You are required, but not limited, to turn in the following source files:

- Assignment6.java (You need **NOT** to change the file, just use it!)
- Order.java (You need **NOT** to change the file, just use it!)
- InputPane.java - **to be completed** (it extends HBox and contains button handler nested class)

- HandlePane.java - **to be completed** (it extends HBox and contains button handler nested class)

2. Requirements to get full credits in Documentation

1. The assignment number, your name, StudentID, Lecture time, and a class description need to be included at the top of each class/file.
2. A description of each method is also needed.
3. Some additional comments inside of methods to explain code that are hard to follow

You can look at the Java programs in the text book to see how comments are added to programs.

3. Skills to be Applied

JavaFX (Check here for JavaFX API https://docs.oracle.com/javase/8/javafx/api/)

(**Note**: *for this GUI application, you cannot use Swing/AWT package, you must implement it by using JavaFX, using Swing/AWT package will result a "0" grade!*)

Classes/Interfaces might be used (**feel free to add any other necessary JavaFX classes)

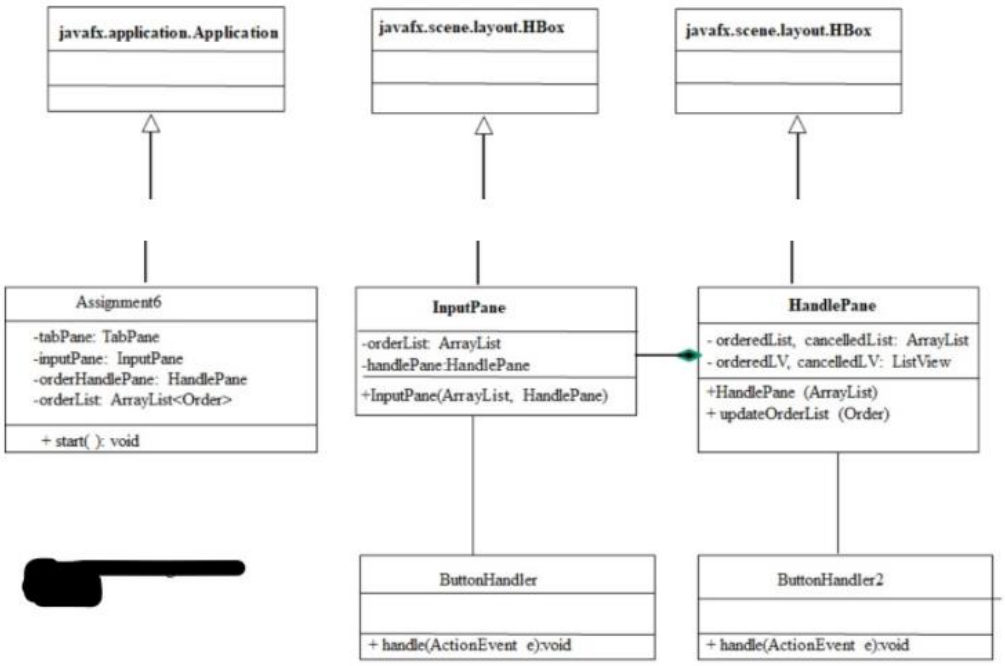
```
javafx.scene.layout.GridPane;
javafx.scene.layout.BorderPane;
javafx.scene.layout.HBox;
javafx.scene.layout.TilePane;

javafx.scene.paint.Color;
javafx.scene.text.Font;

javafx.scene.control.Button;
javafx.scene.control.Label;
javafx.scene.control.TextField;
javafx.scene.control.TextArea;
javafx.scene.control.ListView;
javafx.scene.control.SelectionMode;
javafx.collections.FXCollections;
javafx.collections.ObservableList;

javafx.geometry.HPos;
javafx.geometry.Pos;
javafx.geometry.Insets;
javafx.geometry.Orientation;

javafx.event.ActionEvent;
javafx.event.EventHandler;

java.util.ArrayList;
```

4. Program Description

java.util.ArrayList;

4. Program Description

Suggested Class Diagram: (.pdf file is available for this figure)



Write a JavaFX GUI application that simulates a company's order placement/cancellation activities. Your program should contain two tabs. The first tab is labeled "Order Input" and the second tab is labeled "Order Handle". (The size of the applet here is approximately 580 X 400).

