# Getting Ready: An Online Stock Brokerage System

Understand the online stock brokerage system problem and learn the questions to further simplify this problem.

**We'll cover the following**

* [Problem definition](https://www.educative.io/courses/grokking-the-low-level-design-interview-using-ood-principles/getting-ready-the-amazon-online-shopping-system#Problem-definition)
* [Expectations from the interviewee](https://www.educative.io/courses/grokking-the-low-level-design-interview-using-ood-principles/getting-ready-the-amazon-online-shopping-system#Expectations-from-the-interviewee)
  + [Discoverability](https://www.educative.io/courses/grokking-the-low-level-design-interview-using-ood-principles/getting-ready-the-amazon-online-shopping-system#Discoverability)
  + [Visibility](https://www.educative.io/courses/grokking-the-low-level-design-interview-using-ood-principles/getting-ready-the-amazon-online-shopping-system#Visibility)
  + [Order type](https://www.educative.io/courses/grokking-the-low-level-design-interview-using-ood-principles/getting-ready-the-amazon-online-shopping-system#Order-type)
  + [Multiplicity](https://www.educative.io/courses/grokking-the-low-level-design-interview-using-ood-principles/getting-ready-the-amazon-online-shopping-system#Multiplicity)
* [Design approach](https://www.educative.io/courses/grokking-the-low-level-design-interview-using-ood-principles/getting-ready-the-amazon-online-shopping-system#Design-approach)
* [Design patterns](https://www.educative.io/courses/grokking-the-low-level-design-interview-using-ood-principles/getting-ready-the-amazon-online-shopping-system#Design-patterns)

## Problem definition

An **online stock brokerage** system acts as as an intermediary between the buyer and seller during the trade of the stocks. The system facilitates its users to buy and sell stocks online. It enables users to monitor and carry out their transactions and displays performance graphs for the various stocks in their portfolios. It also offers protection for client transactions and notifies them when stock changes reach certain levels.

The automated online stock brokerage system uses computers and the internet to speed up and reduce the cost of traditional stock trading. Additionally, this method provides quicker access to stock information, current market trends, and current stock prices.

The online stock brokerage system

## Expectations from the interviewee

An online stock brokerage system consists of multiple components. Each component has its own functionality and constraints. The following section provides an overview of some of the main expectations that the interviewer will want to hear you discuss in more detail during the interview.

### Discoverability

For an online stock brokerage system, discoverability is one of the key features. You can ask the following questions to know more about the system:

* How do the members search the stock inventory?
* How will the search surface result?

### Visibility

To get a better understanding of how the data is visible to the different users, you may ask the following questions:

* Can every member see the current levels of stock positions at any time?

### Order type

You may ask the interviewer about the types of orders the system should be able to handle by simply asking the question listed below:

* How many types of stock trade orders are the users able to place, for example, a market order, loss order, etc?

### Multiplicity

You may ask the interviewer questions related to the multiplicity of the system. These questions are listed below:

* Can the members have multiple watchlists containing multiple stock quotes?
* Can a member buy multiple lots of the same stock at different times?

## Design approach

We'll design this online stock brokerage system using the bottom-up design approach. For this purpose, we will follow the steps below:

* Identify and design the simple components first, like the stock and stock position.
* Use these small components to design bigger components, such as the order and stock inventory that can be composed of multiple stock items.
* Repeat the steps above until we design the whole stock brokerage system.

## Design patterns

It is always a good practice to discuss the design patterns that the online stock brokerage system falls under, during the interview. Stating the design patterns will give the interviewer a positive impression and shows that the interviewee is well-versed in the advanced concepts of object-oriented design.

The following design patterns can be used to design the online stock brokerage system:

* Singleton design pattern
* Observer design pattern

Let's explore the requirements of the online stock brokerage system in the next lesson.

# Requirements for the Online Stock Brokerage System

Learn about all the requirements for an online stock brokerage system.

**We'll cover the following**

* [Requirement collection](https://www.educative.io/courses/grokking-the-low-level-design-interview-using-ood-principles/getting-ready-the-amazon-online-shopping-system#Requirement-collection)

In this lesson, we’ll list the requirements of the online stock brokerage system. This is a very crucial step since requirements define the scope of a problem, so getting them right from the interviewer and understanding them well will make the design of the rest of the system smooth and easy.

