

## ASSESSMENT BRIEF



<b>Class Group:</b>	CS
<b>Assessor:</b>	Bernie Farrell
<b>Component Title and Code:</b>	6N0736 - Graphical User Interface Programming
<b>Assessment Technique:</b>	Skills Demonstration
<b>Weighting:</b>	30%
<b>Title:</b>	Assessment 1
<b>Issue Date:</b>	17 <sup>th</sup> November 2020
<b>Submission Date:</b>	17 <sup>th</sup> December 2020
<b>Learning Outcomes Assessed:</b>	1, 2, 4, 5, 6, 7, 9 and 10

### Guidelines:

Carry out the skills demonstration in accordance with the guidance provided in this document.

### Assessment Criteria:

Program documentation to include evidence or prototyping and rapid application development. (6 marks)

Application interface to include at least its look, ease of navigation, ease of use, simplicity, consistency, feedback, and a help subsystem. (8 marks)

Quality of application to include at least adherence to user specifications, correctness, user error trapping and feedback, robustness, and adherence to best practice. (12 marks)

Testing of application to include at least suitable test data, expected results and actual results. (4 marks)

<b>Learner Name:</b>	
I confirm that: <ol style="list-style-type: none"> <li>1. I have been provided with information about Cork ETB's assessment and appeals procedures and my responsibilities with regard to assessment</li> <li>2. The assessment work produced by me is all my own original work</li> </ol>	
Learner Signature: Milan Labus	
Date: 17/12/2020	

**Note to Learners:**

- Assessments will not be accepted without this coversheet.
- Plagiarism is the presentation of someone else's ideas, arguments, concepts or work as your own by failing to reference or acknowledge it properly. All such work must be acknowledged. Any learner, who presents another's work as their own, will be investigated in line with Cork ETB Assessment Malpractice procedures and may be awarded a zero grade.
- Learners should keep copies of all assessment submitted, where applicable.

## Vending Machine

A vending machine sells 4 different types of soft drinks. Each drink costs €1.20. The price of the drink is clearly displayed or “Out of stock” is displayed where a particular drink is not available. Users can purchase more than 1 drink at a time.

A user can sign up for an account which will allow them to top-up their vending machine account from their credit card details. They must log into this account to purchase a drink, as each time they buy a drink, their balance is reduced by the amount they spend. Account holders will receive a 5 cent reduction on every drink they purchase. If their balance is below €2.00, they will get a notification after they login.

When a user wants to buy a drink, that doesn't have an account set up, they can log in using a guest account. The vending machine will not dispense any change. They insert their money into the vending machine and select the drink. The user must enter the correct amount or more than the correct amount, then the drink that they selected is dispensed but NO change is given. If a user inserts more money than the price of the drink, a warning message must tell them that “This machine does not give change” and they can choose to still select the drink and forfeit the change, or they can cancel the order. If a customer isn't sure of what to do, they can access a *help* facility at the vending machine.

## Setting up an account

When signing up for an vending machine account, the user enters the following details:

- First name (letters, spaces, no numbers)
- Last name (letters, spaces, no numbers)
- Date of birth (those under 18 years of age cannot legally avail of the service as you must be at least 18 years old to get a credit card by yourself)
- Gender
- Email address
- Pin number (Must be 4 digits)
- Mobile number
- Address (up to 3 lines, but 1 line is sufficient)
- County (if Ireland) and
- Country (if not Ireland).

The user will be created with a username (email address) and password (pin number). The customer must accept the terms and conditions displayed.

Once those details are entered, the customer is invited to enter their:

- Credit card number (16-digit) number.
- Credit card expiry date and
- Credit card security code (a 3-digit number)
- A top-up amount that will be saved on their account (at least €10).

The user name is the email address, the password is the pin number and the credit amount must be saved on a file.

There must be a facility for further top-ups after login that will add to their credit.

## The Manager

The vending machine manager regularly goes to the machine to add more drinks and to remove any coins from the machine. He/She must have a login account. The manager can then add drinks to the machine (up to a maximum of 20 units per drink) and can remove money. The manager can also have a report displayed that indicates:

- The count of each drink sold.
- The amount of revenue generated. This should include the amount generated through sales and the amount generated through unreturned change.
- The amount of money that was removed from the machine.
- The number of users with an account and their total spend.

## Machine Maintenance

If a customer notices that a vending machine is damaged, not allowing entry of coins or drinks are “Out of stock”, they can report it via the Vending Machine screen. They enter a brief description of the fault or the drink that is “Out of stock”. The details are added to the maintenance log.

The manager routinely displays the maintenance log. As each fault is repaired, coin entry facility fixed or restocked with drink, the complaint description, date and time of maintenance are sent to the repaired log and removed from the maintenance log. So this means there are two lists or logs maintained.

## Data Stored

Program needs to remember user name, pin number and account credit for each account.  
Only need to store number of each type of different drink in the Vending Machine.

You are required to design, code and test an application for the Vending Machine using prototyping and rapid application development.

## Submit

### **1. The design of your project.**

This should include all the preliminary work for your project i.e. all the work that you prepared using prototyping and rapid application development, to produce the final product.

### **2. The screen layouts scanned into a word document.**

### **3. The source code copied into a word document.**

This should include the code for all the screens that you have used in your project and any additional code (e.g. module code).

### **4. How you tested your program.**

The testing that you carried out on each implemented prototype should be shown. This should include testing of the functionality of the program and of any other tests e.g. data validation, user/password check etc... that you carried out to ensure that your program is robust. This should be saved into a document.

### **5. The program files**

All files should be zipped to be uploaded to Moodle, include the entire JavaFX package/project and any additional files that you have used in the program.

**Marks will be allocated as shown on the cover page.**

## Login Screen Design

Login Class Design

Login Menu

X

Enter your Username and Pin

username:

Pin:

Confirm

Manager Login

Guest

Create Account

help

On the Login screen I will have 3 Labels one at the top and two above the TextFields I will have for the user to enter their Username and Pin. When they have typed in their Username or Pin they will press either my Confirm or Manager Login Buttons. The Confirm button when press will read using a reader file to find a username and pin that matches what was entered

in the textboxes. If it matches an alert will show saying login was successful and that account is below 2 euro if it is. Also I will have a help button which will pop up an alert explaining how to use the screen

### The Create Account Menu Design

Hand-drawn sketch of a 'Create Account' menu design on lined paper. The form is titled 'Create Account' in the top left corner. It contains the following fields and buttons:

- First Name: [Text Box] [Validate]
- Last Name: [Text Box] [Validate]
- Date of Birth: [Text Box] [Validate]
- Gender: [Dropdown Menu] [Validate]
- Email: [Text Box] [Validate]
- Pin Number: [Text Box] [Validate]
- Mobile Number: [Text Box] [Validate]
- Address Line 1: [Text Box] [Validate]
- Address Line 2: [Text Box] [Validate]
- Address Line 3: [Text Box] [Validate]
- Country: [Dropdown Menu] [Validate]
- Country: [Dropdown Menu] [Validate]
- Submit: [Text Box]
- Cancel: [Text Box]
- Help: [Text Box]

On the Create Account Screen I plan to have Labels followed by a Textfield followed by a validate button for each item and arrange them into a Hbox. Then at the end all arranged into a VBox. I plan to have a submit button which when pressed will check if all of the fields have been filled in then writes to the file using my writer file.

The way I will validate what is entered into the textfield is by having methods that will ensure:

- That first name and last name are strings
- That date of birth is a valid Date and will pop up an alert informing the user that they have to be over 18
- Gender will be a drop down Combo box with male and female options
- Email will accept only strings into the Textfield
- Pin Number will only accept an Integer
- Mobile number will only accept numbers between 0-9
- Address line 1 will have to be filled with a string but address lines 2 and 3 are optional
- Country will be a drop down combo box with 6 countries to choose from
- County will have a drop down combo box with 5 counties and an other option if the user is not in Ireland

In the corner I will have a cancel button to exit the menu and a help button which will give instructions to the user.

The final Submit button will check if all the fields have been filled in and then will write them to the txt file using my writer class



## Credit Card Details Screen Design

Hand-drawn UI design for a Credit Card Details screen. The screen has a title bar "Credit Card Details" with a close button "X". Below the title bar is a heading "Credit Card Details". There are three input fields, each with a "Validate" button: "Enter your 16 Digit Credit Card Number", "Enter the date of Expiry", and "Enter 3 Digit Security Code:". Below these is a "Top Up" button. At the bottom are two buttons: "Drink Menu" and "Help".

I will have A label as a heading and then 3 HBoxes with a Label, Textfield and validated button. I will arrange these hboxes into a vbox including the top up button which when pressed will go to the top up screen. I will have a button to take the user to the drinks menu and a help button. I will put all of this into a borderPane with a top centre and bottom.

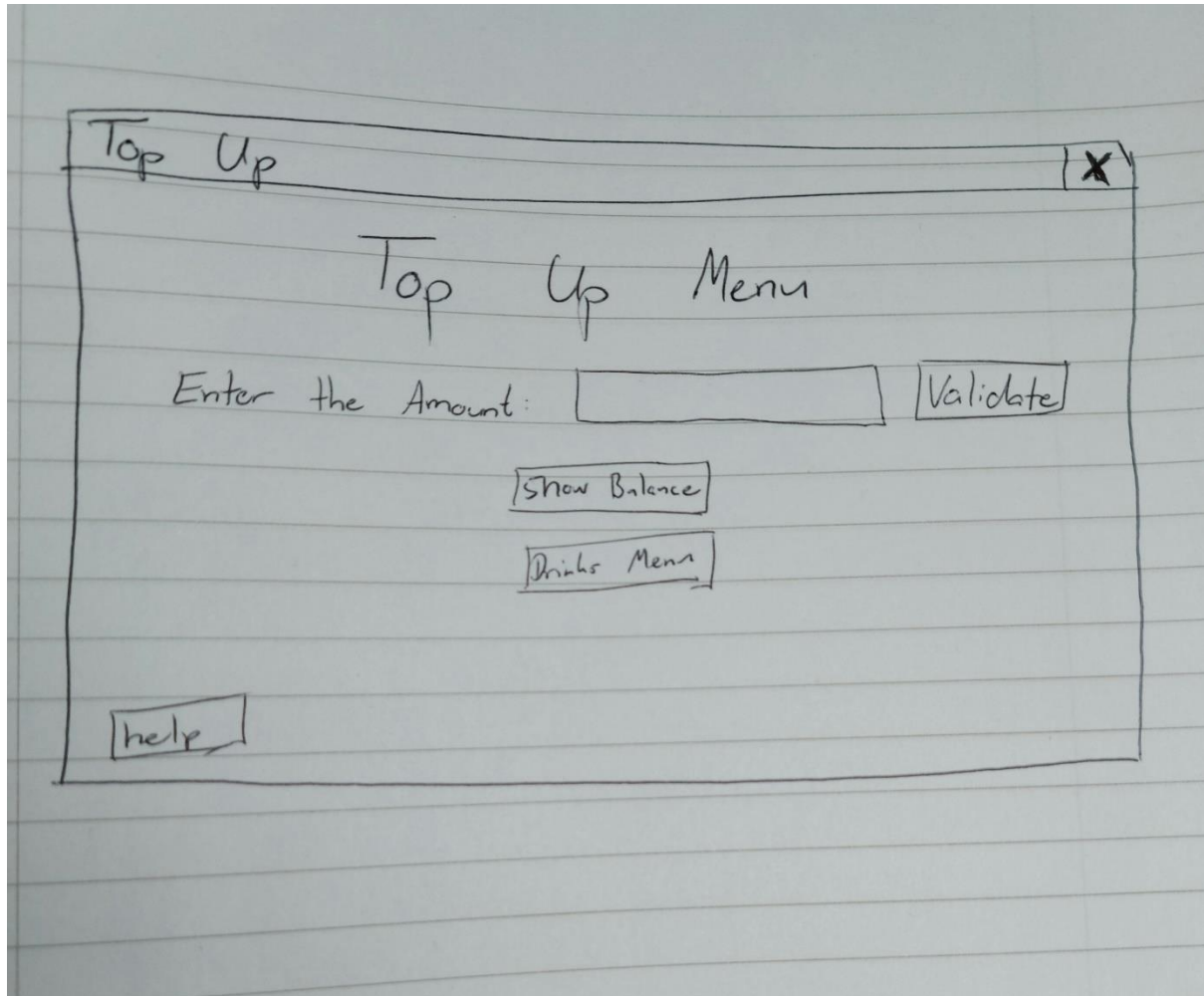
The way the 3 textfields will be validated will be:

- the Enter your 16 digit card number will accept only a Long
- The date of entry will accept only a date
- The 3 digit security code will accept only an Int

When the validate has been pressed if the correct value has been entered a validation successful alert will show and if its wrong a wrong input alert will show

The help button will show an alert explaining how to use the menu.

## Top Up Screen Design and Testing



I plan to have a label at the top, then under it a HBox that contains a label, TextField to enter the top up amount followed by a validate button. Then a show balance button which will display the amount of money the user has in their account and a button called Drinks menu that will take the user to the drinks menu and also a help button

The user will enter the amount they wish to top up by and then the validate button will check if a double (by use of a method) has been entered then that amount will be added to the account balance.

## Accounts Drinks Menu Design

A hand-drawn menu design on lined paper. The menu is titled "Drinks Menu" in a box with a close button "X" in the top right corner. Below the title, it says "Please Select Which Drinks you would like to please". There are four rows of drink options, each with the drink name, price "€1.2", a text input field, and an "Add" button. The drinks listed are Coke, Fanta, Sprite, and Pepsi. Below these options are four buttons: "Complete Order", "Show Balance", "Top Up", and "Exit". In the bottom right corner, there are two buttons: "Help" and "Repeat".

Drinks Menu X

Please Select Which Drinks you would like to please

Coke: Price €1.2  Add

Fanta: Price €1.2  Add

Sprite: Price €1.2  Add

Pepsi: Price €1.2  Add

Complete Order

Show Balance

Top Up

Exit

Help Repeat

I plan to have a heading label, followed by 4 hboxes that will contain a label, textfields and add button for each drink, the user will enter the quantity of each drinks entered in the textfield beside that drinks name then press add to add it to the total. Once they have entered all the drinks they want and have pressed add they press complete order. What will happen then is the price of each other order of drinks is made by multiplying the total of drinks by the price. This is done for each drink and at the end the total is found by add together all of the drink total prices. For the discount I will multiply the amount of drinks buy the discount then deduct that number from the final price for example 4 total drinks will mean 20cents gets deducted from the final cost. The show Balance button will display the account balance, the top up button will take you to the top up Screen and the exit button exits the enter window and takes you back to the login screen.

How I plan to do the validation and calculations:

First the add buttons will make sure there is more coke in stock than the person is ordering if there is less an out of stock alert will appear if there is enough then the drink stock will get deducted by the number of coke ordered, the discount will be calculated and applied as stated above, the price with the discount applied will be calculated by the method stated above and a message saying how many of that particular drink you have bought will be output to the console. At the end all of these drink total prices are simply added together and to get the total cost then an if statement will check if there is enough money in the account and the total cost will get added to the machine revenue and the account balance will get deducted by this total cost.

There will also be a report button that will take the Customer to window where they can write a complaint and a help button that will display a alert explaining everything in further detail.

## Confirm Purchase

A hand-drawn UI mockup for a 'Payment' dialog box. The dialog has a title bar with the word 'Payment' on the left and a close button 'X' on the right. The main content area contains the text 'your total is: TotalCost' in blue ink. Below this is the question 'Do you wish to complete purchase?'. At the bottom, there are three buttons: 'Yes', 'No', and 'Help', each enclosed in a rectangular box.

Payment X

your total is: TotalCost

Do you wish to complete purchase?

Yes

No

Help

I plan to display the total cost at the top and then have as yes and no button, This display method will be a Boolean because I will have a Boolean variable called answer and if the yes button is pressed then the answer returns as true and no is pressed then it returns as false. Back in the accounts drinks menu a true result will complete the order and a false result will exit the stage and cancel the order.

## Complaint Menu for Customer Design

The sketch shows a rectangular window titled 'Report' with a close button 'X' in the top right corner. Inside the window, the text 'Report Menu' is centered. Below this, there is a label 'Type your Complaint Here:' followed by a rectangular text input field. Underneath the input field is a 'submit' button. At the bottom of the window, there are two buttons: 'exit' on the left and 'help' on the right.

This window shows when the customer presses the report button on the accountdrinksMenu and here I will have a textfield where the customer can type in a complaint and press submit.



I will have an exit button and help button. I have make the submit button check if a string has been entered and will show an error if its not.

## Guest Drinks Menu Design

Drinks Order X

Guest Menu

Please Enter Money:  Verify

Coke: Price €1.20  Add

fanta: Price €1.20  Add

Sprite: Price €1.20  Add

Pepsi: Price €1.20  Add

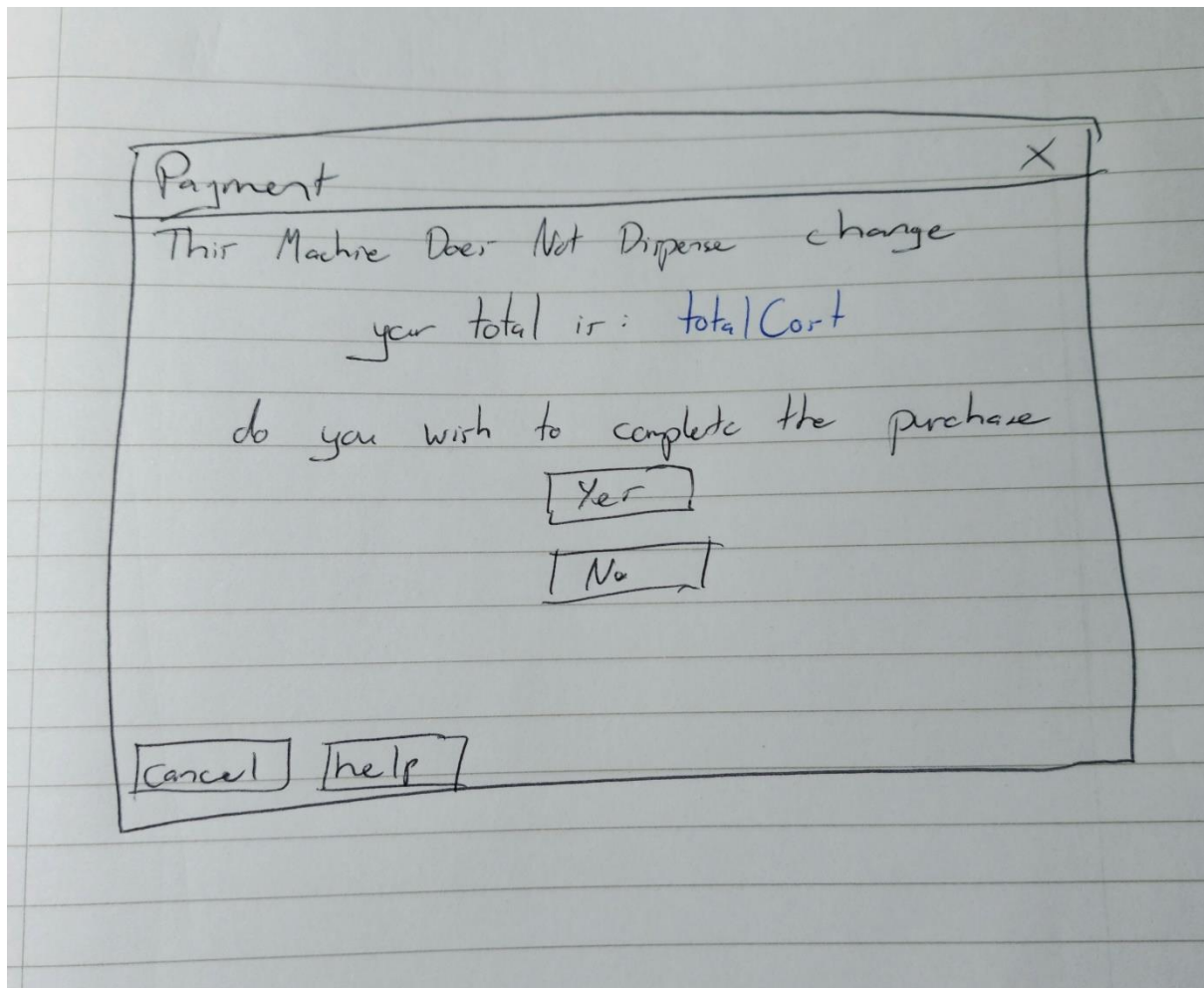
Complete Order

Report

Cancel help

In the guests drinks menu I plan to have a very similar layout to the account drinks menu with a similar process for finding the total price with the lack of a discount calculation. Also instead of checking if there is enough money this window will check if more money was entered than the drinks cost. There will also display an alert if less than 1.20 was entered saying its not enough to buy a drink and an alert that will say there is no change given by the machine if more than 1.20. the rest is very similar to the account drinks menu stock is updated and the revenue is update after a purchase as well as the change left by the customer is taken into account.

