| **Project – External Documentation** |  |
| --- | --- |
| <ISYS6197003>  <Business Application Development> |
| <**Odd**/Even/Compact> Semester Year <2023/2024> |

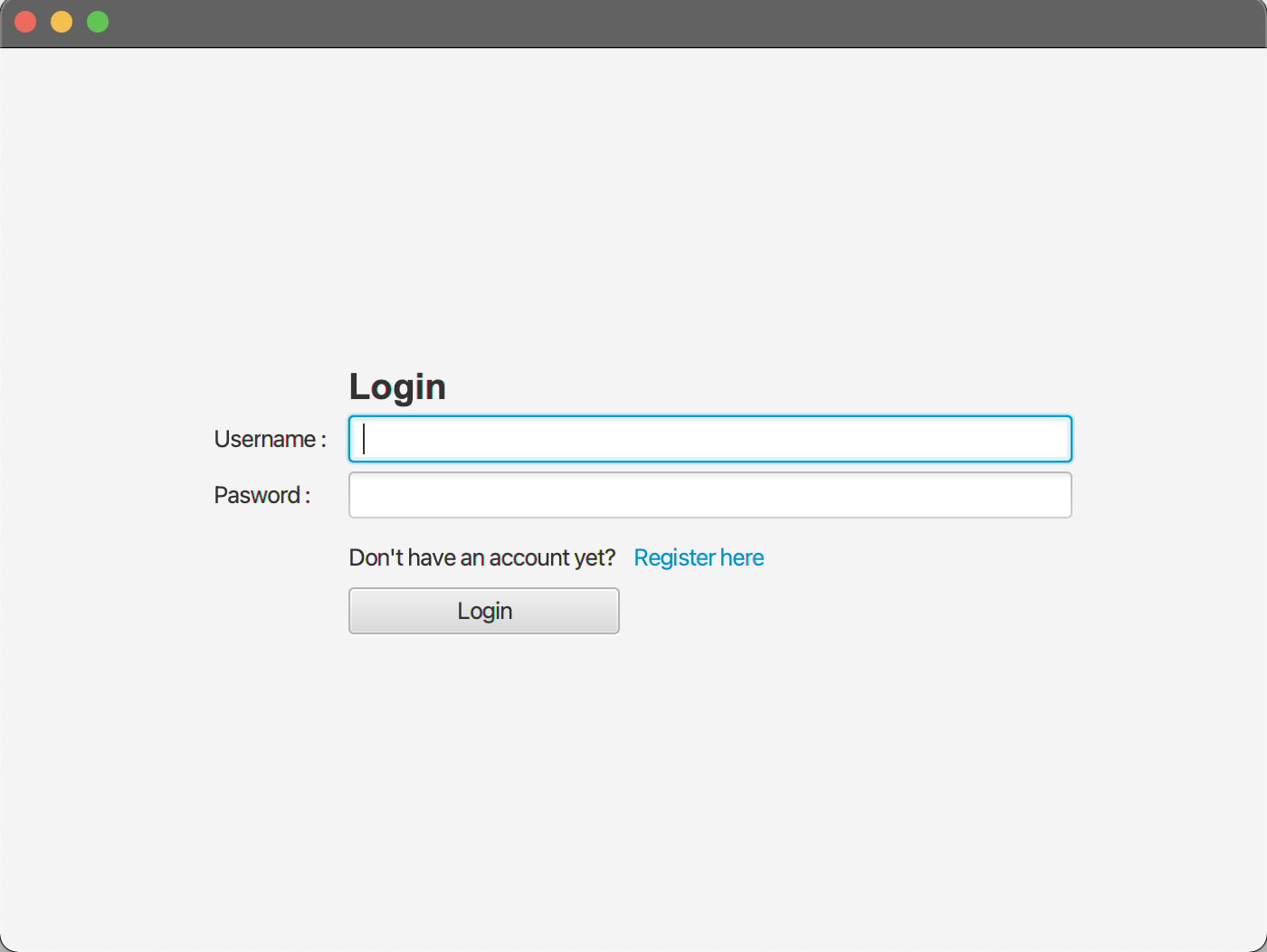
* **Project Title**

SeRuput Teh Business Application Development

* **Introduction**

SeRuput Teh is an expanding store specializing in premium tea products. SeRuput Teh aims to enhance the customer experience by launching a digital platform. The owner has given you their trust to develop a Java application to manage SeRuput Teh’s inventory and transactions. The program should be made using Java Programming Language, with MySQL Database Engine. Ensure that you didn’t use regex, Java FXML, JavaFX Scene Builder, or any library used outside the class materials in your project or it will affect your score.

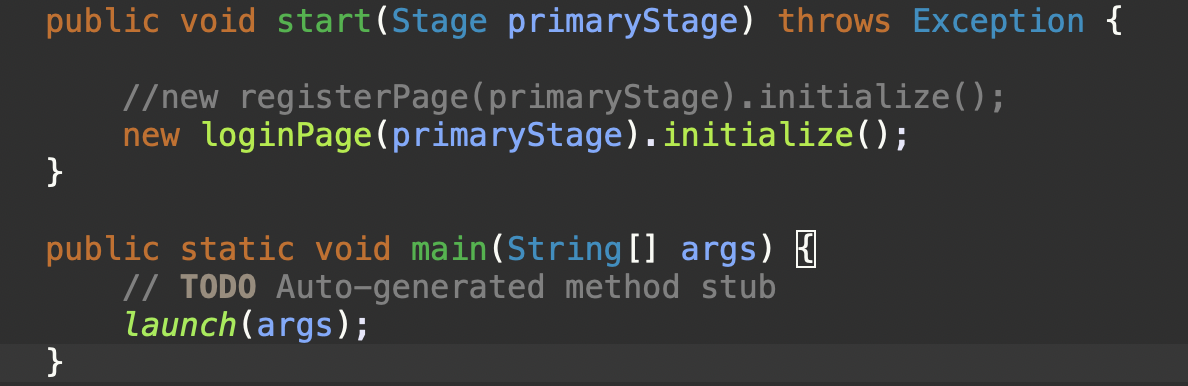
* **Report / Documentation**

1. Login Scene  
   

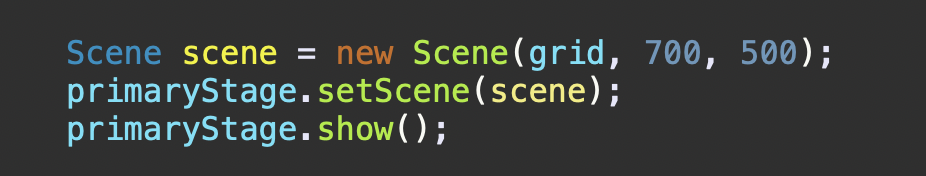
The Login Scene is the initial interface users encounter upon launching the application. Users enter their credentials (username and password) to log in. Upon successful login, users are directed to their role-specific page (customer or admin). If users do not have an account, they can click a “register here” to navigate to the registration page.

* 1. **Component**

To create the Login Scene interface we use JavaFX as our language. The first step is to **create a stage named “PrimaryStage ''** so we can add some components to it.



After we create the stage we **generate the scene for the login page named "scene".**

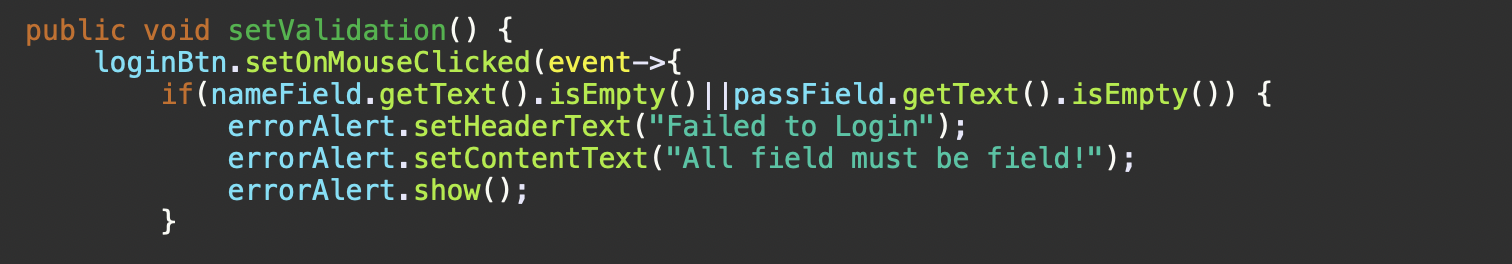


Next **add and initialize components within the gridPane**, such as labels to create a text interface (e.g., nameLbl, passLbl, title, registerInvite), textField and PasswordField so user can fill in their username and password(nameField & passField), and a flowPane called “ registerHook” to handle registration redirection(registerInvite & registerLink).

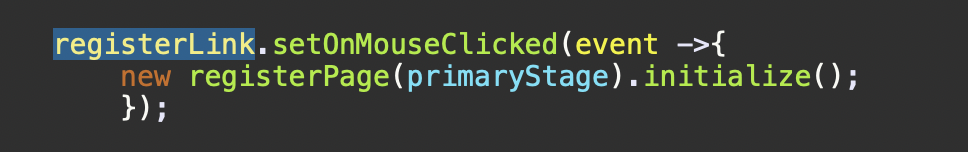


The next step is to **input and arrange all of the components into the gridPane** so the layout becomes neater. The X axis is used to arrange the column meanwhile the Y axis is used to arrange the row.

