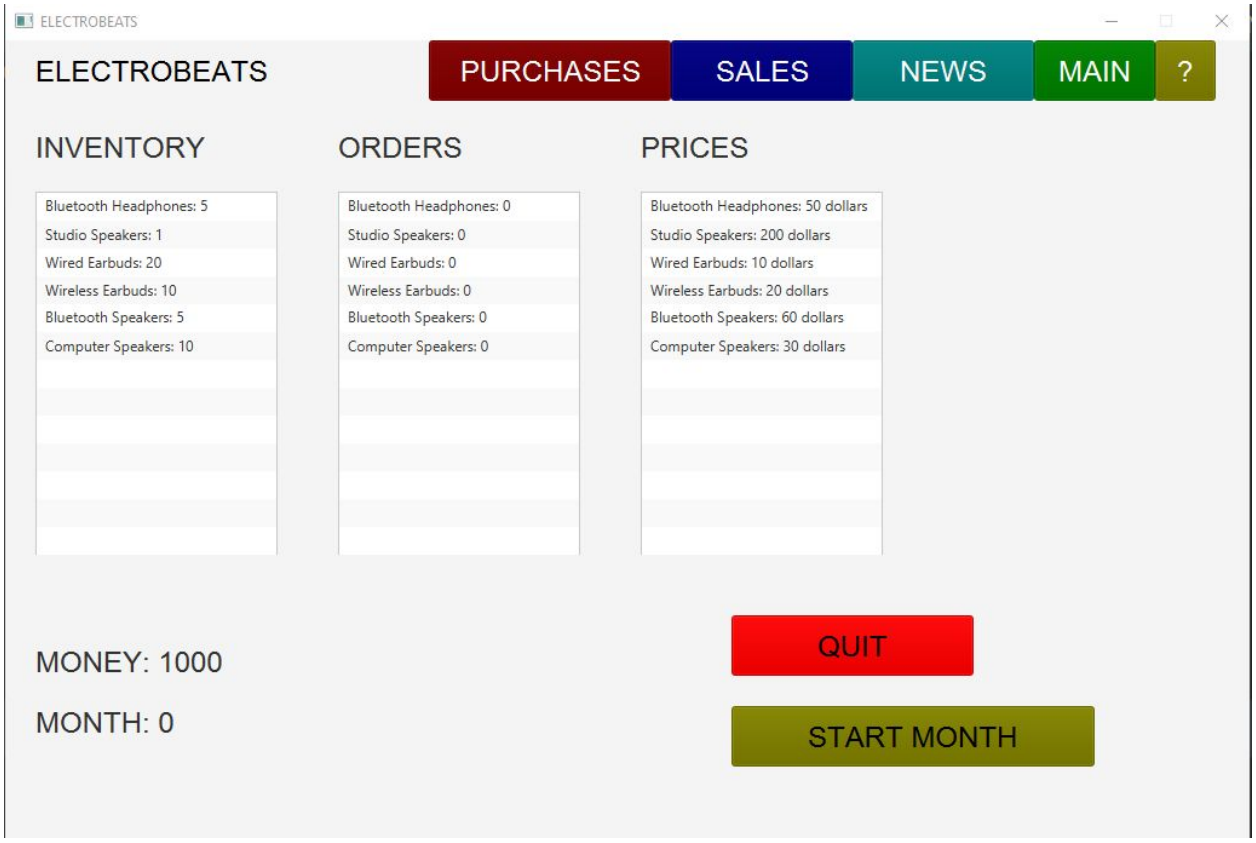
Jerry Deng
101181884

ELECTROBEATS

My project is to simulate a store that sells some headphones and speakers.
It allows the user to purchase inventory, keep track of inventory, control prices, and to sell products based on an estimate.
The goal of this game is to make as much profit as possible within 12 months.