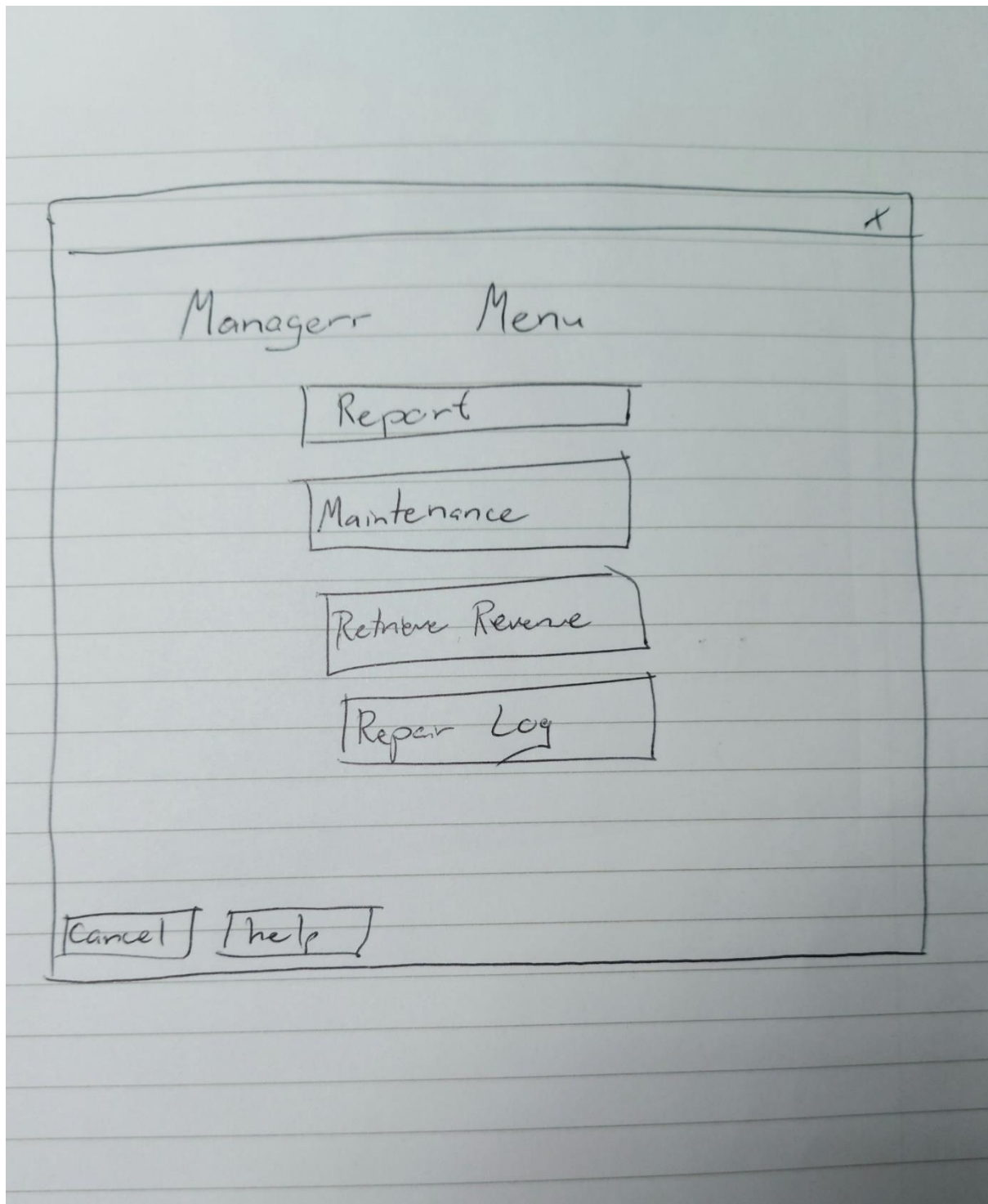
### No change and payment window Design



This window shows when you press complete order on the guest drinks menu and works the same way as the complete order class by returning a value true or false based upon the yes and no buttons and also displays the no change warning again.



## Mangers Menu Design



In the managers menu I will have Main 4 buttons :

-Report: Will show a report to the managers to include revenue, drinks, change left , customer complaints etc..

-Maintenance: Which will take the manager to a window where they can add drinks to the machine

-Retrieve Revenue: Which will take the manager to window where they can take money out of the machine

-repair Log: Will display a lot of how many of each drink the manager added to the machine and amount of revenue removed

Also there will be a back and help button.

The Design of these 4 windows will look as follows:

The image shows four hand-drawn wireframes for a software application, each with a title bar and a close button (X).

- Maintenance**: Contains the text "Please Enter Drinks to Add:". Below this are four rows, each with a drink name ("Coke", "Fanta", "Sprite", "Pepsi") followed by "in stock:" and an input field. To the right of each input field is an "Add" button. At the bottom left is a "Submit" button. At the bottom are three buttons: "Cancel", "help", and "Complaint".
- Report Menu**: Contains the text "Each Drink Sold:". Below this are five rows, each with a drink name ("Coke", "Fanta", "Sprite", "Pepsi", "Revenue") followed by a colon and an input field. Below these are three more rows with labels "Change left:", "Revenue Removed:", and "Number of users:", each followed by an input field. At the bottom are two buttons: "back" and "help".
- Repair Log**: Contains the text "Repair Log". Below this are four rows, each with a drink name ("Coke", "Fanta", "Sprite", "Pepsi") followed by "Added:" and an input field. At the bottom are two buttons: "back" and "help".
- Retrieve Revenue**: Contains the text "Revenue Removal". Below this is the text "Enter Amount you wish to remove" followed by an input field and a "remove" button. Below that is a button labeled "Show Balance". At the bottom are two buttons: "back" and "help".

The 4 windows Design Explained:

-Report: This window will contain label with all of the information in the order as shown in the diagram above and will have 2 buttons a back button and help button

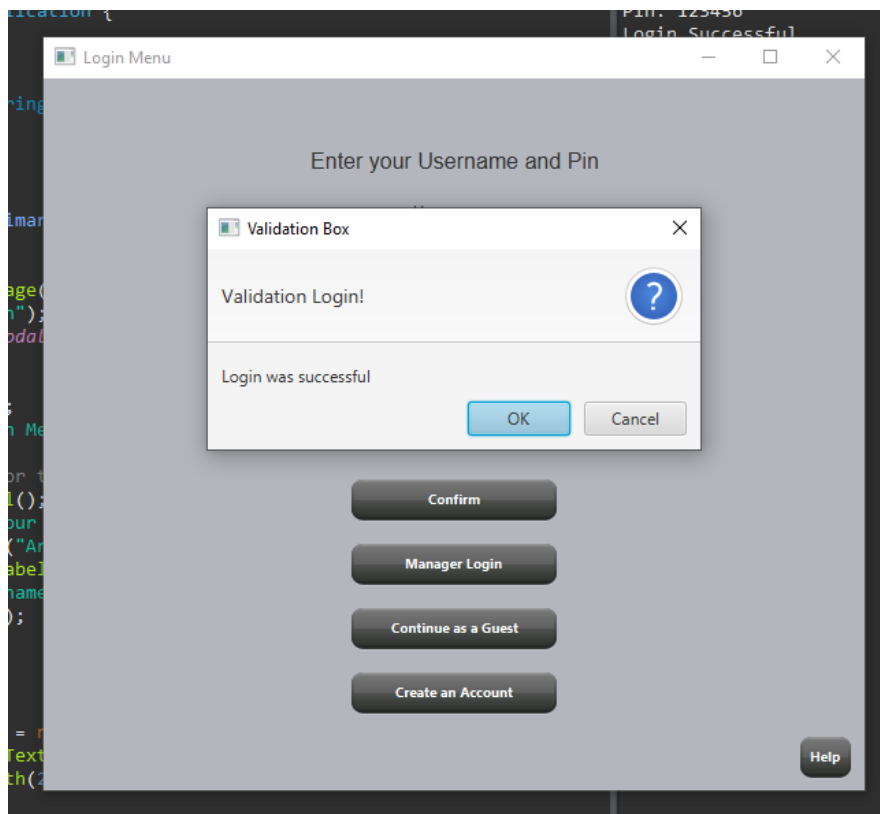
-Maintenance: will display how much of each drink is in stock and will allowed the manager to type in how many he/she wants to add, an error alert will show if they want to add enough drinks to put the drinks in the machine above 20 because only 20 can fit in the machine. There will be a complain button where a window will pop up showing any complaints the customers have written.

-Retrieve Revenue: This window will allow the Manager to type in how much money they want to take out and then press remove. If they enter a number greater than the revenue in the machine an error message will display. To avoid this I have added a show balance button to show how much revenue is In the machine.

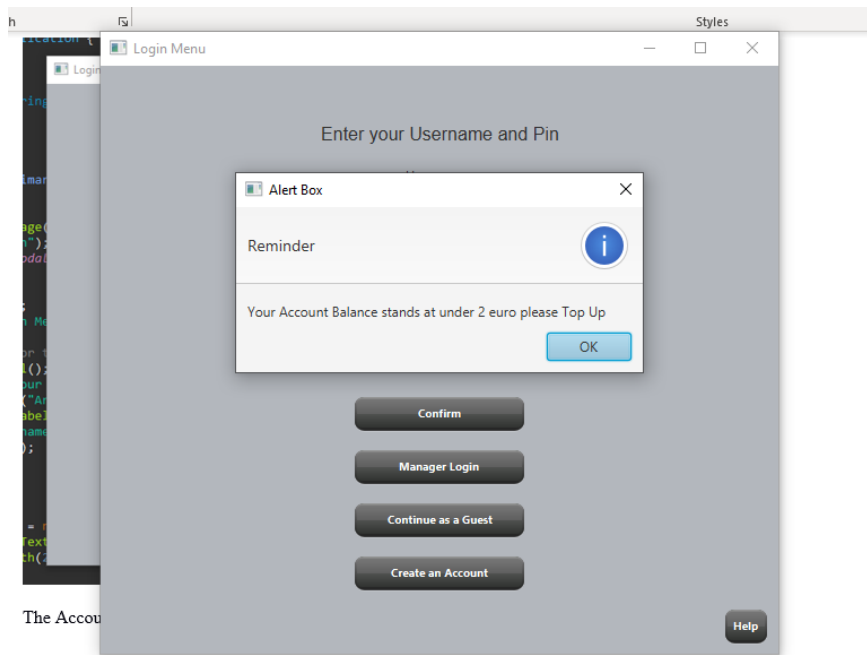
-Repair log: Will just show How many drinks the manager added and how much money the manager took out.

# Testing

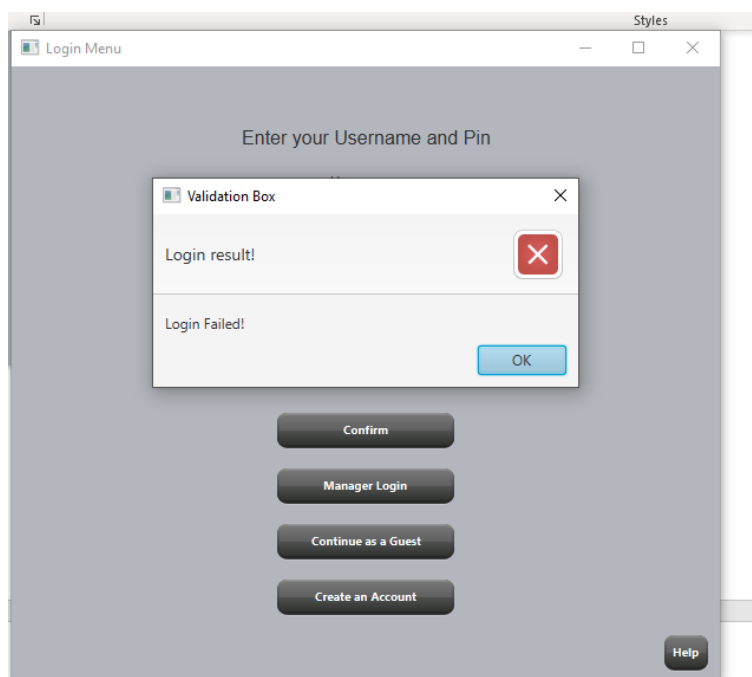
Correct login for customer:



The Account Balance being Below 2 euro if it is

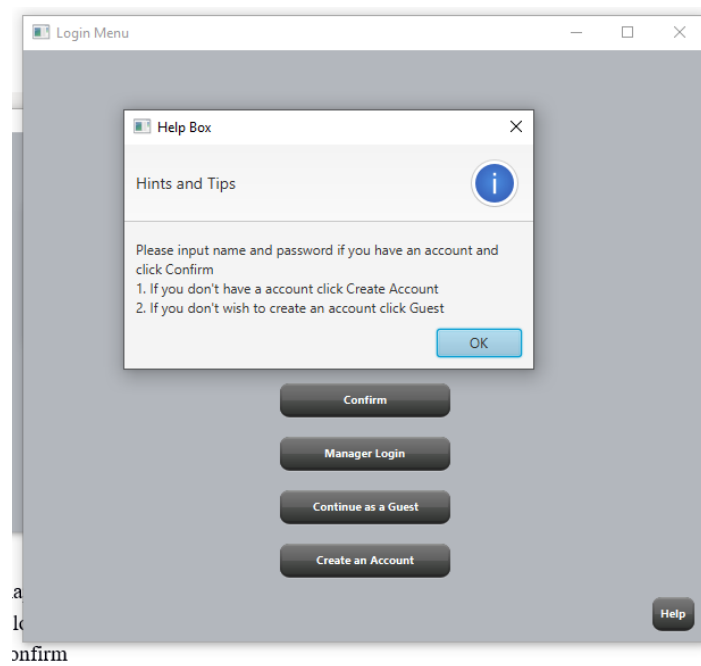


The same will happen when a manager logs in except no below 2 euro alert will be shown only successful login. Now I will test what happens when I enter an incorrect username and pin and press Confirm

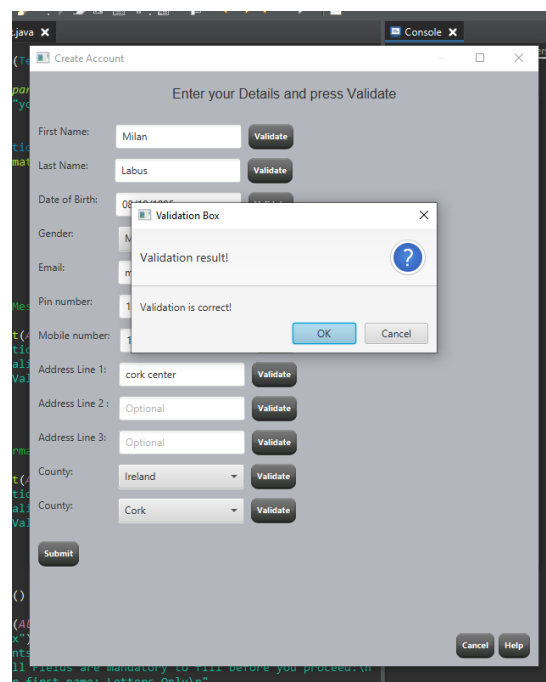


As we can see an Login failed alert shows up for an incorrect login. If a user does not have an account, they have the options between a create Account Button which when pressed takes the user to the Create Account Window and a guest account button which will take the user to a Drinks menu for guests.

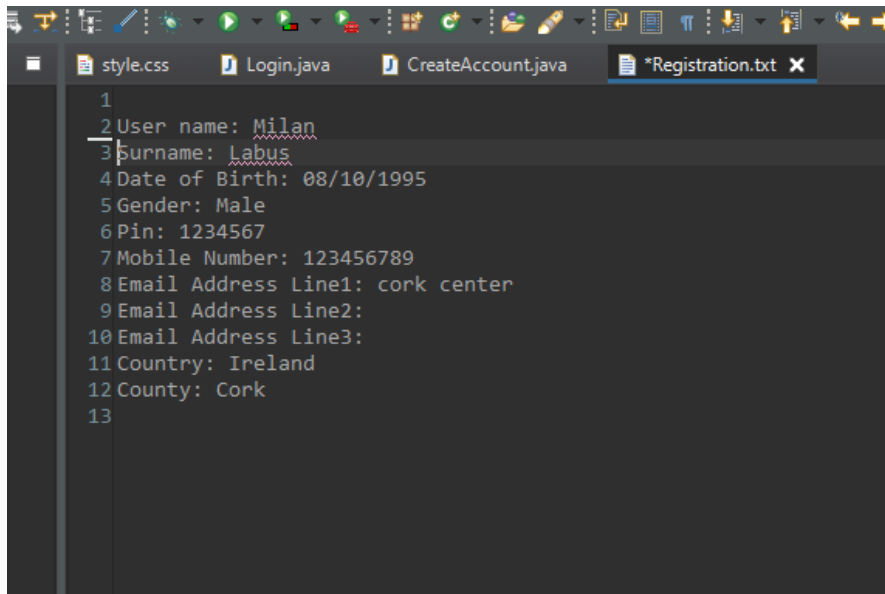
The help Button when Pressed is Functional



Here if have completed all of the fields in create account and pressed submit to test the functionality of the window

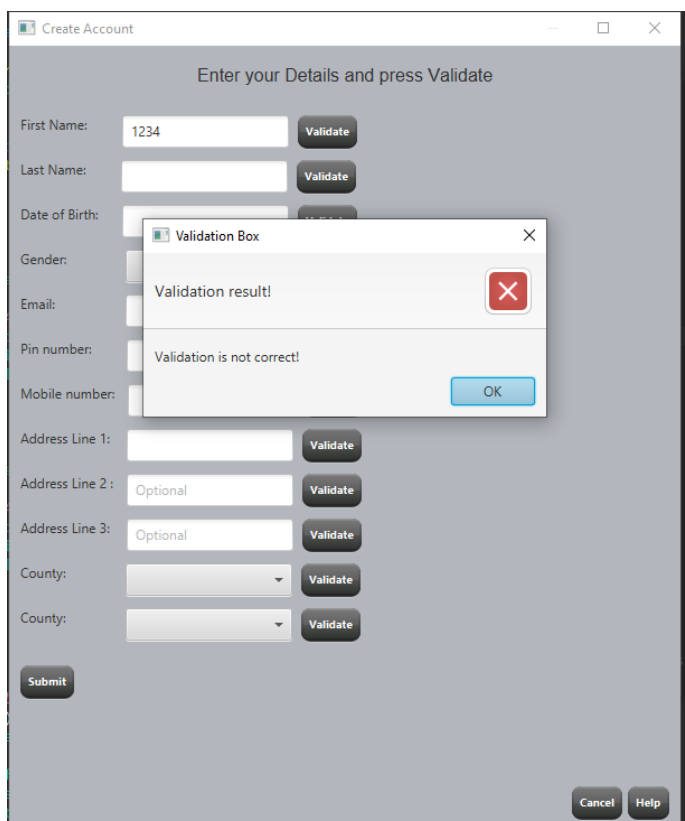


Everything I entered was successfully written to the Registration txt file. The create Account class is Fully functioning.