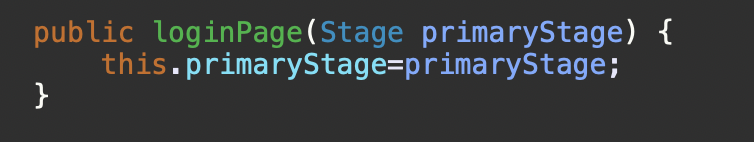


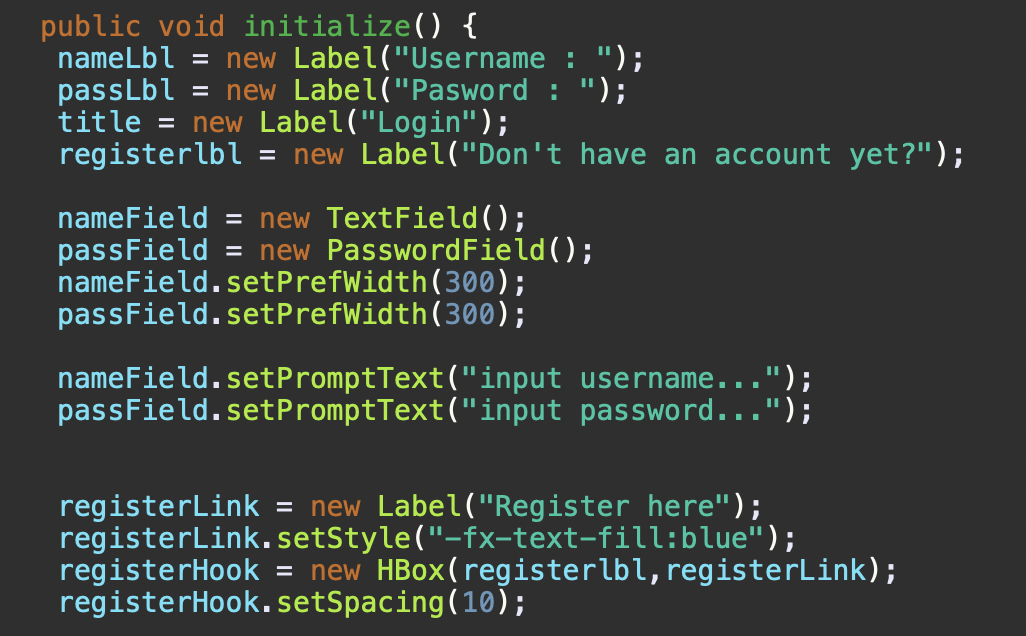
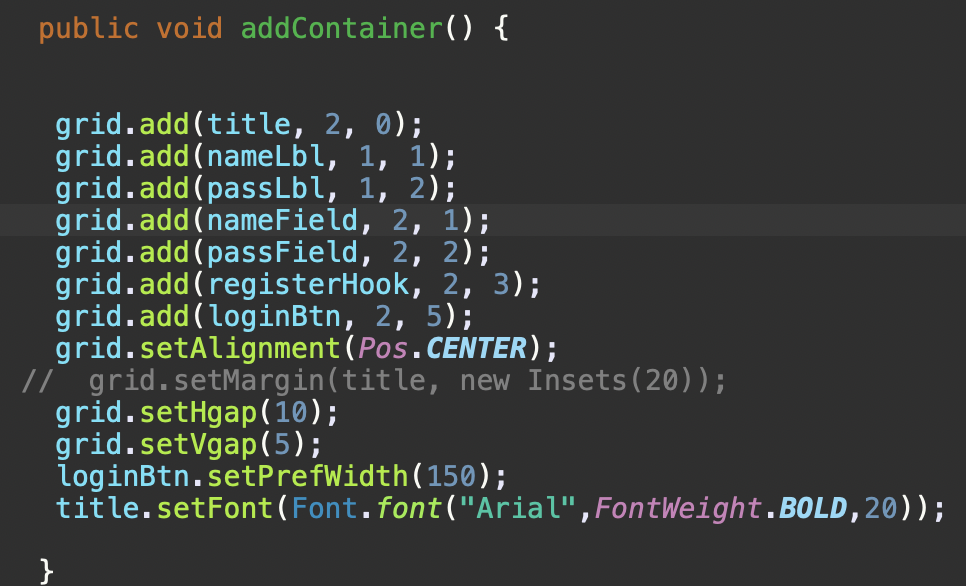
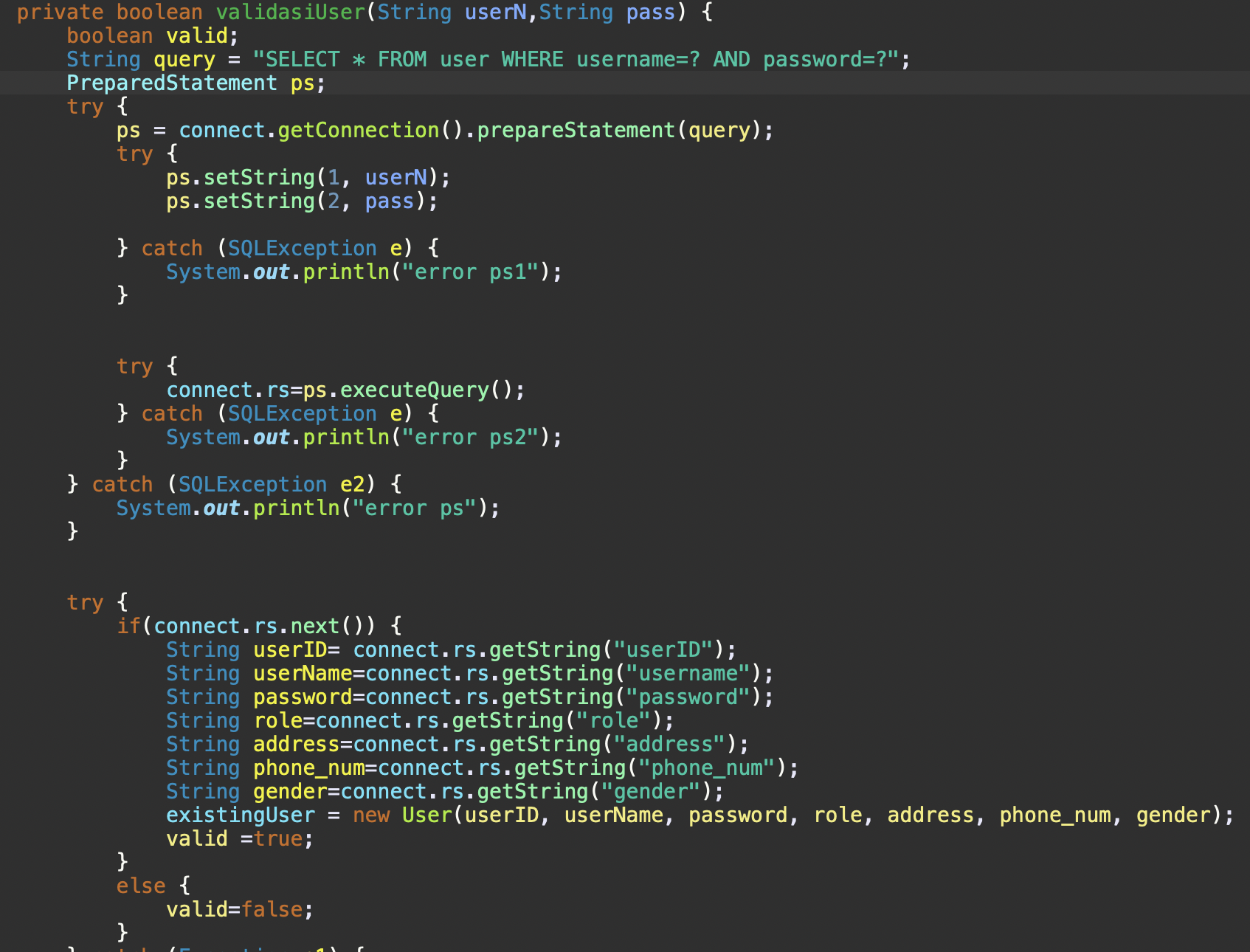
* 1. **Event Handler**In the Login Scene, two event handlers are implemented. The first, associated with the loginBtn, resides in the setValidation function. When users click the loginBtn, this function is triggered. The setValidation function includes multiple validations (elaborated in the method section). If the credentials pass validation, the user successfully logs in.

The second event handler is linked to the registerLink. Clicking this label directs users to the register page, facilitating the registration process.

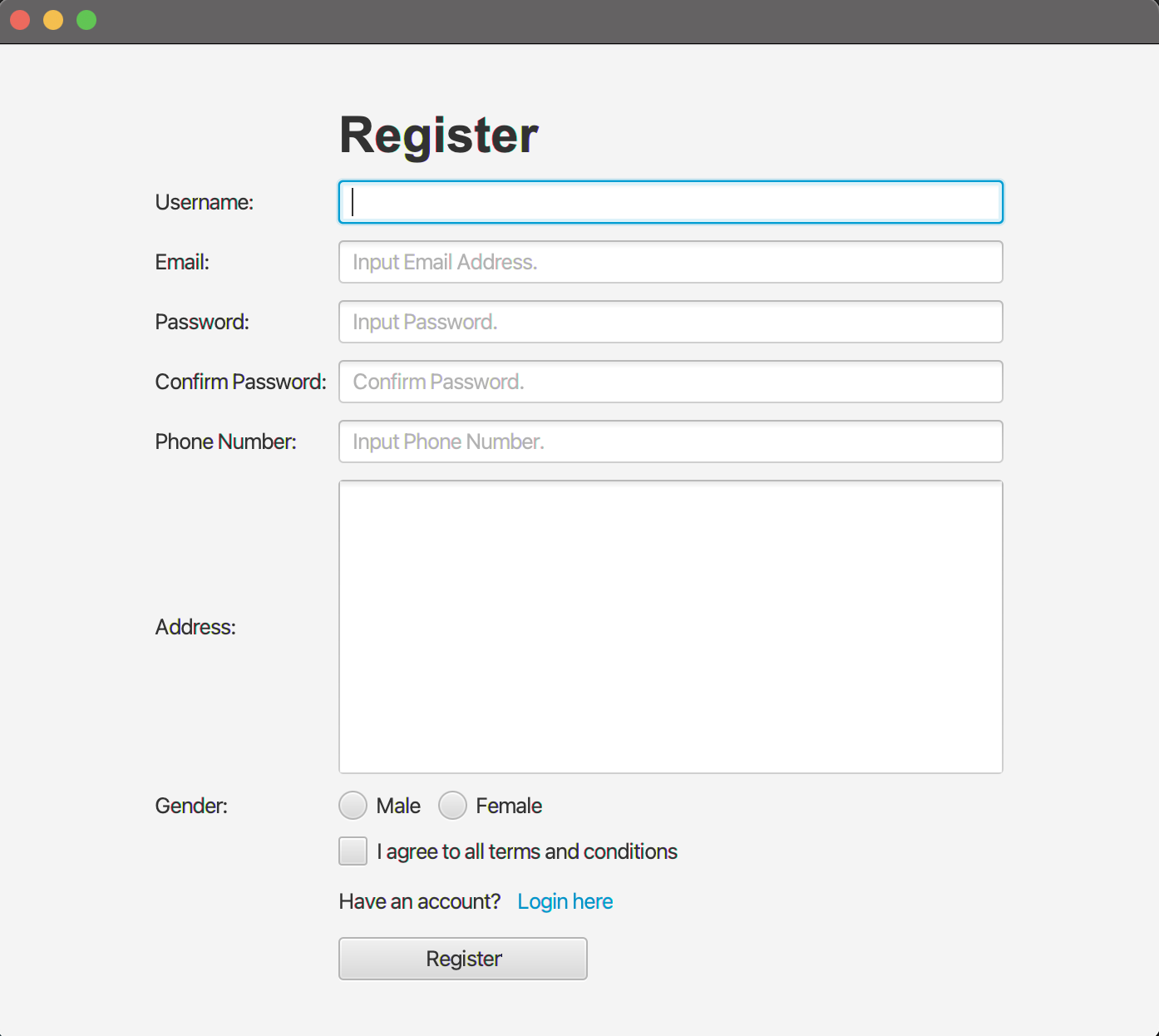
.

* 1. **Method**
     1. loginPage method is used for setting up the loginPage as the primaryStage for this application. So the first interface that the user will interact with is the loginPage.

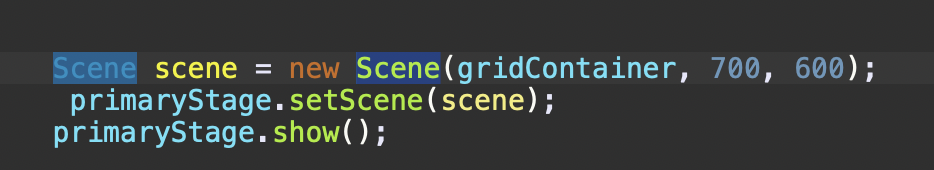


* + 1. The initialize method is used for initializing all of the components into objects so it could be used to layouting the LoginPage interface.  
       
    2. The addContainer function is used for inserting all of the components that had been initialized into a single gridPane so it could be laid out.  
       
    3. validaUser is a method for validating whether the password and username match in the database or not.  
       
    4. The setValidation function is used to compile all of the validation to see whether the user’s input is correct or not. The validation is triggered when the login Button is clicked then the function will check for any false input. If the input is invalid the system will pop up an error message and the login fails. The validation includes checking if the input is complete/not empty, and checking the user roles(customer or admin).  
       

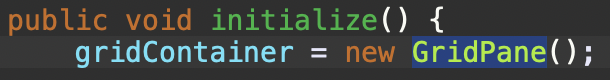
1. Register Scene

  
The Register Scene is where users register for application access by clicking the “register here label” on the loginPage. During registration, users input personal details such as name, address, email, password, etc. Successful validation allows user registration, while validation failure prompts an error message. Users with existing accounts can navigate to the login page by clicking the “login here label” at the bottom.

* 1. Components  
     The Register Scene is filled with alot of components, first of all there are the scene components for generating the registerPage.



Next, we create a gridPane to encapsulate all of the components for the registerPage.