Main Page



This is the main page of the application. It shows the title and menu bar to navigate to other pages at the top and contains a summary of your store's inventory, pending orders, and prices charged to customers. It has a quit button to exit the application and a 'start month' button that allows the user to proceed through the month and see the results.

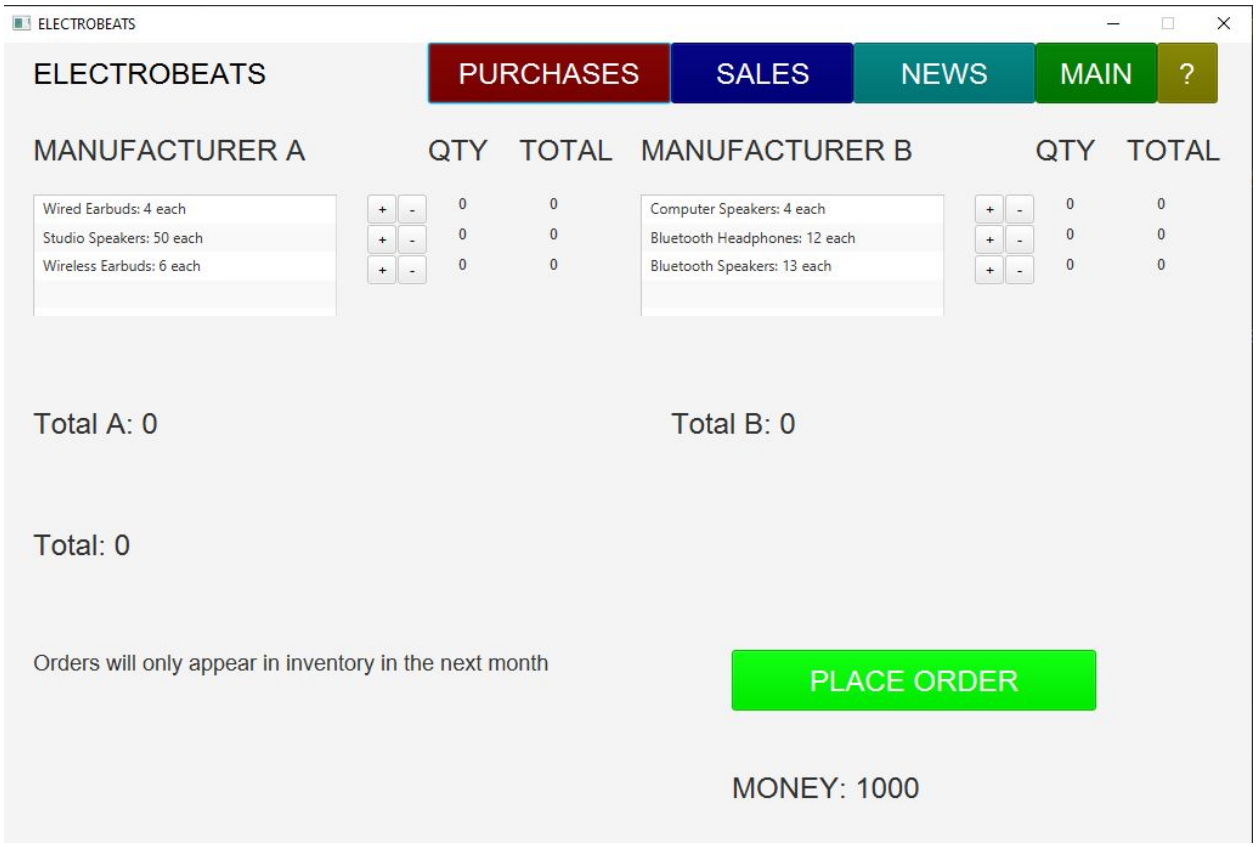
Course Concepts Covered

- GUI

    - Lots of GUI in this page, with listviews, labels, buttons
    - Multiple panes stacked (menu pane and main pane)
- OOP Principles
    - Fetching data from a “Store” class that is not shown which contains the attributes of inventory, prices, and orders
    - Proper encapsulation with getter and setter methods from various classes when inventory is changed
    - Parent class of Pane, using Pane methods/attributes
    - Money value is synced with both the store and the purchase page pane through instance methods
- Algorithm/Complexity
    - Lots of accessing data from other classes (ie other panes and the store class) to fill in the various data
- Data structures
    - HashMaps used to get prices, orders, and inventory from product
    - Converting HashMaps into various array data types to use in listviews