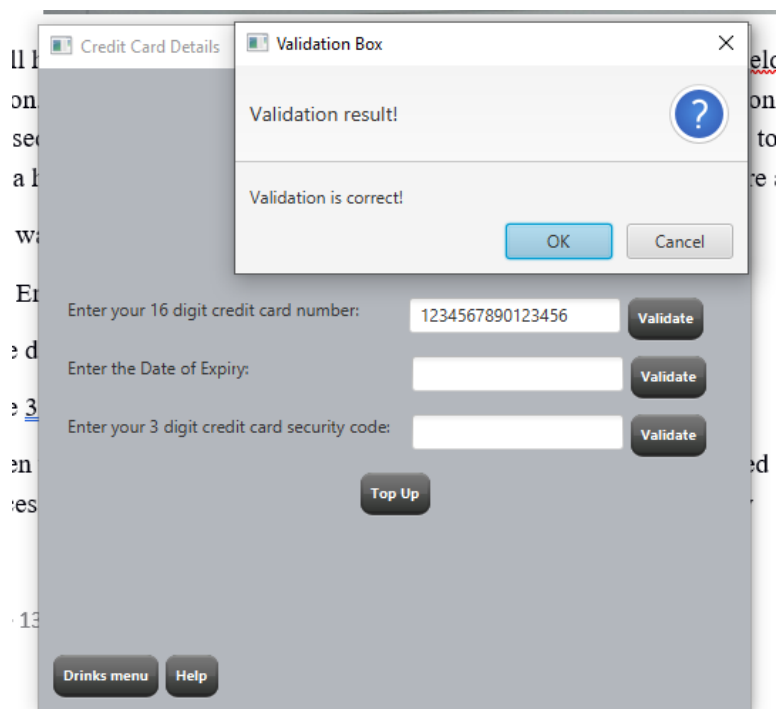
```
1
2 User name: Milan
3 Surname: Labus
4 Date of Birth: 08/10/1995
5 Gender: Male
6 Pin: 1234567
7 Mobile Number: 123456789
8 Email Address Line1: cork center
9 Email Address Line2:
10 Email Address Line3:
11 Country: Ireland
12 County: Cork
13
```

Here I entered a wrong input (putting a string in the first name textfield) to test the error trapping and a warning alert successfully popped up

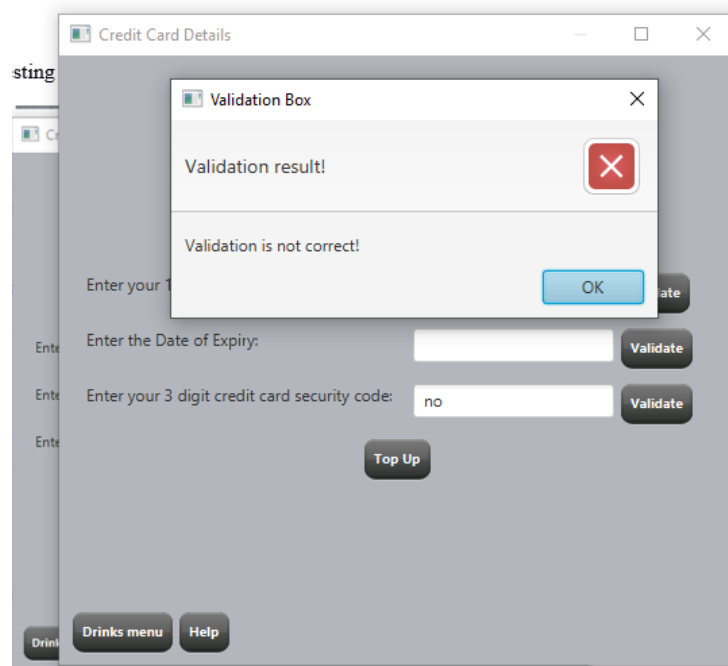




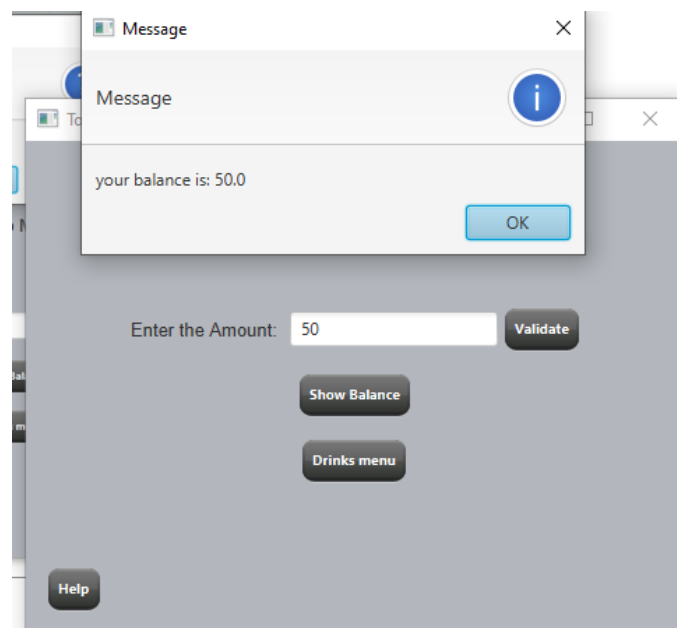
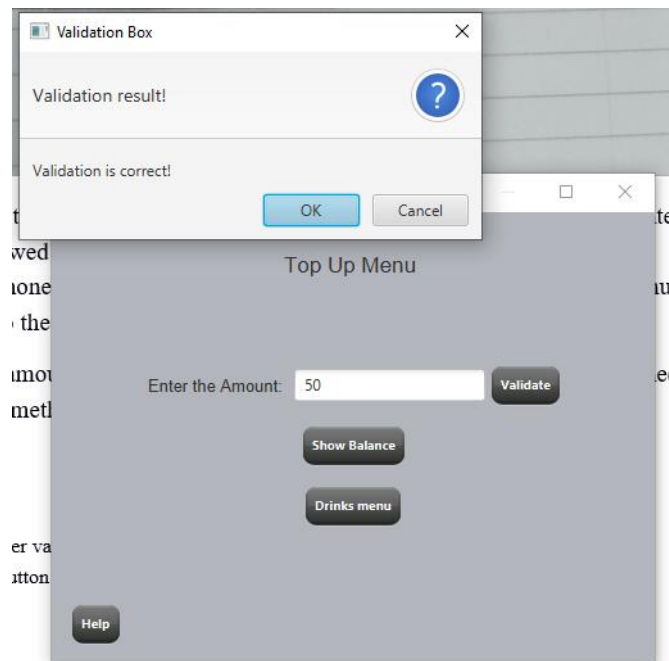
Testing the validate button when a correct information is entered



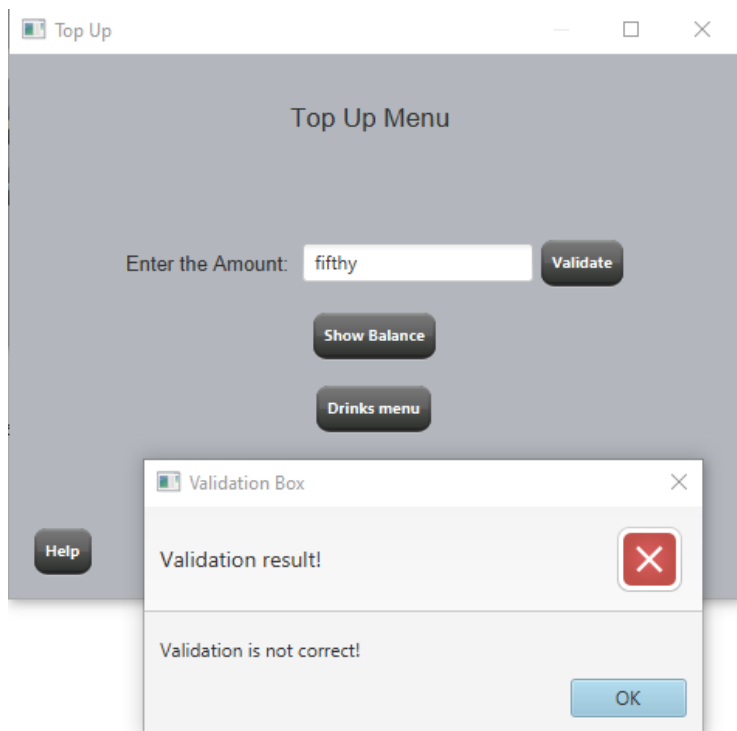
And when the wrong information is entered



In the Top Up menu To test it I will enter a number value and see if the account balance has been increased by that amount by pressing the show balance button after I have validated.

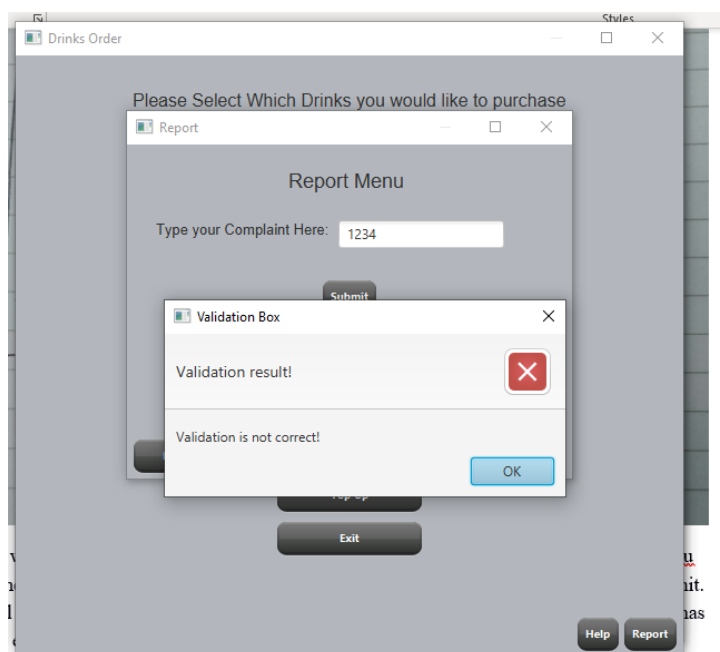


The test was successful this Screen works now I will test what happens when a string is entered instead of a number



The screen test has been completed It works perfectly

In the customer complaint window Now I will test by typing a number instead of a string



I will test by typing a number instead of a string

An error message shows up successfully.

Here in confirm purchase window I have pressed yes to test if it returns true:

```
Console X
Login (!) [Java Application] C:\Users\milan\.p2\pool\plugins\org.eclipse.justj.openjdk.hot
User name: Milan
Pin: 123456
Login Successful
You logged in with username: Milan
And with password: 123456
Your new Balance is: €50.0

Coke bought: 2

Fanta bought: 2

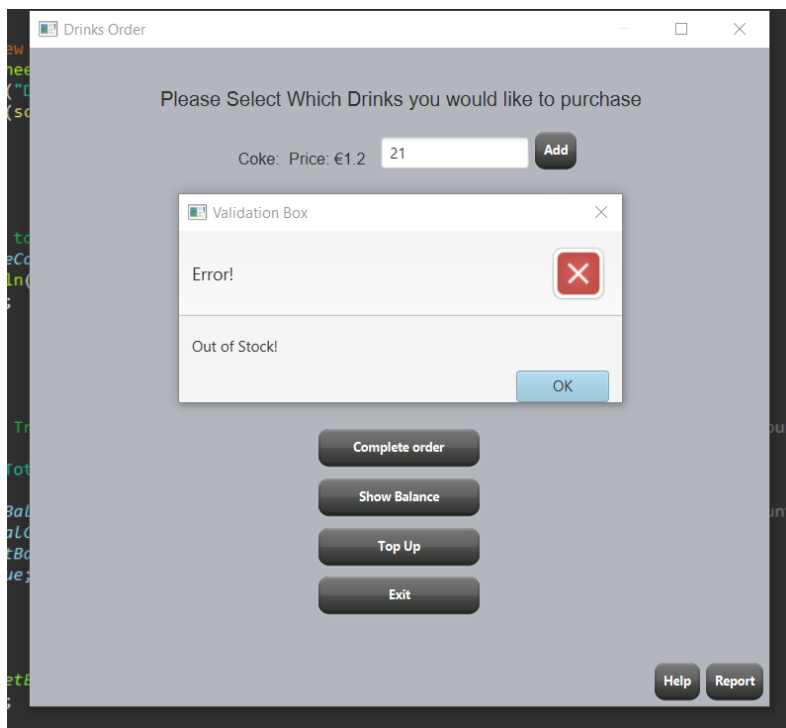
Sprite bought: 2

Pepsi bought: 2

Total Cost: 9.2
```

Yes it does it gives the receipt after clicking yes

In this first test of accounts drinks menu I will try to order more coke than is in stock



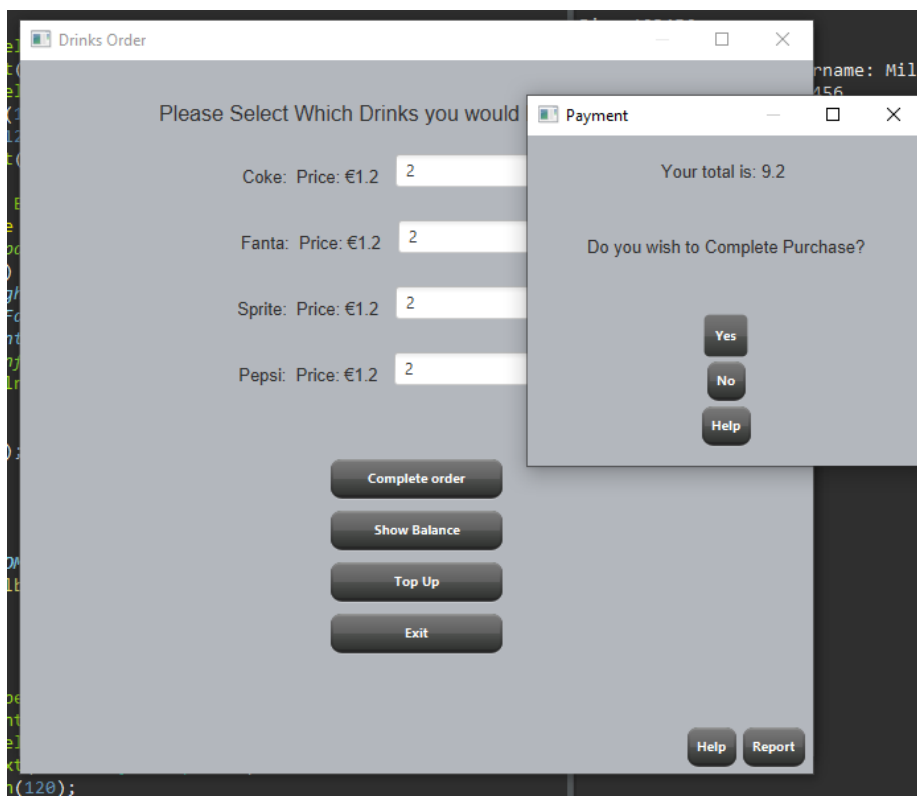
An error message has successfully appeared

For this next test I will add 50 euro into the account and order two of each drink

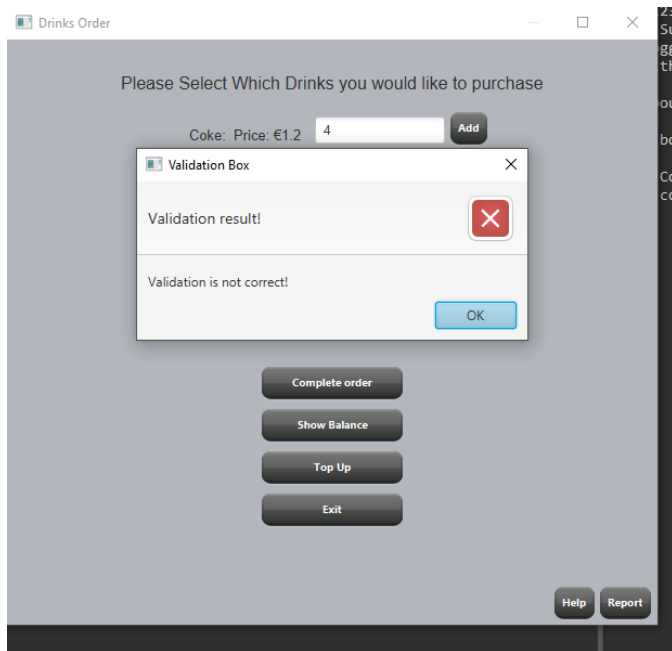
Expected result:

$$\begin{array}{l}
 2 \text{ Coke} * \text{Price (1.20)} = 2.40 \\
 \text{Discount (.05)} * 2 \text{ Coke} = 0.10 \\
 2.40 - 0.10 = 2.30 \text{ (Coke total price)} \\
 \\
 2 \text{ Fanta} * \text{Price (1.20)} = 2.40 \\
 \text{Discount (.05)} * 2 \text{ Fanta} = 0.10 \\
 2.40 - 0.10 = 2.30 \text{ (Fanta total price)} \\
 \\
 2 \text{ Pepsi} * \text{Price (1.20)} = 2.40 \\
 \text{Discount (.05)} * 2 \text{ Pepsi} = 0.10 \\
 2.40 - 0.10 = 2.30 \text{ (Pepsi total price)} \\
 \\
 2 \text{ Sprite} * \text{Price (1.20)} = 2.40 \\
 \text{Discount (.05)} * 2 \text{ Sprite} = 0.10 \\
 2.40 - 0.10 = 2.30 \text{ (Sprite total price)} \\
 \\
 2.30 * 4 = 9.2 \\
 \\
 \underline{\text{Expected total} = 9.2}
 \end{array}$$

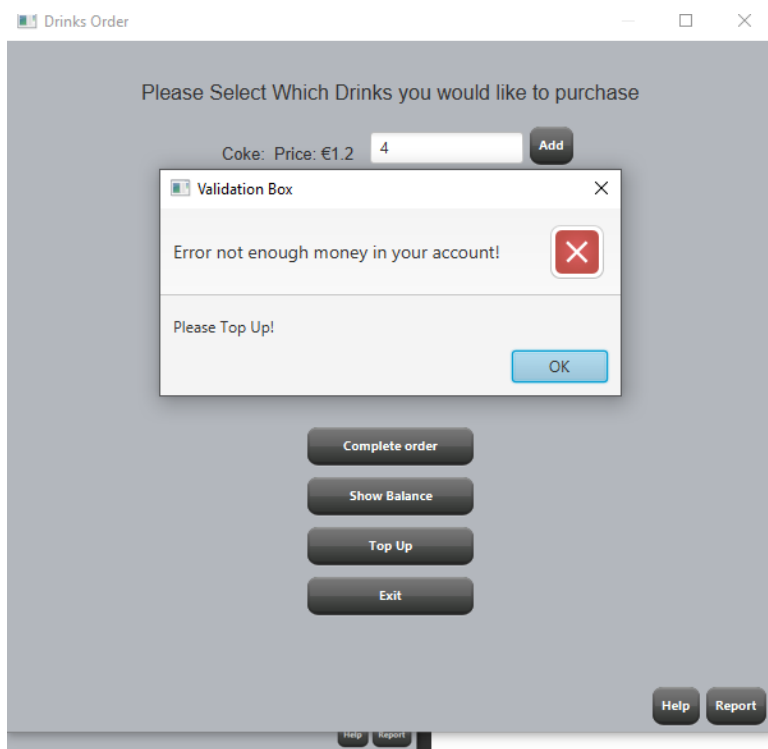
Actual result:



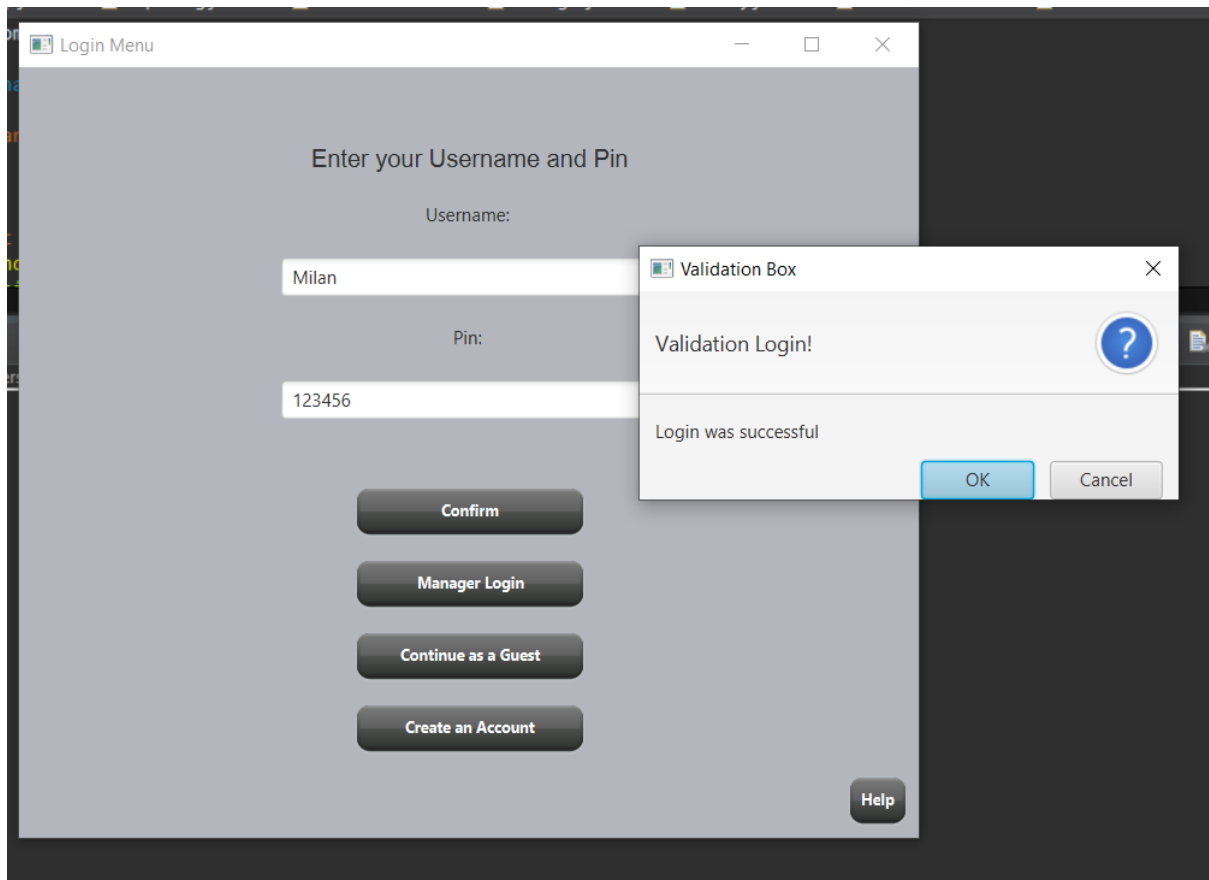
Now I will have my account balance as zero and try to order a drink