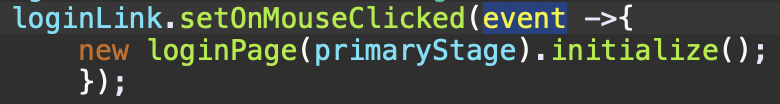
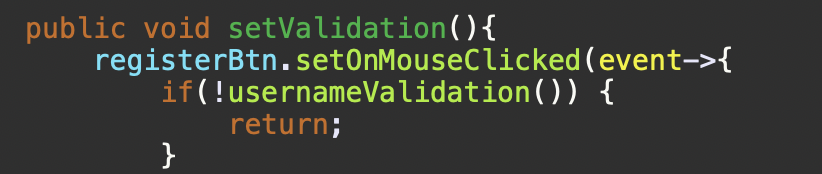
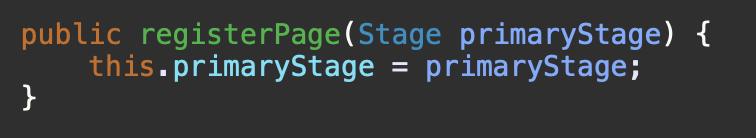
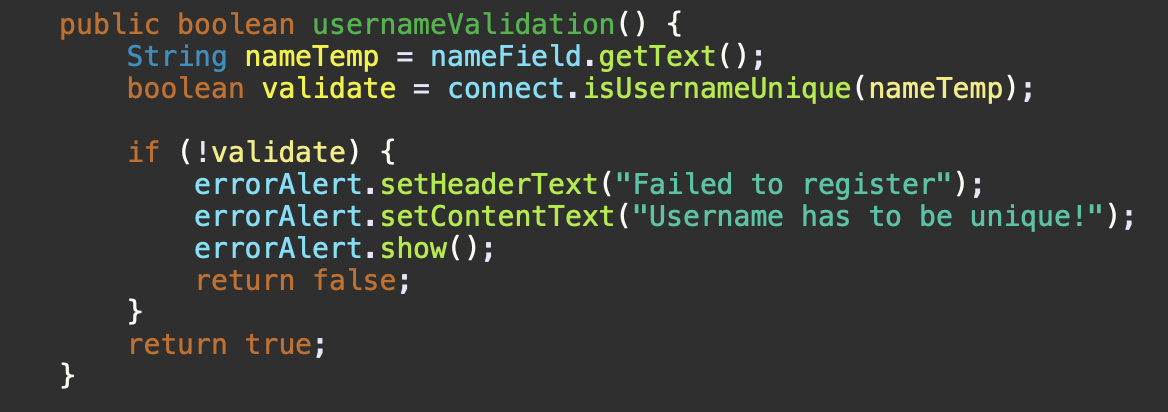
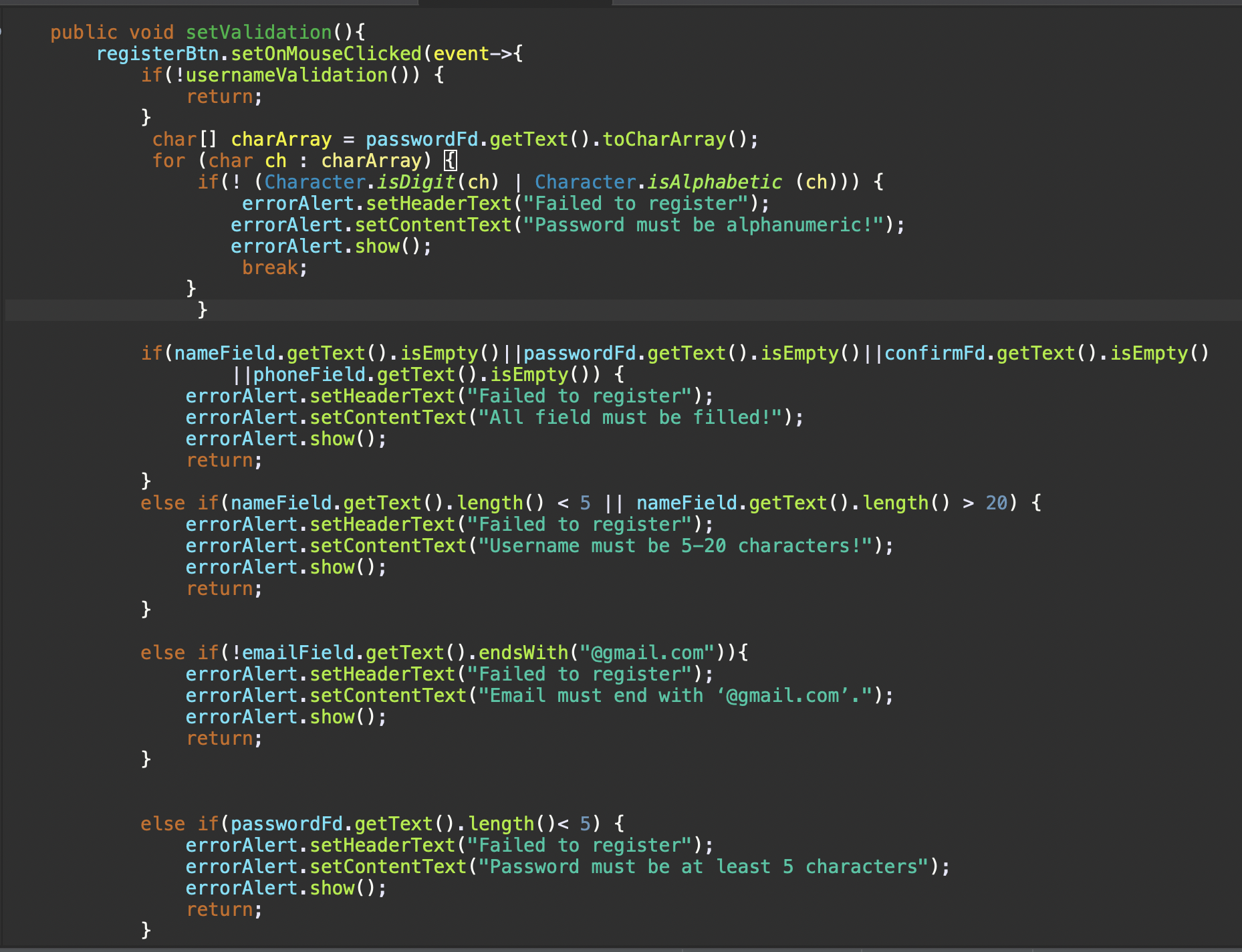


After that **we start to initializing all of the components** such as labels to create a text interface (e.g., nameLbl, passLbl, etc), textField, PasswordField, textAddress so users can fill in their personal info(emailField, nameField, passwordFD), radioButton so users can pick their gender, Flowpane to encapsulate the radioButton, Checkbox so the users can check the agree with terms and conditions, lastly alerts to display a pop up message to the users can be about login successful or an error message.