Jerry Deng
101181884

Purchases Page



The purchases page still has the menu bars at the top. It allows the user to place an order of various items that will arrive in inventory the next month. The user has two manufacturers to choose from, and can only place one order every month. Note that not all items may be available at time of purchase.

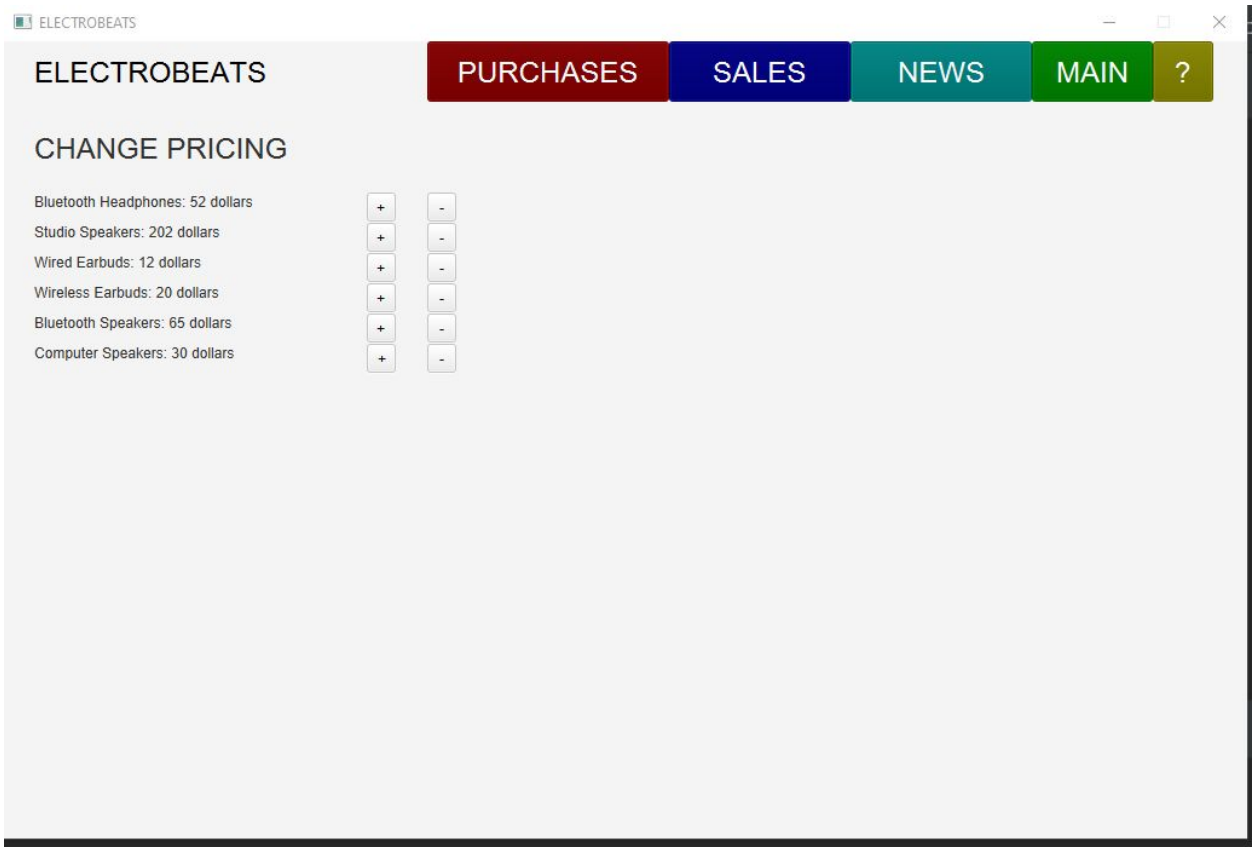
Course Concepts Covered

- GUI
    - Lots of Buttons, labels, and listviews
    - Lots of event listeners for the buttons
    - Place order event listener integrates with the main class Run which allows purchases ListView of Main Page to be synced. It also disables the button until the next month
    - Object Layouts
- OOP Principles
    - Proper Encapsulation with getter and setter methods

    - Inheritance from extending Pane class and using Pane methods for layout
    - Sending and fetching data between other classes (ie, syncing the money label with the main page)
- Data Structures
    - hashmap used here to track the order as the user adds item