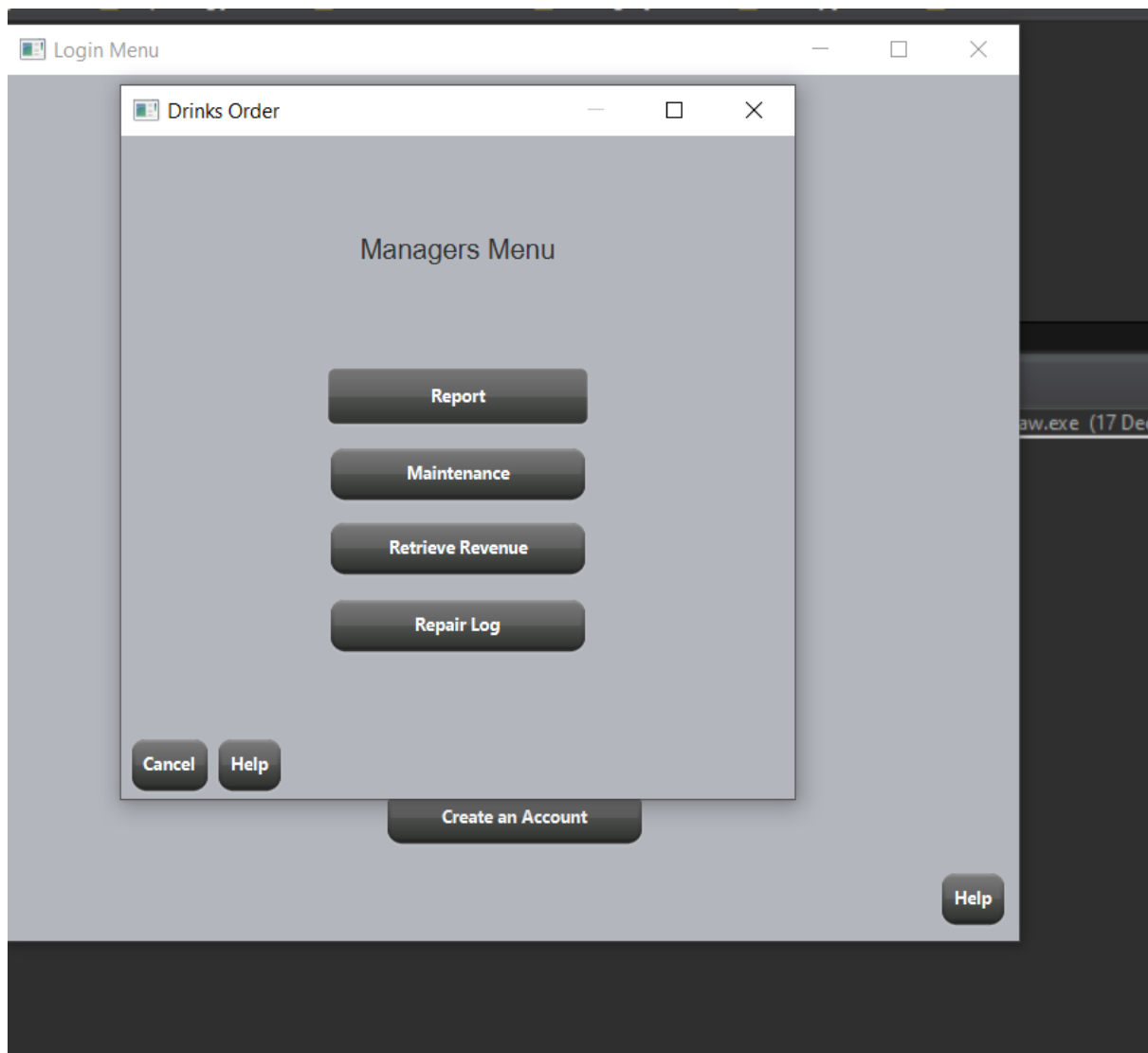


Not successful



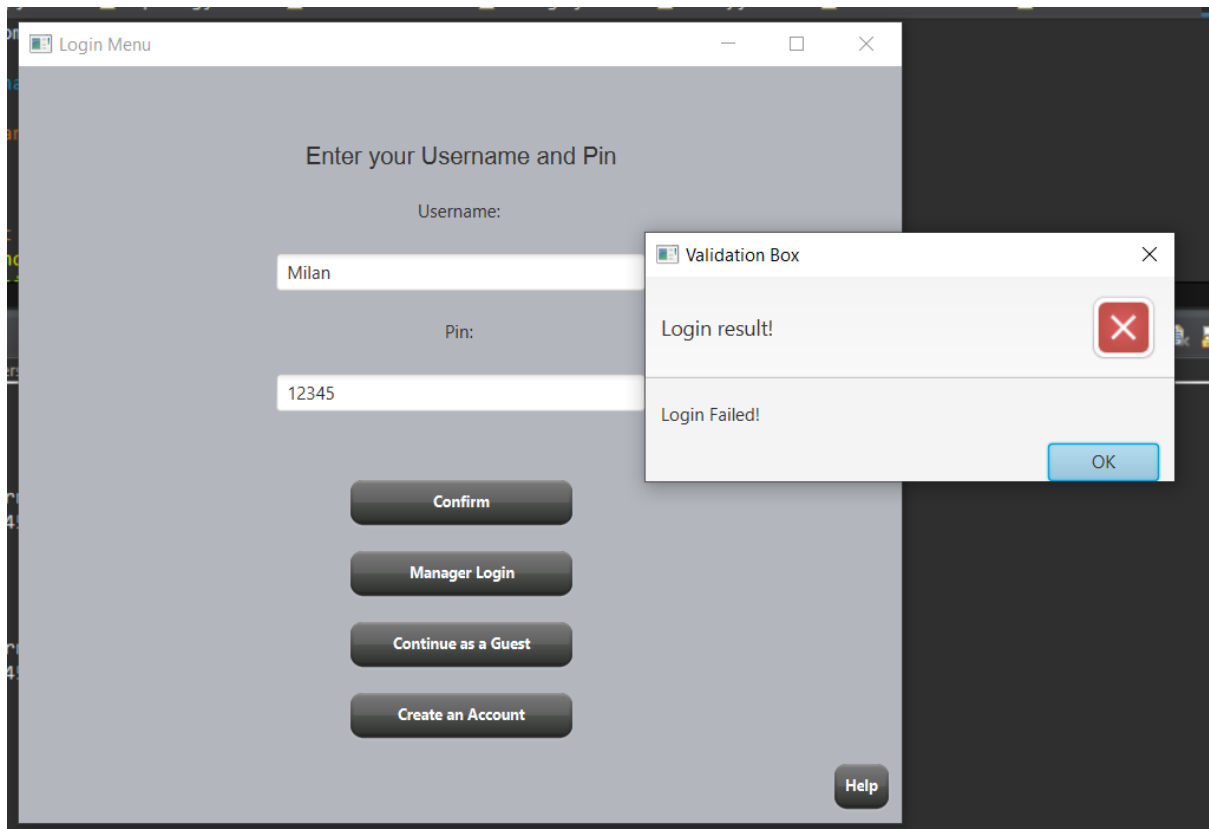
Now I will show correct and incorrect logins for Manager



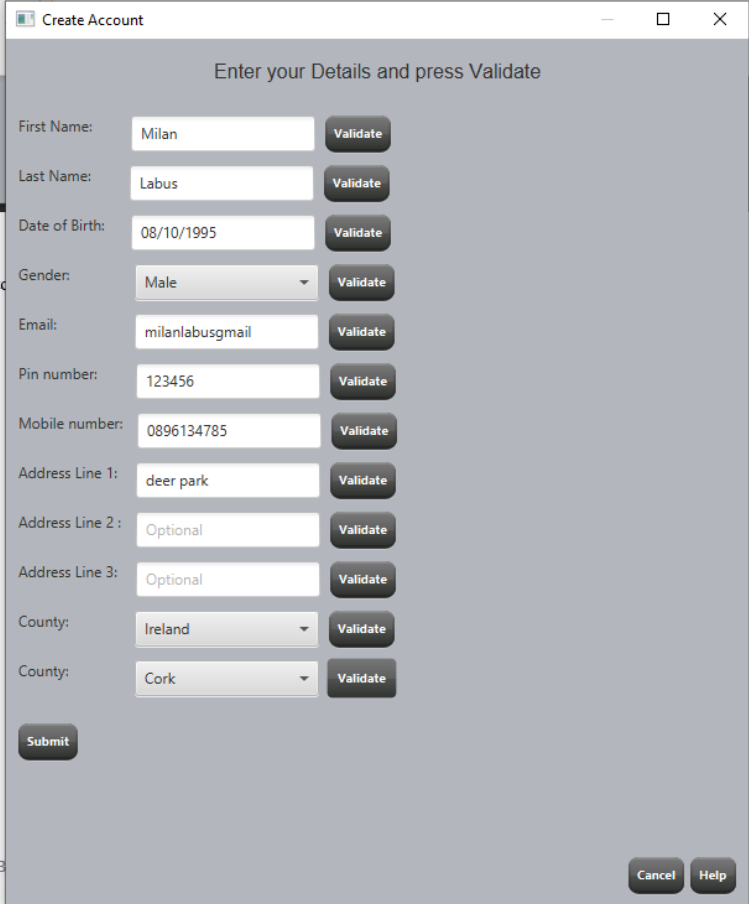




And now the incorrect manager login:



Now to show an account being successfully created and written to txt file



The screenshot shows a 'Create Account' dialog box with the title 'Create Account' and a subtitle 'Enter your Details and press Validate'. The dialog contains the following fields and buttons:

- First Name:
- Last Name:
- Date of Birth:
- Gender:
- Email:
- Pin number:
- Mobile number:
- Address Line 1:
- Address Line 2:
- Address Line 3:
- County:
- County:

At the bottom left is a  button. At the bottom right are  and  buttons.

'' AaBbCcD AaBbCc AaBbCcD AaBbCc AaBbCc AaBbCcD AaBbCcD AaBbCcD A

Create Account

Enter your Details and press Validate

First Name: Milan Validate

Last Name: Labus Validate

Date of Birth: 08/10/1985 Validate

Gender: M Validate

Email: m Validate

Pin number: 1 Validate

Mobile number: 0 Validate

Address Line 1: deer park Validate

Address Line 2: Optional Validate

Address Line 3: Optional Validate

County: Ireland Validate

County: Cork Validate

Submit

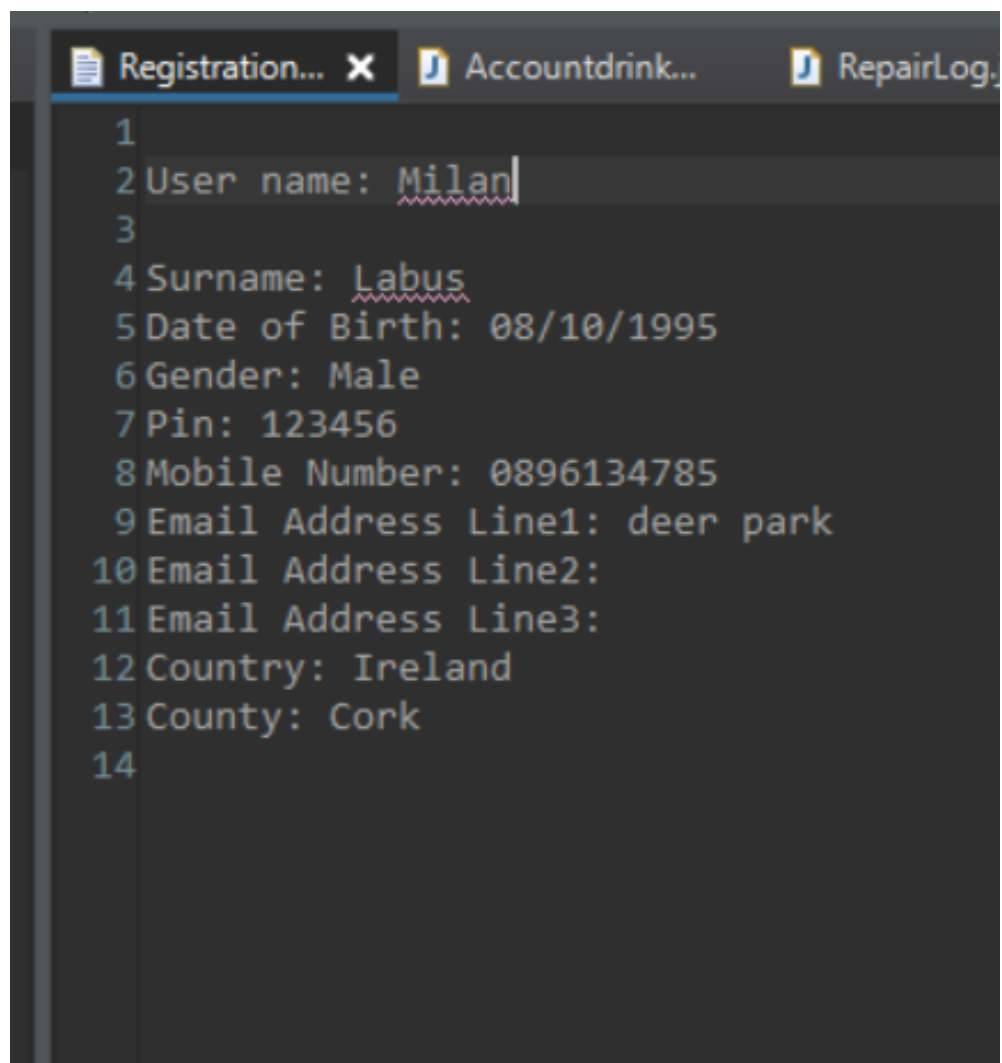
Validation Box

Validation result!

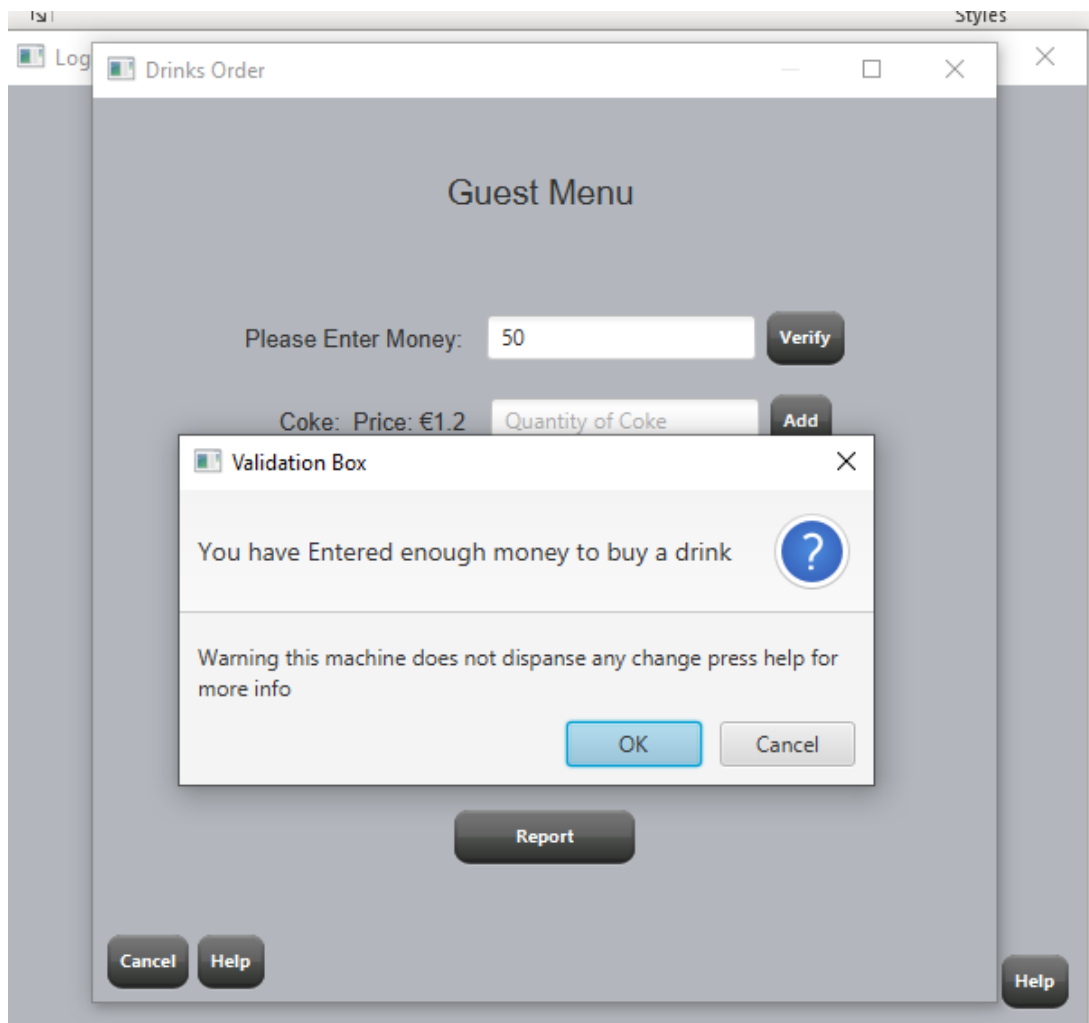
Validation is correct!

OK Cancel

Cancel Help



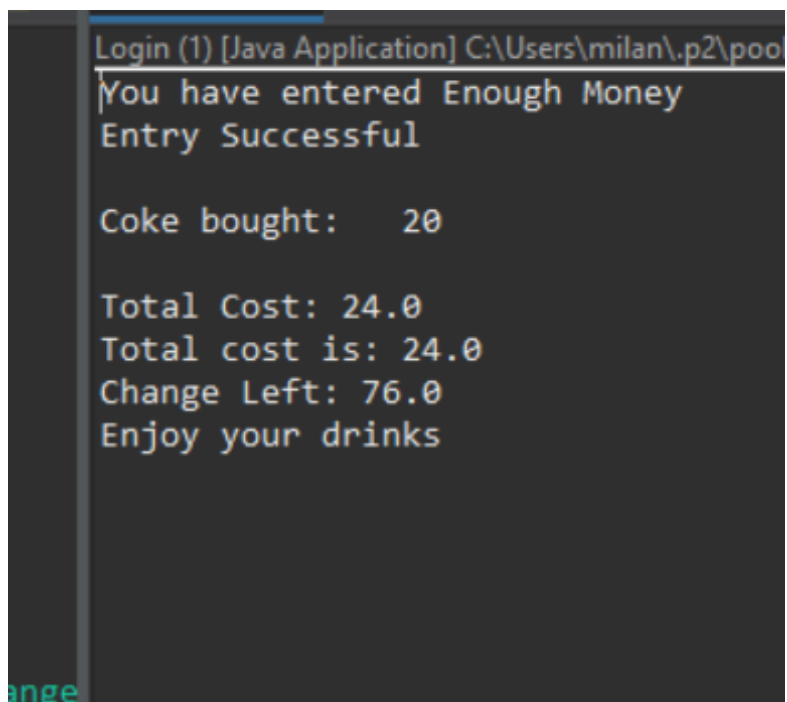
Now to show what happens when a guest enters more than 1.20 and pays with more than 1.20



```
You have entered Enough Mon  
Entry Successful  
  
Coke bought: 2  
Fanta bought: 2  
Sprite bought: 2  
Pepsi bought: 2  
  
Total Cost: 9.6  
Total cost is: 9.6  
Change Left: 40.4  
Enjoy your drinks
```

A user trying to buy a drink that's not available:

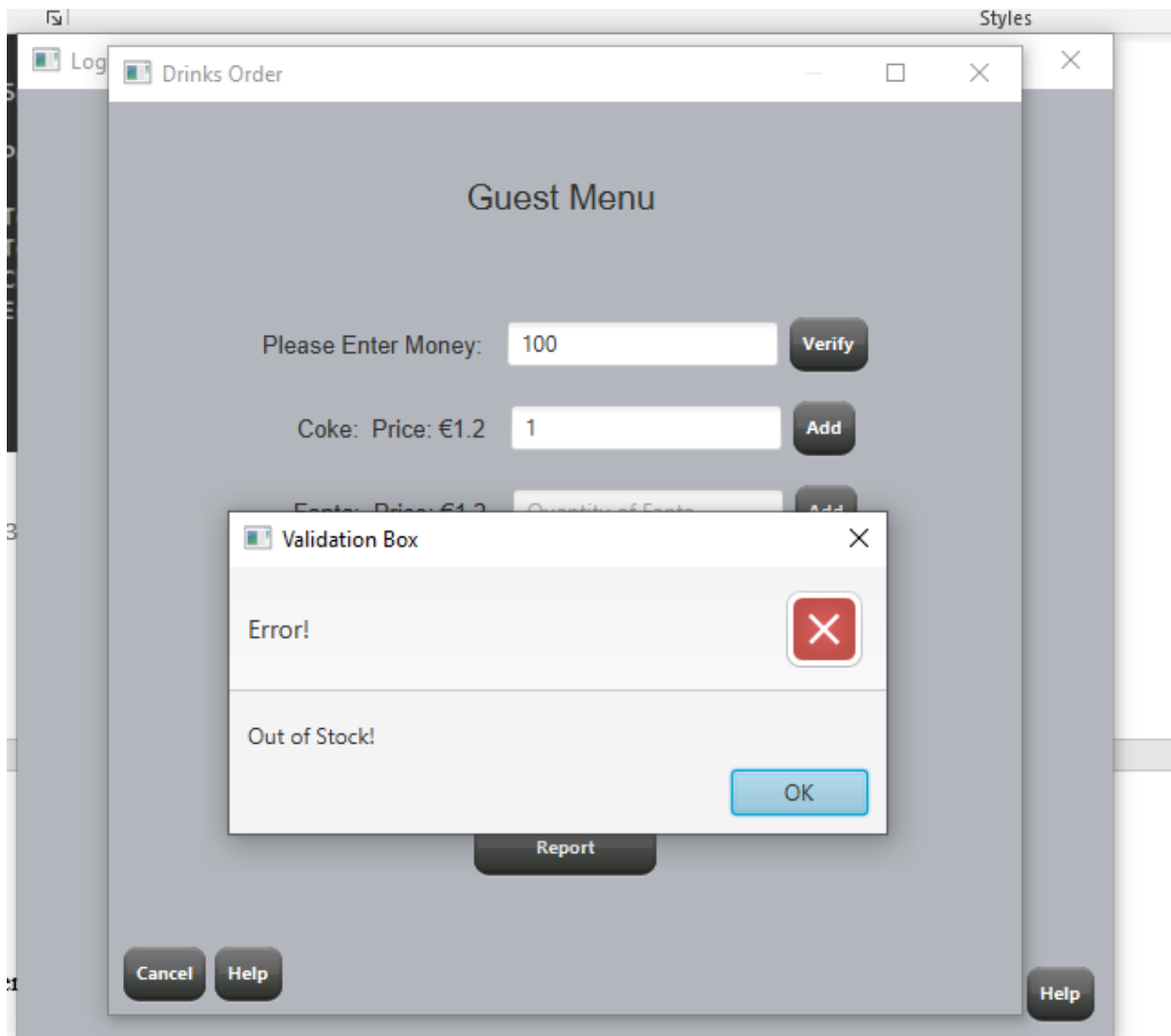
Here I have bought 20 cokes previously bringing the stock down to zero so I try to buy 1 again to test it