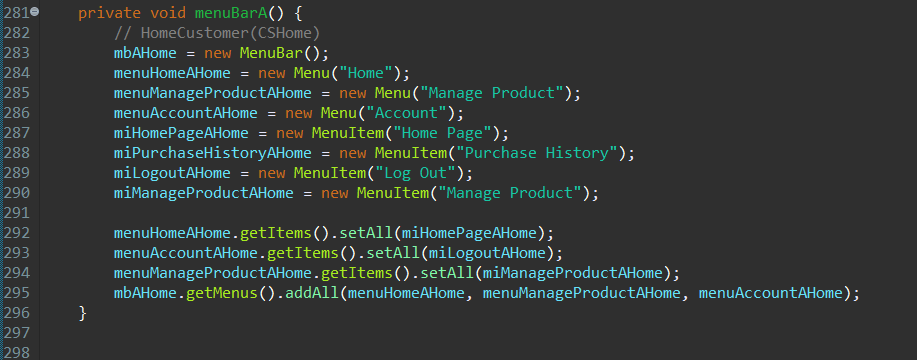
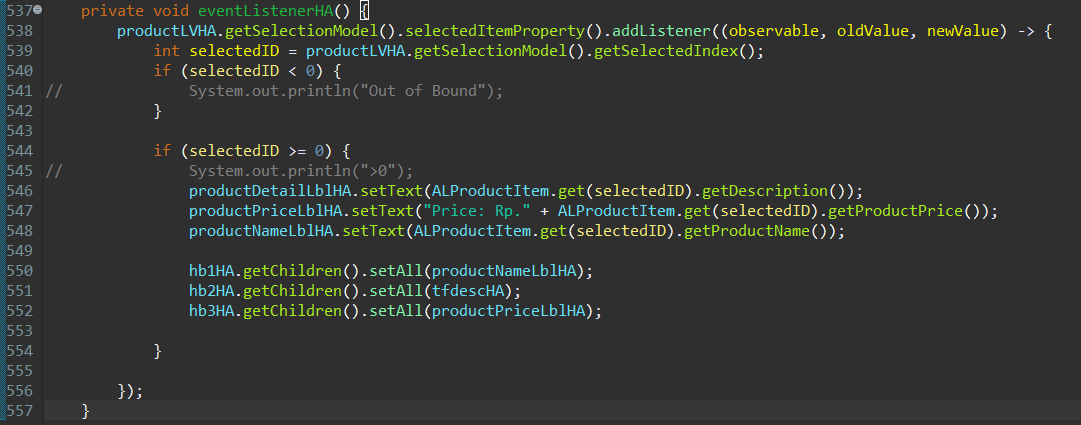
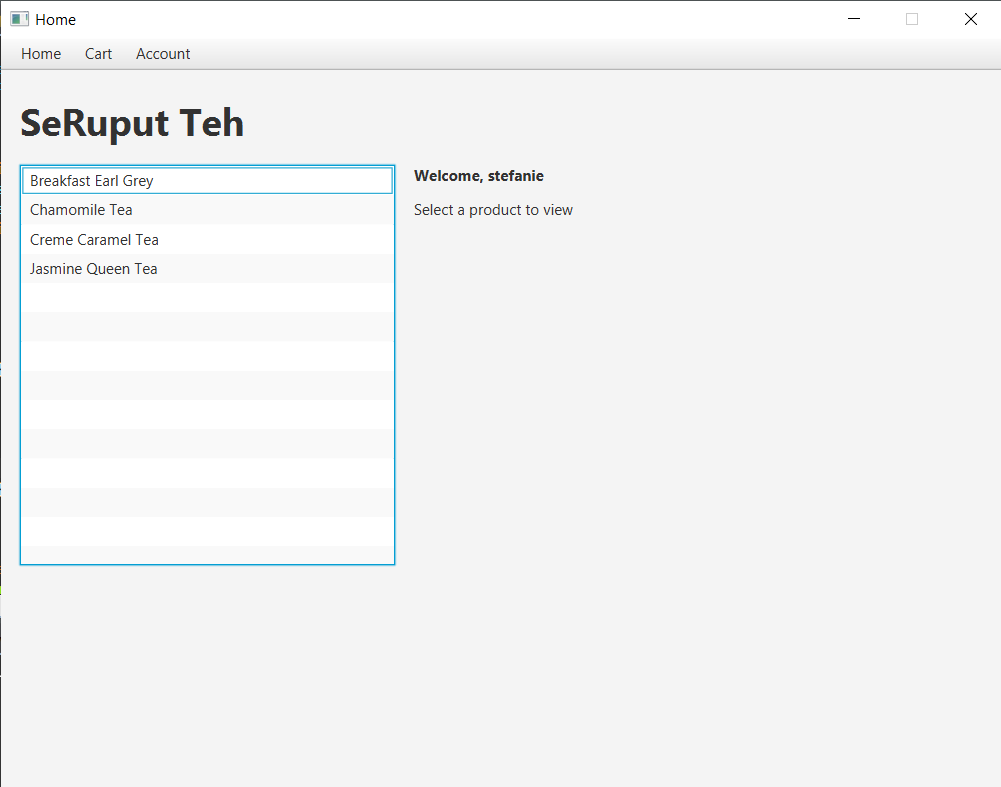
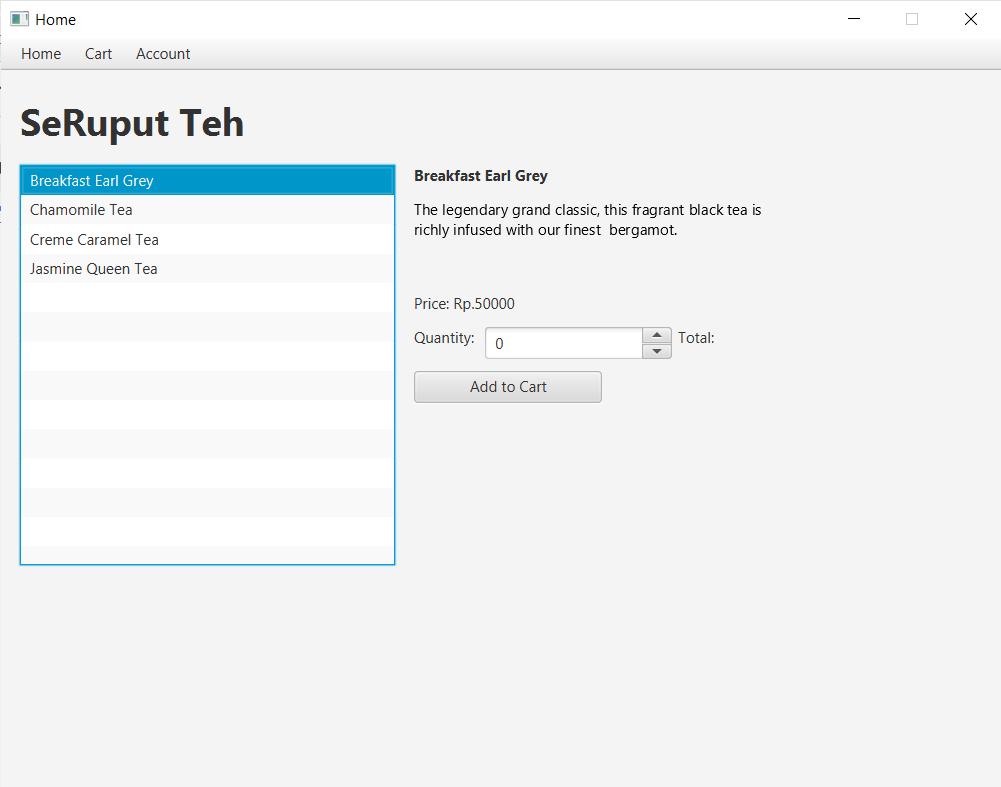
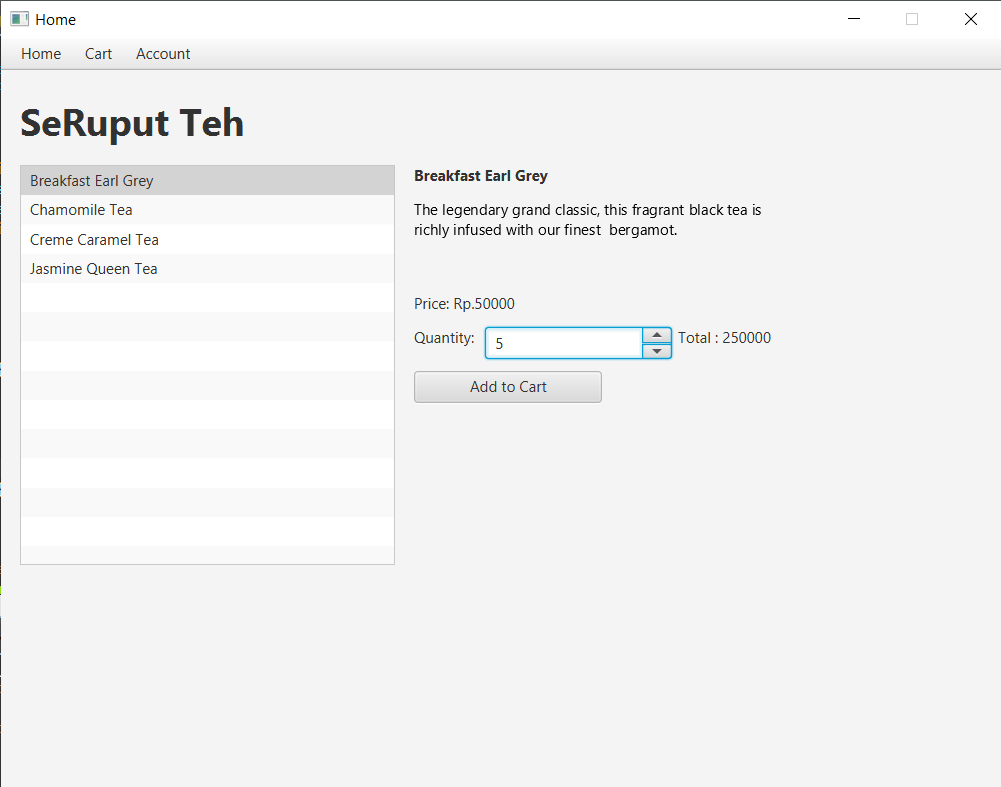
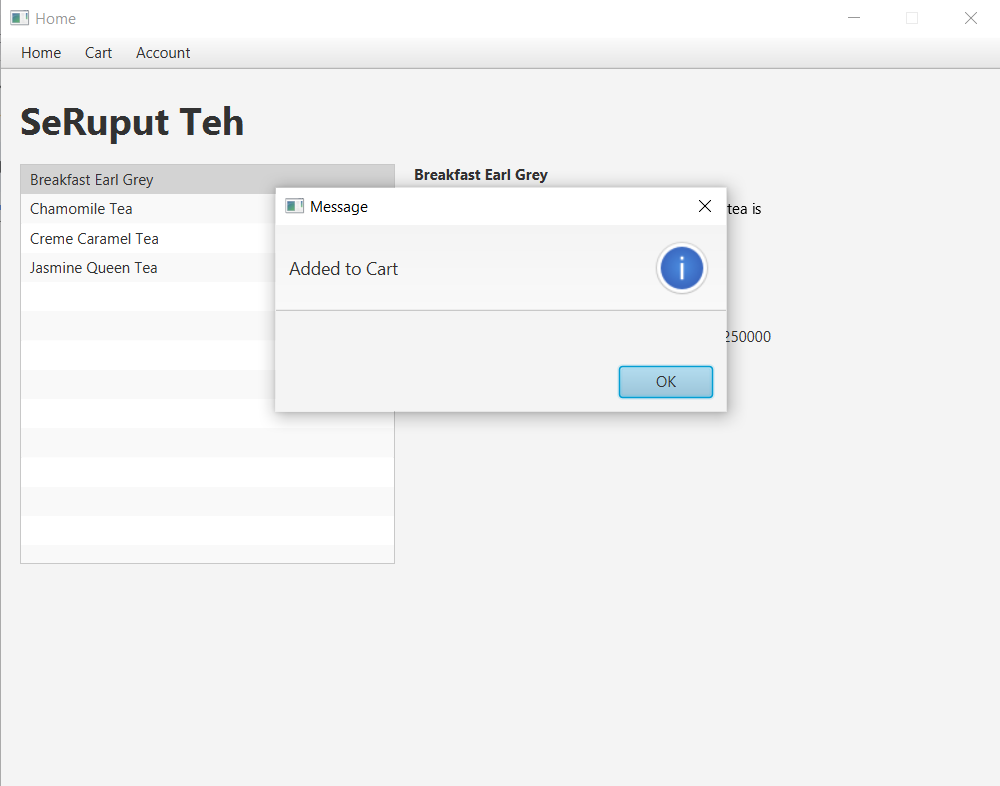
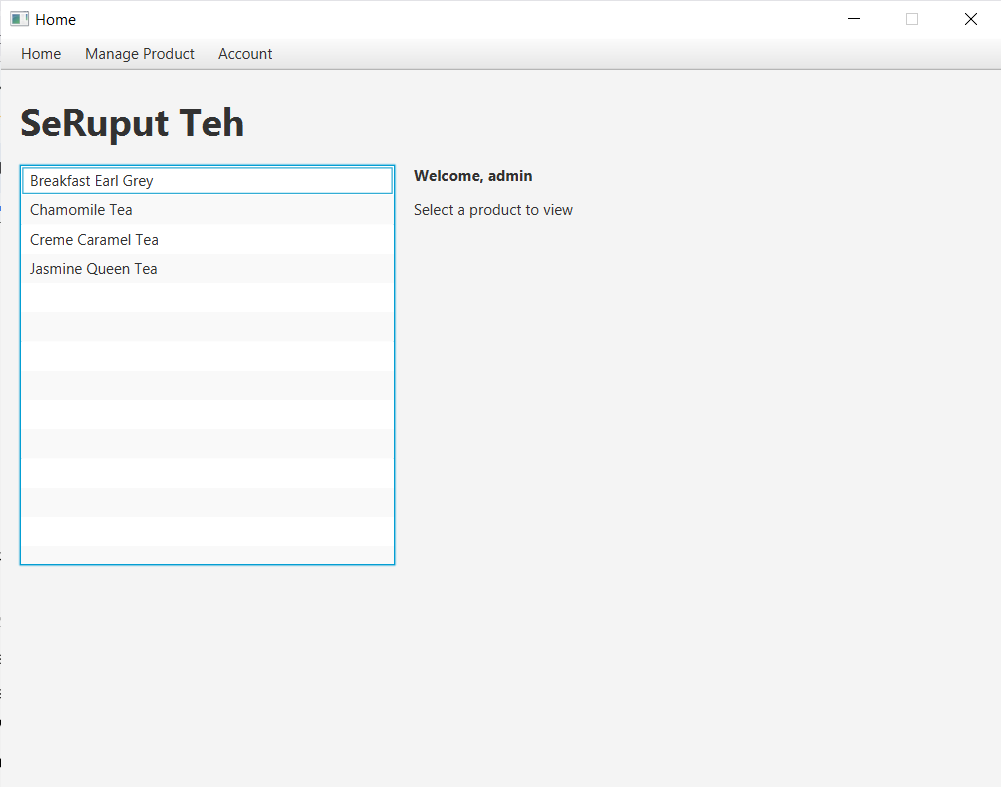
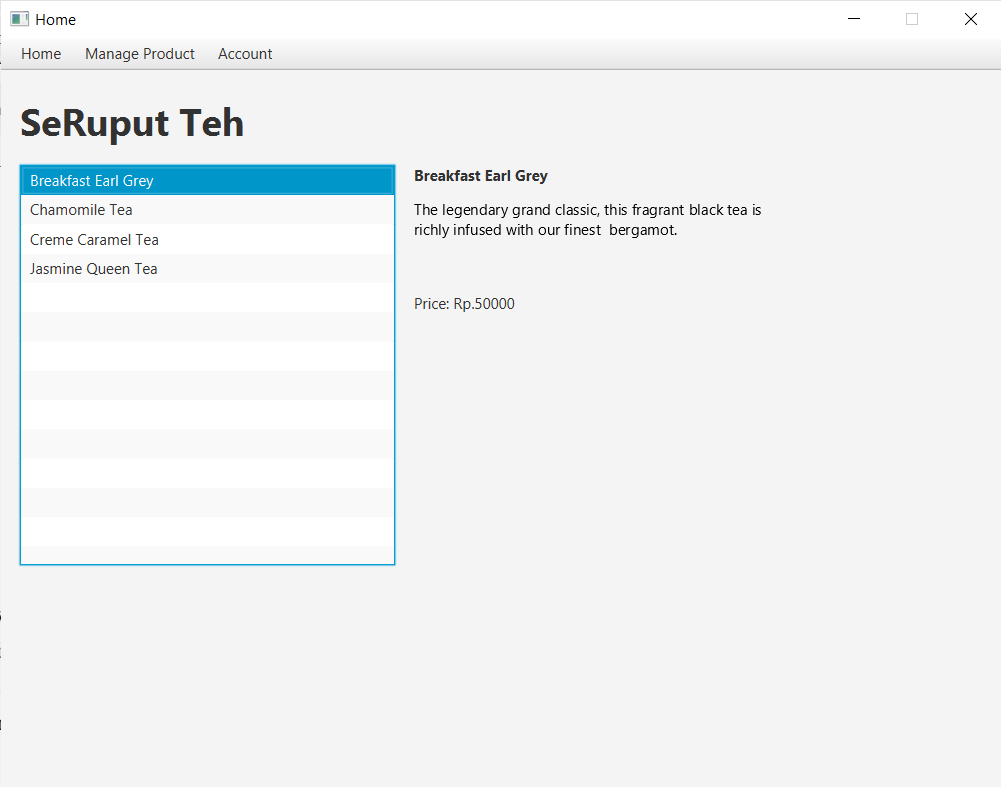
Finally, we input all of the objects into a gridPane that previously had been created.



* 1. Event Handler  
     There are two eventHandlers, the first one is located in the login here label so if a user clicks it they will be directed into the loginPage.  
       
     The next event handler is on the registerBtn, so when the user clicks the button the validation function will be triggered. The setValidation function includes multiple validations (elaborated in the method section). If the credentials pass validation, the user successfully registered themselves.  
     
  2. Method:
     1. registerPage method is used for setting up the registerPage into the primaryStage.  
        
     2. The initialize method is used for initializing all of the components into objects so it could be used to layouting the RegisterPage interface.  
        
     3. The addContainer function is used to add all of the components into a gridPane so the components can be laid out.  
        
     4. The usernameValidation function is used to validate whether a username is unique or not. If not then an error message will pop up, but if its unique the username passes this validation.  
        
     5. setValidation function is a function that contains all of the validation for the registerPage such as validation like username unique, valid format, text field, etc. If the input doesn’t comply with one of the validation then an error message will pop up. If the user passed all of the validation then the user is registered to the system.  
         

1. Home Scene
   1. Components

The Home scene is the main page of the SeRuput Teh digital platform. The Home Scene is available for both customers and admin. Although, there is a slight difference between the GUI for customers and admins. In this scene, the user can view all products that are available in the database on a ListView. Customers can click on one of the products on the list to view the details of the selected product including a button to add the product to the cart. Admin can also view all products that are available in the database on the ListView but they can’t add products to cart.