    - Various hashmap-to-array conversions to fill listviews
- Algorithm/Complexity
    - Lots of thought into randomly generating manufacture items and prices within a certain range for each product
    - Lots of various update methods to sync the purchases page with the store and the main page
    - The page has to reload after every month with different manufacturer items and prices
    - It has to make sure that the user doesn't enter a negative quantity, so conditional statements are implemented
    - Orders have to be in inventory by the next month, so a hashmap is needed for the current inventory and current orders and has to be updated accordingly

Jerry Deng
101181884

Sales Page



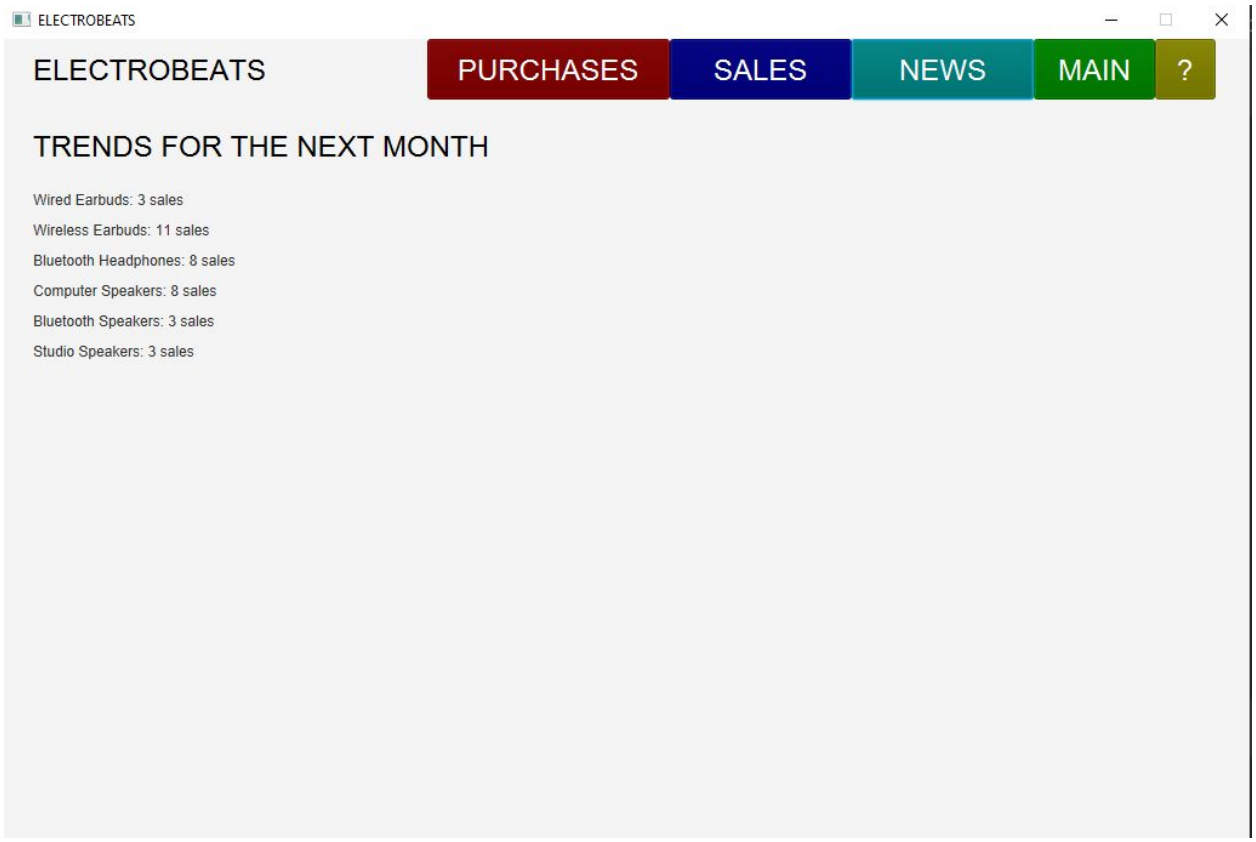
The sales page is where the user can change the pricing of its products to change how many customers will buy it.
NOTE that I did not create an algorithm that changes customer behaviour due to pricing, just a simple threshold if any product costs more than $250, it will not get sold