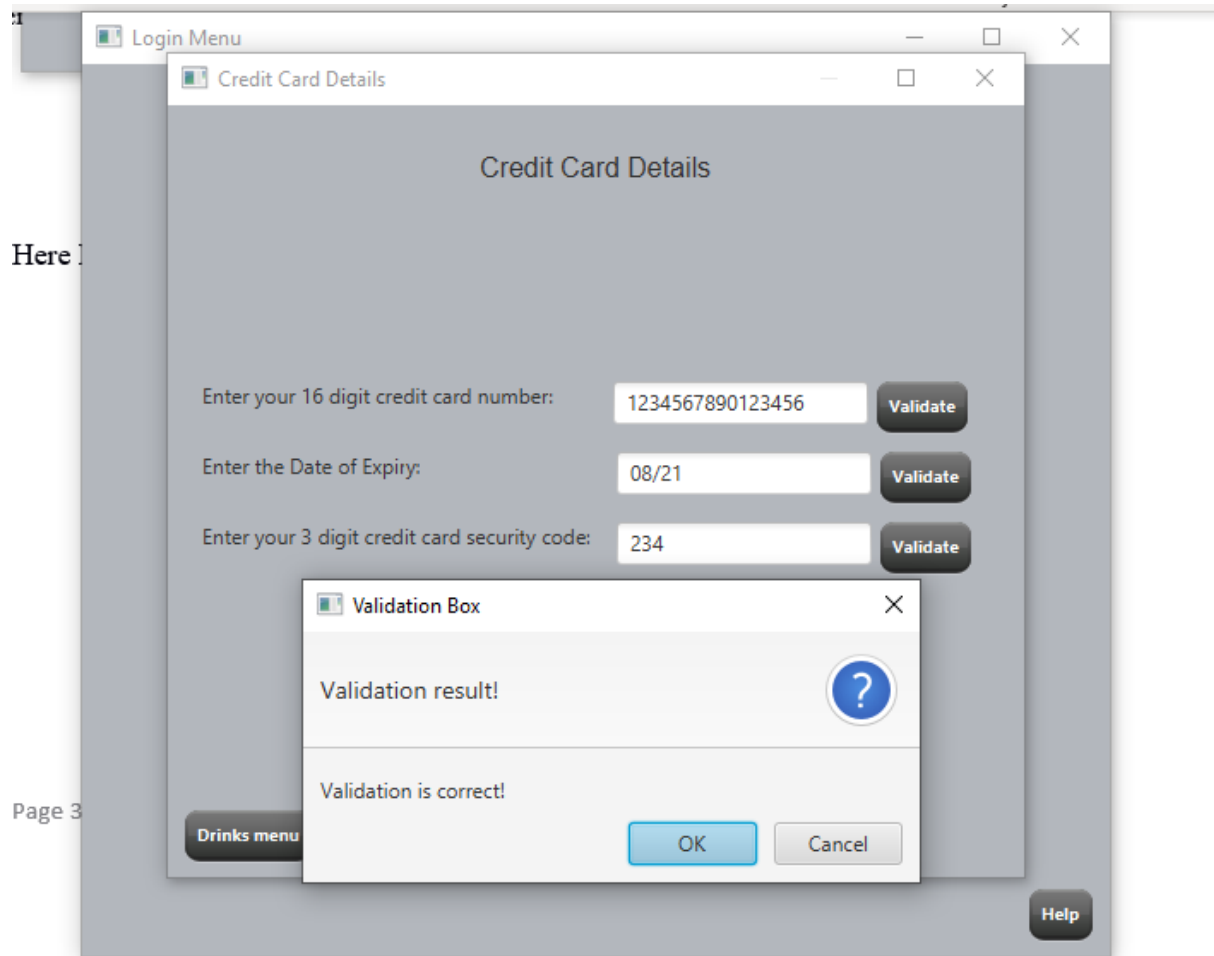
```
Login (1) [Java Application] C:\Users\milan\.p2\poo
You have entered Enough Money
Entry Successful

Coke bought: 20

Total Cost: 24.0
Total cost is: 24.0
Change Left: 76.0
Enjoy your drinks
```

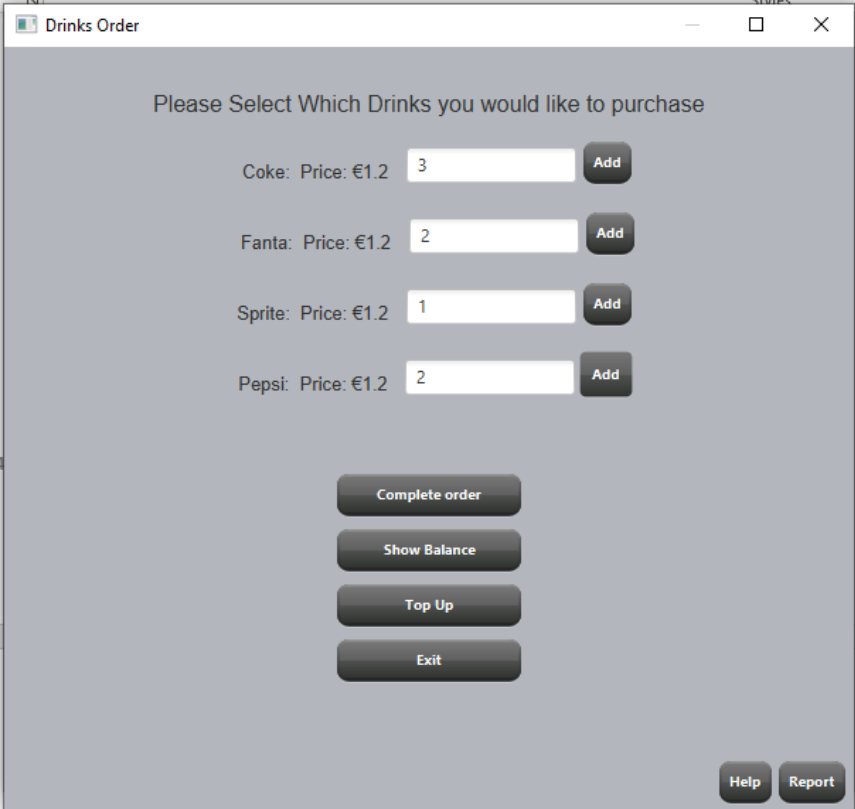


Here I will show a successful credit card validation





Now I will show drinks count updating after purchase



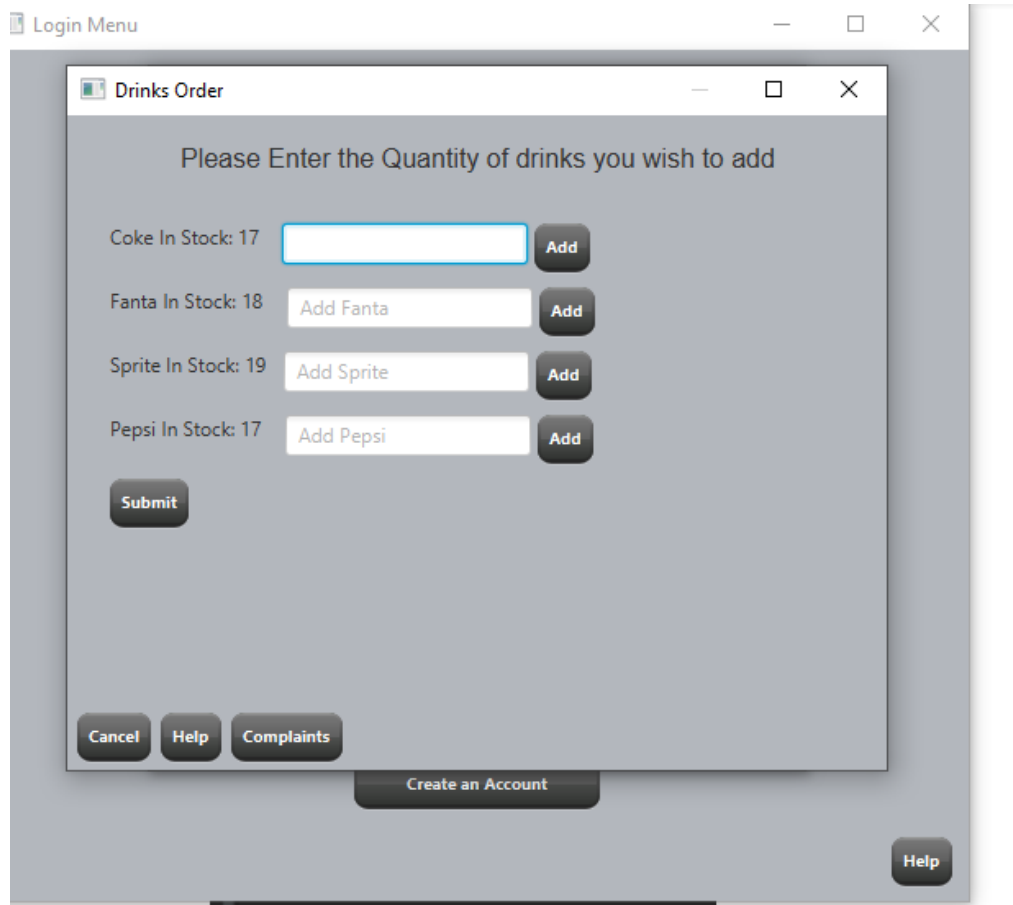
The screenshot shows a window titled "Drinks Order" with a light gray background. At the top, it says "Please Select Which Drinks you would like to purchase". Below this, there are four rows of drink selection:

- Coke: Price: €1.2 with a text input field containing "3" and an "Add" button.
- Fanta: Price: €1.2 with a text input field containing "2" and an "Add" button.
- Sprite: Price: €1.2 with a text input field containing "1" and an "Add" button.
- Pepsi: Price: €1.2 with a text input field containing "2" and an "Add" button.

Below the drink selection, there are four buttons stacked vertically: "Complete order", "Show Balance", "Top Up", and "Exit". In the bottom right corner, there are two more buttons: "Help" and "Report".

Now I will show drinks count updating after purchase

```
Coke bought: 3
Fanta bought: 2
Sprite bought: 1
Pepsi bought: 2
Total Cost: 9.2
```



As we see here in the maintenance window the drinks have updated successfully

Now I will show the Manager adding drinks and removing revenue

**Drinks Order**

Please Enter the Quantity of drinks you wish to add

Coke In Stock: 17	<input type="text" value="2"/>	Add
Fanta In Stock: 18	<input type="text" value="1"/>	Add
Sprite In Stock: 19	<input type="text" value="Add Sprite"/>	Add
Pepsi In Stock: 17	<input type="text" value="2"/>	Add

Submit

Cancel Help Complaints

**Report Menu**

**Validation Box**

Validation result!

Validation is correct!

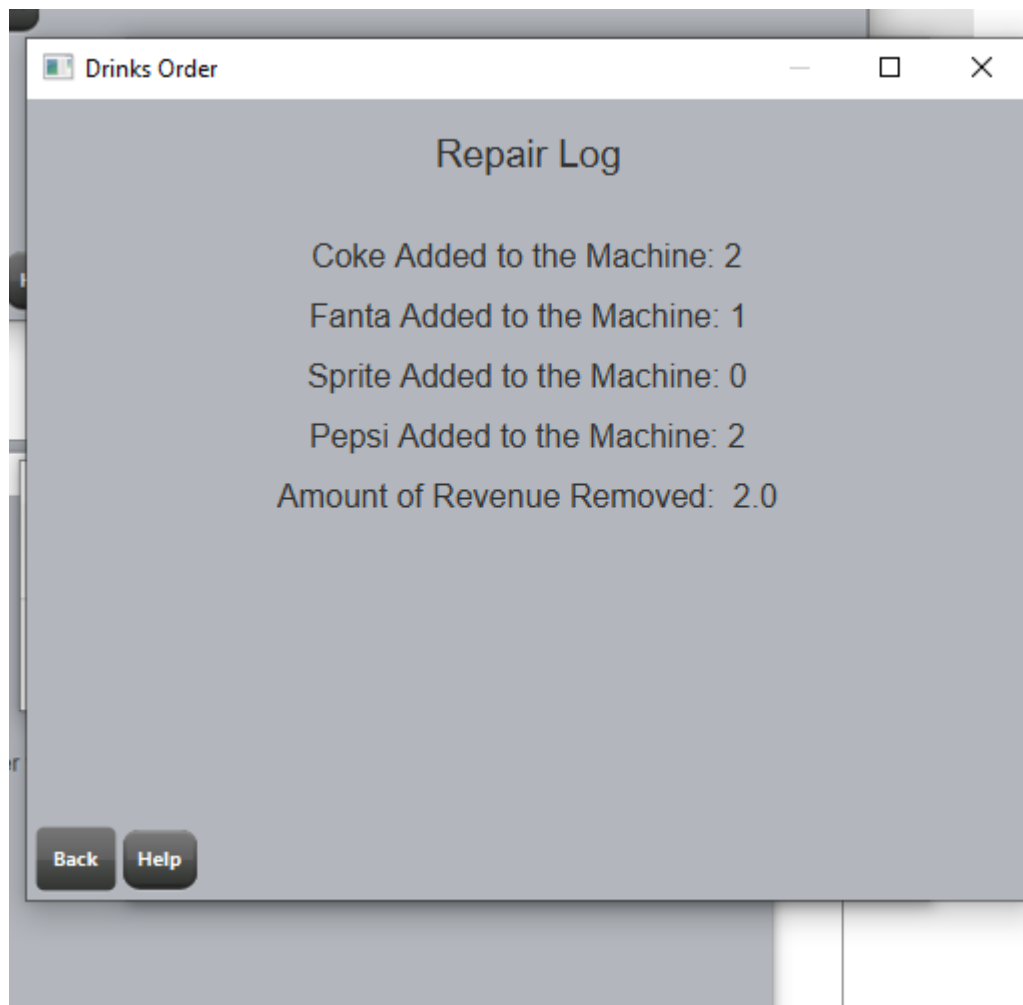
OK Cancel

Enter the Amount you Wish to Remove:

Remove

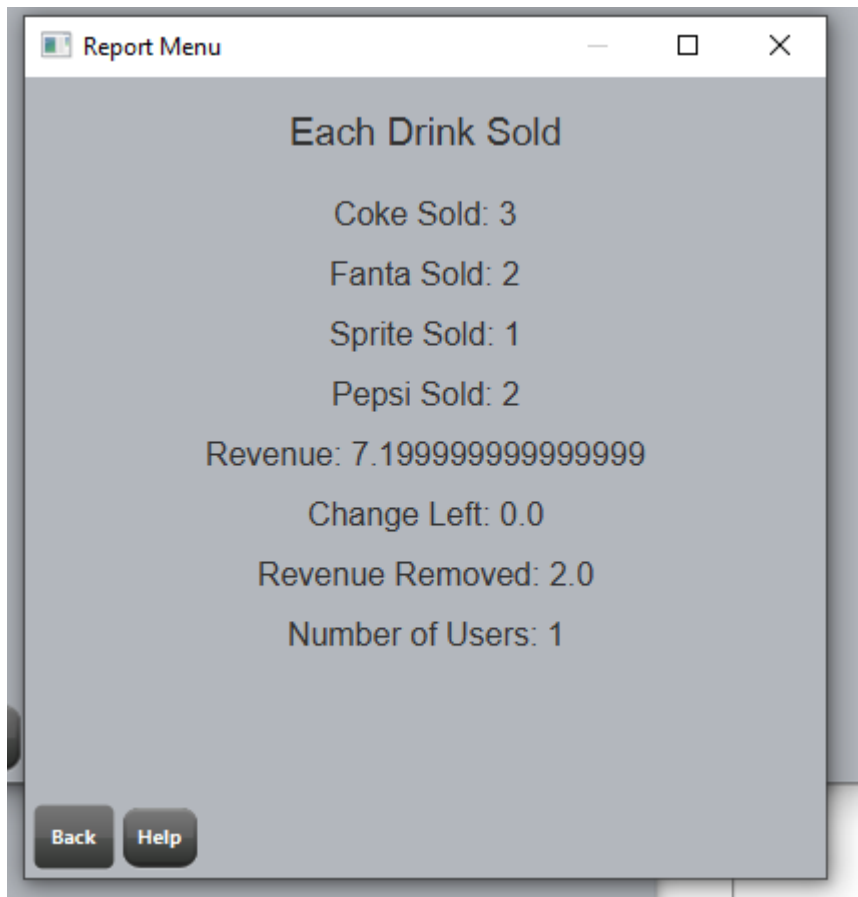
Show Balance

Back Help



As we can see here everything works successfully

And we can now test if all of this shows up on the report



Yes it does

Now lets be 100€ sure and test it again from the guests drinks menu to see if change left works

Log

### Drinks Order

## Guest Menu

Please Enter Money:

Coke: Price: €1.2

Fanta: Price: €1.2

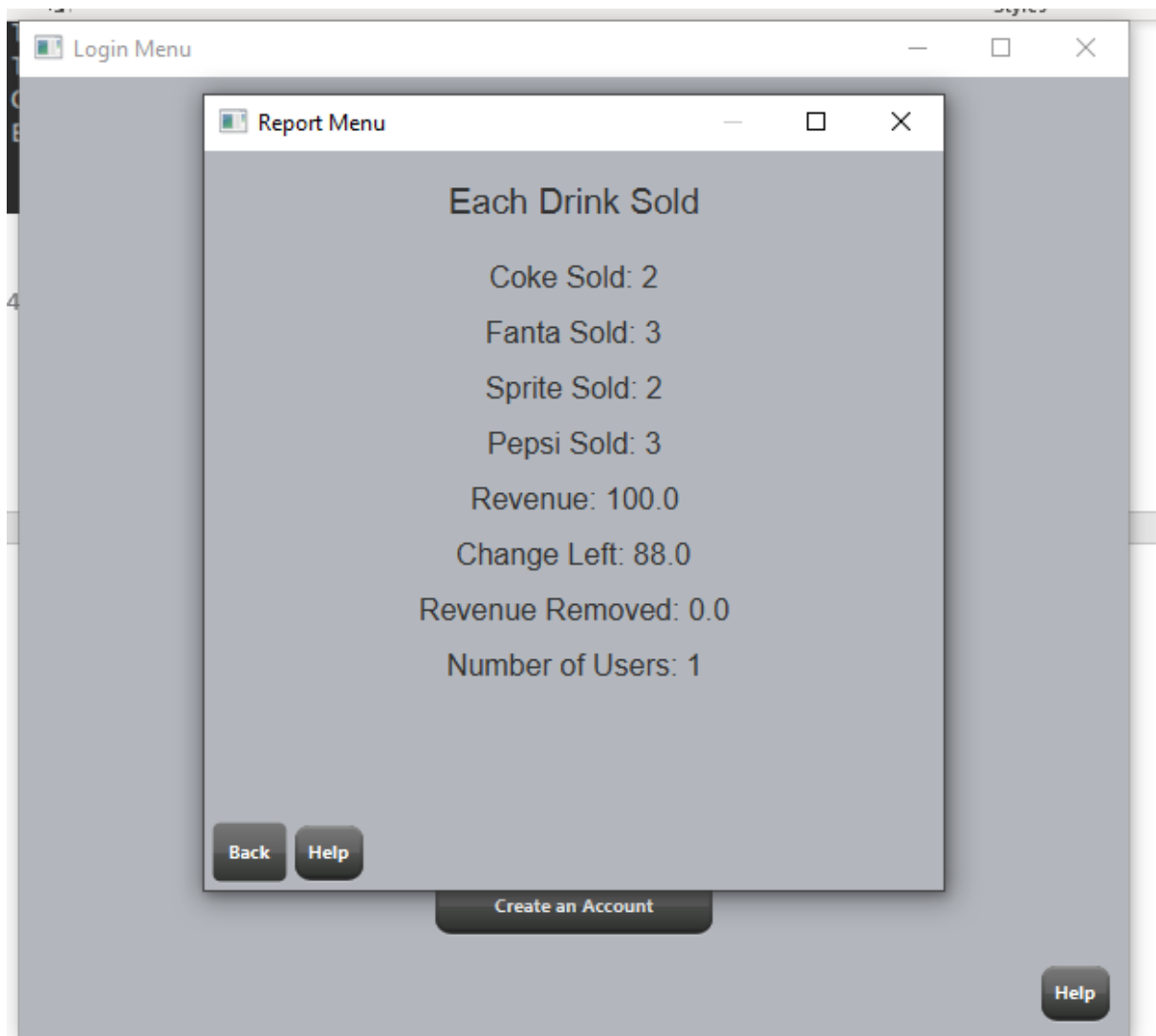
Sprite: Price: €1.2

Pepsi: Price: €1.2

```
Login (1) [Java Application] C:\Users\milan\.p2\po
You have entered Enough Money
Entry Successful

Coke bought: 2
Fanta bought: 3
Sprite bought: 2
Pepsi bought: 3

Total Cost: 12.0
Total cost is: 12.0
Change Left: 88.0
Enjoy your drinks
```