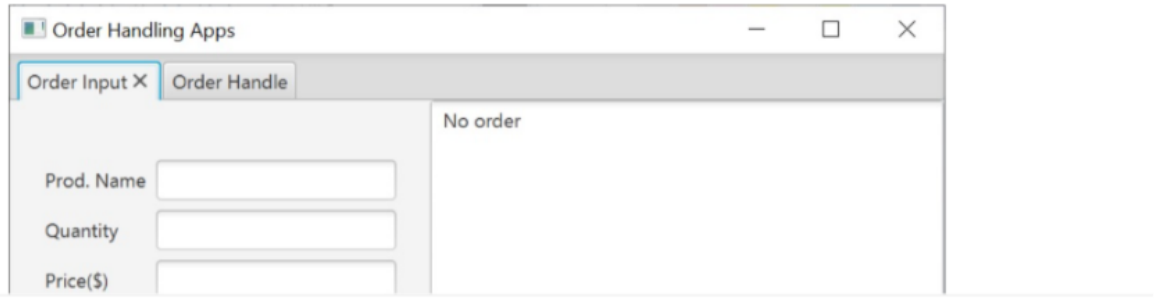
The section under the "Order Input" tab should be divided into two parts:

The left part contains three labels, three textfields, and a button for a user to enter an Order's information. The right part shows "No Order" at the beginning (it is done using TextArea).

Write a JavaFX GUI application that simulates a company's order placement/cancellation activities. Your program should contain two tabs. The first tab is labeled "Order Input" and the second tab is labeled "Order Handle". (The size of the applet here is approximately 580 X 400).

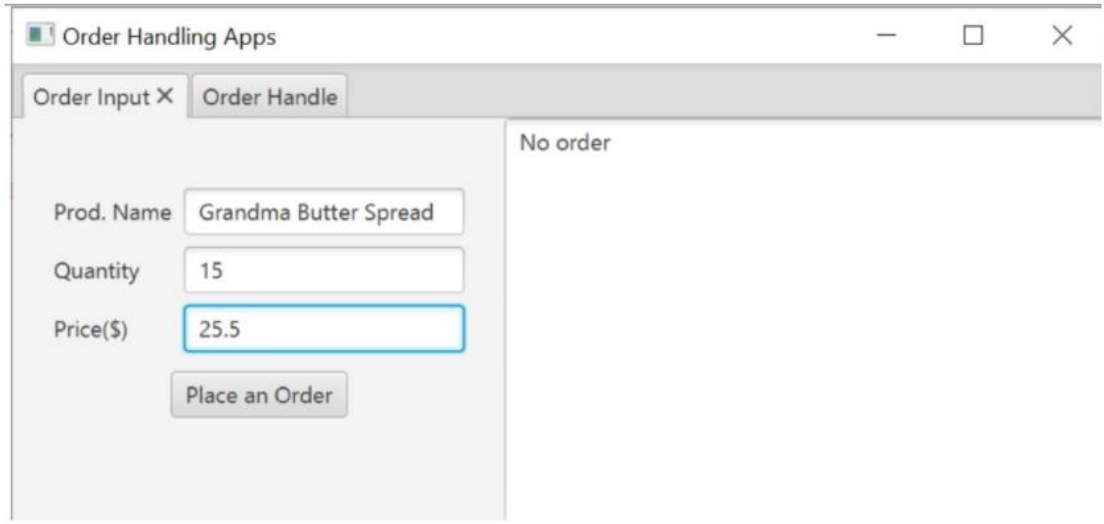
The section under the "Order Input" tab should be divided into two parts:

The left part contains three labels, three textfields, and a button for a user to enter an Order's information. The right part shows "No Order" at the beginning (it is done using TextArea).

