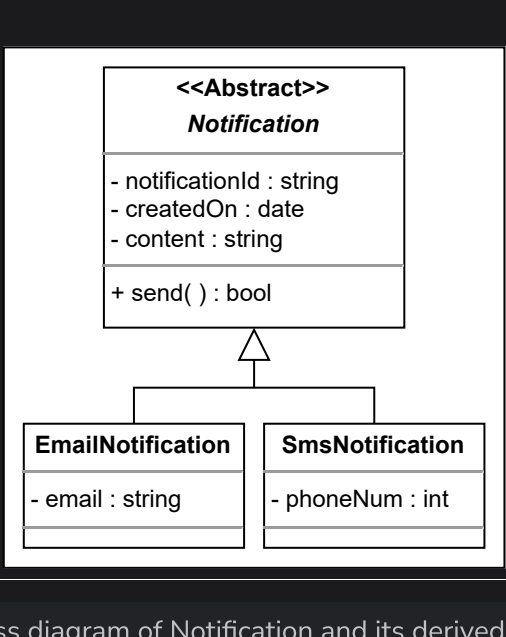
R6: Online Stock Brokerage System

R6: The system should allow the user to make deposits and withdrawals using checks, wire transfers, or electronic bank transfers.

Notification

`Notification` is an abstract class. This class is responsible for sending notifications when trade orders are executed. Every notification has an ID, creation date, and content in it. The notification can either be an SMS notification or an email notification.

The `SmsNotification` class requires the phone number of the member to send a notification. The `EmailNotification` class needs the email address of the member to send a notification. The relationship diagram of these classes is shown here:



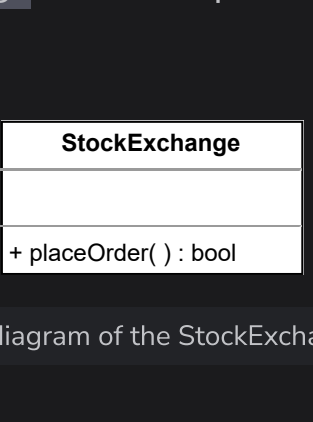
The class diagram of Notification and its derived classes

R4: Online Stock Brokerage System

R4: Every time a trade order is carried out, the system should be able to notify users.

Stock exchange

The stock brokerage system will get all stocks from the stock exchange and their current pricing. The `StockExchange` class is responsible for creating orders for trading stocks on the stock exchange.

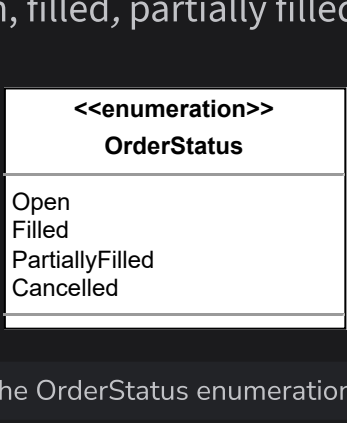


The class diagram of the StockExchange class

Enumerations and custom data types

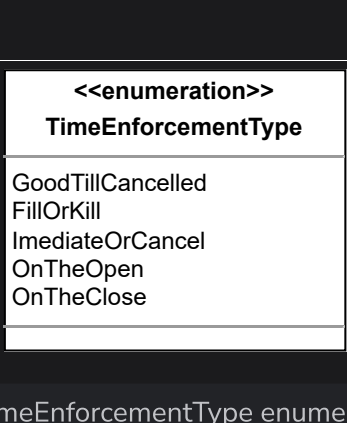
The following provides an overview of the enumerations and custom data types used in this problem:

- `OrderStatus`: We need to create an enumeration to keep track of the status of the order, whether it is open, filled, partially filled, or canceled.