We’ll use the notational convention to identify each requirement with a unique label "Rn", where "R" is short for Requirement and "n" is a natural number.

## Requirement collection

The following are the requirements that we have defined for the online stock brokerage system:

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

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

**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.

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

**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).

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

We've identified our requirements for the problem, and in the next lesson, we will define different use cases for the online stock brokerage system.

# Use Case Diagram for the Online Stock Brokerage System

Learn how to define use cases and create the corresponding use case diagram for the online stock brokerage system.

**We'll cover the following**

* [System](https://www.educative.io/courses/grokking-the-low-level-design-interview-using-ood-principles/getting-ready-the-amazon-online-shopping-system#System)
* [Actors](https://www.educative.io/courses/grokking-the-low-level-design-interview-using-ood-principles/getting-ready-the-amazon-online-shopping-system#Actors)
  + [Primary actors](https://www.educative.io/courses/grokking-the-low-level-design-interview-using-ood-principles/getting-ready-the-amazon-online-shopping-system#Primary-actors)
  + [Secondary actors](https://www.educative.io/courses/grokking-the-low-level-design-interview-using-ood-principles/getting-ready-the-amazon-online-shopping-system#Secondary-actors)
* [Use cases](https://www.educative.io/courses/grokking-the-low-level-design-interview-using-ood-principles/getting-ready-the-amazon-online-shopping-system#Use-cases)
  + [Member](https://www.educative.io/courses/grokking-the-low-level-design-interview-using-ood-principles/getting-ready-the-amazon-online-shopping-system#Member)
  + [Admin](https://www.educative.io/courses/grokking-the-low-level-design-interview-using-ood-principles/getting-ready-the-amazon-online-shopping-system#Admin)
  + [System](https://www.educative.io/courses/grokking-the-low-level-design-interview-using-ood-principles/getting-ready-the-amazon-online-shopping-system#System)
* [Relationships](https://www.educative.io/courses/grokking-the-low-level-design-interview-using-ood-principles/getting-ready-the-amazon-online-shopping-system#Relationships)
  + [Generalization](https://www.educative.io/courses/grokking-the-low-level-design-interview-using-ood-principles/getting-ready-the-amazon-online-shopping-system#Generalization)
  + [Associations](https://www.educative.io/courses/grokking-the-low-level-design-interview-using-ood-principles/getting-ready-the-amazon-online-shopping-system#Associations)
  + [Include](https://www.educative.io/courses/grokking-the-low-level-design-interview-using-ood-principles/getting-ready-the-amazon-online-shopping-system#Include)
* [Use case diagram](https://www.educative.io/courses/grokking-the-low-level-design-interview-using-ood-principles/getting-ready-the-amazon-online-shopping-system#Use-case-diagram)

Let's build the use case diagram of the online stock brokerage system and understand the relationship between its different components.

First, we’ll define the different elements of our online stock brokerage, followed by the complete use case diagram of the system.

## System

Our system is a "stock brokerage."

## Actors

Now, we’ll define the main actors of our online stock brokerage system.

### Primary actors

* **Member:**This actor 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 three ways. It can also create and update accounts, perform login and logout, and cancel membership.

### Secondary actors

* **Admin:** This can create and update accounts, perform login and logout, cancel membership, and block or unblock members.
* **System:** This is responsible for notifying the member about the order status and transactions. It can also fetch stock quotes from the stock exchange.

## Use cases

In this section, we will define the use cases for the online stock brokerage system. We have listed down the use cases according to their respective interactions with a particular actor.

**Note:**You will see some use cases occurring multiple times, because they are shared among different actors in the system.

### Member

* **Create account:**To create a new account inthe online stock brokerage system
* **Cancel membership:**To cancel an old online stock brokerage system membership
* **Update account:**To update account details in the online stock brokerage system
* **Login/Logout:** To log in and out of the online stock brokerage system account
* **Cancel order:**To cancel any buying or selling stock order in the online stock brokerage system
* **View stock positions:** To view live positions and pricing of stocks in the stock exchange
* **Add stock to wishlist:**To add a stock to a member's wishlist
* **Create/update wishlist**: To add and remove stocks from the existing or new wishlist
* **Search stock inventory:** To search for a stock in the stock inventory
* **Place order**: To place an order to buy or sell the order
* **Deposit/withdraw money:**To deposit or withdraw money to and from your account