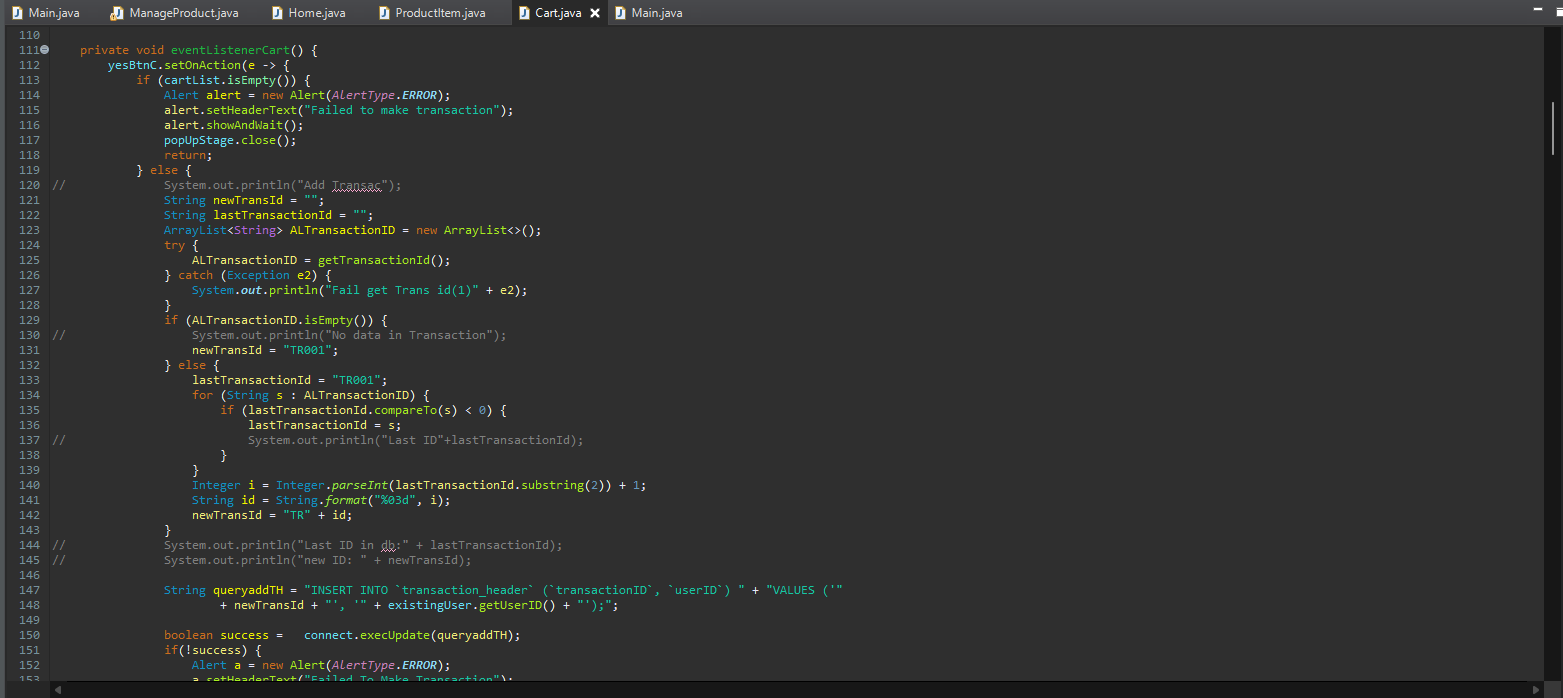
* 1. Event Handler
     1. Menu Event Handling:   
        In the Customer Home page, the menu bar contains; Home, Cart, and Account are among the choices available on the menu bar are menu items that have event listeners linked to them so that they can be used to start activities such as going to the home page, handling products, or logging out. In the Admin Home page, the menu bar contains; Home, Manage Product, and Account.   
        Code(Customer):  
        Code(Admin):  
        
     2. ListView Selection:  
        Monitors item selection in the cart's ListView. Displays detailed information about the selected product, including its description, price, and allows quantity adjustment.   
        Code:  
          
        Code(Admin):  
        
     3. Add to Cart button:  
        If the selected product is not in the current user’s cart, the selected product with the preferred quantity from the spinner will be added to the current user’s cart. The data will also be stored on the Cart table in the database. If the selected product is already in the current user’s cart, the data of the product in the Cart table will be updated. The quantity of the product will be updated to the sum of the quantity in the cart plus the value of the spinner. Display an Information Alert if the product is successfully added to the cart and an Error Alert if failed.   
        Code:  
          
          
        
  2. Methods:
     1. generateObservableListHC & generateObservableListHA  
        Generate the observable list of the available product from the database by putting it into the “productNameListHC” or “productNameListHA” arraylist depending on the user’s role so it can be accessed in ListView.
     2. getProductData  
        Add the product datas from the database by putting it into the “ALProductItem” arraylist so it can be easily accessed later.
     3. getCartData  
        Add the cart datas from the database by putting it into the “ALCartItem” arraylist so it can be easily accessed later.
  3. Execute
     1. Welcoming/first entering Home page (Customer):  
        
     2. Select a product in ListView (Customer):  
        
     3. Quantity Spinner (Customer):  
        
     4. Add to cart (Customer):  
        
     5. Welcoming/first entering Home page (Admin):  
        
     6. Select a product in ListView (Admin):  
        

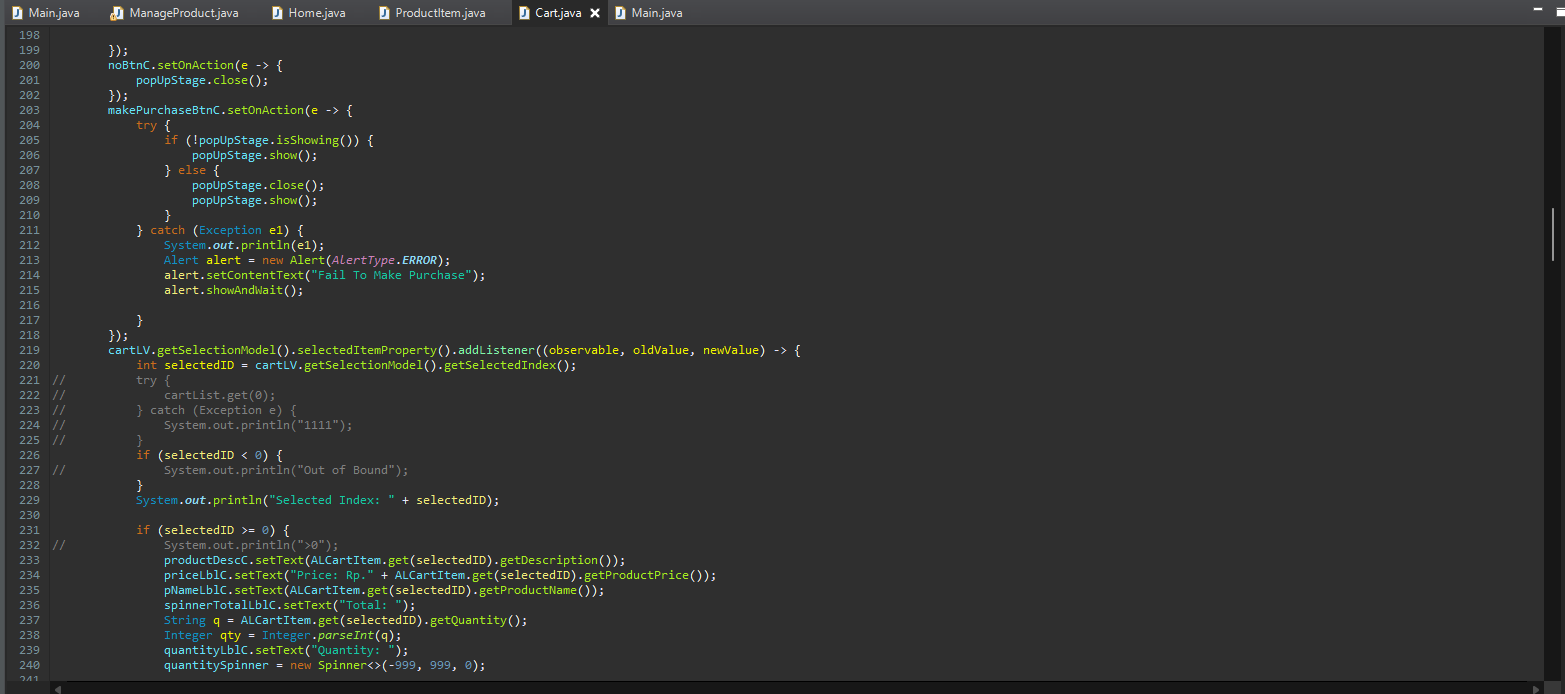
1. Customer Cart Scene
2. Components



The Cart GUI has a menu bar with options like "Home," "Cart," and "Account." The main screen shows your name, the total price of items in your cart, and order details. Your selected items are listed neatly, displaying product names, prices, and quantities in a ListView. You can easily change quantities or remove items using buttons. The primary layout, a BorderPane, structures the interface into distinct regions, with a menu bar at the top and the main content in the center. Within the center, a GridPane is utilized to arrange components systematically in rows and columns. The horizontal layout, HBox, is employed for organizing elements like labels and buttons side by side, providing a clean and linear display. Conversely, the vertical layout, VBox, is employed in the pop-up window to stack components, such as labels and buttons, ensuring a clear and organized presentation.

1. Event Handler

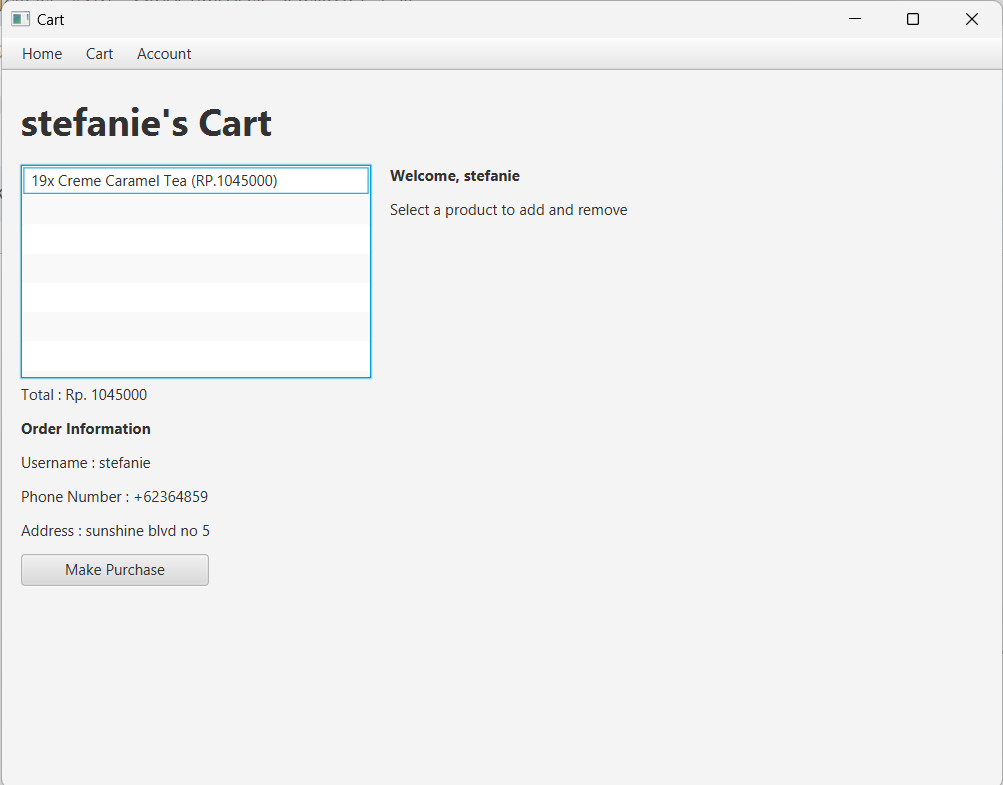




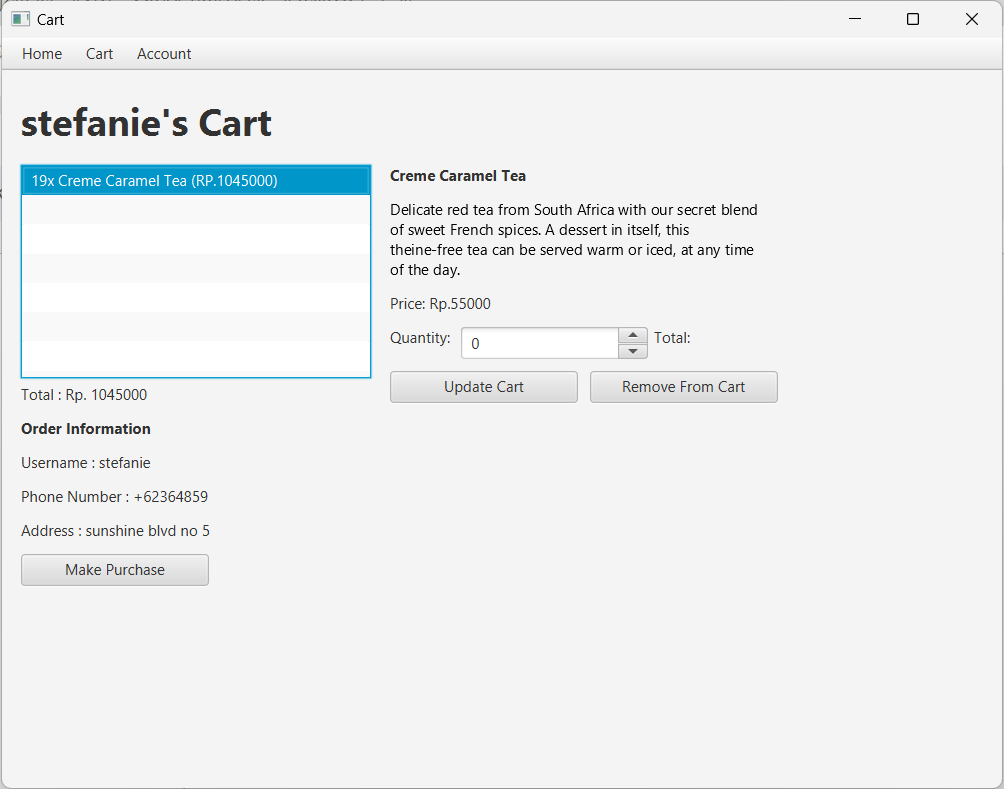
* 1. Menu Event Handling:
     1. Responds to menu items like "Home Page," "My Cart," and "Log Out."
     2. Navigates the user to different sections of the application based on their selection.
  2. Cart ListView Selection:
     1. Monitors item selection in the cart's ListView.
     2. Displays detailed information about the selected product, including its description, price, and allows quantity adjustment.
  3. Update Cart Button:
     1. Modifies the quantity of selected items in the cart.
     2. Updates the cart's total price dynamically based on the changes.
  4. Remove From Cart Button:
     1. Removes the selected item from the cart.
     2. Refreshes the cart display and total price accordingly.
  5. Make Purchase Button:
     1. Triggers a pop-up window to confirm the purchase.
     2. Executes the transaction by updating the database with the user's purchase details.
     3. Clears the cart and provides an informational alert upon a successful purchase.
  6. Pop-Up Window (Yes/No Buttons):
     1. Manages the pop-up window for purchase confirmation.
     2. Allows the user to confirm or cancel the purchase with "Yes" and "No" buttons, respectively.
  7. When user has no product in cart



