Stock Visualizer

Authors: Ryan Xu, Leor Porat
Revision: 5/6/22

<u>Introduction</u>:
[In a few paragraphs totaling about ½ page, introduce the high-level concept of your program. What this looks like depends a lot on what type of thing you are making. An introduction for an *application* will look different than one for a *game*. In general, your introduction should address questions like these:
What does your program do?
What problem does it solve? Why did you write it?
What is the story?
What are the rules? What is the goal?
Who would want to use your program?
What are the primary features of your program?

Our stock program draws a stock chart for an inputted ticker and has the ability to automatically calculate a price target from uploaded company financial information. The program also provides tools for the user to analyze the chart, like drawing lines or displaying the close price of a certain day. The reason we created this is because it saves a lot of time for both retail and institutional investors who want to calculate a precise price target. Many times, investing programs are overly complicated, especially for the average retail investor, so our program provides a sleek & effective interface that provides all the tools the average investor needs to succeed. Picture this: a teenager realizes the importance of financial literacy and decides that he wants to learn how to invest. He ponders over which program to use, until he stumbles upon Ryan & Leor's super awesome mega cool Stock Visualizer program that has all of the features that he needs, even those not available to the public, like the DCF calculator, for free. All investors would want to use our program because it has different functions that are suitable for beginners and also experts. This ranges from people wanting to learn how to invest and need a simple tool to get the information they need, to advanced investors looking to supplement their investment strategy with precise-to-the-cent price targets and improve their trading profitability by decreasing their response time to new financial information. The main features of our program are the stock chart, being able to draw on the chart to mark different things (and after use an eraser to delete the things drawn), and finally a revolutionary price target (DCF) calculator with a built in automatic financial information extractor.

<u>Instructions</u>:
[Explain how to use the program. This needs to be **specific**:
Which keyboard keys will do what?

- Keys can be used to search different stock tickers. For example, if you want to look at Apple's stock, you would use your keys to type AAPL, Apple's stock ticker, in lower or upper case. Past this, no further keyboard strokes or shortcuts are needed.

Where will you need to click?

- There will be one central screen where all the features will be contained. You can click the search box which will allow you to type tickers to view the graphs of different stocks. Past this, you can select different tools on the side panel. For example, if you click the line tool then you'll be able to draw lines on the chart, if you click the eraser tool you'll be able to erase the lines you previously drew, and if you click the pointer tool you'll be able to find the closing price for a specific day. With the default tool selected, you can click and drag the graph to select the time frame that you want to view.

Will you have menus that need to be navigated? What will they look like?

- The program will have one central screen as it is all that is needed, so no navigation between menus is required.

Do actions need to be taken in a certain order?

- Yes. In order to use one of the tools, one must click on the button corresponding to the tool first. In order to calculate a price target, one must upload a txt file named data.txt to the program.

Features List (THE ONLY SECTION THAT CANNOT CHANGE LATER):

Must-have Features:

[These are features that we agree you will *definitely* have by the project due date. A good final project would have all of these completed. At least 5 are required. Each feature should be fully described (at least a few full sentences for each)]

- Extraction of stock data using the AlphaVantage API
  - AlphaVantage is a leading open-source stock market data api provider. In order to accomplish this, we will need to find a suitable Java wrapper for the API and make sure it's compatible with Maven. Then we need to set up the Maven project file inside of eclipse and import the jar. This function will enable the creators and users of the program to access real time stock data.
- Visualization of data (line chart)
  - Normally, when users analyze stock data, they look at charts, which are essentially just graphs created through the stock's performance over a certain period of time. Our program will use extracted data to create these charts for our users to easily spot patterns. Users will be able to make observations based on the chart that would be impossible with data.
- User interface
  - We will have an interactive user interface to make it easier for the user to navigate the program. This will include buttons to switch between the tools used (switch between lines, erase tool, etc), switch between the stock shown, or can be used to hopefully drag the graph.
- Drawing Lines + Erase
  - The ability to mark lines on the stock charts to analyze chart patterns and trends (e.g. drawing your own support + resistance lines). Lines will be drawn by two clicks,

    - connecting the points clicked. These lines can be deleted with the eraser tool, which works by clicking on the line desired after selecting the tool.
- Expanding & Contracting
    - The ability to zoom in and out on a chart to make more intricate or general observations about the data. This will be done using the scroll wheel of the mouse, where scrolling up will zoom out the graph and scrolling down will zoom the graph in.