### Admin

* **Create account:**To create a new account inthe online stock brokerage system
* **Cancel membership:**To cancel an old online stock brokerage system membership
* **Update account:**To update account details in the online stock brokerage system
* **Login/Logout:** To log in and out of the online stock brokerage system account
* **Block/unblock member:** To block and unblock a member from the online stock brokerage system

### System

* **Fetch stocks quotes from stock exchange:**To fetch stock price from the stock inventory in the online stock brokerage system
* **Send order status change notification:** To notify about the change in the order status in the online stock brokerage system
* **Send deposit/withdrawal status change notification:**To notify about the deposit and withdrawal of money

## Relationships

This section describes the relationships between and among actors and their use cases.

### Generalization

* The "Electronic bank transfer," "Wire transfer," and "Check Transfer" use cases are used for transactions. Hence, they have a generalization relationship with the "Transaction" use case.
* When an order is placed, it could be the market, limit, stop-loss or stop-limit order. Therefore, the "Place market order," "Place limit order," "Place stop-limit order," and "Place stop-loss order" use cases have a generalization relationship with the "Place order" use case.

### Associations

The below table shows the association relationship between actors and their use cases.

|  |  |  |
| --- | --- | --- |
| **Member** | **Admin** | **System** |
| Create account | Create account | Fetch stocks quotes from stock exchange |
| Cancel membership | Cancel membership | Send order status change notification |
| Update account | Update account | Send deposit/withdrawal status change notification |
| Login/Logout | Login/Logout |  |
| Cancel order | Block/unblock member |
| View stock positions |  |
| Add stock to wishlist |
| Create/update wishlist |
| Search stock inventory |
| Place order |
| Deposit/withdraw money |

### Include

The "Deposit/withdraw money" use casehas an include relationship with the "Transaction" use case, since a money transaction takes place when money is deposited or withdrawn.

## Use case diagram

Here's the use case diagram of the online stock brokerage system:

A screenshot of a computer screen

Description automatically generated

# 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**

* [Components of a stock brokerage system](https://www.educative.io/courses/grokking-the-low-level-design-interview-using-ood-principles/getting-ready-the-amazon-online-shopping-system#Components-of-a-stock-brokerage-system)
  + [Account](https://www.educative.io/courses/grokking-the-low-level-design-interview-using-ood-principles/getting-ready-the-amazon-online-shopping-system#Account)
  + [Watchlist](https://www.educative.io/courses/grokking-the-low-level-design-interview-using-ood-principles/getting-ready-the-amazon-online-shopping-system#Watchlist)
  + [Stock](https://www.educative.io/courses/grokking-the-low-level-design-interview-using-ood-principles/getting-ready-the-amazon-online-shopping-system#Stock)
  + [Search and stock inventory](https://www.educative.io/courses/grokking-the-low-level-design-interview-using-ood-principles/getting-ready-the-amazon-online-shopping-system#Search-and-stock-inventory)
  + [Stock position](https://www.educative.io/courses/grokking-the-low-level-design-interview-using-ood-principles/getting-ready-the-amazon-online-shopping-system#Stock-position)
  + [Stock lot](https://www.educative.io/courses/grokking-the-low-level-design-interview-using-ood-principles/getting-ready-the-amazon-online-shopping-system#Stock-lot)
  + [Order](https://www.educative.io/courses/grokking-the-low-level-design-interview-using-ood-principles/getting-ready-the-amazon-online-shopping-system#Order)
  + [Order part](https://www.educative.io/courses/grokking-the-low-level-design-interview-using-ood-principles/getting-ready-the-amazon-online-shopping-system#Order-part)
  + [Deposit and withdraw money](https://www.educative.io/courses/grokking-the-low-level-design-interview-using-ood-principles/getting-ready-the-amazon-online-shopping-system#Deposit-and-withdraw-money)
  + [Transfer money](https://www.educative.io/courses/grokking-the-low-level-design-interview-using-ood-principles/getting-ready-the-amazon-online-shopping-system#Transfer-money)
  + [Notification](https://www.educative.io/courses/grokking-the-low-level-design-interview-using-ood-principles/getting-ready-the-amazon-online-shopping-system#Notification)
  + [Stock exchange](https://www.educative.io/courses/grokking-the-low-level-design-interview-using-ood-principles/getting-ready-the-amazon-online-shopping-system#Stock-exchange)
  + [Enumerations and custom data types](https://www.educative.io/courses/grokking-the-low-level-design-interview-using-ood-principles/getting-ready-the-amazon-online-shopping-system#Enumerations-and-custom-data-types)
    - [Address](https://www.educative.io/courses/grokking-the-low-level-design-interview-using-ood-principles/getting-ready-the-amazon-online-shopping-system#Address)
* [Relationship between the classes](https://www.educative.io/courses/grokking-the-low-level-design-interview-using-ood-principles/getting-ready-the-amazon-online-shopping-system#Relationship-between-the-classes)
  + [Association](https://www.educative.io/courses/grokking-the-low-level-design-interview-using-ood-principles/getting-ready-the-amazon-online-shopping-system#Association)
    - [One-way association](https://www.educative.io/courses/grokking-the-low-level-design-interview-using-ood-principles/getting-ready-the-amazon-online-shopping-system#One-way-association)
    - [Two-way association](https://www.educative.io/courses/grokking-the-low-level-design-interview-using-ood-principles/getting-ready-the-amazon-online-shopping-system#Two-way-association)
  + [Composition](https://www.educative.io/courses/grokking-the-low-level-design-interview-using-ood-principles/getting-ready-the-amazon-online-shopping-system#Composition)
  + [Inheritance](https://www.educative.io/courses/grokking-the-low-level-design-interview-using-ood-principles/getting-ready-the-amazon-online-shopping-system#Inheritance)
* [Class diagram of the online stock brokerage system](https://www.educative.io/courses/grokking-the-low-level-design-interview-using-ood-principles/getting-ready-the-amazon-online-shopping-system#Class-diagram-of-the-online-stock-brokerage-system)
* [Design pattern](https://www.educative.io/courses/grokking-the-low-level-design-interview-using-ood-principles/getting-ready-the-amazon-online-shopping-system#Design-pattern)