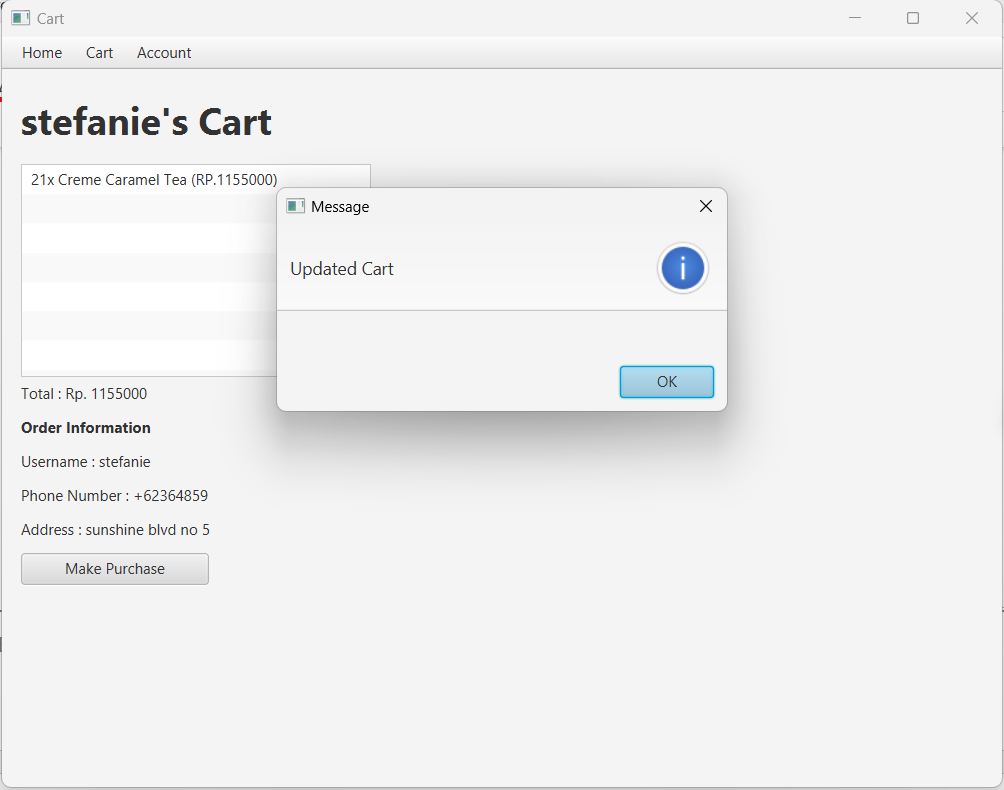
* 1. When the user has product in cart



* 1. When a product is selected

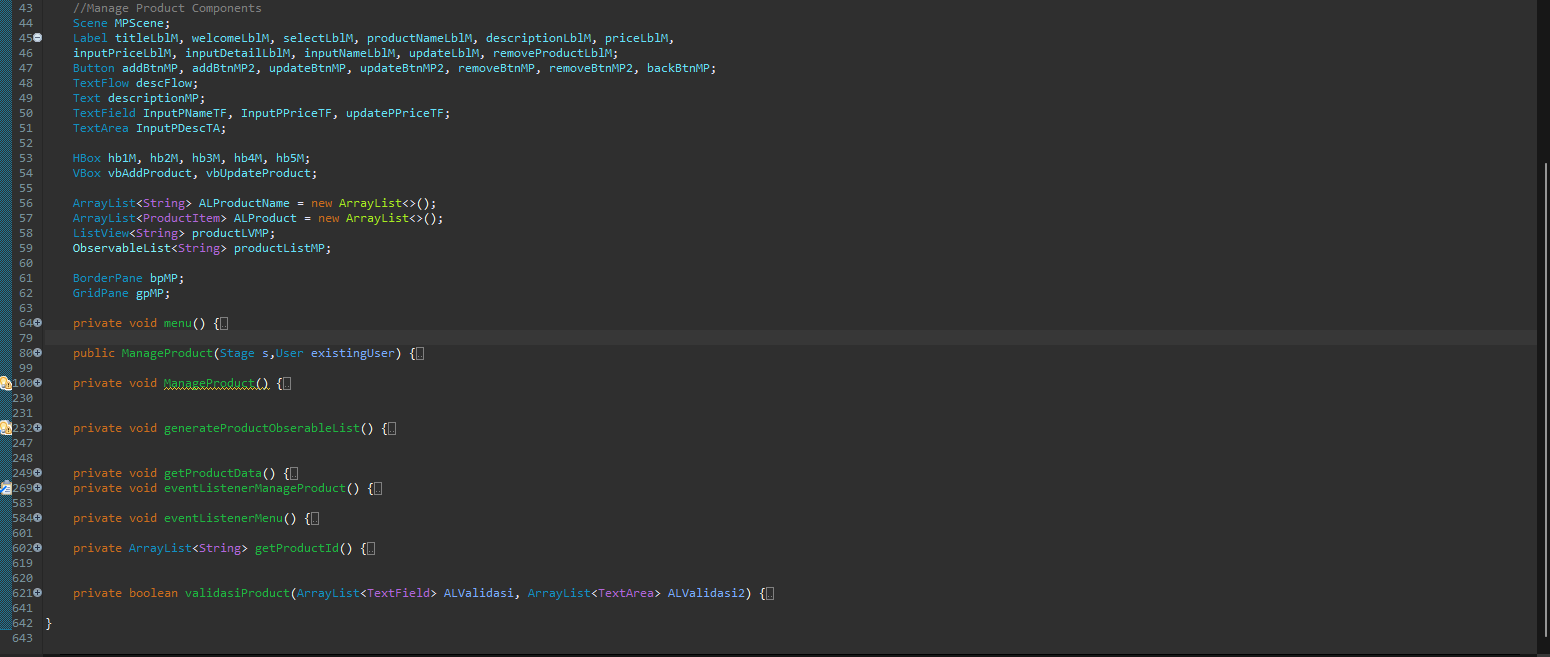


* 1. When product quantity updated



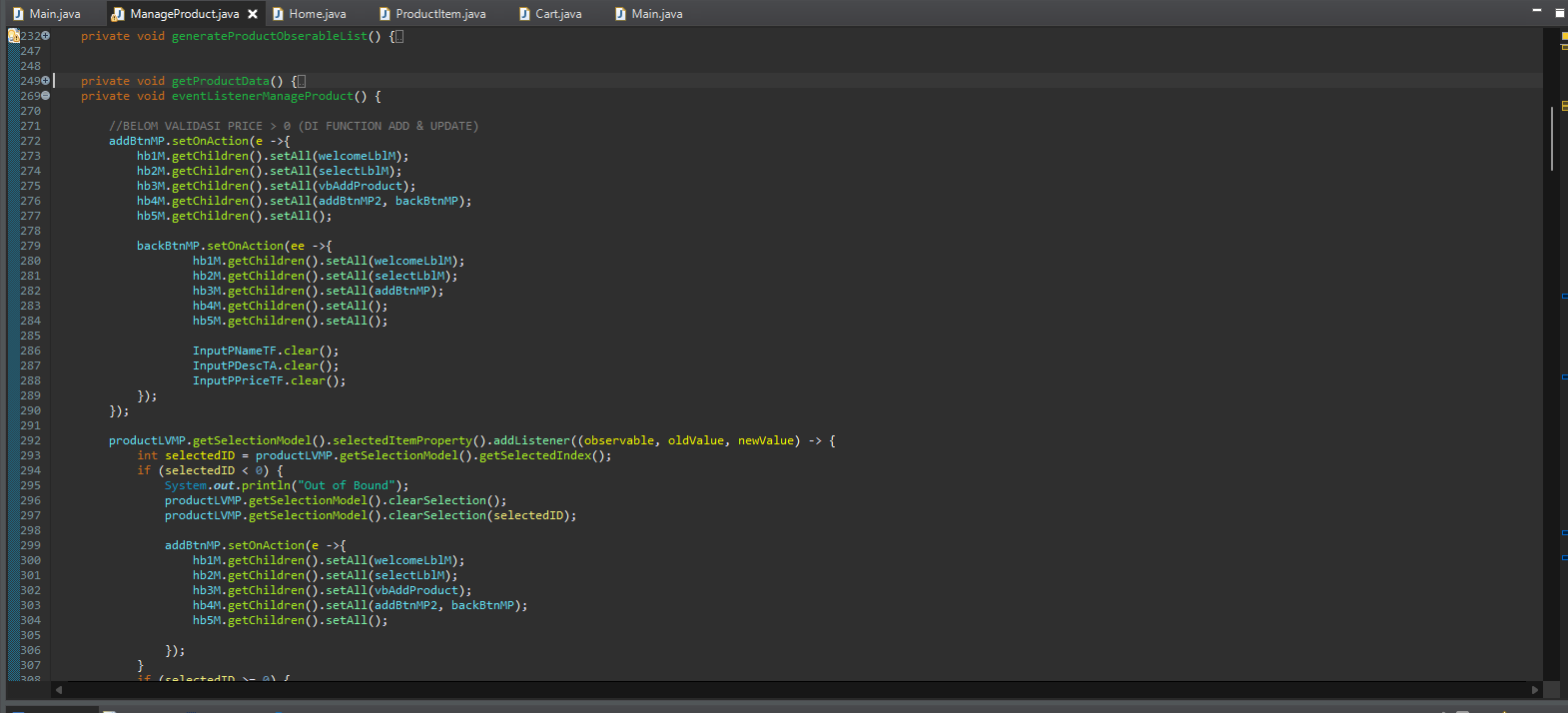
* 1. When user clicked make transaction

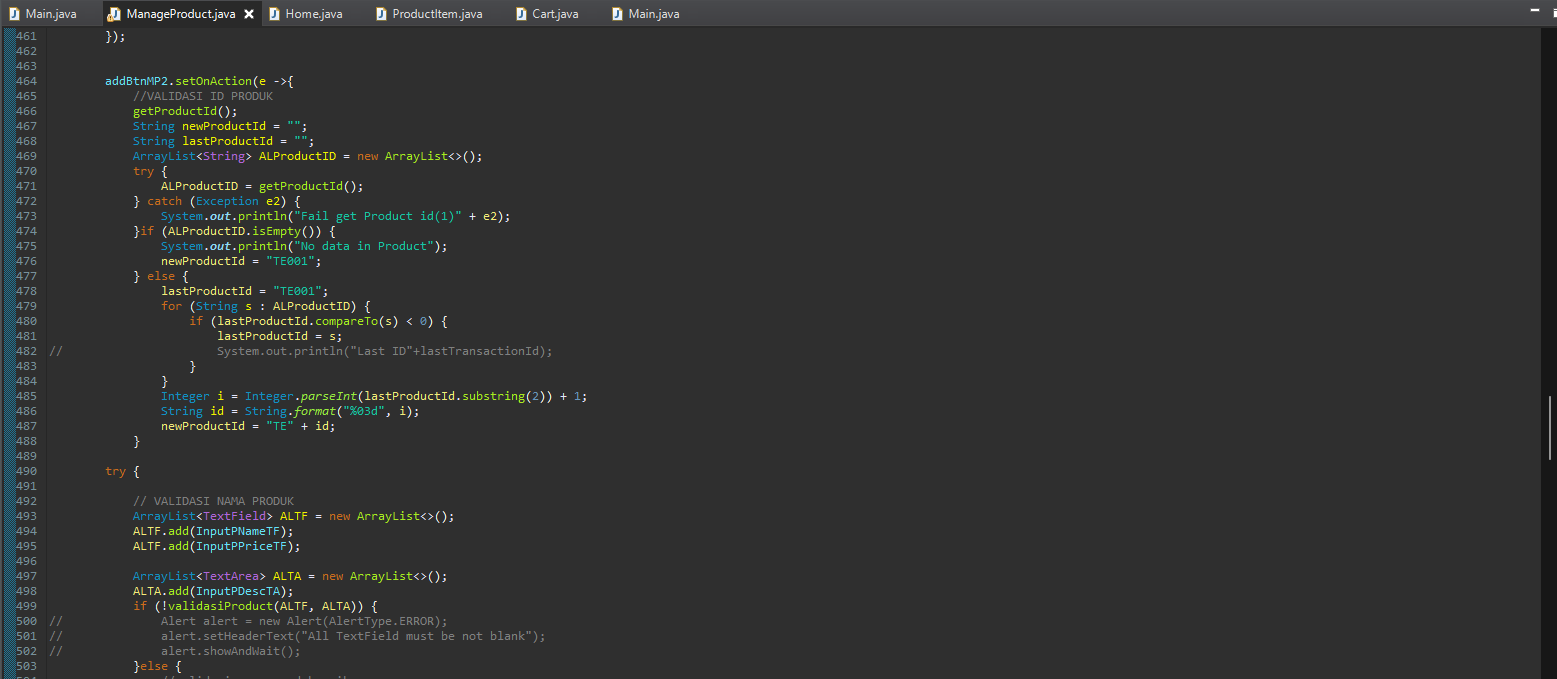
1. Manage Product Scene
2. Component



The Manage Product GUI is a user interface designed in JavaFX for managing products. It includes a menu bar with options like Home, Manage Product, and Account. Labels are used for titles and prompts, and buttons facilitate actions such as adding, updating, removing products, and navigating back. Input fields (TextFields and TextArea) allow users to enter product details like name, price, and description. The product list is displayed in a ListView, enabling users to select a product for updating or removal. TextFlow and Text components format and display product descriptions. Layout containers (GridPane, HBox, VBox, BorderPane) organize the visual structure. Alerts are utilized for providing information and handling errors. CSS styling enhances visual appeal, and event listeners are attached to buttons and the product ListView to manage user interactions effectively. Overall, the GUI offers a user-friendly experience for administrators or managers to handle product-related tasks seamlessly.

