Team 01

1. Dante Pasionek → Dpasi314
2. Preston Kelley → pkelley
3. Walker Schmidt → walkyd12

Vision

PaperTrader - An alternative to traditional stock market simulators
Creating a user-friendly trading environment to provide real-time information to gain valuable trading suggestions

Project Description

A python based stock market simulator, web application that allows users to trade in real time with fake currency. Traders are able track their investments and find optimal trading strategies.

Index

List of Features Implemented

USER REQUIREMENTS		Priority
UCR-001	As an **Administrator** I want to be able to selectively set up which stocks are available to investors to provide a small sample of real-world investing scenarios.	Highest

UCR-002	As a **Trader** I want to be able to view all of my investment history from the current point in time to my first transaction to be able to track my total monetary investments over time.	Normal
UCR-003	As a **Trader** I want to be able to buy available stock in a quantity that I specify	Highest
UCR-004	As a **Trader** I want to be able to sell available stock in a quantity that I specity so I'm able to divest in assets to increase my financial capital.	Highest
UCR-005	As a **Trader** I want to be able to add money to my account so that I'm able to buy stock, similar to an investment I would make in real life.	Normal
UCR-006	As an **Administrator** I want to be able to search for a Trader by name to view their current information.	Low
UCR-007	As a **Trader** I want to be able to view my personal portfolio in order to access information regarding my investments.	Normal
UCR-008	As a **Trader** I want a trading algorithm to help determine which stocks I should invest in based on their current trends to help me make better real-life decisions.	High

NON-**F**UNCTIONAL REQUIREMENTS		Priority
UCNF-001	**Performance:** The site should have at least 90% uptime during active trading hours to help maintain accurate results.	Normal

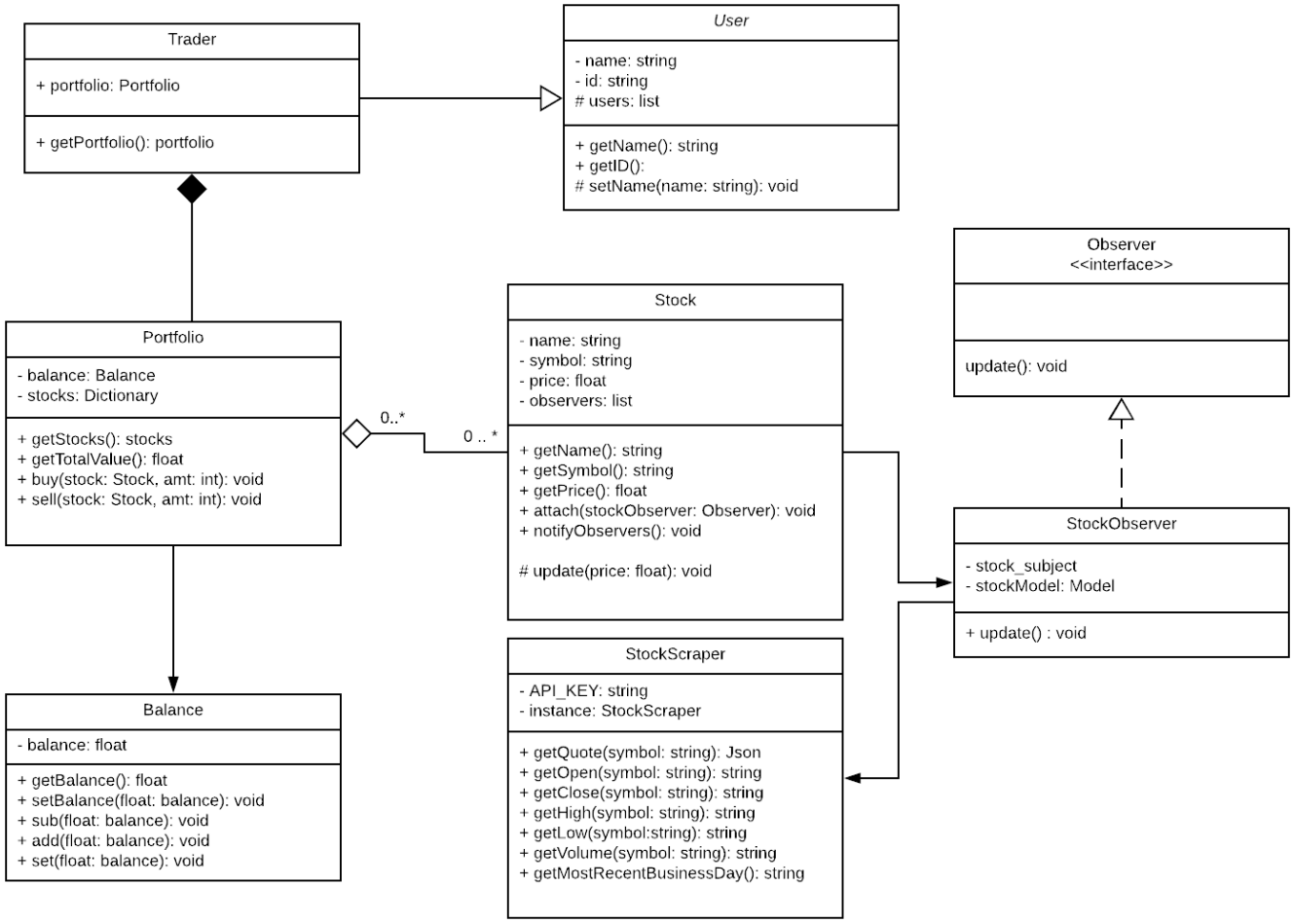
FUNCTIONAL REQUIREMENTS		Priority
UCF-001	The **System** will prevent Traders from purchasing stock when their balance amount is less than the price of the amount of stock attempting to be purchased.	Normal
UCF-002	The **System** will prevent Traders from selling stock when the quantity they wish to sell is greater than the total quantity in their portfolio	Normal