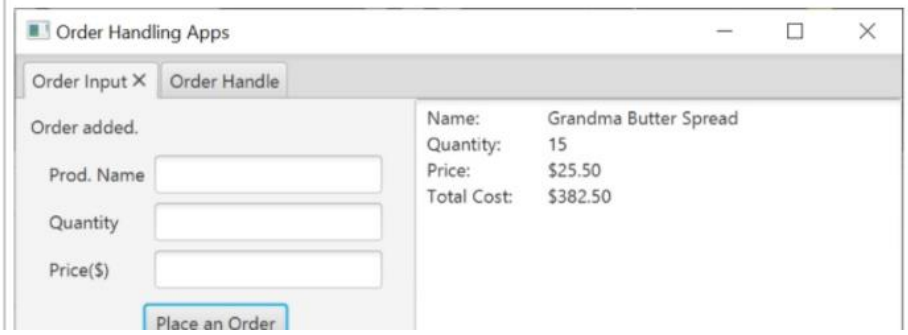


A user can enter an Order's information in the textfields, and push "Place an Order" button (see below).

A user can enter an Order's information in the textfields, and push "Place an Order" button (see below).

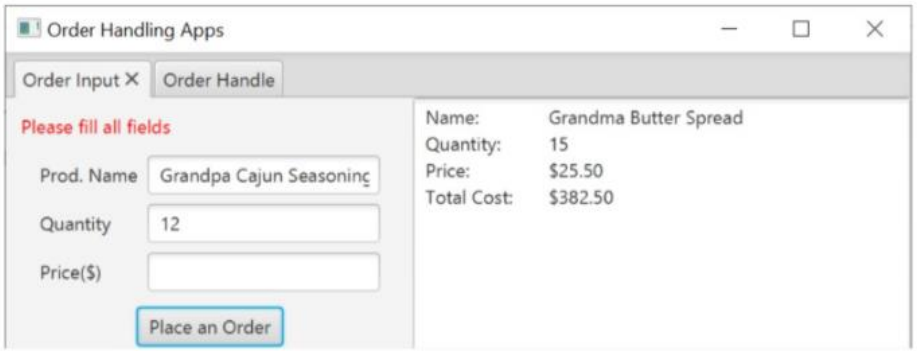


Then the Order's information should appear on the right hand side of the pane. A message "Order added." should also appear at the top of the pane. Note: (1) The total cost of the order should also be computed and shown on the right hand side, you should use the relevant mutator in **Order** class to compute this. (2) After an Order is successfully added to the right hand side, all text fields are cleared with no texts and are ready for user to enter another order's information.

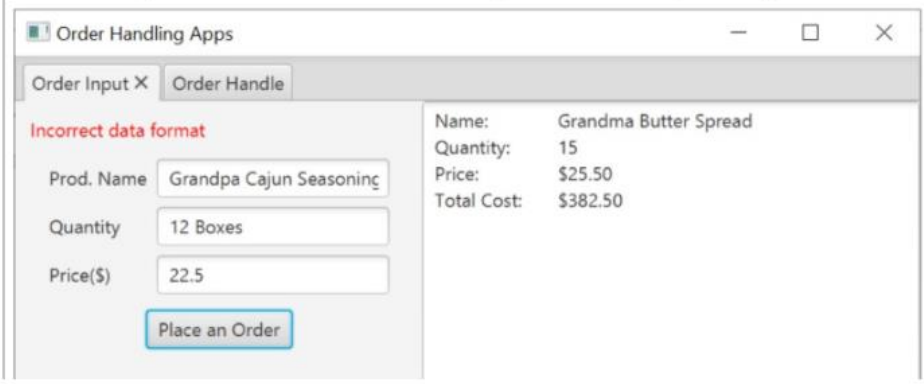


Error handling:

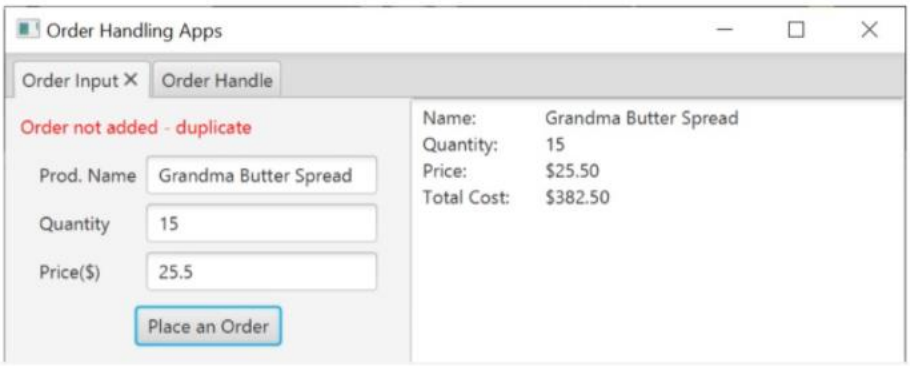
1. If a user forgets to enter a text into any of the four text fields and pushes "Place an Order" button, show a message "Please fill all fields" with **red color**, and nothing should be added to the right hand side pane.



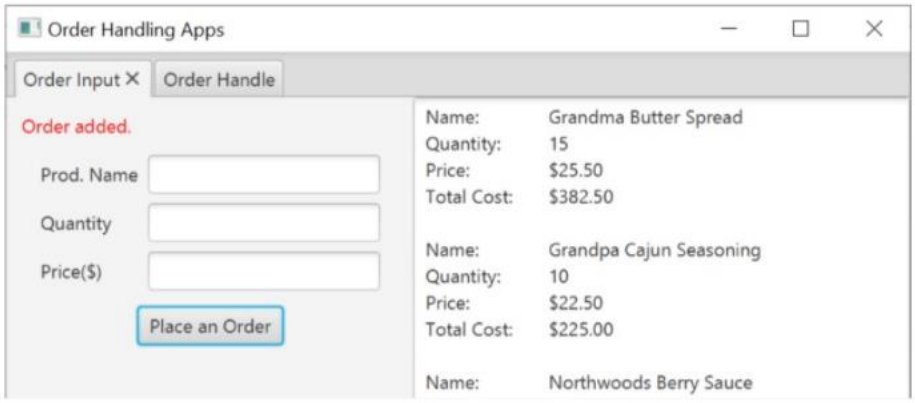
2. If a user enters wrong information into the text fields and pushes "Place an Order" button, such as enter **12 Boxes** instead of a number 12 into the "Quantity" text field, show a message "incorrect data format" with **red color**, and nothing should be added to the right hand side panel.



3. If a user enters an Order's information that is already listed on the right hand side pane, and pushes the "Place an Order" button, show a message "Order not added - duplicate" with **red color** and nothing should be added to the right hand side pane. **Orders with the same name, quantity and price are considered the same (or duplicated).** Note: as long as one of the input is different, we will treat it as a different order.