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

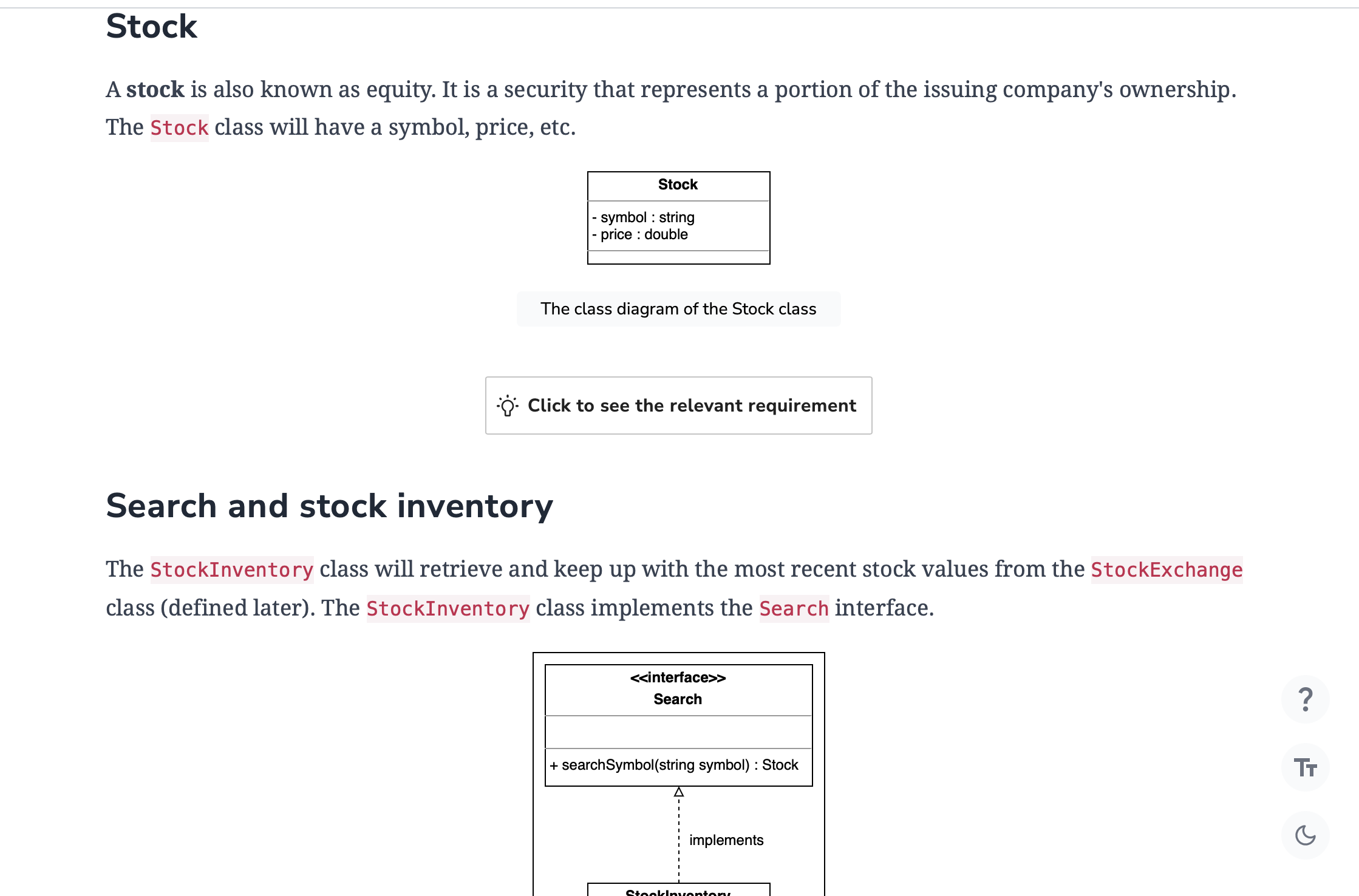
Account is an abstract class that is used to store the account information of a person. This class has members like account ID, password, account status,address,email,andphone number. There can be two types of accounts, i.e., Admin and Member.