BUSINESS REQUIREMENTS		Priority
UCB-001	A **User** can be identified by a name and separate static user identification number.	High

All use cases colored in Green are features that have been implemented into the final working version of PaperTrader.

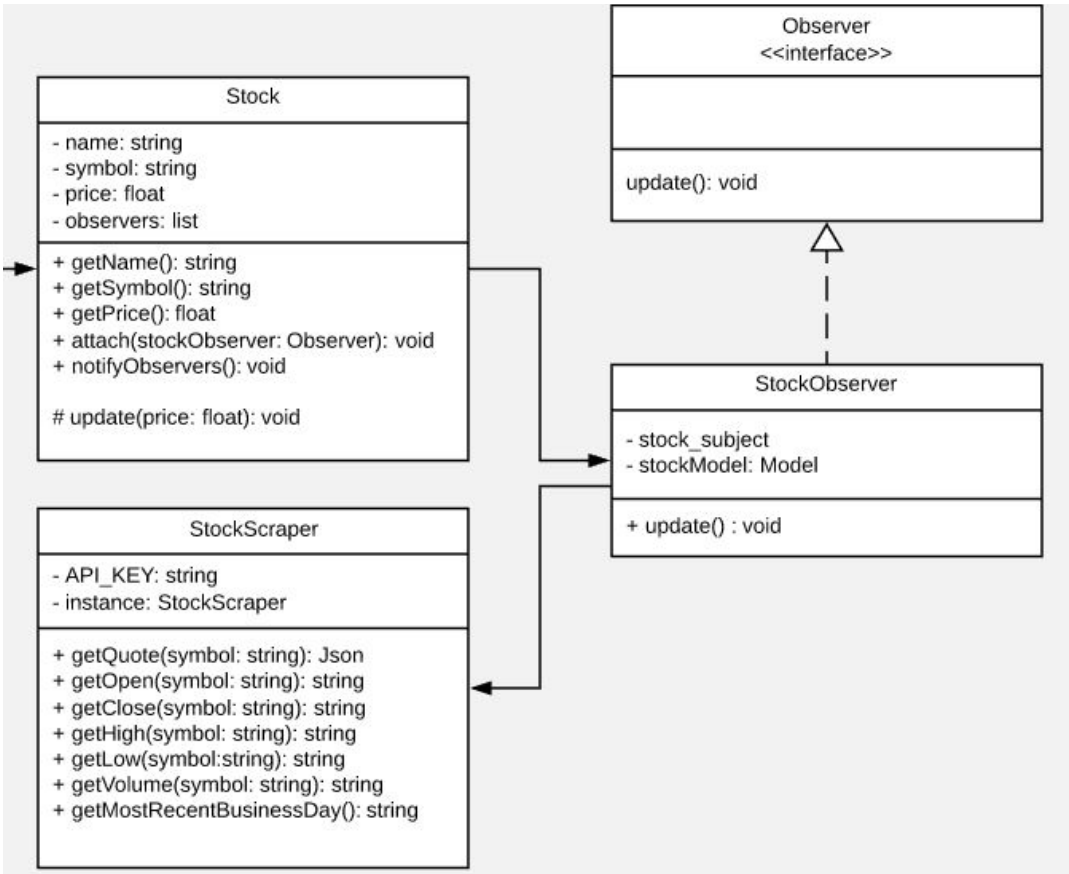
Class Diagrams (Part 2)