Want-to-have Features:
[These are features that you would like to have by the project due date, but you're unsure whether you'll hit all of them. A good final project would have perhaps half of these completed. At least 5 are required. Again, fully describe each.

- Earnings report extractor and DCF calculator
    - The extractor will be able to, when a txt file correctly named data.txt is uploaded to the folder, extract relevant financial data and input it into the complicated formula for discounted cash flow operations to get a price target. The DCF formula is incredibly complicated and thus extremely tedious to do manually, whether it be gathering the data or plugging it into the formula, so the extractor saves a ton of time and effort.
- Click for value
    - The user will be able to click on the chart to see the value of the stock at that point. Through this, users will be able to better analyze the chart. Since clicking the exact pixel is very hard, the user will be able to click whatever y value at a given x value for convenience.
- Moving averages
    - Moving averages are smoothed out lines formed from taking the average of closing prices over a set frequency. They are extremely useful for technical analysis as it allows investors to filter out market noise and be notified when a trend is beginning. The moving average option will be a button, but will have options for lengths.
- Candle chart
    - Candlestick charts, compared to the conventional line chart, contain much more information. While containing the closing price that line charts solely display, candle charts also display the opening price, intraday high, and intraday low. We would use the API to implement this feature but we would have to use rectangles + lines for each candle to draw it out. The user would be able to press a button at the bottom right corner to use this chart.
- MACD, RSI
    - MACD stands for the Moving Average Convergence Divergence while the RSI stands for Relative Strength Index. These are arguably the two most common and effective technical indicators: MACD measures a change in trend while the RSI approximates the relative valuation. Both will be in the form of a button, and the graphs for these would pop up in a space below the chart (and thus compact the chart vertically).

Stretch Features:
[These are features that we agree a *fully complete version of this program would have, but that you probably will not have time to implement*. A good final project does not necessarily need to have any of these completed at all. At least 3 are required. Again, fully describe each.]

- Automated technical analysis (support & resistance at least)
    - Automated technically analysis will take the data and analyze it to produce an outcome of how it should be treated. In the case of support & resistance lines, the program should be able to recognize where those lines should be drawn, and if prompted to, draw them for the user (they would be treated as drawn lines). Support and resistance lines are simply an example, and we hope to implement further forms of technical analysis.
- Fully automated trading algorithm
    - The trading algorithm would be a button that opens up a panel for a multitude of required inputs to determine its functionality. For example, the user may be a young investor who would like to take on more risk than the average investor so he would aim for a beta greater than 1. The user would also have to specify the desired frequency of trades. The algorithm would automatically conclude the success ratio, the average profit for each trade, and alpha, the amount it beats the market.
- Automatic earnings report extractor
    - By automatic, I mean somehow porting earnings data into the folder the second it comes out through automation. The user would just have to check a box next to the corner of the stock chart of a stock to enable automatic extraction. This would save investors precious minutes of time as the market reacts to information extremely quickly. Even a one second advantage cumulates into massive profits.
- Watchlist
    - This would exist as a tab to the left of the stock chart, enabling users to save their favored stocks in a list for future ease of access. In order to add it, users would click the plus sign on the top left corner of the list and search the ticker. There can be multiple different lists, and they can each be renamed at the top by clicking the pencil icon next to the name.

<u>Class List</u>:
[This section lists the Java classes that make up the program and very briefly describes what each represents. It's totally fine to put this section in list format and not to use full sentences.]

- StockVisualizer.java: hosts the main method
- DrawingSurface.java: represents a PApplet that draws the stock chart UI
- DataExtractor.java: represents an extractor of data that takes a txt file and parses it into a string
- DcfCalculator.java: represents a DataExtractor that calculates for a price target using discounted cash flow
- TrendlineCalculator.java: automatically finds trendlines in the stock chart (prospective)

- Button.java: Represents a button that can be used to activate/deactive the different features of the program (e.g. button for erasing, drawing lines, etc).

Credits:
[Gives credit for project components. This includes both internal credit (your group members) and external credit (other people, websites, libraries). To do this:

- Ryan: Brainstormed project idea, completed README, made UML,
    - Will create the data extractor, design the DCF formula and incorporate it into the chart
- Leor: Revised README,
    - Will program the user interface and the stock chart. For the user interface, this includes programming the buttons and implementing their functionality into the program, making sure tools like drawing or erasing work and are completely functional with the other features, adding areas for stock tickers & displaying the corresponding chart, etc..
- Credits
    - AlphaVantage
    - Crazzyghost on github
    - Maven
    - Zios bank DCF calculator