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.

A screenshot of a computer

Description automatically generated



A screenshot of a computer

Description automatically generated

A screenshot of a computer screen

Description automatically generated

A screenshot of a computer

Description automatically generated

A screenshot of a computer

Description automatically generated

A screenshot of a computer screen

Description automatically generated

A screenshot of a computer

Description automatically generated

A screenshot of a computer

Description automatically generated

A screenshot of a computer

Description automatically generated

A screenshot of a computer

Description automatically generated

### Inheritance

The following classes show an inheritance relationship:

* Both Admin and Member extend the Account class.
* The MarketOrder, LimitOrder, StopLimitOrder, and StopLossOrderclasses 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:

A diagram of a data flow

Description automatically generated

## 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.

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.

# Sequence Diagram for the Online Stock Brokerage System

Visualize the sequence diagram for selling stock in an online stock brokerage system.

**We'll cover the following**

* [Selling a stock](https://www.educative.io/courses/grokking-the-low-level-design-interview-using-ood-principles/getting-ready-the-amazon-online-shopping-system#Selling-a-stock)

Sequence diagrams are a great way to understand the interactions between different entities and objects in the system. There can be different sequence diagrams that we can create for the online stock brokerage system. In this lesson, we will create a sequence diagram for the following interaction:

* **Selling a stock:** A member wants to sell a stock to the stock market.

## Selling a stock

The sequence diagram for selling a stock should have the following actors and objects that will interact with each other:

* **Actor:** Member
* **Objects:** StockInventory and StockExchange

Here are the steps involved in selling a stock:

1. A member selects a stock and its selling parameters: order type, stock quantity, time price limit, and time enforcement.
2. If the stock is available in the stock inventory:
   1. The stock quantity is deducted from the stock inventory.
   2. The stock order details will be sent to the stock exchange.
   3. The stock exchange acknowledges it.
   4. The member will be notified about the order placement.
3. Else if, enough stock is not available:
   1. The member will be notified about stock unavailability.

Based on the order above, the sequence diagram for selling a stock in an online stock brokerage system is provided below.

A screenshot of a computer screen

Description automatically generated

Next, let's look at the activity diagrams for an online stock brokerage system to understand the control flow of the system.

# Activity Diagram for the Online Stock Brokerage System

Create an activity diagram for the online stock brokerage system.