Course Concepts Covered:

- GUI
  - Using buttons, labels, panes
  - Setting event listeners that responds to pricing
  - Update other pages like main page to sync with the pricing changes
- OOP
  - Extends Pane class
  - Sending and transferring data between various class instances through class instance methods
  - Encapsulation with getter methods for the buttons so the main class Run can sync the pricing changes with the main page

- Data types/structures
    - Using hashmaps to sync pricing with the store and Main page
- Complexity
    - Mainly syncing with other classes so all classes have the same data
    - Pricing is limited as it can't go less than 0

News Page

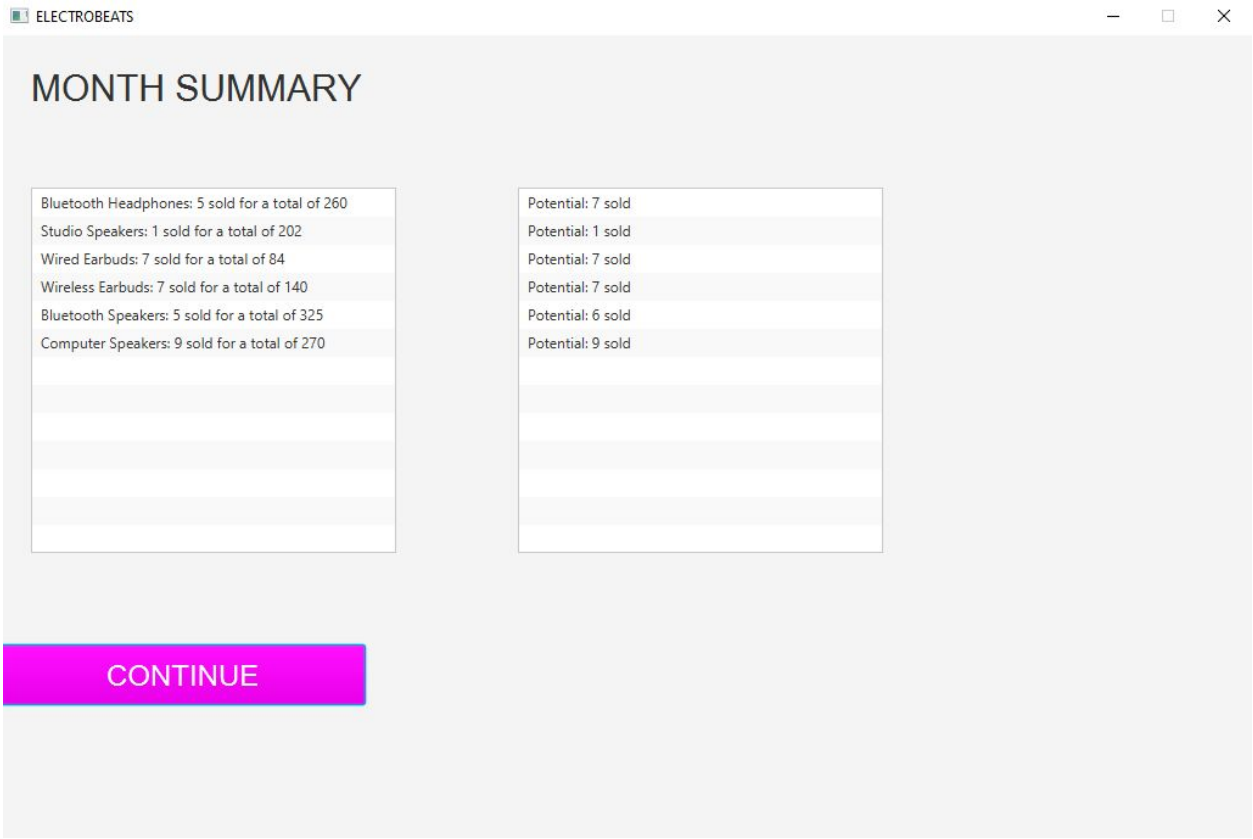


The news page shows the trends for the next month with a +-5 accuracy.
Ex: Wired Earbuds forecasts 3 sales, but it could be anywhere from 0 to 8.
NOTE that the forecast avoids negative numbers.

- GUI
    - Using various labels and updating them accordingly after every month
- OOP
    - Accessing important information (realForecast and forecast) from the store class through proper encapsulation
- Data types
    - Hashmaps to store the realForecast and the forecast shown to user

    - Accessing data from arrays and hashmaps to display on GUI
- Complexity
    - Randomly generates potential sales and forecasted sales through methods and verification

Month Summary Page (after clicking Start Month on Main Page)



After the user clicks the Start Month button on the main page, a pane shows up showing the month summary, with the total sales for each product based on the user's pricing as well as the potential sales if the user had enough in stock.
Pressing the continue button redirects the user to the main page and increments the month by 1 and updates inventory and orders.