Class Diagrams (Final)



In the end, not too much changed from our original class diagram. The most notable change is the removal of the Administrator class. In the end, Django provided a plethora of functionalities, many of which we thought we would have to design in our original concept. In addition, the Portfolio model created in Django removes the need of having a PortfolioHandler class. Overall, the structure of the other classes did not change much (besides the addition of the StockObserver class). Designing upfront did help create these classes to a point where modification was unnecessary. For example, the Stock class was only modified to add the observer, but otherwise has not changed. Same can be said for the User, Trader, Portfolio and Balance class. Our initial ideas, for the most part, ended up working out very well.

Design Pattern



We primarily focused on using the Observer design pattern. It's helpful in many different contexts but provides the most help when updating prices at the end of a stock day. Users want to be informed of how their money fluctuates over time, and the easiest way to do this is to update the prices. However, it's hard to always keep a list of stocks on hand, so the easiest way to update these prices is with Observer, then as we need, we can update the information. Similarly, when we add the intraday price in the future, this will require another implementation of Observer (since intraday is called through the API, and likely we'll want to batch-update these stocks).

What have we learned?

What have you learned about the process of analysis and design now that you have stepped through the process to create, design and implement a system?

Walker - Being a part of a real project this semester has opened my eyes to the usefulness of design oriented thinking. In many cases before this class as I began getting into programming, I would code to complete a specific task or goal and when that was completed I would put it in a folder and never touch it again. As this isn't what happens in the industry, I am extremely glad to have learned so much about object oriented programming. Throughout the course project and class lectures, I was able to learn not only that these systems were gonna be reused and built upon in the real world, but also the design approaches necessary to ensure that all of the systems I program will be accomplishing this task and their own tasks in the best way possible. I learned that thinking about the design and planning ahead can save a lot of time down the road when beginning to program. Once the design was made for our project, the programming seemed very straightforward and was just an implementation of the design. This class and project have given me the tools to design elegant systems not by learning implementations necessarily, but learning broad concepts to help me approach future design problems.

Preston - Designing is truly an art. I can see how people who have designed systems many, many times get the knack for design and are able to predict what might be useful in the future by making present decisions smarter. Normally when I code, even a larger system, I know things are going to change so I wouldn't bother to document or make a class diagram. However, design patterns and documentation/diagrams can help speed up the process of changes in the future. Investing a bit more time initially, although may seem counterintuitive, will save tons of time down the road, even in a project that is pivoting and constantly changing. Having only general ideas of class structure and interaction leads to pretty bad spaghetti code and having to redo a bunch of classes later on. Knowing

good design patterns to use and investing time in documentation are too good rules of thumb for future-proofing a project, and have saved me so much time in this current project. The rest of design comes with good experience, so I can't wait to use what I've learned now and apply it to future projects to continue learning and eventually get the professional eye for design!

Dante - Over this past semester, many things stood out to me about how important analysis and design are. Most notably, is that there usually is an easier way to solve a problem than I might initially think. Design patterns had really influenced the way I looked at this project, and where I can apply them and how effective will they be. The importance of analyzing and designing a solution to a problem is invaluable to these kinds of projects. If one doesn't analyze the project (or analyze it well) then in the future it's likely that the original idea will need to be changed. However, when done properly analyzing the project and it's hurdles, then designing effective solutions makes the development process a lot easier. It's essentially getting the hardest part of the project out of the way before you even start it! I'm excited to be able to use the skills I've developed over the past few months as I go out into industry.