Zhiwei Liu

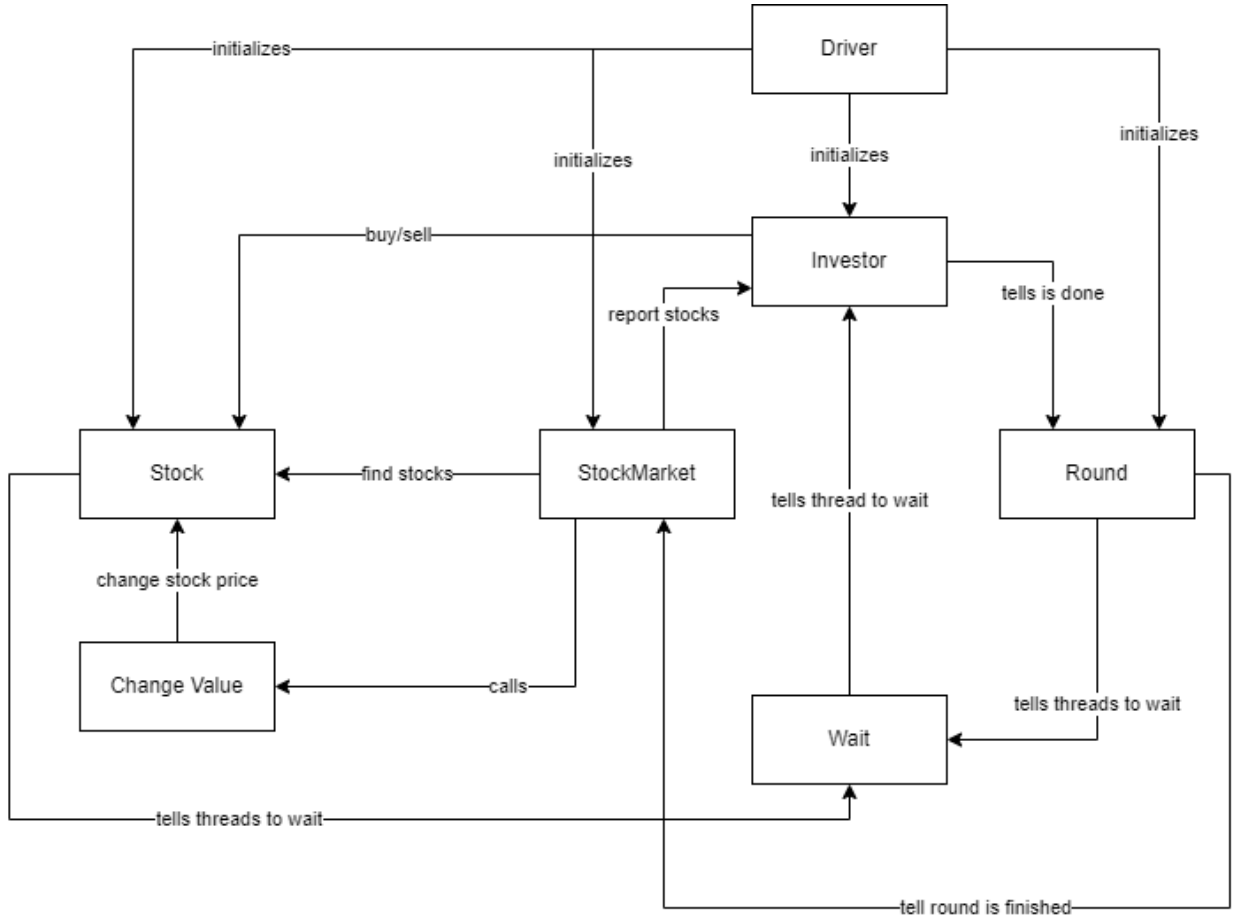
Overview

This project is focused on creating a stock market application using multiple threads in Java. Multiple threads of Investors will attempt to interact with Stocks that have a limit on how many threads can access it at a time.

System Architecture

The main function the user interacts with is the Driver class. The Driver class is where all the parameters of the simulation is decided upon by the user. Once all the parameters have been set, the Driver class initializes all the objects that are needed. Then the Driver calls every Investor thread, calling *start()* and then *join()* on all of them. Within the Investor threads, they loop through the main algorithm that simulates a stock market. The Investors first check if they own any Stocks, and if they do, how much they spent on average for each Stock. If there is something there, then the Investor looks at the current value of those owned Stock listed on the Stock Market. If the value is higher than the average, then the Investor sells all the Stocks they own of that Stock. Otherwise, if the Investor doesn't own any Stock, or the Stock's current value is less than the average the Investor spent buying that Stock, the Investor moves on to seeing if there is any Stock they can buy. The Investor has a budget value, which is determined by dividing their current money by 2. The Investors look through the Stock Market, looking for Stocks who are within budget. If there are Stocks under budget, the Investor then buys as much of the cheapest Stock they can. Otherwise, if the Investor can't do anything, they just don't do anything for this Round. The Investor thread then calls on the Round class, indicating that the current thread is done acting. If all Investor threads are done actings, then the Round class will call on the Stock Market class to change the values of every Stock then notify all the Investor threads. Otherwise, if there are still Investor threads running, the thread that is done acting will be told to wait.

System Block Diagram



Stock Market:
This class has a list of all the Stock objects. When the round is done, it will go through the list, calling on each instance of Stock to change their value.

Round:
This is a class that handles the changing of rounds. A round is determined by when all the Investor threads have done an action for this iteration. The Round class has a hashmap of every Investor thread that keeps track of whether that thread is done acting. When an Investor thread is done, it will notify the Round class. If every Investor thread is done acting, then the Round class will tell the Stock Market that the round is finished and to change the Stock values. Otherwise, the class will tell the Investor thread to wait for all the other threads to finish.

Stock:
This is the stock class. Each Stock object contains a value associated with it. This value will change at the end of every loop. Each Stock instance can only interact with one Investor thread at a time. If more than one Investor thread tries to interact with the Stock object, they will have to wait in a Linked List called queue. Once the Investor thread currently interacting with the Stock object is finished, the other threads are notified and the thread in the first position of the queue is up.

Change Stock:
This is a method within the Stock class. This method serves to change the value of the Stock at the end of each round. Each Stock has a 50% change to either increase or decrease in value by a random percentage. Each Stock object has a hard minimum value of 1.

Investor:
Each Investor object will be a thread. The goal of each thread is to reach a certain designated moneyGoal value. Each thread will either sell, buy, or do nothing for each round. When the Investor thread tries to sell, it will look through its owned Stocks and see if that Stock's current value is greater than the average amount the Investor thread spent buying those stocks. Each thread has a budget variable that dictates what Stock objects it can buy. The budget variable is one-half the current money the Investor thread has.

Wait:
Each Stock object can only interact with one Investor thread at a time. If the Stock is currently busy, then the Investor thread will have to wait their turn. When the Investor has finished their actions for the round, then they will have to wait for the other Investors to finish.

Driver:
This is the class that the user will interact with to set up the program. The variables and parameters that other classes will use are set by the user here.