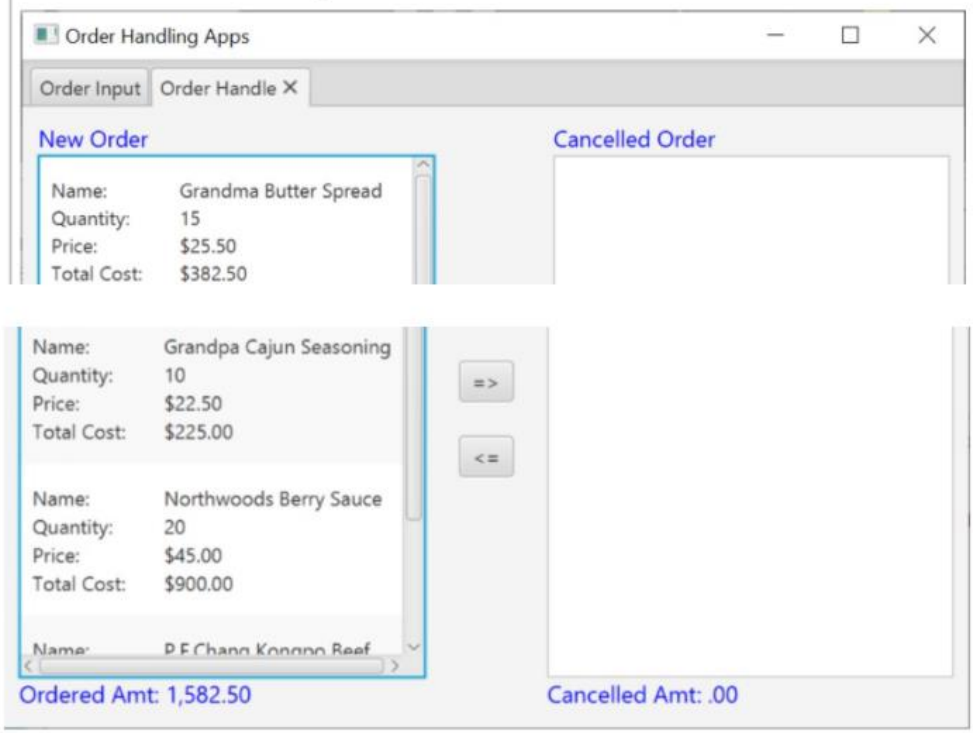


4. After entering several Orders' information, the GUI window will have the following appearance.

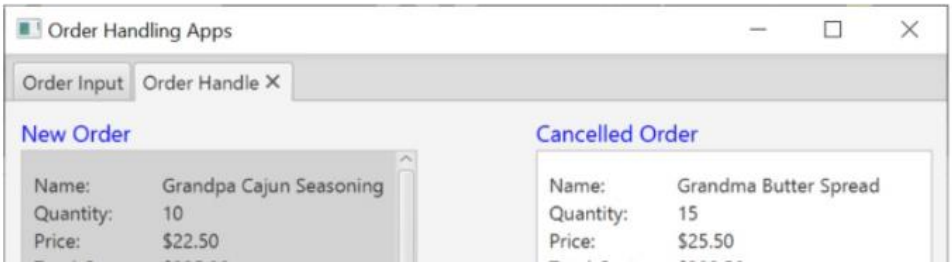


5. Under the "Order Handle" tab, a user can trace a new or cancelled order. The layout should be as the following. The left part will have a list of "New Order" where these new orders are created under "Order Input" tab, *i.e.* at beginning it should contain exactly the same order list as the one under the tab "Order Input". At the bottom it also display the total amount of all orders listed under "New Order"

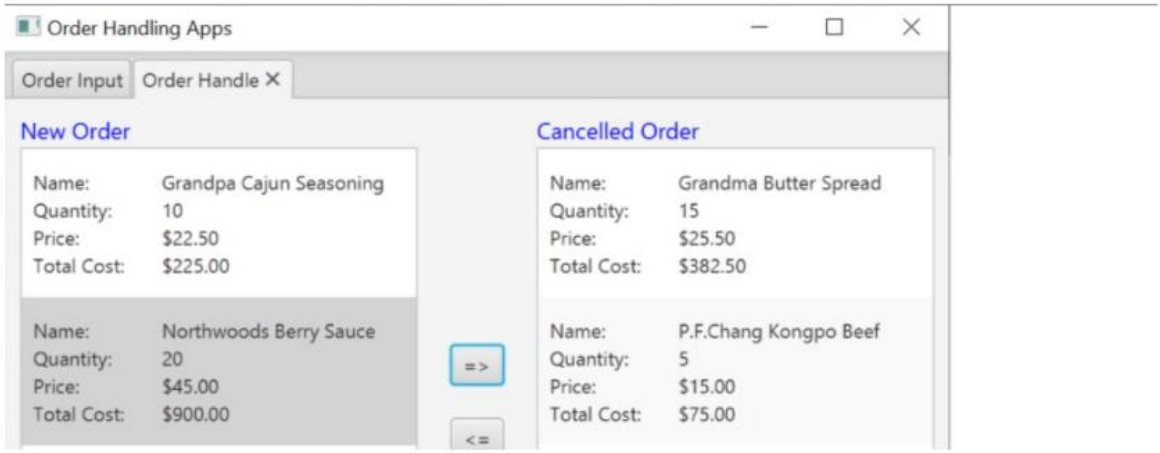
In the middle, there are two buttons, namely "=>" and "<=".