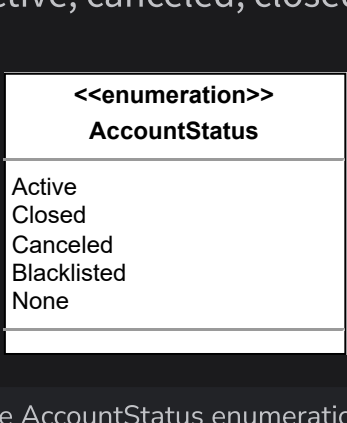
The OrderStatus enumerations

- `TimeEnforcementType`: We need to create an enumeration for the time enforcement type, whether it is good till canceled, fill or kill, immediate or cancel, on the open, or on the close.



The TimeEnforcementType enumerations

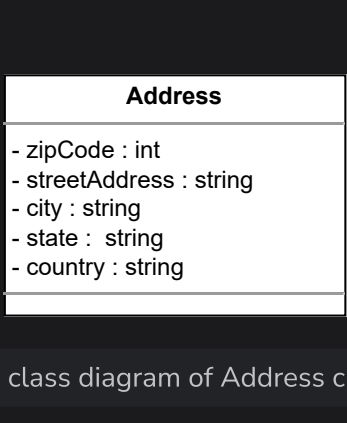
- `AccountStatus`: We need to create an enumeration to keep track of the status of the account, whether it is active, canceled, closed, blacklisted, or none.



The AccountStatus enumerations

Address

We also need to create a custom data type, `Address`, that will store the location of the user.



The class diagram of Address class

Relationship between the classes

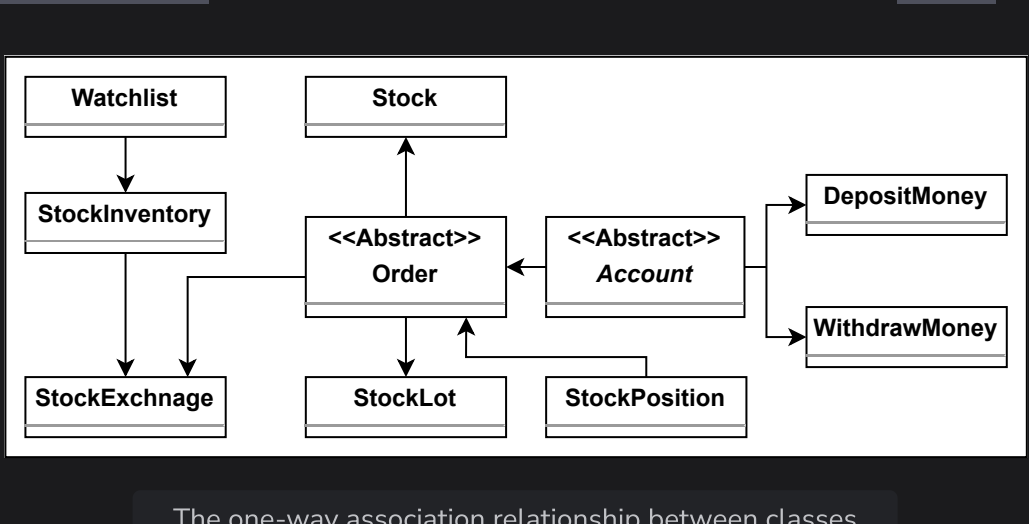
Now, we'll discuss the relationships between the classes we have defined above in our online stock brokerage system.

Association

The class diagram has the following association relationships:

One-way association

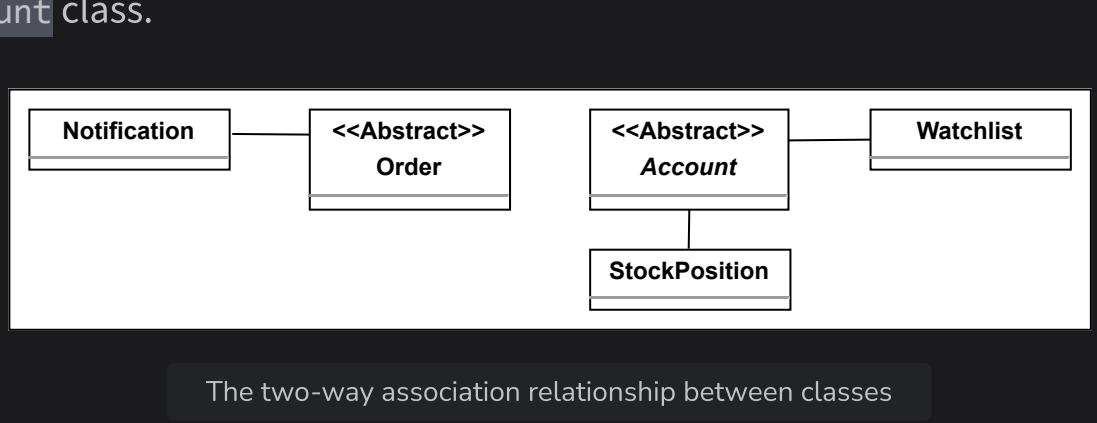
- The `StockInventory` class has a one-way association with `Watchlist` and `StockExchange`.
- The `Order` class has a one-way association with `Stock`, `StockExchange`, and `StockLot`.
- The `Account` class has a one-way association with `Order`, `DepositMoney`**,** and `WithdrawMoney`**.**
- The `StockPosition` class has a one-way association with the `Order` class.



The one-way association relationship between classes

Two-way association

- The `Notification` class has a two-way association with the `Order` class.
- Both `Watchlist` and `StockPosition` have a two-way association with the `Account` class.

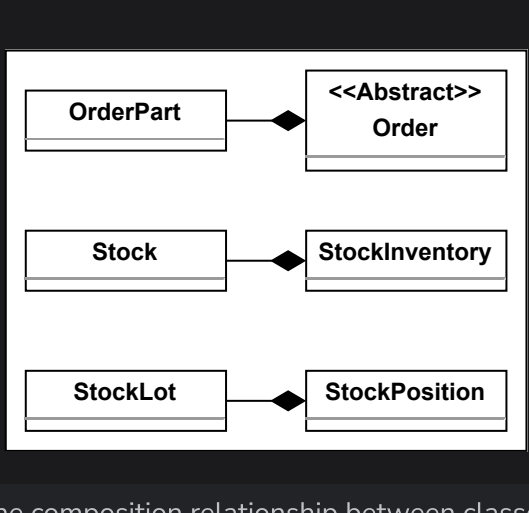


The two-way association relationship between classes

Composition

The class diagram has the following composition relationships:

- The `Order` class is composed of `OrderPart`.
- The `StockInventory` class is composed of `Stock`.
- The `StockPosition` class is composed of `StockLot`.