**We'll cover the following**

* [Buying a stock](https://www.educative.io/courses/grokking-the-low-level-design-interview-using-ood-principles/getting-ready-the-amazon-online-shopping-system#Buying-a-stock)
  + [States](https://www.educative.io/courses/grokking-the-low-level-design-interview-using-ood-principles/getting-ready-the-amazon-online-shopping-system#States)
  + [Actions](https://www.educative.io/courses/grokking-the-low-level-design-interview-using-ood-principles/getting-ready-the-amazon-online-shopping-system#Actions)

Activity diagrams are a great way to visualize the flow of messages from one activity to the other in the system. There can be different activity diagrams that we can create for our online stock brokerage system. In this lesson, we will create an activity diagram for the following activity.

## Buying a stock

The states and actions involved in this activity diagram are listed below.

### States

**Initial state:**The member selects a stock to buy.

**Final state:**There are two final states present in this activity diagram, shown below:

* The member successfully buys a stock from the online stock brokerage system.
* Due to the unavailability of a sufficient deposit, members aren't able to buy the stocks.

### Actions

The member selects the stock to buy, order type, stock quantity, time price limit, and time enforcement. Then, the system will check if the member has a sufficient deposit in the account. If available, the system then deducts funds from the account, sends order details to the stock exchange, and notifies the member about order success when it receives acknowledgment from the stock exchange.

Based on the order above, the activity diagram for buying a stock at the online stock brokerage system is provided below:

A diagram of a stock brokerage system

Description automatically generated

# Code for the Online Stock Brokerage System

Write the object-oriented code to implement the design of the ‘online stock brokerage system’ problem.

**We'll cover the following**

* [Online stock brokerage system classes](https://www.educative.io/courses/grokking-the-low-level-design-interview-using-ood-principles/getting-ready-the-amazon-online-shopping-system#Online-stock-brokerage-system-classes)
  + [Enumerations and custom data type](https://www.educative.io/courses/grokking-the-low-level-design-interview-using-ood-principles/getting-ready-the-amazon-online-shopping-system#Enumerations-and-custom-data-type)
  + [Account](https://www.educative.io/courses/grokking-the-low-level-design-interview-using-ood-principles/getting-ready-the-amazon-online-shopping-system#Account)
  + [Watchlist and stock](https://www.educative.io/courses/grokking-the-low-level-design-interview-using-ood-principles/getting-ready-the-amazon-online-shopping-system#Watchlist-and-stock)
  + [Search and stock inventory](https://www.educative.io/courses/grokking-the-low-level-design-interview-using-ood-principles/getting-ready-the-amazon-online-shopping-system#Search-and-stock-inventory)
  + [Stock position and stock lot](https://www.educative.io/courses/grokking-the-low-level-design-interview-using-ood-principles/getting-ready-the-amazon-online-shopping-system#Stock-position-and-stock-lot)
  + [Order](https://www.educative.io/courses/grokking-the-low-level-design-interview-using-ood-principles/getting-ready-the-amazon-online-shopping-system#Order)
  + [Transfer money](https://www.educative.io/courses/grokking-the-low-level-design-interview-using-ood-principles/getting-ready-the-amazon-online-shopping-system#Transfer-money)
  + [Notification](https://www.educative.io/courses/grokking-the-low-level-design-interview-using-ood-principles/getting-ready-the-amazon-online-shopping-system#Notification)
  + [Stock exchange](https://www.educative.io/courses/grokking-the-low-level-design-interview-using-ood-principles/getting-ready-the-amazon-online-shopping-system#Stock-exchange)
* [Wrapping up](https://www.educative.io/courses/grokking-the-low-level-design-interview-using-ood-principles/getting-ready-the-amazon-online-shopping-system#Wrapping-up)

We've reviewed different aspects of the online stock brokerage system and observed the attributes attached to the problem using various UML diagrams. Let’s explore the more practical side of things, where we will work on implementing the online stock brokerage system using multiple languages. This is usually the last step in an object-oriented design interview process.

We have chosen the following languages to write the skeleton code of the different classes present in the online stock brokerage system:

* Java
* C#
* Python
* C++
* JavaScript

## Online stock brokerage system classes

In this section, we’ll provide the skeleton code of the classes designed in the class diagram lesson.

**Note:** For simplicity, we are not defining getter and setter functions. The reader can assume that all class attributes are private and accessed through their respective public getter methods and modified only through their public method functions.