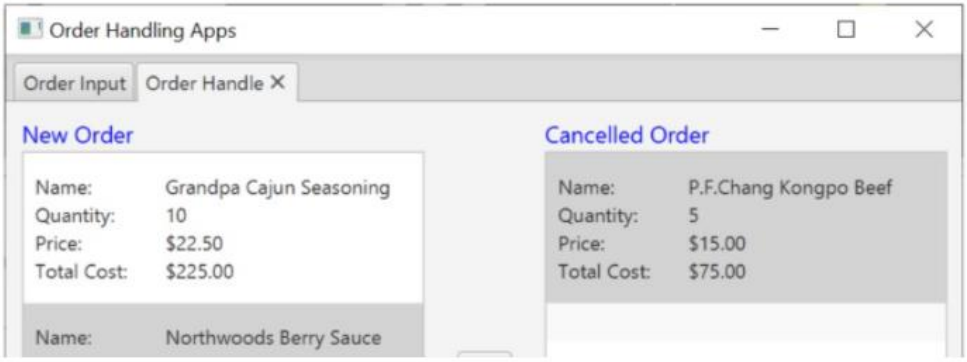
6. The "=>" button allows us to cancel a new order, *i.e.* to move a new order from the left hand side "New Order" list to the right hand side "Cancelled Order" list. We can first pick a new order from the left, then press the "=>" button, it will then be removed from the "New Order" list and placed under "Cancelled Order" list. The relevant "Ordered Amt" and "Cancelled Amt" should be updated accordingly. See below for one example that we removed the first order "Grandma Butter Spread" from the "New Order" list to the "Cancelled Order" list, notice the change of "Ordered Amt" and "Cancelled Amt".



We can continue using "=>" button to move an order from "New Order" list to "Cancelled Order" list until the "New Order" list is empty. See below for another example where we moved "P.F.Chang Kongpo Beef" from the "New Order" list to "Cancelled Order" list (cancelled this order).



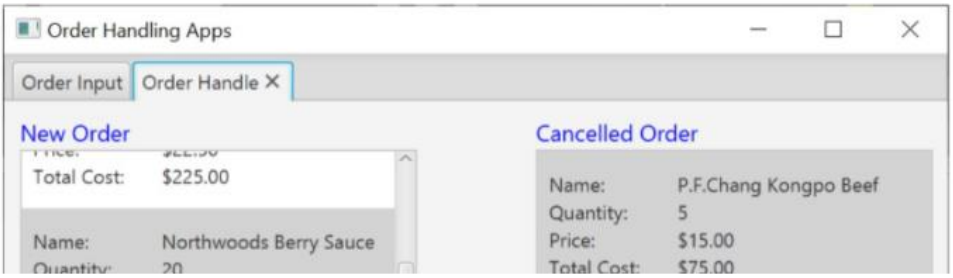
7. The "<=" button allows us to uncancel an order, *i.e.* move a cancelled order from the right hand side "Cancelled Order" list to the left hand side "New Order" list. We can first pick an order from the right, then press the "<=" button, it will then be removed from the "Cancelled Order" list and placed at the end under the "New Order" list. The relevant "Ordered Amt" and "Cancelled Amt" should also be updated accordingly. See below for one example that we removed the first order "Grandma Butter Spread" from the "Cancelled Order" list to the "New Order" list.



8. An user can switch back and forth between "Order Input" and "Order Handle" tabs. For example, assume we switched back to "Order Input" tab and enter the following order info. there:

Prod. Name: Papa John's Pizza
Quantity: 2
Price: 12.50

The resulting "Order Handle" tab should be as follows. Note above order is listed under "New Order" list and the total "Ordered Amt" is updated also.