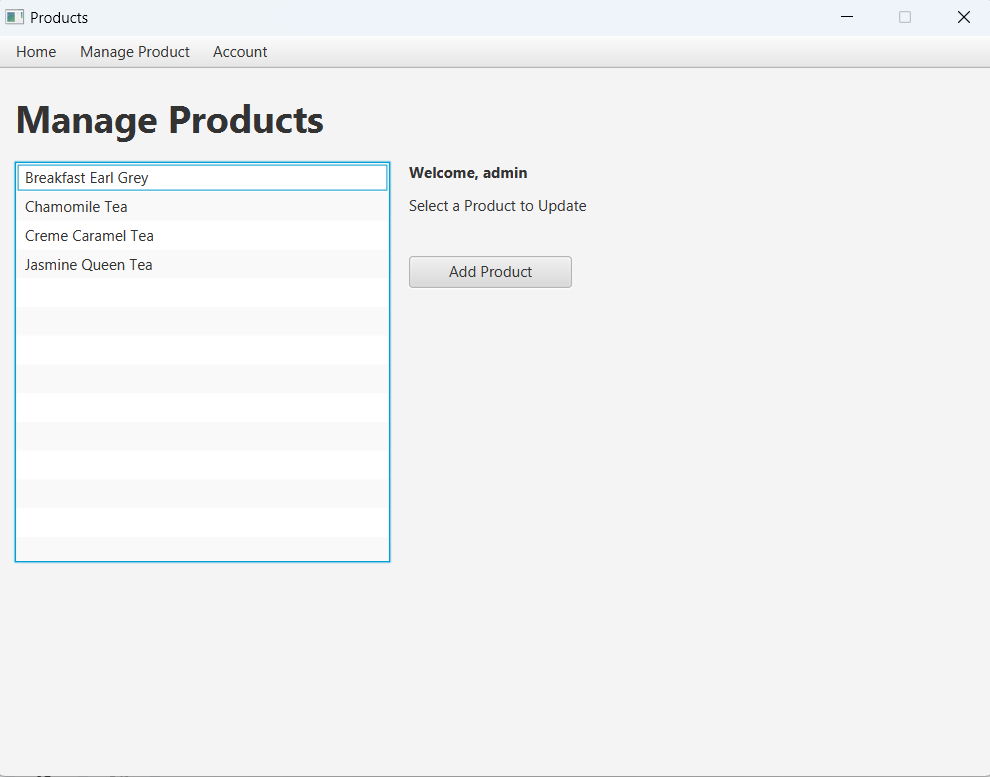
1. Event Handler



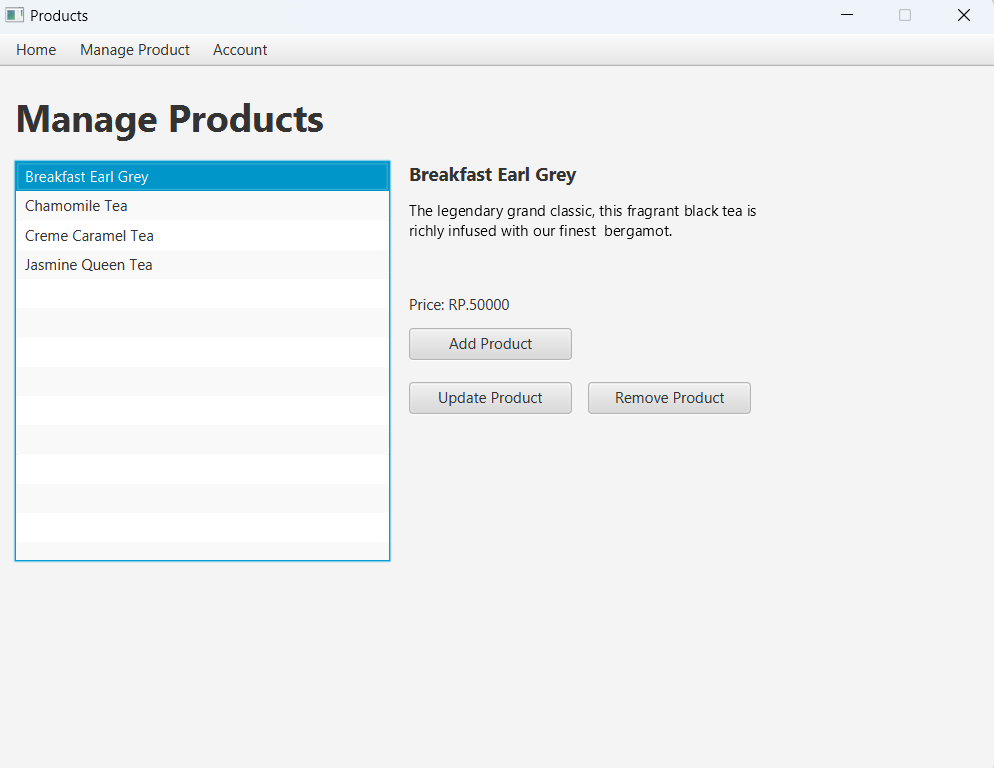


* 1. Menu Event Handling:
     1. The menu bar includes options like Home, Manage Product, and Account.
     2. Event listeners are attached to these menu items to trigger actions, such as navigating to the home page, managing products, or logging out.
  2. Product Selection Event Handling:
     1. The product ListView allows users to select a product from the list.
     2. When a product is selected, corresponding details are displayed, and options to add, update, or remove the product become available.
  3. Add Product Event Handling:
     1. Clicking the "Add Product" button reveals input fields for entering a new product's name, price, and description.
     2. Additional buttons allow users to confirm the addition or navigate back.
     3. Validations ensure that mandatory fields are not blank and that the product name is unique.
  4. Update Product Event Handling:
     1. Clicking the "Update Product" button displays input fields for modifying the product's price.
     2. Users can confirm the update or go back.
     3. Validations check for a positive price input.
  5. Remove Product Event Handling:
     1. Selecting the "Remove Product" option reveals a confirmation message.
     2. Users can confirm the removal or go back.
     3. Checks are in place to handle cases where the product is already in a customer's cart.
  6. Back Button Event Handling:
     1. The "Back" button allows users to navigate back to the previous state, canceling ongoing operations.
  7. General Error Handling:
     1. Alerts are used to communicate success or failure messages, guiding users and ensuring a smooth user experience.

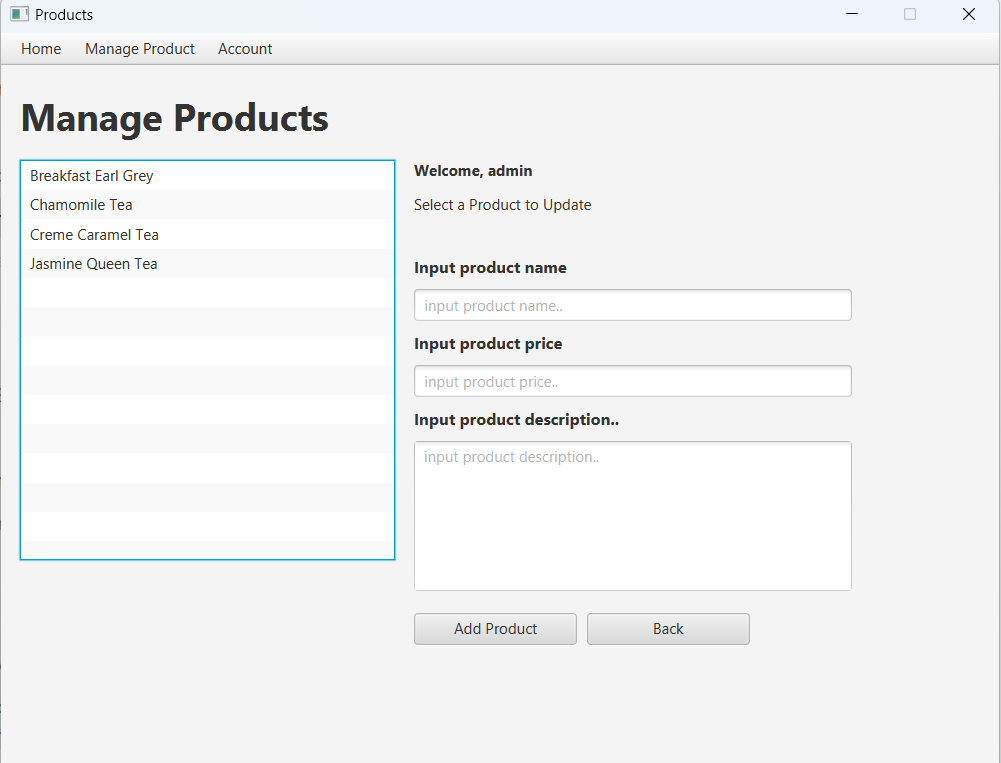
1. Execute (Run)
   1. Manage Product Form (no selected product)



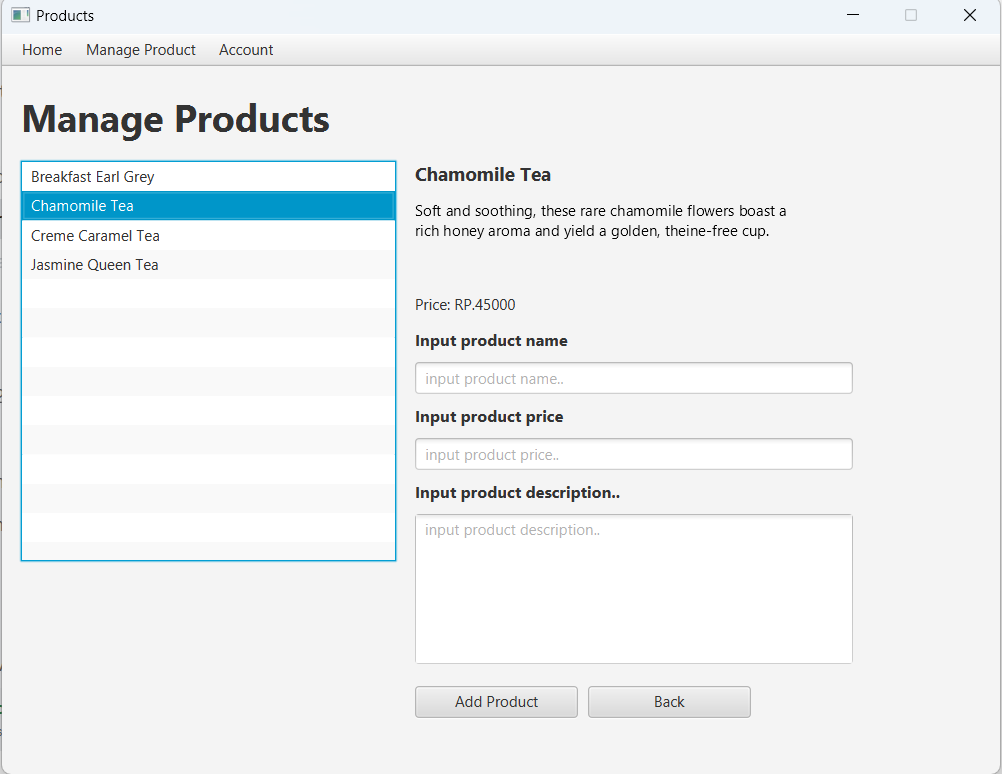
* 1. Manage Product Form (a product is selected)