### Enumerations and custom data type

First of all, we will define all the enumerations required in the stock brokerage system. According to the class diagram, there are three enumerations used in the system i.e. OrderStatus, TimeEnforcementType and AccountStatus.The code to implement these enumerations and custom data types is as follows:

**Note:** JavaScript does not support enumerations, so we will be using the Object.freeze() method as an alternative that freezes an object and prevents further modifications.

Java

// Enumeration

enum OrderStatus {

OPEN,

FILLED,

PARTIALLY\_FILLED,

CANCELED

}

enum TimeEnforcementType {

GOOD\_TILL\_CANCELED,

FILL\_OR\_KILL,

IMMEDIATE\_OR\_CANCEL,

ON\_THE\_OPEN,

ON\_THE\_CLOSE

}

enum AccountStatus {

ACTIVE,

CLOSED,

CANCELED,

BLACKLISTED,

NONE

}

// Custom Address data type class

public class Address {

private int zipCode;

private String address;

private String city;

private String state;

private String country;

}

### Account

The Account class will be an abstract class, which will have the actors, Admin and Member, as child classes. The definition of these classes is given below:

// Account is an abstract class

public abstract class Account {

private String id;

private String password;

private String name;

private AccountStatus status;

private Address address;

private String email;

private String phone;

public abstract boolean resetPassword();

}

public class Member extends Account {

private double availableFundsForTrading;

private Date dateOfMembership;

private HashMap<string, StockPosition> stockPositions;

private HashMap<Integer, Order> activeOrders;

public ErrorCode placeSellLimitOrder(string stockId, float quantity, int limitPrice, TimeEnforcementType enforcementType);

public ErrorCode placeBuyLimitOrder(string stockId, float quantity, int limitPrice, TimeEnforcementType enforcementType);

public void callbackStockExchange(int orderId, List<OrderPart> orderParts, OrderStatus status);

public boolean resetPassword(){

// definition

}

}

public class Admin extends Account {

public boolean blockMember();

public boolean unblockMember();

public boolean cancelMembership();

public boolean resetPassword(){

// definition

}

}

### Watchlist and stock

A Watchlist class is a list of stocks that an investor keeps an eye on, to profit from price drops. The Stock class is an equity or a security that represents a portion of the issuing company's ownership.

Java

C#

Pytho

public class Watchlist {

private String name;

private List<Stock> stocks;

public List<Stock> getStocks();

}

public class Stock {

private String symbol;

private double price;

}

A screenshot of a computer

Description automatically generated

public class StockPosition {

private String symbol;

private double quantity;

}

public class StockLot {

private String iotNumber;

private Order buyingOrder;

public double getBuyingPrice();

}

### Order

Members can place stock trading orders when they want to sell or acquire StockPosition.

public class OrderPart {

private double price;

private double quantity;

private Date executedAt;

}

// Order is an abstract class

public abstract class Order {

private String orderNumber;

public boolean isBuyOrder;

private OrderStatus status;

private TimeEnforcementType timeEnforcement;

private Date creationTime;

private HashMap<int, OrderPart> parts;

public void setStatus(OrderStatus status);

public boolean saveInDatabase();

public void addOrderParts(OrderParts parts);

}

public class LimitOrder extends Order {

}

public class StopLimitOrder extends Order {

}

public class StopLossOrder extends Order {

}

public class MarketOrder extends Order {

}

### Transfer money

Members should be able to deposit and withdraw money either via Check, Wire, or ElectronicBank transfer.

// TransferMoney is an abstract class

public abstract class TransferMoney {

private int id;

private Date creationDate;

public int fromAccount;

private int toAccount;

public abstract boolean initiateTransaction();

}

public class ElectronicBank extends TransferMoney {

private String bankName;

public boolean initiateTransaction(){

// definition

}

}

public class Wire extends TransferMoney {

private int wire;

public boolean initiateTransaction(){

// definition

}

}

public class Check extends TransferMoney {

private String checkNumber;

public boolean initiateTransaction(){

// definition

}

}

public class DepositMoney {

private int transactionId;

public boolean initiateTransaction(){

// definition

}

}

public class WithdrawMoney {

private int transactionId;

}

A screenshot of a computer

Description automatically generated

A screenshot of a computer

Description automatically generated

We've explored the complete design of an online stock brokerage system in this chapter. We've looked at how a basic online stock brokerage system can be visualized using various UML diagrams and designed using object-oriented principles and design patterns.