Error Action Handling.

In case user picks an order from the "New Order" list, but press the "<-" button, or user picks an order from the "Cancelled Order" list, but press the "->" button, nothing should happen, *i.e.* your program should consider the cases, and shall not thrown any exceptions!

5. Class description

InputPane

InputPane class extends HBox. It should contain at least the following instance variable:

5. Class description

InputPane

InputPane class extends HBox. It should contain at least the following instance variable:

Attribute name	Attribute type	Description
orderList	ArrayList	a list of **Order** objects.
handlePane	HandlePane	an object of HandlePane.

This class should have a constructor:

public InputPane(ArrayList<Order> list, HandlePane pane)

where the parameters "list" and "pane" represents the orderList and handlePane that will be passed from the Assignment6 class. The constructor layouts and organizes components in this pane. You will be adding more variables (components) than what is listed here.

This class contains a nested class called **ButtonHandler** that implements EventHandler<ActionEvent> interface. Thus the ButtonHandler needs to override the following abstract method:

public void handle(ActionEvent e)

HandlePane

HandlePane class extends VBox. It should contains at least the following instance variable. **Hint:** for the two scrollable lists shown on HandlePane, consider using ListView objects.

Attribute name	Attribute type	Description
OrderedList	ArrayList<Order>	A list of Orders that show under "New Order" list to the left of "Order Handle" tab. It should contain exactly the same group of Orders entered in InputPane.
cancelledList	ArrayList<Order>	A list of Orders that show under "Cancelled Order" to the right of the "Order Handle" tab.

This class should have a constructor:

public HandlePane(ArrayList<Order> list)

For HandlePane, you will need to add more variables (components) that is necessary, such as ListView, Button and Label, etc.

public void updateOrderList(Order newOrder)

This method refreshes the ListView of "New Order" whenever there's a new order added in InputPane, it will also update the "Ordered Amt". You will need to update the underline ObservableList object to achieve this.

This class contains a nested class called **ButtonHandler2** that implements EventHandler<ActionEvent> interface. Thus the ButtonHandler2 needs to override the following handle method when user press the "=>" or "<=" button. See the UML class diagram for the parameter and return type of this method.

public void handle(ActionEvent e)

6. Assignment6 class

Assignment6 class extends Application. It contains the start() method to instantiate all instance variables and adds its components to itself. It also sets its size. It contains at least following instance variables:

6. Assignment6 class

Assignment6 class extends Application. It contains the start() method to instantiate all instance variables and adds its components to itself. It also sets its size. It contains at least following instance variables:

Attribute name	Attribute type	Description
orderList	ArrayList	A list of Orders. It will be used in both InputPane and HandlePane. *i.e.* the two pane share a common data set.
inputPane	InputPane	An object of InputPane.
handlePane	HandlePane	An object of HandlePane.
tabPane	TabPane	An object of TabPane. It will contain inputPane and HandlePane under each of the two tabs.

Note: Assignment6.java will generate the two tabs for you and you need NOT to change it.

7. Misce.

- Your program's layout need not to be EXACTLY the same as above shows, but should be similar. In other words, each GUI components and their relevant layout, position, size should be similar to what we show above.
- We recommend that you design/set up the layouts in InputPane and HandlePane first, when the layout is ready, you can then consider handling the events generated from buttons or ListView.
- **There are no test cases for this assignment**, we will download, run/test your application GUI individualy.

8. Submission

1) For this GUI program, there are not test cases. Our TA will manually test each student's submission.

2) To submit, login to our Canvas course website, from left hand side control panel, click on gradescope, it will bring you to a seperate gradescope.com website. Submit ONLY the source code file(s), for this assignments, you will need to submit the following four (4) source code files, make sure your file's name is EXACTLY the same as stated below, if you submit a wrong file name, your program might fail all test cases!

- **Assignment6.java**
- **Order.java**
- **InputPane.java**
- **HandlePane.java**