Jerry Deng
101181884

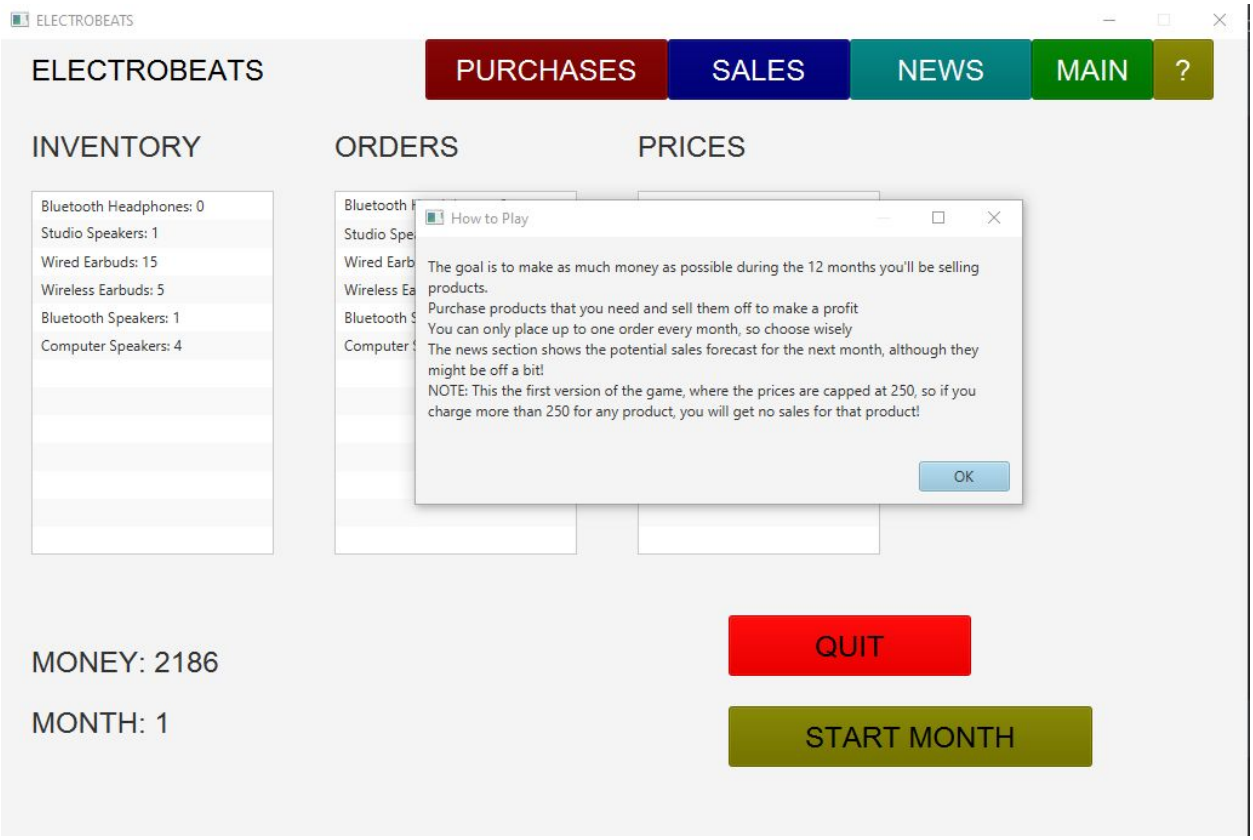
Course Concepts

- GUI
    - Working with listviews and pane layouts
    - Passing event listeners that leads to other panes and different event listeners through buttons
- OOP
    - Extends pane, uses Pane layout
    - Lots of fetching data from other classes through getter methods (ie, takes the potential sales form a HashMap variable from the Store class)
    - Interfaces working in the backend to update the money
- Data structures
    - Hashmap-to-array conversions to work with listviews (splitting hashmaps into arrays of their keys and values and then combining them into another modified array to display on GUI)
    -
- Algorithms/Complexity
    - Uses potential sales and real sales generator algorithms, where the real sales are predetermined randomly within a range, with the potential forecast shown to user taking in the real sales randomly generated and adds a +-5 random variance to it.
    - All relevant variables must be updated after that month since the orders hashmap will have to be reset, the previous orders hashmap needs to be added into the current inventory hash, the current inventory hash needs to change according to the products sold
    - GUI Pages have to be updated accordingly
        - Main Page needs updated inventory and orders
        - Purchases page needs to be reset
        - News page needs to be rerandomized for the next month
    - Money has to be updated accordingly to products sold and updated for the store class and many page classes that shows Money
    - Inventory has to be checked if it can sell products (ie verify that there is at least 5 wired earbuds in inventory if customers order 5 wired earbuds)