The composition relationship between classes

Inheritance

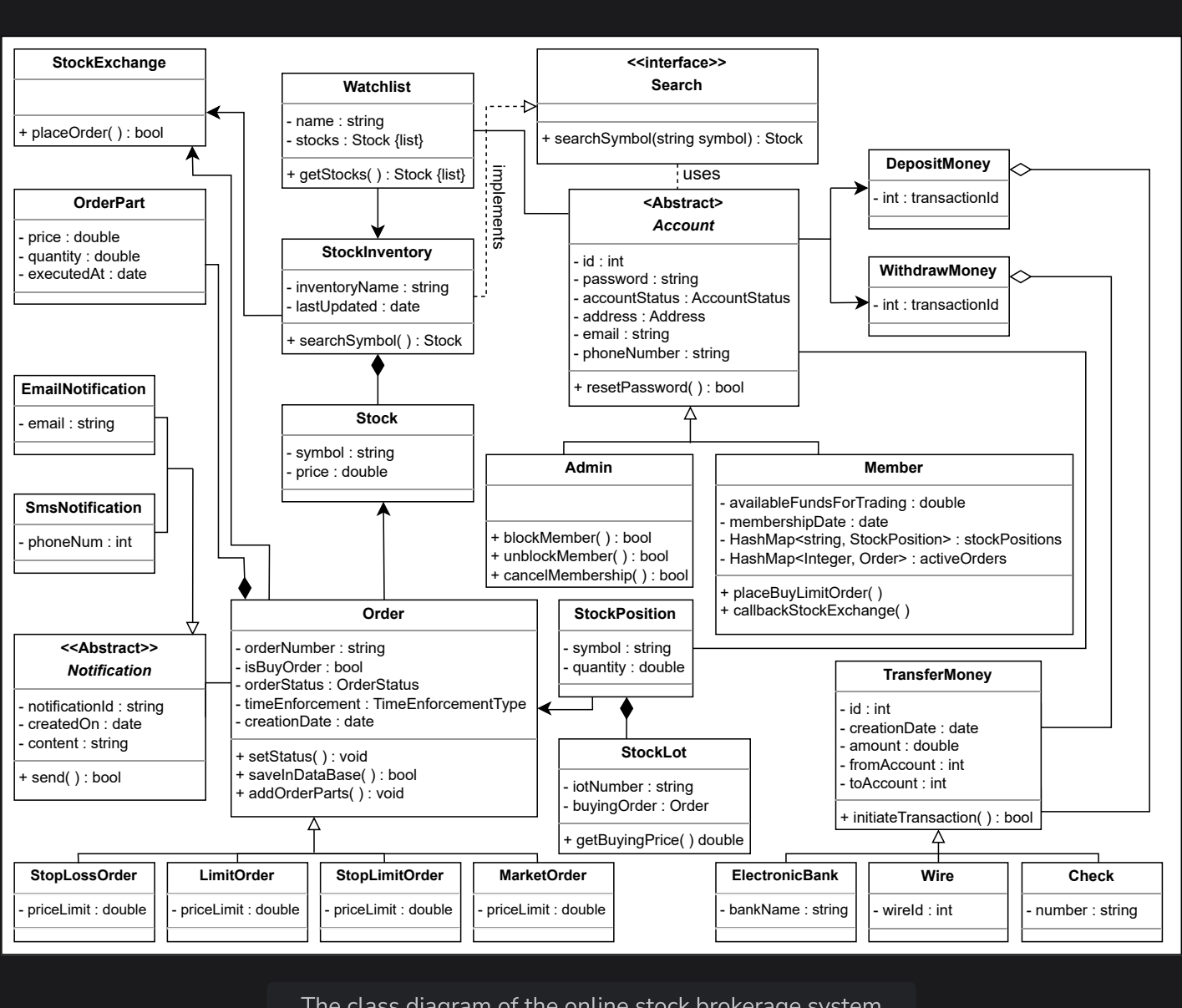
The following classes show an inheritance relationship:

- Both `Admin` and `Member` extend the `Account` class.
- The `MarketOrder`, `LimitOrder`, `StopLimitOrder`, and `StopLossOrder` classes extend the `Order` class.
- The `ElectronicBank`, `Wire`, and `Check` classes extend the `TransferMoney` class.
- The `SmsNotification` and `EmailNotification` classes extend the `Notification` class.
- The `StockInventory` class implements the `Search` interface.

> **Note:** We have already discussed the inheritance relationship between classes in the component section above one by one.

Class diagram of the online stock brokerage system

Here is the complete class diagram for our online stock brokerage system:



The class diagram of the online stock brokerage system

Design pattern

In the online stock brokerage system, there is going to be only one instance of the stock exchange, which encloses all the information and relations relating to the stock exchange. Therefore, we use the Singleton design pattern to ensure that only one instance for the class is created and this instance has a global point of access.

We can also use the Observer design pattern for our online stock brokerage system. Since the user has set buying and selling limits, the system observes stock prices, and when a stock reaches the specified price it buys and sells the stock automatically. Therefore, there is a need for an observer who observes the price of stock all the time.

AI-powered trainer

At this stage, everything should be clear. If you encounter any confusion or ambiguity, feel free to utilize the interactive AI-enabled widget below to seek clarification. This tool is designed to assist you in strengthening your understanding of the concepts.

Powered by AI | 20 Prompts Remaining

Prompt AI Widget

Our tool is designed to help you to understand concepts and ask any follow up questions. Ask a question to get started.

Enter Prompt Here...

We have successfully designed the class diagram for the online stock brokerage system. Let's move to the next lesson to see how we can construct the sequence diagram for the system.