Yes everything works perfectly testing was successful fully

Source code:

```
import javafx.application.Application;
import javafx.geometry.Insets;
import javafx.geometry.Pos;
import javafx.scene.Scene;
import javafx.scene.control.Alert;
import javafx.scene.control.Button;
import javafx.scene.control.Label;
import javafx.scene.control.TextField;
import javafx.scene.control.Alert.AlertType;
import javafx.scene.layout.BorderPane;
import javafx.scene.layout.HBox;
import javafx.scene.layout.VBox;
import javafx.scene.text.Font;
import javafx.stage.Modality;
import javafx.stage.Stage;
```

```
public class Login extends Application {
```

```
    public static void main(String[] args) {
        launch(args);
    }
```



@Override

```
public void start(Stage primaryStage) {
```

```
    Stage window1 = new Stage();//Creating the Stage
```

```
    window1.setTitle("Login");
```

```
    window1.initModality(Modality.APPLICATION_MODAL);
```

```
    window1 = primaryStage;
```

```
    window1.setTitle("Login Menu");//Creating the Title
```

```
    //Creating the Label for the Username and pin Textfields
```

```
    Label login = new Label();
```

```
    login.setText("Enter your Username and Pin");
```

```
    login.setFont(new Font("Arial", 16));
```

```
    Label username = new Label();
```

```
    username.setText("Username: ");
```

```
    Label pin = new Label();
```

```
    pin.setText("Pin: ");
```

```
    //username Textfield
```

```
    TextField userNameText = new TextField();
```

```
userNameText.setPromptText("Username");  
userNameText.setMaxWidth(250);
```

```
// creating Pin Textfield
```

```
TextField enterPin = new TextField();  
enterPin.setPromptText("Pin");  
enterPin.setMaxWidth(250);
```

```
//Login Details Confirm Button when the Textfields have been  
filled
```

```
Button confirm = new Button();  
confirm.setText("Confirm");  
confirm.setMaxWidth(150);  
confirm.setOnAction(e -> {
```

```
ReaderFile.readTheFile("C:\\Users\\milan\\OneDrive - Cork  
College of  
Commerce\\Fx\\VendingMachineAssignement\\src\\Registration.txt"); //  
Reading from my file
```

```
String checkUsername = ReaderFile.username; // Finding  
username what was created
```

```
String checkPin = ReaderFile.pin; // Finding password what  
was created
```

```
        if (userNameText.getText().equals(checkUsername) &&  
enterPin.getText().equals(checkPin)) { //If the username and pin match it takes  
you to the drinks menu
```

```
        CreateAccount.getLoginConfirmation(); // get Alert  
Box if login is correct
```

```
        System.out.println("You logged in with username: " +  
checkUsername + "\n" + "And with password: " + checkPin);
```

```
        belowTwoWarning();
```

```
        AccountdrinksMenu.display();
```

```
    }
```

```
    else
```

```
        CreateAccount.getLoginError(); // Get Error Alert  
Box if information are incorrect
```

```
    });
```

```
//Login Details Confirm Button for the Manager Login
```

```
Button manager = new Button();
```

```
manager.setText("Manager Login");
```

```
manager.setMaxWidth(150);
```

```
manager.setOnAction(e -> {
```

```
        ReaderFile.readTheFile("C:\\Users\\milan\\OneDrive - Cork  
College of  
Commerce\\Fx\\VendingMachineAssignement\\src\\Registration.txt"); //  
Reading from the file
```

```
String checkUsername = ReaderFile.username; // Finding
Username what was created
```

```
String checkPin = ReaderFile.pin; // Finding password what
was created
```

```
if (userNameText.getText().equals(checkUsername) &&
enterPin.getText().equals(checkPin)) {
```

```
    CreateAccount.getLoginConfirmation(); // get Alert
Box if login is correct method from CreateAccount class
```

```
    System.out.println(
        "You logged in with username: " +
checkUsername + "\n" + "And with password: " + checkPin);
```

```
    Manager.display(); //Displaying the Managers Menu
```

```
}
```

```
else
```

```
    CreateAccount.getLoginError(); // Get Error Alert
Box if information are incorrect
```

```
});
```

```
//The Create Account button
```

```
Button createAccount = new Button();
```

```
createAccount.setText("Create an Account");
```

```
createAccount.setMaxWidth(150);
```

```

createAccount.setOnAction(e ->{
    CreateAccount.display();
});

```

menu

//Creating a Guest Login Button that will take the user to the drinks

```

Button guestLogin = new Button();
guestLogin.setText("Continue as a Guest");
guestLogin.setMaxWidth(150);
guestLogin.setOnAction(e -> {
    GuestDrinksMenu.display();
});

```

Button help = new Button("Help");

help.setOnAction(e -> {

getHelp(); //The help button calling my getHelp method

which will show an alert

```

});

```

//Making a HBox for the help button

```

HBox h = new HBox();

```

h.setAlignment(Pos.BOTTOM\_RIGHT); //Alligning my Hbox to

the bottom right

```

h.getChildren().addAll(help);

```

//Creating a VBox for all of the textfields and Labels

```

VBox v = new VBox();

v.setSpacing(19);

v.setPadding(new Insets(50, 14, 14, 14));

v.setAlignment(Pos.CENTER);

v.getChildren().addAll(login, username, userNameText, pin,
enterPin);

```

```

//Making a VBox that will contain all of the buttons

VBox menuButtons = new VBox();

menuButtons.setSpacing(18);

menuButtons.setPadding(new Insets(1,10, 10,10));

menuButtons.setAlignment(Pos.CENTER);

menuButtons.getChildren().addAll( confirm,manager,guestLogin,
createAccount, h);

```

```

//Creating a BorderPane for this menu

BorderPane borderpane = new BorderPane();

borderpane.setTop(v);

borderpane.setBottom(menuButtons);

```

```

//Making my scene and setting the size of it

Scene sceneA = new Scene(borderpane, 600, 520);

sceneA.getStylesheets().add("style.css");

```

```

        window1.setScene(sceneA);

        window1.show();

    }

```

```

public static Alert belowTwoWarning() {

    //Alert displaying that your account Balance is under 2 Euro

    Alert alert = new Alert(AlertType.INFORMATION);

    alert.setTitle("Alert Box");

    alert.setHeaderText("Reminder");

    alert.setContentText("Your Account Balance stands at under 2 euro
please Top Up");

    alert.showAndWait();

    return alert;

}

```

```

public static Alert getHelp() {

    //An alert that will Display when the help button is Pressed

    Alert alert = new Alert(AlertType.INFORMATION);

    alert.setTitle("Help Box");

    alert.setHeaderText("Hints and Tips");

    alert.setContentText("Please input name and password if you have
an account and click Confirm\n"

```

```
Account\n"
+ "1. If you don't have a account click Create
Guest"
+ "2. If you don't wish to create an account click
);
alert.showAndWait();

return alert;
}

}
```



```

import javafx.geometry.Insets;
import javafx.geometry.Pos;
import javafx.scene.Scene;
import javafx.scene.control.Alert;
import javafx.scene.control.Button;
import javafx.scene.control.ComboBox;
import javafx.scene.control.Label;
import javafx.scene.control.TextField;
import javafx.scene.control.Alert.AlertType;
import javafx.scene.layout.BorderPane;
import javafx.scene.layout.HBox;
import javafx.scene.layout.VBox;
import javafx.scene.text.Font;
import javafx.stage.Modality;
import javafx.stage.Stage;

public class CreateAccount {

    //Making all of the personal details Private
    private static TextField firstNameEnter;
    private static TextField surnameText;
    private static TextField DOBText;
    private static ComboBox<String> genderCombo;
    private static TextField PinText;
    private static TextField emailText;
    private static TextField mobileText;
    private static TextField addressText;
    private static TextField addressText2;
    private static TextField addressText3;
    private static ComboBox<String> countryCombo;
    private static ComboBox<String> countyCombo;
    static ComboBox<String> getEmpty;
    static Scene SceneB;
    static Scene SceneC;

    public static void display() {

        Stage window2 = new Stage();
        window2.setTitle "Create Account";
        window2.initModality (Modality.APPLICATION_MODAL);
        //Stage details
        Label l = new Label();
        l.setText("Enter your Details and press Validate");
        l.setFont(new Font("Arial", 16));
    }
}

```

```

//The First Name
Label firstName = new Label();
firstName.setText("First Name:      ");
firstNameEnter = new TextField();
firstNameEnter.setMaxHeight(170);
Button validateFirstName = new Button();
validateFirstName.setText("Validate");
validateFirstName.setOnAction(e -> {

    //creating an if statement to make sure first name a
string
    if(isAlphabetic(firstNameEnter)) {
        getConfirmationBox(); //Display Alert Box if the
if statement is met
    }else
        getErrorMessage(); //If not Displaying an Error
Box via an alert in a Method
    });
    HBox h = new HBox();
    h.getChildren().addAll(firstName, firstNameEnter,
validateFirstName);
    h.setSpacing(8);

//Create new label for the user's last name
Label surname = new Label();
surname.setText("Last Name:      ");
surnameText = new TextField();
surnameText.setMaxHeight(170);
//Creating validate button for the second name
Button validateSecondName = new Button(); //Creating new
Button
validateSecondName.setText("Validate");
validateSecondName.setOnAction(e -> {
    //again an if statement to make sure surname is a string
    if(isAlphabetic(surnameText)) {
        getConfirmationBox();
    }else
        getErrorMessage();
    });
    HBox h1 = new HBox();
    h1.getChildren().addAll(surname, surnameText,
validateSecondName);
    h1.setSpacing(8);

//The Date of Birth
Label DOB = new Label();
DOB.setText("Date of Birth:      ");
DOBText = new TextField();
DOBText.setMaxHeight(170);

```

```

Button validateDOB = new Button();
validateDOB.setText("Validate");
validateDOB.setOnAction(e -> {
    overEighteens();
});
HBox h2 = new HBox();
h2.getChildren().addAll(DOB, DOBText, validateDOB);
h2.setSpacing(8);

```

```

//Creating Section for the Gender
Label gender = new Label();
gender.setText("Gender:\t\t ");
//Creating new ComboBox for the gender
genderCombo = new ComboBox<>();
genderCombo.setPrefWidth(149);
genderCombo.setMaxHeight(170);
genderCombo.getItems().addAll( //populating the drop down

```

combo

```

    "Male", "Female"
);
Button submitG = new Button();
submitG.setText("Validate");
submitG.setOnAction(e -> {
    genderCombo.getValue();
});
HBox h3 = new HBox();
h3.getChildren().addAll(gender, genderCombo, submitG);
h3.setSpacing(8);

```

```

//Creating Email Label
Label email = new Label();
email.setText("Email:\t\t ");
//Creating new TextField to enter email
emailText = new TextField();
emailText.setMaxHeight(170);
Button emailValidation = new Button();
emailValidation.setText("Validate");
emailValidation.setOnAction(e -> {
    //again an if statement to make sure surname is a string
    if(isAlphabetic(emailText)) {
        getConfirmationBox();
    } else
        getErrorMessage();
});
HBox h4 = new HBox();
h4.getChildren().addAll(email, emailText, emailValidation);
h4.setSpacing(8);

```

```

//Creating new Label for the Pin
Label Pin = new Label();
Pin.setText("Pin number: ");
PinText = new TextField();
PinText.setMaxHeight 170;
Button PinValidation = new Button();
PinValidation.setText("Validate");
PinValidation.setOnAction( e -> {
    //creating an if statement to make sure its a Number
    if(isInt(PinText)) {

        getConfirmationBox(); //Display Alert Box
    }else
        getErrorMessage(); //Display Error Box

});

HBox h5 = new HBox();
h5.getChildren().addAll(Pin,PinText, PinValidation);
h5.setSpacing(8);

//Creating new Label for phone number
Label mobile = new Label();
mobile.setText("Mobile number: ");
//Textfield for the user to input his phone number
mobileText = new TextField();
mobileText.setMaxHeight 170;
Button mobileValidation = new Button();
mobileValidation.setText("Validate");
mobileValidation.setOnAction( e -> {
    //creating an if statement to make sure its a Date
    if(isDigitOnly(mobileText)) {

        getConfirmationBox(); //Display Alert Box
    }else
        getErrorMessage(); //Display Error Box

});

HBox h6 = new HBox();
h6.getChildren().addAll(mobile,mobileText, mobileValidation);
h6.setSpacing(8);

//Creating address Label
Label address1 = new Label();

```

```

address1.setText("Address Line 1: ");
addressText = new TextField();
addressText.setMaxHeight(170);
Button address1Validation = new Button();
address1Validation.setText("Validate");
address1Validation.setOnAction(e -> {
    //again an if statement to make sure address is a string
    if(isAlphabetic(addressText)) {
        getConfirmationBox();
    }else
        getErrorMessage();
});
HBox h7 = new HBox();

h7.getChildren().addAll(address1,addressText,address1Validation);
h7.setSpacing(8);

//Creating new Label for the second line of the address
Label address2 = new Label();
address2.setText("Address Line 2 : ");
//Creating new TextField for the addressLine
addressText2 = new TextField();
addressText2.setPromptText("Optional");
addressText2.setMaxHeight(170);
Button address2Validation = new Button();
address2Validation.setText("Validate");
address2Validation.setOnAction(e -> {
    //again an if statement to make sure address is a string
    if(isAlphabetic(addressText2)) {
        getConfirmationBox();
    }else
        getErrorMessage();
});
HBox h8 = new HBox();

h8.getChildren().addAll(address2,addressText2,address2Validation);
h8.setSpacing(8);

//Creating new Label for the third line of the address
Label address3 = new Label();
address3.setText("Address Line 3: ");
//Creating new TextField for the addressLine
addressText3 = new TextField();
addressText3.setPromptText("Optional");
addressText3.setMaxHeight(170);
Button address3Validation = new Button();
address3Validation.setText("Validate");
address3Validation.setOnAction(e -> {
    //again an if statement to make sure address is a string
    if(isAlphabetic(addressText3)) {

```

```

        getConfirmationBox();
    }else
        getErrorMessage();
    });
    HBox h9 = new HBox();
    h9.getChildren().addAll(address3,addressText3,
address3Validation);
    h9.setSpacing(8);

    //Label for Country Drop down Box
    Label country = new Label();
    country.setText("Country:\t\t ");
    countryCombo = new ComboBox<>();
    countryCombo.getItems().addAll(
        "Ireland", "Uk", "France" , "Germany", "Italy",
"Spain"
    );
    countryCombo.setPrefWidth(149);
    countryCombo.setMaxHeight(170);
    Button submitC = new Button();
    submitC.setText("Validate");
    submitC.setOnAction(e-> countryCombo.getValue());
    HBox h10 = new HBox();
    h10.getChildren().addAll(country, countryCombo,submitC);
    h10.setSpacing(8);

    //Label for County Drop down Box
    Label county = new Label();
    county.setText("County:\t\t ");
    countyCombo = new ComboBox<>();
    countyCombo.setPrefWidth(149);
    countyCombo.getItems().addAll(
        "Cork", "Limerick", "Clare" , "Dublin", "Kerry",
"Other"
    );
    countyCombo.setMaxHeight(170);
    Button submitCounty = new Button();
    submitCounty.setText("Validate");
    submitCounty.setOnAction(e-> countyCombo.getValue());
    county.setText("County:\t\t ");
    HBox h11 = new HBox();
    h11.getChildren().addAll(county, countyCombo,submitCounty);
    h11.setSpacing(8);

    //back button
    Button cancel = new Button("Cancel");
    cancel.setOnAction(e -> {
        window2.close();
    });

```

```

        Button help = new Button("Help");
        help.setOnAction(e -> { //If the button is clicked, you will
get help window with small description
            getHelp();
        });
        //Create new HBox for the help button
        HBox bottom = new HBox();
        bottom.setAlignment(Pos.BOTTOM_RIGHT); //Positioning of the
button
        bottom.getChildren().addAll(cancel, help);
        bottom.setPadding(new Insets(10, 10, 10, 10));
        bottom.setSpacing(5);

        //button to submit all of the details you have entered
        Button finalSubmit = new Button("Submit");
        finalSubmit.setAlignment(Pos.BOTTOM_CENTER);

        finalSubmit.setOnAction(e -> {

            //An IF statement to make sure all of the Info was
entered correctly
            if (isAlphabetic(firstNameEnter) &&
isAlphabetic(surnameText) && isAlphabetic(emailText) && isInt(PinText) &&
isDigitOnly(mobileText)
                && isAlphabetic(addressText)) { //If
validation of first name and PPSN is correct, then create new account
                getConfirmationBox();
                CreditCard.display(); //Opening the Credit Card
Menu
                pw.write(); //Writing the information in writer
class
                window2.close(); //Close the current window

            } else if (!isAlphabetic(firstNameEnter) || !
isAlphabetic(surnameText) && !isAlphabetic(emailText) && !isInt(PinText)
&& !isDigitOnly(mobileText)
                && !isAlphabetic(addressText)) { //If
Validation is not correct, get Error message and try again
                Alert confirm = new Alert(AlertType.ERROR);
                confirm.setTitle("ERROR");
                confirm.setHeaderText("Validation result!");
                confirm.setContentText("Validation is not correct!
Please check if your first name or pps number is correct! Validate
everything before you click submit!");
                confirm.showAndWait();

```

```

        System.out.println("Wrong input");
    }
});

VBox heading = new VBox(); //Creating a VBox
heading.setAlignment(Pos.TOP_CENTER); //Making it in the
middle
heading.setPadding(new Insets(16,16,16,16));
heading.getChildren().addAll(l);

VBox v = new VBox(); //Creating a VBox
v.setAlignment(Pos.BASELINE_LEFT); //Making it Left
v.setPadding(new Insets(10,10,10,10));
v.getChildren().addAll(h, h1, h2, h3, h4, h5, h6, h7, h8, h9,
h10,h11,finalSubmit); //Adding all elements to the VBox
v.setSpacing(11);

BorderPane borderpane = new BorderPane();
borderpane.setTop(heading); //Setting the top
borderpane.setCenter(v); //Setting the Center
borderpane.setBottom(bottom); //Setting the Bottom

//Scene sceneB = new Scene(v,900,800); //Set scene
Scene sceneB = new Scene(borderpane, 600, 700);
sceneB.getStylesheets().add("style.css");
window2.setScene(sceneB);
window2.showAndWait();
}
}

```

```

public static void getWidth(TextField text) { //Method what will set
width
    text.setPrefWidth(200);
    text.setMaxWidth(200);
}
}

```



```
        public static TextField FirstNameTextField() { //getter for the
First Name because it is private, it will be accessible in the other
classes
```

```
        return firstNameEnter;
    }
```

```
        public static TextField getLastName() { //getter for the last name
because it is private, it will be accessible in the other classes
```

```
        return surnameText;
    }
```

```
        public static TextField getDob() { //getter for the DOB because it
is private, it will be accessible in the other classes
```

```
        return DOBText;
    }
```

```
        public static ComboBox<String> getGender() { //getter for the gender
because it is private, it will be accessible in the other classes
```

```
        return genderCombo;
    }
```

```
        public static TextField getEmail() { //getter for the email because
it is private, it will be accessible in the other classes
```

```
        return emailText;
    }
```

```
        public static TextField getAddress() { //getter for the address
because it is private, it will be accessible in the other classes
```

```
        return addressText;
    }
```

```
        public static TextField getAddressLine2() { //getter for the address
because it is private, it will be accessible in the other classes
```

```
        return addressText2;
    }
```

```
        public static TextField getAddressLine3() { //getter for the address
because it is private, it will be accessible in the other classes
```

```
        return addressText3;
    }
```

```
        public static ComboBox<String> getCountry() { //getter for the
country because it is private, it will be accessible in the other classes
```

```

        return countryCombo;
    }

    public static ComboBox<String> getCounty(){ //getter for the county
because it is private, it will be accessible in the other classes
        return countyCombo;
    }

    public static TextField getMobile() { //getter for the mobile number
because it is private, it will be accessible in the other classes
        return mobileText;
    }

    public static TextField getPin() { //getter for the mobile number
because it is private, it will be accessible in the other classes
        return PinText;
    }

