**Project Outline**

**Overview:**

**Screen 1/ Homepage**

Screen 1 firstly as our main page with

* the title, group number
* an image and
* a start button at the bottom

**Screen 2**

* Has 3 queries
* Query number is typed in the text widget
* Radio buttons, normal buttons and sliders selections made by user related to the respective query
* When these buttons are clicked, we shift to the next screen displaying the result of the selected query

**Screen 3**

* Displays the graphical result to the selected query
* allows the user to go back to Screen 2 to select another query which the user wants to inspect
* The user doesn’t have access to home page directly from Screen3

Have 5 classes,2 subclasses and the main

* Constants
* Screen
* Graphs
* Widgets: Mostly a common widget implementation has been applied to all the user interface which has been cumbersome but makes everything neater
* Radio Buttons
* Text Widgets
* Stock Data

As a group we divided the work among ourselves weekly to make sure we achieved the weekly goals comfortably and helped each other through the whole process making sure none of us faced any major issues due to the sudden change in the circumstances.

**QUERIES (Ryan Ting)**

We implemented 3 queries:

1. To find the most moved(monetarily) share in a single day
2. The price change of the shares of a company over the past 3 months
3. The companies with the largest change in share price between 2 dates chosen by the user

**Query 1**

To implement query one, I parsed through the data and found stocks traded on the day that was inputted by the user. We also allowed the user to choose how many companies they wanted to view. When getting the date, we multiplied the closing share price of that day with the volume traded. We then stored that and the ticker of the company in a linked list. I chose to use linked lists due to its time complexity. The

search time is linear, and the search and remove time of singly linked lists are constant. This was a problem I encountered as using other data structures would take too long as there are a lot of data going through it, so to solve this issue I used singly linked lists as they had all the functionality I needed. This data then went to the scale method, which I will speak about later as all query methods go through it

**Query 2**

To implement query 2, it was similar to query one, I found the ticker that matched the one inputted by the user, then found the 90 most recent entries in the dataset and put them in a linked list, along with another linked list that had all the dates in the same index.

We first decided to make the query span across the time the company was on the market, but there was a problem. I could not fit all the information on a screen. When I obtained all the data, I drew the graph and nothing came out, I then found out that there were way too many days, to the point where, if I would put all data on the screen, the x size of the data point would be rounded to zero by processing, we then resolved this y only making it 3 months

**Query 3**

For query 3, I first checked if there was data on a company which had traded on the two dates chosen by the user. If the program found that, it would calculate the percentage change between the two dates of that company. Then it would store the data and tickers in two linked lists like the others.

**Scale method**

This method is to scale the data into the correct size to draw the graph. This is because the data is normally too big, all query methods go straight to this method. It takes in the data, tickers (both linked lists), the graph title, and the preface (e.g. % or $). This constantly divides the data by 10 until the data is the correct size to draw onto the screen

**Draw method (for graph class)**

For all three graphs, I used the same draw method, this comes with some problems. For example, I needed to deal with their either being 5 bars or 90, this made it harder to create the method but made it so I could use one method for all queries which is more efficient. Bar charts were used in all queries, this method is just to draw the charts and labels. All query methods are led to the scale method which is led to this method.

**User Interface and Screen appearances**(Khushboo)

**Screen 1**

This screen has lines drawn using line function, with stroke weight and stroke color.

A random wave has been created using map () to calculate y position of the point, the sin angle which has been incremented for every point location by 0.03.

Also, the points appear to be changing color because of the use of color Mode function with the change of hue, saturation and base by 100 for every dot within every frame.  Then it displays the text “Programming Project with the Group number on the left side “in the font type “Apple Mungo”. There is a button on the right which leads to the second screen.

**Screen 2**

Screen 2, firstly looking at the appearance

A lot of time has been spent on the appearance of the program

 Background of white color with on the left side empty rectangles appear with a green outline appear constantly, are sketched overlapping at random coordinates which are of varied dimensions.