Other Aspects of App

How to play

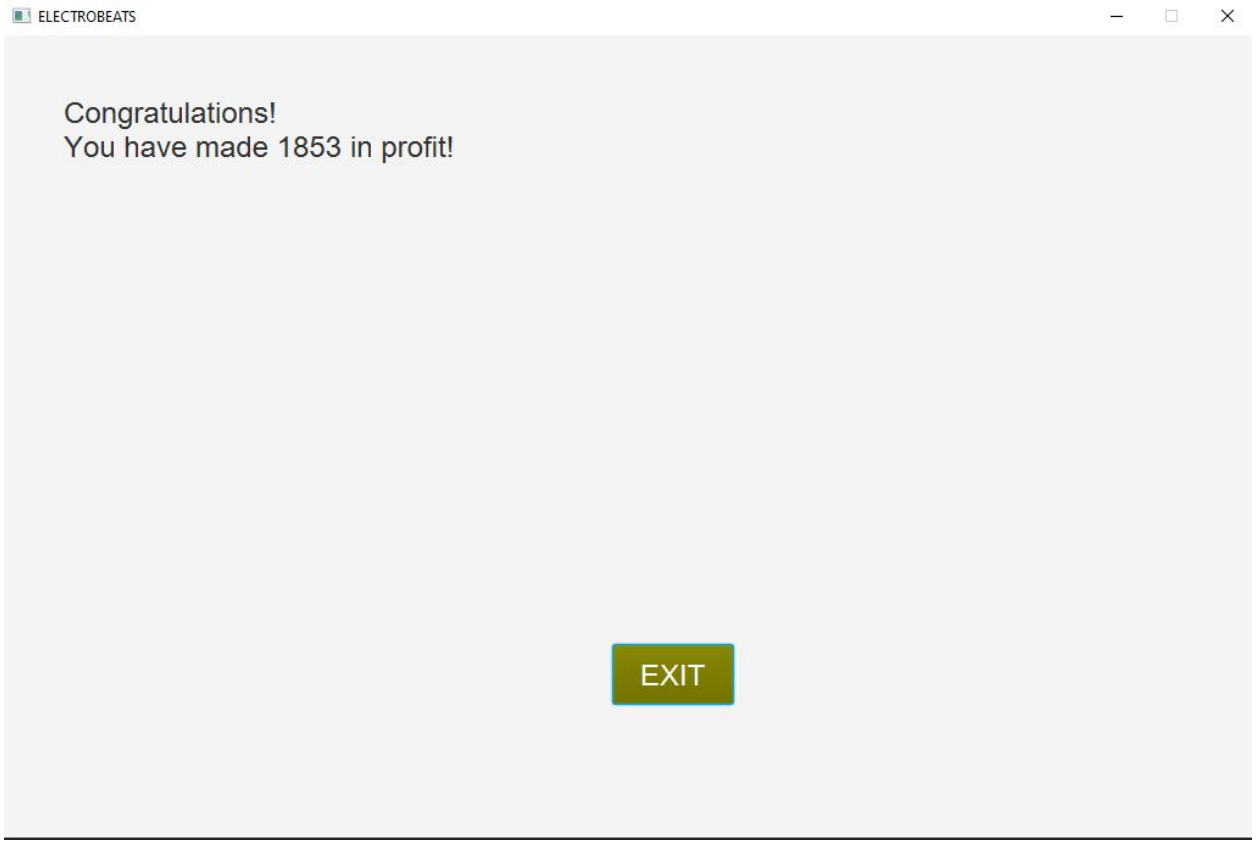
Jerry Deng
101181884



When the question mark button is clicked, a short dialog appears to give the user intuition about the app.

Jerry Deng
101181884

Victory Page



Shows the ending of the game and lets the user know his profit based on his ending money value.

Improvements

- The main improvement I could do is to optimize a lot of my classes. Many of my pane classes have methods to change/update the store variables, which could be an inaccuracy as it is more memory efficient to change the store instance variables directly from the store class than from an intermediate pane class. My reasoning for this is that 1) there are so many things to change about the store (inventory, orders, real forecast, projected forecast, etc.) and that the store variables like inventory will have to be changed in the Main Pane class anyways since it is shown on the pane.
- I could also add more complexity by having the number of products sold be based on the price (ie 1 more wired earbuds gets sold for every $1 drop under $10)

Jerry Deng
101181884

- I could also calculated the value of the liquidated inventory at the end of the 12 months through keep track of existing inventory through queue data structures that keeps track of the price of every product sold