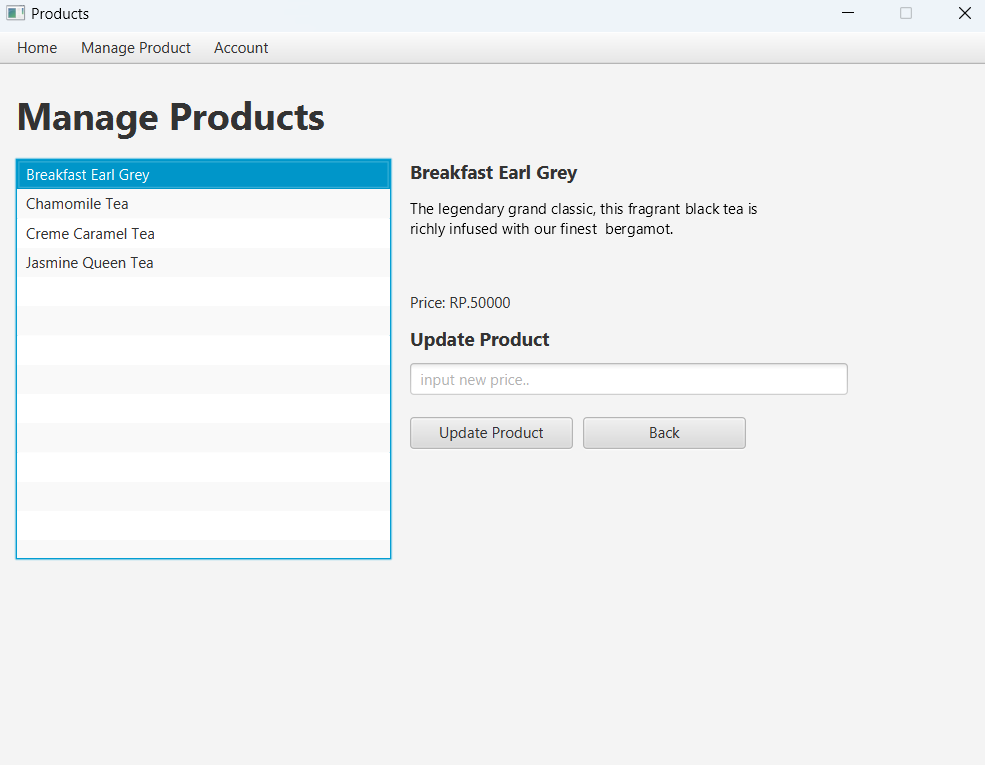
* 1. Manage Product Form (no product is selected and add product button clicked)



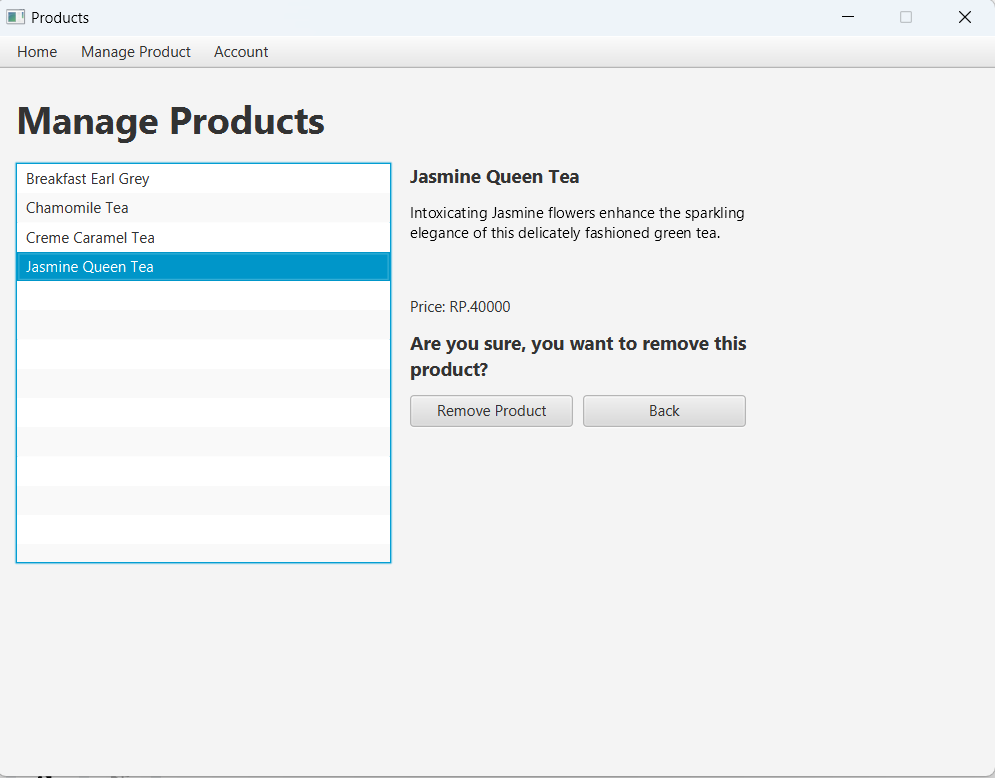
* 1. Manage Product Form (a product is selected and add product button clicked)



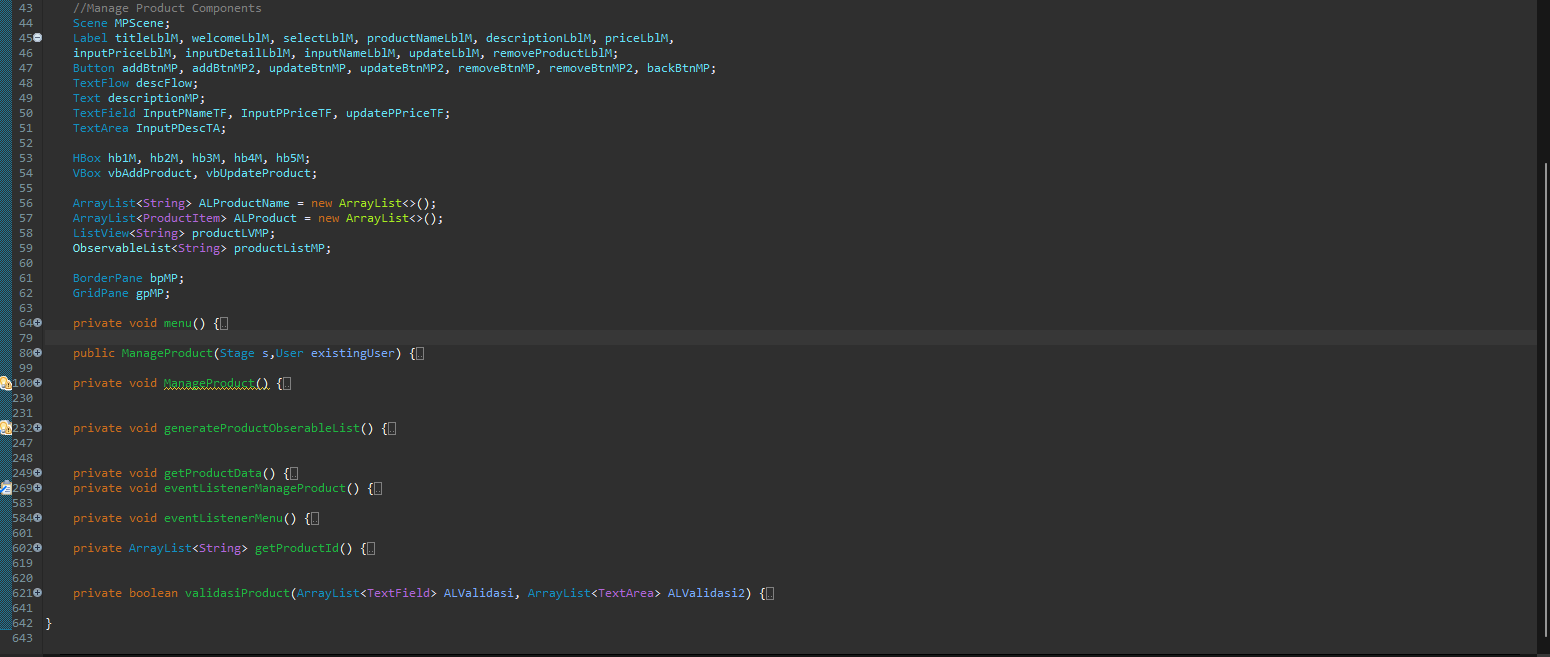
* 1. Manage Product Form (a product is selected and update product button is clicked)

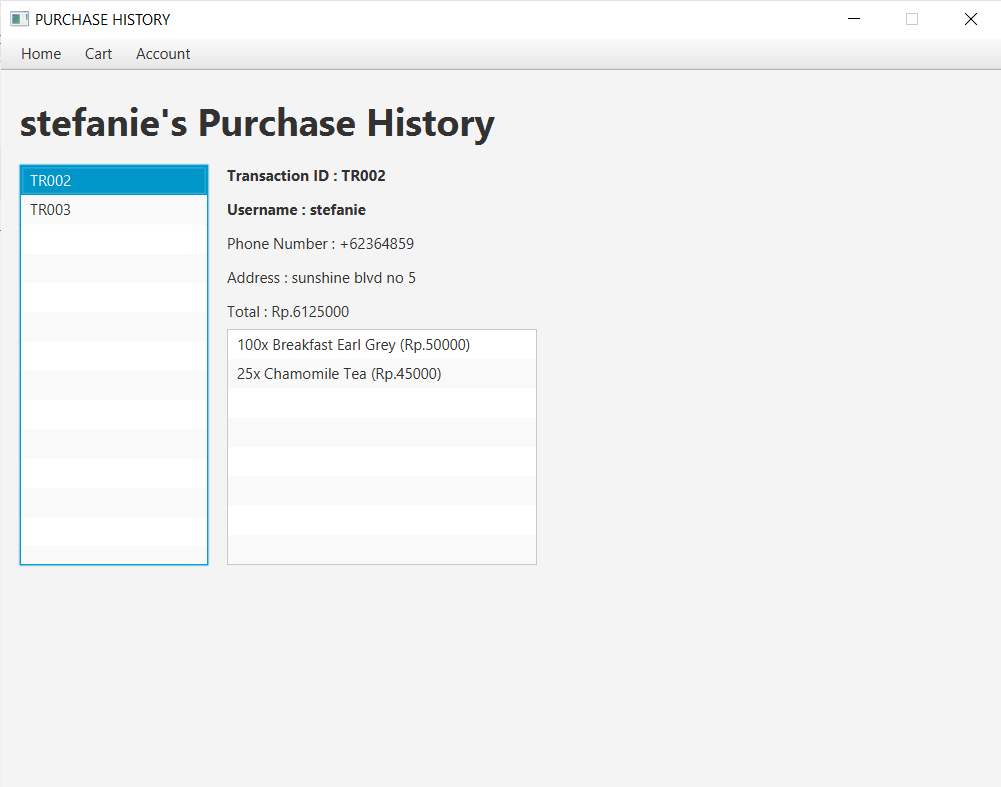


* 1. Manage Product Form (a product is selected and remove button is clicked)



* 1. Code



1. Transaction History Scene
   1. Component
      1. The Transaction Scene is a page that is only available to Customers. This scene is used to display all the transactions that had been done by the current user. All the transactions will be displayed on a ListView that Users can interact with. When the ListView is selected, it will show details of the selected transaction. When the user first enters purchase history, display an empty message. if the user has not made any transactions or purchases yet. If the user has made any transactions or purchases,
   2. Event Handler:
      1. Menu Event Handling:  
         Responds to menu items like "Home Page," "My Cart," and "Log Out." Navigates the user to different sections of the application based on their selection, such as going to the home page, handling products, or logging out.  
         Code:  
         
      2. TransactionID Selection Event Handling:  
         The product ListView allows users to select a product from the list. When one of the transactionID is selected, the page will show details of the transaction, such as; address, quantity, total price, etc.  
         Code:  
         
   3. Method:
      1. generateObservableIDPH  
         Generate the observable list of a transactionID by putting it into the “transactionIDListPH” arraylist so it can be accessed in ListView.
      2. generateObservableDetailPH  
         Generate the observable list of a transaction (individual item with their quantity and price) by putting it into the “transactionDetailListPH”arraylist so it can be accessed in ListView.
      3. getTotalPriceCart  
         Calculate the total price of a transaction.
      4. getTransactionData  
         Add the transaction datas from the database by putting it into the “ALPurchases” arraylist so it can be easily accessed later.
   4. Execute:
      1. Purchase History (no selected transaction):  
         
      2. Purchase History (selected a transaction):  
         

* **Reference**
* Introduction to Java Programming. 12ed. Liang,
* **Group Member**
* <2602066625> – <Ervan Fernando Wijaya>
* <2602063610> – <Dwi Kurniawan Aliputra>
* <2602054133> – <Georgius Kenny Gunawan>
* <2602061832> – <Yosua Samuelpin>