* For the squares to appear at random and of a random size the random () function has been used to select the coordinates while for different sizes a constant has been subtracted and added from height and the width.  Every time screen 2 was returned to the squares used to overlap the widgets and the other displayed graphics to ensure that didn’t happen a white background was recreated every time the user returns to the screen
* Then on top of that we have an image of an electric bulb with a little box displaying the user the query options and what needs to be entered by the user in the text widget if the user wants that query’s graph to be displayed
* For the bulb image PImage () has been used
* Then the green square displaying a brief about the queries has been done through using the triangle() and recto() functions with the text displayed inside them using the text() function
* Then on the right we have the text widget which shows up the label “Please enter a valid query”  with a search button on the right to it. The user enters the text widget backspaces, then enters either in lower or uppercase without pressing any other key  and then presses search then as to other options related to that query appear
* Text widget  is a subclass of the widget class which extends the widget class. The text widget works as it checks whether the user's mouse pointer is within the widget area and if it is then it changes the focus to true and then according to what key user presses accordingly appends the label. It needs to be clicked on then made sure only the required alphabet keys are pressed and no other
* Then user presses the search widget which displays down the further options for user to select
* The radio buttons appear for query 1 and query 2; further selection of number of companies and which company respectively
* Radio buttons is a subclass of the widget class which extends the widget class . The radio buttons appear as a small square on the left to their label text ; when the user clicks inside the box it marks the radio button with a small black square. Every radio button has been clearly displayed with its own coordinate and text on the right. Each of them is identified clicked and has an event number associated with it telling the program which one has been selected
* The sliders appear for query 1 and query 3;dates
* There was an issue with sliders appearing on every screen and not still being displayed after a query has been performed and the user has returned to screen 2 to combat that .show() and .hide() functions have been used
* There is a run widget which takes the user to the required query graph and home widget to screen 1

A close up of a map

Description automatically generatedA close up of a map

Description automatically generatedA close up of text on a white background

Description automatically generatedA screenshot of a cell phone

Description automatically generatedScreen 2

A picture containing computer

Description automatically generated Screen 1

**Backend and functionality (Peadar)**

My job on the project was to take care of any backend functionality for the project, this included the reading in of data from the data sets. Early on efficiency was a real issue for the project as I was loading data in using the loadString() function native to Processing. Unfortunately this led to long loading times even with smaller data sets, I decided to employ the use of the loadTable() function and linking up with ryan we were able to read in all the data efficiently and store it in the stockData Class.

From there I did a bit of cleanup work on some of the functions in main, adding in switch statements for efficiency wherever I could. My next job was working on Radio buttons along with Khushboo. As I am very well aware that UI design is a weakness of mine, I worked on functionality on these radio buttons for queries. In the earlier weeks when we were working on displaying queries, I linked the query buttons to their associated screens. When working on query one I designed the uniqueTicker() function which reads through the entire data set to find a user inputted number of tickers and returns this number of unique tickers in arraylist format.

For query two I was again working on functionality, with Khushboo again working on Ui design and the position of the buttons I ensured that once the user made a selection for query 2 this selection was saved as a string unless the selection was changed by the user.

**Selecting subset of data and stockData class** (Brian Bredican)

The first task I had was to create the stockData class alongside Ryan. The function of this class was to get getter methods for all the stock data. We created String getter methods for the tickers and the dates and floats for the open Price, close Price, adjusted Price, low, high, volume, date and then finally a display method which prints the values when called. Once we planned our approach to this class we didn’t run into many problems.

 Next I had to implement selection of data subsets. I decided to use sliders for the dates of the stocks for queries 1 and 3. At first, I tried to create a slider class from scratch. However, I ran into a number of difficulties with this in particular when trying to create multiple sliders for day, month and year. I also ran into problems retrieving data from the sliders. Eventually the team and I decided to implement a library function to create the sliders called controlp5. This means we only had to worry about their implementation. We created a draw function for the arraylist of sliders on the screen class. After this we implemented six different sliders in total which for query 1 and 3 to create the date parameters for the graph functions. The integers we obtained from the sliders then converted into a string of year/month/day and passed into the graph function. One problem we faced was the slides remaining on screens after the query had been run. We eventually solved this with the use of a .show and .hide method.