```

```

    public static boolean isDigitOnly(TextField text) { //Method what
will set input to be only digits
        String phone = text.getText();

        if(!phone.matches("[0-9]+")) {
            throw new NumberFormatException("Invalid Input!");
        }else
        System.out.println("Correct Input!");
        return true;
    }

```

```

    public static boolean isAlphabetic(TextField text) { //Method for
validation, settin regex to accept only alphabetic and single apostrophe
        String field = text.getText();

        if(!field.matches("[a-zA-Z\\s']+")) {
            return false;
        }else
        System.out.println("Correct Input");
        return true;
    }

```

```

    public static boolean isInt(TextField text){
        try {

```

```

        int Pin = Integer.parseInt(text.getText());
        System.out.println("your pin is:" + Pin);
        return true;
    }
    catch (NumberFormatException e) {
        throw new NumberFormatException("Invalid Input!");
    }
}

public static void getErrorMessage() { //Method what will give you
alert box(confirmation)

    Alert alert1 = new Alert(AlertType.ERROR);
    alert1.setTitle("Validation Box");
    alert1.setHeaderText("Validation result!");
    alert1.setContentText("Validation is not correct! ");

    alert1.showAndWait();
}

public static void getConfirmationBox() { //Method what will give
you alert box(error)

    Alert alert1 = new Alert(AlertType.CONFIRMATION);
    alert1.setTitle("Validation Box");
    alert1.setHeaderText("Validation result!");
    alert1.setContentText("Validation is correct!");
    alert1.showAndWait();
}

public static Alert getHelp() { //Method what will give you alert box
with explanation what the user needs to do in the window

    Alert alert = new Alert(AlertType.INFORMATION);
    alert.setTitle("Help Box");
    alert.setHeaderText("Hints and Tips");
    alert.setContentText("All Fields are mandatory to fill before
you proceed:\n"
        + "1. Enter your first name: Letters Only\n"
        + "2. Enter your second name: Letters Only\n"
        + "3. Date of Birth: dd/mm/yy\n"
        + "4. Mobile Number: Digits Only\n"
        + "5. Pin: Digits Only\n"
        + "6. Address at least one line needed\n"
        + "7. Please Select one of the countries and click
validate"
    );
    alert.showAndWait();
}

```

```

        return alert;
    }

    public static void getLoginError() { //Login failed alert
        System.out.println("Login Error");
        Alert alert1 = new Alert(AlertType.ERROR);
        alert1.setTitle("Validation Box");
        alert1.setHeaderText("Login result!");
        alert1.setContentText("Login Failed! ");
        alert1.showAndWait();
    }

    public static void getLoginConfirmation() { //Login successful alert
        System.out.println("Login Successful");
        Alert alert1 = new Alert(AlertType.CONFIRMATION);
        alert1.setTitle("Validation Box");
        alert1.setHeaderText("Validation Login!");
        alert1.setContentText("Login was successful");
        alert1.showAndWait();
    }

    public static void overEighteens() { //An alert reminding a user
that they have to be Over 18 to Register
        Alert alert = new Alert(AlertType.INFORMATION);
        alert.setTitle("Alert");
        alert.setHeaderText("You must be over 18 to Register");
        alert.setContentText("You must be over 18 because you will
need a Credit card to complete Registration\n"
+ "If you are Over 18 press OK if you are under 18
press cancel and login as guest"

        );
        alert.showAndWait();
    }

}

import javafx.application.Platform;

```

```

import javafx.geometry.Insets;
import javafx.geometry.Pos;
import javafx.scene.Scene;
import javafx.scene.control.Alert;
import javafx.scene.control.Button;
import javafx.scene.control.Label;
import javafx.scene.control.TextField;
import javafx.scene.control.Alert.AlertType;
import javafx.scene.layout.BorderPane;
import javafx.scene.layout.HBox;
import javafx.scene.layout.VBox;
import javafx.scene.text.Font;
import javafx.stage.Modality;
import javafx.stage.Stage;

public class CreditCard {

    private static TextField creditEnter;
    private static TextField expiryDateEnter;
    private static TextField securityEnter;

    public static void display() {

        Stage window3 = new Stage(); //Creating the Stage
        window3.setTitle("Credit Card Info");
        window3.initModality(Modality.APPLICATION_MODAL);

        Label label2 = new Label("Credit Card Details");
        label2.setFont(new Font("Arial", 16));

        //getting the user to enter their 16 digit card number and
validating it
        Label creditNumber = new Label();
        creditNumber.setText("Enter your 16 digit credit card number:");

        creditEnter = new TextField();
        Button validteCardNumber = new Button();
        validteCardNumber.setText("Validate");
        validteCardNumber.setOnAction(e -> {
            if(isLong(creditEnter)) {
                getConfirmationBox();
            }else
                getErrorMessage();
        });

        HBox h1 = new HBox();
        h1.getChildren().addAll(creditNumber,creditEnter,
validteCardNumber);
        h1.setSpacing(5);

        //getting the user to enter the credit card Expiry date

```



```

        Button goBack = new Button("Drinks menu");
        goBack.setOnAction(e -> {
            AccountdrinksMenu display();
            window3.close();
            //Platform.setImplicitExit(false);
        });

        Button help = new Button("Help");
        help.setOnAction(e -> {
            getHelp();
        });

        VBox heading = new VBox(); //Creating a VBox
        heading.setAlignment(Pos.TOP_CENTER); //Making it in the
middle
        heading.setPadding(new Insets(16,16,16,16));
        heading.getChildren().addAll(label2);
        heading.setAlignment(Pos.CENTER);

        HBox Bottom = new HBox();
        Bottom.getChildren().addAll(goBack,help);
        Bottom.setAlignment(Pos.BOTTOM_LEFT);
        Bottom.setSpacing(5);

        VBox v2 = new VBox(10);
        v2.setAlignment(Pos.BASELINE_LEFT);
        v2.setPadding(new Insets(10,10,10,10));
        v2.getChildren().addAll(h1, h2, h3, topUpbtn);
        v2.setAlignment(Pos.CENTER);
        v2.setSpacing(12);

        Insets insets = new Insets(10);

        BorderPane borderpane = new BorderPane();
        borderpane.setTop(heading); //Setting the top
        BorderPane.setMargin(heading, insets);
        borderpane.setCenter(v2); //Setting the Center
        BorderPane.setMargin(v2, insets);
        borderpane.setBottom(Bottom); //Setting the Bottom
        BorderPane.setMargin(Bottom, insets);

        Scene SceneC = new Scene(borderpane, 500, 450);
        SceneC.getStylesheets().add("style.css");
        window3.setTitle("Credit Card Details");
        window3.setScene(SceneC);
        window3.show();
    }

```

```
public static TextField CreditNumberTextField() { //getter for the  
First Name because it is private, it will be accessible in the other  
classes
```

```
return creditEnter;
```

```
}
```

```
public static TextField expiryDateTextField() { //getter for the  
First Name because it is private, it will be accessible in the other  
classes
```

```
return expiryDateEnter;
```

```
}
```

```
public static TextField securityCodeTextField() { //getter for the  
First Name because it is private, it will be accessible in the other  
classes
```

```
return securityEnter;
```

```
}
```

```
public static boolean isLong TextField text){ //Method to check if a  
Long has been Entered
```

```
try {
```

```
long card = Long.parseLong(text.getText());  
System.out.println("your pin is:" + card);
```

```
return true;
```

```
}
```

```
catch (NumberFormatException e) {
```

```
getErrorMessage();  
throw new NumberFormatException("Invalid Input!");
```

```
}
```

```
}
```

```
public static boolean isInt TextField text){ //Method to check if an  
Interger was entered into the textfield
```

```
try {
```

```
int code = Integer.parseInt(text.getText());  
System.out.println("your pin is:" + code);  
getConfirmationBox();  
return true;
```

```
}
```

```
catch (NumberFormatException e) {
```

```
getErrorMessage();  
throw new NumberFormatException("Invalid Input!");
```

```
}
```



```
}
```

```
public static void getErrorMessage() { //Method what will give you  
alert box(confirmation)
```

```
Alert alert1 = new Alert(AlertType.ERROR);  
alert1.setTitle("Validation Box");  
alert1.setHeaderText("Validation result!");  
alert1.setContentText("Validation is not correct! ");  
  
alert1.showAndWait();
```

```
}
```

```
public static void getConfirmationBox() { //Method what will give  
you alert box(error)
```

```
Alert alert1 = new Alert(AlertType.CONFIRMATION);  
alert1.setTitle("Validation Box");  
alert1.setHeaderText("Validation result!");  
alert1.setContentText("Validation is correct!");  
alert1.showAndWait();
```

```
}
```

```
public static Alert getHelp() { //Alert for the help box
```

```
Alert alert = new Alert(AlertType.INFORMATION);  
alert.setTitle("Help Box");  
alert.setHeaderText("Hints and Tips");  
alert.setContentText("Please Fill all fields to Continue\n"  
+ "1. Enter your 16 digit credit card: numbers  
only\n"  
+ "2. Enter the expiry date: dd/mm/yyyy\n"  
+ "3. Enter the 3 digit security code found at the  
back\n"  
+ "4. If you wish to top up your account click Top  
Up\n"  
+ "5. If you wish to go to the drinks menu click  
Drink Menu"  
);  
alert.showAndWait();
```

```
return alert;
```

```
}
```

```
}
```

```
import javafx.application.Platform;
import javafx.geometry.Insets;
import javafx.geometry.Pos;
import javafx.scene.Scene;
import javafx.scene.control.Alert;
import javafx.scene.control.Button;
import javafx.scene.control.Label;
import javafx.scene.control.TextField;
import javafx.scene.control.Alert.AlertType;
import javafx.scene.layout.BorderPane;
import javafx.scene.layout.HBox;
import javafx.scene.layout.VBox;
import javafx.scene.text.Font;
import javafx.stage.Modality;
import javafx.stage.Stage;
```

```
public class TopUp {
```

```
    public static double accountBalance;
    private static TextField topUpEnter;
    public static double TopUpEnter;
```

```
    public static void display() {
```

```

Stage window4 = new Stage();

window4.setTitle("Credit Card Info");

window4.initModality(Modality.APPLICATION_MODAL);

Label topUpLbl = new Label();//Creating the label at the top of the window
topUpLbl.setText("Top Up Menu");
topUpLbl.setFont(new Font("Arial", 17));
topUpLbl.setPadding(new Insets(15,10,5,5));


Label amountLbl = new Label();
amountLbl.setText("Enter the Amount: ");
amountLbl.setFont(new Font("Arial", 13));
topUpEnter = new TextField(); //Textfield to enter the amount to top up by
Button ValidateBtn = new Button ();
ValidateBtn.setText("Validate");
ValidateBtn.setOnAction(e -> {
    if(isDouble(topUpEnter)) { //Validating that a
        addToBalance();

    }else
        getErrorMessage();
});

HBox h1 = new HBox();
h1.getChildren().addAll(amountLbl,topUpEnter, ValidateBtn);
h1.setAlignment(Pos.CENTER);

```

```
h1.setSpacing(5);
```

```
Button goToMenu = new Button("Drinks menu");//button to go back to the  
drinks menu
```

```
goToMenu.setOnAction(e -> {  
    window4.close();  
    Platform.setImplicitExit(false);  
});
```

```
Button showBalance = new Button("Show Balance");  
showBalance.setOnAction(e -> {  
    ShowBalance();  
});
```

```
Button help2 = new Button("Help");//Help Button  
help2.setOnAction(e -> {  
    getHelp();  
});
```

```
VBox v2 = new VBox(10);//Creating VBox  
v2.setPadding(new Insets(10,10,10,10));
```

```

v2.getChildren().addAll( h1,showBalance,goToMenu);
v2.setSpacing(18);
v2.setAlignment(Pos.CENTER);

```

```

Insets insets = new Insets(10);

```

Stage

```

BorderPane borderpane = new BorderPane();//Creating borderpane for the

```

```

borderpane.setTop(topUplbl);
BorderPane.setAlignment(topUplbl, Pos.CENTER);
BorderPane.setMargin(topUplbl, insets);
borderpane.setCenter(v2);
BorderPane.setMargin(v2, insets);
borderpane.setBottom(help2);
BorderPane.setMargin(help2, insets);
borderpane.setPadding(new Insets(6));

```

```

Scene SceneC = new Scene(borderpane,470, 350);
SceneC.getStylesheets().add("style.css");//link to my external css
window4.setTitle("Top Up");
window4.setScene(SceneC);
window4.show();

```

```
}
```

```
public static void ShowBalance() { //Button to show how much is in the Account  
Balance
```

```
    Alert alert = new Alert(Alert.AlertType.INFORMATION);  
    alert.setContentText("your balance is: " + accountBalance);  
    alert.show();
```

```
}
```

```
public static double addToBalance() { //Method to add the Top Up Entered to the  
account Balance
```

```
    double TopUpEnter;
```

```
//    String text1 = topUpEnter.getText();  
    TopUpEnter = Double.parseDouble(topUpEnter.getText());  
    accountBalance += TopUpEnter;  
    System.out.println("Your new Balance is: €" + accountBalance);  
    return accountBalance;  
}
```

```
public static boolean isDouble(TextField text){//Method to verify if its a double
```

```

        try {
            TopUpEnter = Double.parseDouble(text.getText());
            getConfirmationBox();
            return true;
        }
        catch(NumberFormatException e) {
            getErrorMessage();
            throw new NumberFormatException("Invalid Input!");
        }
    }
}

```

```

public static TextField getTopUpEnter() { //Getter for topUPEnter
    return topUpEnter;
}

```

```

public static void getErrorMessage() { //Method what will give you alert
box(confirmation)

```

```

    Alert alert1 = new Alert(AlertType.ERROR);
    alert1.setTitle("Validation Box");
    alert1.setHeaderText("Validation result!");
    alert1.setContentText("Validation is not correct! ");

```

```

        alert1.showAndWait();
    }

```

```

    public static void getConfirmationBox() { //Method what will give you alert
    box(error)

```

```

        Alert alert1 = new Alert(AlertType.CONFIRMATION);
        alert1.setTitle("Validation Box");
        alert1.setHeaderText("Validation result!");
        alert1.setContentText("Validation is correct!");
        alert1.showAndWait();
    }

```

```

    public static Alert getHelp() { //Alert for the help

```

```

        Alert alert = new Alert(AlertType.INFORMATION);
        alert.setTitle("Help Box");
        alert.setHeaderText("Hints and Tips");
        alert.setContentText("Enter an amount in numbers and press validate to add\n"
            + "1. If you wish to go to the menu for drinks press Drink
Menu\n"
            + "2. If you wish to see the amount of money in your account
press Show Balance\n"
        );

```



```
    alert.showAndWait();
```

```
    return alert;
```

```
}
```

```
}
```

```
import javafx.geometry.Insets;  
import javafx.geometry.Pos;  
import javafx.scene.Scene;  
import javafx.scene.control.Alert;  
import javafx.scene.control.Button;
```

```

import javafx.scene.control.Label;
import javafx.scene.control.TextField;
import javafx.scene.control.Alert.AlertType;
import javafx.scene.layout.BorderPane;
import javafx.scene.layout.HBox;
import javafx.scene.layout.VBox;
import javafx.scene.text.Font;
import javafx.stage.Modality;
import javafx.stage.Stage;

public class AccountdrinksMenu extends TopUp {

    //Variables
    static TextField amountCoke;
    static TextField amountFanta;
    static TextField amountSprite;
    static TextField amountPepsi;
    int quantity;
    static Button btnSelect;
    Button btnClear;
    Label lblcoke, lblfanta, lblsprite, lblpepsi;
    boolean checkingOut = true;
    static double price = 1.20;
    static double discount = 0.05;
    static int CokeBought;
    static int FantaBought;
    static int SpriteBought;
    static int PepsiBought;
    static int coke = 20;
    static int fanta = 20;
    static int sprite = 20;
    static int pepsi = 20;
    static double priceCoke;
    static double priceFanta;
    static double priceSprite;
    static double pricePepsi;
    static double cokeDiscount;
    static double fantaDiscount;
    static double spriteDiscount;
    static double pepsiDiscount;
    static double Revenue;
    static double totalCost;

    public static void display() {

        Stage window4 = new Stage();
    }

```

```

window4.setTitle("Drinks Menu");
window4.initModality(Modality.APPLICATION_MODAL);

Label menu = new Label(); //Label for the Top
menu.setText("Please Select Which Drinks you would like to
purchase");
menu.setFont(new Font("Arial", 16));
menu.setAlignment(Pos.TOP_CENTER);

//For Buying a Coke
Label lblcoke = new Label(String.format("    Coke: " +
"Price: €" + price + " ")); //Label showing price of Coke
lblcoke.setFont(new Font("Arial", 13));
amountCoke = new TextField(); //Textfield to enter the
quantity they wish to purchase
amountCoke.setPromptText("Quantity of Coke");
amountCoke.setPrefWidth(120);
amountCoke.setMaxWidth(120);

Button verifyCoke = new Button("Add"); //Button to add cokes
to total(basket)
verifyCoke.setOnAction(e ->{

    CokeBought = Integer.parseInt(amountCoke.getText());
//Parsing the textfield into an Int
    if(CokeBought < coke) { //IF statement checking if there
is more coke in stock than ordered
        coke -= CokeBought;
        cokeDiscount = CokeBought * discount; //Seeing how
much the discount will be by multiplying it by cokes ordered
        priceCoke = (CokeBought * price) - cokeDiscount;
//Find prices of cokes ordered and applying discount
        CreditCard.getConfirmationBox();
        System.out.println("\nCoke bought: " +
CokeBought);

    } else {
        outOfStockAlert();
    }
});

HBox h1 = new HBox(); //h for everything in ordering a coke
h1.setAlignment(Pos.BOTTOM_CENTER);
h1.getChildren().addAll(lblcoke, amountCoke, verifyCoke);
h1.setSpacing(5);

//For buying a Fanta
Label lblfanta = new Label(String.format("    Fanta: " +
"Price: €" + price + " "));

```

```

lblfanta.setFont(new Font("Arial", 13));
amountFanta = new TextField();
amountFanta.setPrefWidth(120);
amountFanta.setMaxWidth(120);
amountFanta.setPromptText("Quantity of Fanta");

Button verifyFanta = new Button("Add");
verifyFanta.setOnAction(e ->{
    FantaBought = Integer.parseInt(amountFanta.getText());
    if(FantaBought < fanta) { //Same method used for Coke
        fanta -= FantaBought;
        fantaDiscount = FantaBought * discount;
        priceFanta = (FantaBought * price) - fantaDiscount;
        CreditCard.getConfirmationBox();
        System.out.println("\nFanta bought: " +
FantaBought);

    }else {
        outOfStockAlert();
    }

});
HBox h2 = new HBox();
h2.setAlignment(Pos.BOTTOM_CENTER);
h2.getChildren().addAll(lblfanta, amountFanta, verifyFanta);
h2.setSpacing(5);

//For buying a Sprite
Label lblsprite = new Label(String.format("Sprite: " +
"Price: €" + price + "  "));
lblsprite.setFont(new Font("Arial", 13));
amountSprite = new TextField();
amountSprite.setPromptText("Quantity of Sprite");
amountSprite.setPrefWidth(120);
amountSprite.setMaxWidth(120);
Button verifySprite = new Button("Add");
verifySprite.setOnAction(e ->{
    SpriteBought = Integer.parseInt(amountSprite.getText());
    if(SpriteBought < sprite) { //Same Method as Coke
        CreditCard.getConfirmationBox();
        sprite -= SpriteBought;
        spriteDiscount = SpriteBought * discount;
        priceSprite = (SpriteBought * price) - spriteDiscount;
        System.out.println("\nSprite bought: " +
SpriteBought);

    }else {
        outOfStockAlert();
    }
}

```

```

    ));
    HBox h3 = new HBox();
    h3.setAlignment(Pos.BOTTOM_CENTER);
    h3.getChildren().addAll(lblsprite, amountSprite,
verifySprite);
    h3.setSpacing(5);

    //For buying Pepsi
    Label lblpepsi = new Label(String.format(" Pepsi: " +
"Price: €" + price + " "));
    lblpepsi.setFont(new Font("Arial", 13));
    amountPepsi= new TextField();
    amountPepsi.setPromptText("Quantity of Pepsi");
    amountPepsi.setPrefWidth(120);
    amountPepsi.setMaxWidth(120);

    Button verifyPepsi = new Button("Add");
    verifyPepsi.setOnAction(e ->{

        PepsiBought = Integer.parseInt(amountPepsi.getText());
        if(PepsiBought<pepsi) { //Same method as Coke
            CreditCard.getConfirmationBox();
            pepsi -=PepsiBought;
            pepsiDiscount = PepsiBought*discount;
            pricePepsi = (PepsiBought*price)-pepsiDiscount;
            System.out.println("\nPepsi bought: " +
PepsiBought);

        }
        else {
            outOfStockAlert();
        }

    });

    HBox h4 = new HBox();
    h4.setAlignment(Pos.BOTTOM_CENTER);
    h4.getChildren().addAll(lblpepsi, amountPepsi, verifyPepsi);
    h4.setSpacing(5);

    Button addToCart = new Button("Complete order");//Making a
complete order button
    addToCart.setOnAction(e -> {
        boolean result = ConfirmPurchase.display(totalCost);
//Linking a confirm purchase class that will return the answer

        if(result == true ) { //If the yes is clicked then we go
ahead with the order
            Transaction();

```

```

        totalCost =
priceCoke+priceFanta+priceSprite+pricePepsi; //Find total cost of the
Order
        System.out.println("\nTotal Cost: " + totalCost);
        //Revenue+=totalCost; //Adding the total cost to
the machine revenue
        TopUp.accountBalance-=totalCost; //Deducting the
total cost from the Account Balance
    }
    else {
        window4.close(); //If the no Button is pressed it
exits the stage
    }

});
addToCart.setPrefWidth(130);

Button showBalance = new Button("Show Balance"); //Button to
show the account Balance
showBalance.setOnAction(e -> {
    TopUp.ShowBalance();
});
showBalance.setPrefWidth(130);

Button topUp = new Button("Top Up"); //Button to go to top up
menu
topUp.setOnAction(e -> {
    TopUp.display();
});
topUp.setPrefWidth(130);

Button exit = new Button("Exit"); //Button to exit the Menu
exit.setOnAction(e -> {
    window4.close();
});
exit.setPrefWidth(130);

Button report = new Button("Report"); //Button if the user
wants to report something
report.setOnAction(e -> {
    Complaints.display();
});

Button help = new Button("Help");
help.setOnAction(e -> {
    getHelp();
});

HBox h = new HBox();
h.setAlignment(Pos.BOTTOM_RIGHT);

```

```

        h.getChildren().addAll(help, report);
        h.setSpacing(5);

        VBox textfields = new VBox(10); //Creating Textfield
        textfields.setAlignment(Pos.BASELINE_CENTER);
        textfields.setPadding(new Insets(10, 10, 10, 10));
        textfields.getChildren().addAll(menu, h1, h2, h3, h4);
        textfields.setSpacing(21);

        VBox menuButtons = new VBox(); //Creating VBox
        menuButtons.setAlignment(Pos.BASELINE_CENTER);
        menuButtons.setPadding(new Insets(15, 15, 15, 15));

        menuButtons.getChildren().addAll(addToCart, showBalance, topUp, exit, h)
;
        menuButtons.setSpacing(10);

        Insets insets = new Insets(10);

        BorderPane borderpane = new BorderPane(); //Creating BorderPane
        borderpane.setTop(textfields);
        BorderPane.setMargin(textfields, insets);
        borderpane.setCenter(menuButtons);
        BorderPane.setMargin(menuButtons, insets);
        borderpane.setBottom(h);
        borderpane.setPadding(new Insets(6));

        Scene sceneD = new Scene(borderpane, 600, 540);
        sceneD.getStylesheets().add("style.css");
        window4.setTitle("Drinks Order");
        window4.setScene(sceneD);
        window4.show();
    }

    public static double totals() { //Method to calculate and return
total Cost
        totalCost = priceCoke+priceFanta+priceSprite+pricePepsi;

        return totalCost;
    }

```

```
        public static double Transaction() { //method to add total cost
to revenue and deduct from Account balance
```

```
        if(TopUp.accountBalance>=totalCost) { //Checking if the total
cost is less than whats in Account Balance
        Revenue+=totalCost;
```

```
        return Revenue;
```

```
    }
    else {
        CreditCard.getErrorMessage();
        outOfMoney();
    }
```

```
    return Revenue;
}
```

```
        public static void outOfMoney() { //An alert to show when the order
total is greater than account balance
```

```
        Alert alert1 = new Alert(AlertType.ERROR);
        alert1.setTitle("Validation Box");
        alert1.setHeaderText("Error not enough money in your account!
");
```

```
        alert1.setContentText("Please Top Up! ");
        alert1.showAndWait();
    }
```

```
        public static void outOfStockAlert() { //Alert to show when a drink
is out of stock
```

```
        Alert alert1 = new Alert(AlertType.ERROR);
        alert1.setTitle("Validation Box");
        alert1.setHeaderText("Error! ");
        alert1.setContentText("Out of Stock! ");
```

```
        alert1.showAndWait();
    }
```

```
        public static Alert getHelp() { //help alert
```



```

        Alert alert = new Alert(AlertType.INFORMATION);
        alert.setTitle("Help Box");
        alert.setHeaderText("Hints and Tips");
        alert.setContentText("Please Enter the quantity of drinks you
want in the textfield beside it and press add\n"
+ "1. Once you are ready to check out press
Complete Order\n"
+ "2. If you wish to see how much money is in your
account press Show Balance\n"
+ "3. Enter the 3 digit security code found at the
back\n"
+ "4. If you wish to top up your account click Top
Up\n"
+ "5. If you wish to exit the program press the
Exit button"
        );
        alert.showAndWait();

        return alert;
    }
}

```

```

import javafx.geometry.Insets;
import javafx.geometry.Pos;
import javafx.scene.Scene;

```

```
import javafx.scene.control.Alert;
import javafx.scene.control.Button;
import javafx.scene.control.Label;
import javafx.scene.control.Alert.AlertType;
import javafx.scene.layout.BorderPane;
import javafx.scene.layout.VBox;
import javafx.scene.text.Font;
import javafx.stage.Modality;
import javafx.stage.Stage;
```

```
public class ConfirmPurchase extends AccountdrinksMenu{
```

```
    static boolean answer;
```

```
    public static boolean display(Double totalCost) {
```

```
        Stage window = new Stage();
```

```
        window.initModality(Modality.APPLICATION_MODAL);
```

```
        window.setTitle("Payment");
```

```
        window.setMinWidth(250);
```

```
        Label Total = new Label(); //Label to display order total
```

```
        Total.setText("Your total is: " + totals());
```

```
        Total.setAlignment(Pos.CENTER);
```

```
        Total.setFont(new Font("Arial", 13));
```

```
Total.setPadding(new Insets(13,10,5,5));
```

```
Label doYouWish = new Label(); //Creating Label
```

```
doYouWish.setText("Do you wish to Complete Purchase?");
```

```
doYouWish.setFont(new Font("Arial", 13));
```

```
//Making my Yes and No Buttons
```

```
Button yesButton = new Button("Yes");
```

```
Button noButton = new Button("No");
```

```
yesButton.setOnAction(e -> { //Yes button which will return true if clicked
```

```
    answer=true;
```

```
    window.close();
```

```
});
```

```
noButton.setOnAction(e -> { //No button which will return false is clicked
```

```
    answer=false;
```

```
    window.close();
```

```
});
```

```
Button help = new Button("Help");
```

```
help.setOnAction(e -> {
```

```
    getHelp();
```

```
});
```

```
VBox buttons = new VBox();
```

```

        buttons.getChildren().addAll(yesButton, noButton, help);

        buttons.setSpacing(5);

        buttons.setAlignment(Pos.CENTER);


        Insets insets = new Insets(10);


        BorderPane borderpane = new BorderPane();

        borderpane.setTop(Total);

        BorderPane.setAlignment(Total, Pos.CENTER);

        borderpane.setCenter(doYouWish);

        BorderPane.setMargin(doYouWish, insets);

        BorderPane.setAlignment(doYouWish, Pos.CENTER);

        borderpane.setBottom(buttons);

        BorderPane.setMargin(buttons, insets);


        borderpane.setPadding(new Insets(6));


        Scene scene = new Scene(borderpane, 300, 250);

        scene.getStylesheets().add("style.css");

        window.setScene(scene);

        window.showAndWait();


        return answer;

    }

    public static Alert getHelp() { //Get help\s

```

```

Alert alert = new Alert(AlertType.INFORMATION);

alert.setTitle("Help Box");

alert.setHeaderText("Hints and Tips");

alert.setContentText("Please click one of the Button to continue\n"
                    + "1. Yes if you want to complete the purchase\n"
                    + "2. No if you want to cancel and exit the program"
                    );

alert.showAndWait();

return alert;
}

}

```

```

import javafx.geometry.Insets;

import javafx.geometry.Pos;

import javafx.scene.Scene;

import javafx.scene.control.Alert;

import javafx.scene.control.Button;

import javafx.scene.control.Label;

import javafx.scene.control.TextField;

import javafx.scene.control.Alert.AlertType;

import javafx.scene.layout.BorderPane;

import javafx.scene.layout.HBox;

import javafx.scene.layout.VBox;

import javafx.scene.text.Font;

import javafx.stage.Modality;

import javafx.stage.Stage;

public class Complaints {

    //This is the String where the report will be stored

    private static String Complaint;

    public static void display() {

        Stage window = new Stage();

        window.initModality(Modality.APPLICATION_MODAL);

        window.setTitle("Report");

```

```

Label reportlbl = new Label();//Creating Label
reportlbl.setText("Report Menu");
reportlbl.setFont(new Font("Arial", 18));
reportlbl.setPadding(new Insets(16,10,5,5));

```

```

Label enterlbl = new Label(); //Creating label
enterlbl.setText("Type your Complaint Here: ");
enterlbl.setFont(new Font("Arial", 13));
TextField complaint = new TextField(); //Creating textfield to enter complaint
complaint.setPromptText("Type here");

```

```

Button submit = new Button("Submit");
submit.setOnAction(e -> {

```

```

    if(CreateAccount.isAlphabetic(complaint)) { // if statement checking if a
string was entered

```

```

        Complaint = complaint.getText(); //Making whats entered in the
textbox a string

```

```

        CreateAccount.getConfirmationBox();

```

```

    }

```

```

    else {

```

```

        CreateAccount.getErrorMessage();

```

```

    }

```

```

});

```

```

HBox h = new HBox();
h.getChildren().addAll(enterlbl,complaint);
h.setSpacing(5);

```

```
Button help = new Button("Help"); //help button
help.setOnAction(e -> {
    getHelp();
});
```

```
Button exit = new Button("Exit");//button to exit back
exit.setOnAction( e -> {
    window.close();
});
exit.setPrefWidth(70);
```

```
HBox bottom = new HBox();
bottom.getChildren().addAll(exit,help);
bottom.setAlignment(Pos.BOTTOM_LEFT);
bottom.setSpacing(6);
```

```
VBox textfields = new VBox(10);
textfields.setAlignment(Pos.BASELINE_CENTER);
textfields.setPadding(new Insets(10,10,10,10));
textfields.getChildren().addAll(h,submit);
textfields.setSpacing(18);
```



```
Insets insets = new Insets(10);

BorderPane borderpane = new BorderPane();
borderpane.setTop(reportlbl);
BorderPane.setAlignment(reportlbl, Pos.CENTER);
borderpane.setCenter(textfields);
BorderPane.setMargin(textfields, insets);
borderpane.setBottom(bottom);
borderpane.setPadding(new Insets(6));
```

```
Scene scene = new Scene(borderpane ,400 , 300);
scene.getStylesheets().add("style.css");
window.setScene(scene);
window.showAndWait();
```

```
}
```

```
public static void getHelp() {
    Alert alert = new Alert(AlertType.INFORMATION);
```

```
        alert.setTitle("Help Box");

        alert.setHeaderText("Hints and Tips");

        alert.setContentText("Please Enter the compaint in the box and press Submit");

        alert.showAndWait();
```

```
    }
```

```
    //getter for the string report
```

```
    public static String getComplaint() {

        return Complaint;

    }
```

```
    //setter for the string report
```

```
    public static void setComplaint(String report) {

        Complaint = getComplaint();

    }
```

```
}
```

```
import javafx.geometry.Insets;
import javafx.geometry.Pos;
import javafx.scene.Scene;
import javafx.scene.control.Alert;
```

```

import javafx.scene.control.Button;
import javafx.scene.control.Label;
import javafx.scene.control.TextField;
import javafx.scene.control.Alert.AlertType;
import javafx.scene.layout.BorderPane;
import javafx.scene.layout.HBox;
import javafx.scene.layout.VBox;
import javafx.scene.text.Font;
import javafx.stage.Modality;
import javafx.stage.Stage;

```

```

public class GuestDrinksMenu extends AccountdrinksMenu {

```

```

    //This is the quantity of drinks bought
    static TextField amountCoke;
    static TextField amountFanta;
    static TextField amountSprite;
    static TextField amountPepsi;
    static TextField moneyEnter;
    int quantity;
    static Button btnSelect;
    Button btnClear;
    Label lblcoke, lblfanta, lblsprite, lblpepsi;
    static double price = 1.20;
    static int CokesBought;
    static int FantasBought;
    static int SpritesBought;
    static int PepsisBought;
    static double priceCoke;
    static double priceFanta;
    static double priceSprite;
    static double pricePepsi;
    static double totalDrinks;
    static double totalCost;
    static double MoneyEnter;
    static boolean answer;
    static double ChangeLeft;

```

```

    public static void display() {

```

```

        Stage window4 = new Stage();
        window4.setTitle("Drinks Menu");

```

```

window4.initModality Modality.APPLICATION_MODAL);

//main label
Label guestlbl = new Label();
guestlbl.setText("Guest Menu");
guestlbl.setFont(new Font("Arial", 19));
guestlbl.setPadding(new Insets(23,10,5,5));

//For entering money
Label lblEnter = new Label("Please Enter Money: ");
lblEnter.setFont(new Font("Arial", 13));
moneyEnter = new TextField();
moneyEnter.setPromptText("Enter Amount");

HBox m =new HBox();
m.setSpacing(6);

Button verifyMoney = new Button("Verify");
verifyMoney.setOnAction(e-> {
    if(ValidateMoney(moneyEnter)) {
        getConfirm();
    }
    else {
        moreNeed();
    }
});

HBox h = new HBox();
h.getChildren().addAll(lblEnter, moneyEnter, verifyMoney);
h.setSpacing(6);
h.setAlignment(Pos.CENTER);

//For Buying a Coke
Label lblcoke = new Label(String.format("    Coke: " +
"Price: €" + price + "    "));
lblcoke.setFont(new Font("Arial", 13));
amountCoke = new TextField();
amountCoke.setPromptText("Quantity of Coke");
Button verifyCoke = new Button("Add");
verifyCoke.setOnAction(e ->{
    CokesBought =
Integer.parseInt(amountCoke.getText()); //parsing cokes bought into an int
    if(CokesBought<=AccountdrinksMenu.coke) { //If statement
to see that cokes ordered isnt more than whats in stock
        AccountdrinksMenu.coke-=CokesBought; //deducting
cokes bought from cokes in stock
        priceCoke = CokesBought*price; //finding the
price of all cokes bought
        CreditCard.getConfirmationBox();

```

```

        System.out.println("\nCoke bought: " +
CokesBought);
    }
    else {
        AccountdrinksMenu.outOfStockAlert(); //alert to
show when the drink wanting to be ordered is out of stock
    }

});
HBox h1 = new HBox();
h1.getChildren().addAll(lblcoke, amountCoke, verifyCoke);
h1.setSpacing(6);
h1.setAlignment(Pos.CENTER);

//For buying a Fanta
Label lblfanta = new Label String.format(" Fanta: " +
"Price: €" + price + " ");
lblfanta.setFont(new Font("Arial", 13));
amountFanta = new TextField();
amountFanta.setPromptText("Quantity of Fanta");

Button verifyFanta = new Button("Add");
verifyFanta.setOnAction(e ->{
    FantasBought = Integer.parseInt(amountFanta.getText());
    if (FantasBought <= AccountdrinksMenu.fanta) { //Same as
coke
        AccountdrinksMenu.fanta -= FantasBought;
        priceFanta = FantasBought * price;
        CreditCard.getConfirmationBox();
        System.out.println("\nFanta bought: " +
FantasBought);
    }
    else {
        outOfStockAlert();
    }
});
HBox h2 = new HBox();
h2.getChildren().addAll(lblfanta, amountFanta, verifyFanta);
h2.setSpacing(6);
h2.setAlignment(Pos.CENTER);

//For buying a Sprite
Label lblsprite = new Label String.format(" Sprite: " +
"Price: €" + price + " ");
lblsprite.setFont(new Font("Arial", 13));
amountSprite = new TextField();
amountSprite.setPromptText("Quantity of Sprite");

```

```

        Button verifySprite = new Button("Add");
        verifySprite.setOnAction(e ->{
            SpritesBought =
Integer.parseInt(amountSprite.getText());
            if (SpritesBought<=AccountdrinksMenu.sprite) { //same as
coke

                AccountdrinksMenu.sprite-=SpritesBought;
                priceSprite = SpritesBought*price;
                CreditCard.getConfirmationBox();
                System.out.println("\nSprite bought: " +
SpritesBought);
            }
            else {
                outOfStockAlert();
            }

        });
        HBox h3 = new HBox();
        h3.getChildren().addAll(lblsprite, amountSprite,
verifySprite);
        h3.setSpacing(6);
        h3.setAlignment(Pos.CENTER);

        //For buying Pepsi
        Label lblpepsi = new Label(String.format(" Pepsi: " +
"Price: €" + price + " "));
        lblpepsi.setFont(new Font("Arial", 13));
        amountPepsi= new TextField();
        amountPepsi.setPromptText("Quantity of Pepsi");
        Button verifyPepsi = new Button("Add");
        verifyPepsi.setOnAction(e ->{

            PepsisBought = Integer.parseInt(amountPepsi.getText());
            if (PepsisBought<=AccountdrinksMenu.pepsi) { //same as
coke

                AccountdrinksMenu.pepsi -=PepsisBought;
                pricePepsi = PepsisBought*price;
                CreditCard.getConfirmationBox();
                System.out.println("\nPepsi bought: " +
PepsisBought);
            }
            else {
                outOfStockAlert();
            }

        });

        HBox h4 = new HBox();
        h4.getChildren().addAll(lblpepsi, amountPepsi, verifyPepsi);

```

```

h4.setSpacing(6);
h4.setAlignment(Pos.CENTER);

Button report = new Button("Report");
report.setOnAction(e -> {
    Complaints.display();
});
report.setPrefWidth(100);

```

```

Button addToCart = new Button("Complete order"); //Button to
press when finished ordering
addToCart.setOnAction(e -> {
    if(MoneyEntered>=totalCost) { //checking if the money
eneter is equal or more than the order
        NoChange.display(); //Showing a method to confirm
if order wants to be finished
        totals();
        Transaction(totalCost);
    }

    else {
        System.out.println("Not enough money Entered");
    }
});
addToCart.setPrefWidth(100);

```

```

stage
Button cancel = new Button("Cancel"); //button to exit the

cancel.setOnAction(e -> {
    window4.close();
});

```

```

Button help = new Button("Help");
help.setOnAction(e -> {
    getHelp();
});

```

```

HBox bottom = new HBox();
bottom.getChildren().addAll(cancel,help);
bottom.setSpacing(5);

```

```

VBox v2 = new VBox(10);
v2.setAlignment(Pos.CENTER);
v2.setPadding(new Insets(10,10,10,10));
v2.getChildren().addAll(h, h1, h2, h3, h4, addToCart,report);
v2.setSpacing(17);

```

```

Insets insets = new Insets(10);

BorderPane borderpane = new BorderPane();
borderpane.setTop(guestlbl);
BorderPane.setMargin(guestlbl, insets);
BorderPane.setAlignment(guestlbl, Pos.CENTER);
borderpane.setCenter(v2);
BorderPane.setMargin(v2, insets);
BorderPane.setAlignment(v2, Pos.CENTER);
borderpane.setBottom(bottom);
BorderPane.setAlignment(bottom, Pos.BOTTOM_LEFT);
borderpane.setPadding(new Insets(8));

Scene sceneD = new Scene(borderpane, 500, 500);

window4.setTitle("Drinks Order");
sceneD.getStylesheets().add("style.css");
window4.setScene(sceneD);
window4.show();
}

public static void moreNeed() { //Alert saying not enough money was
entered to purchase one drink

    System.out.println("Entry Error");
    Alert alert1 = new Alert(AlertType.ERROR);
    alert1.setTitle("Validation Box");
    alert1.setHeaderText("Money Entry result!");
    alert1.setContentText("You have not entered enough of the
correct currency! ");
    alert1.showAndWait();
}

public static void getConfirm() { //confirm box
    System.out.println("Entry Successful");
    Alert alert1 = new Alert(AlertType.CONFIRMATION);
    alert1.setTitle("Validation Box");
    alert1.setHeaderText("You have Entered enough money to buy a
drink");
    alert1.setContentText("Warning this machine does not dispense
any change press help for more info");
    alert1.showAndWait();
}

```



```

    }

    public static double totals() { //Method to find the total cost
        totalCost = priceCoke+priceFanta+priceSprite+pricePepsi;
        System.out.println("\nTotal Cost: " + totalCost);
        return totalCost;
    }

    public static void Transaction(double totalCost) { //Method to
conduct Transaction

        System.out.println("Total cost is: " + totalCost);

        if(MoneyEnter>=totalCost) {
            AccountdrinksMenu.Revenue+=MoneyEnter; //Adding total
cost to revenue
            ChangeLeft = MoneyEnter-totalCost; //This is how much
change was left by the machine
            System.out.println("Change Left: " + ChangeLeft);
            System.out.println("Enjoy your drinks");
        }
        else {
            CreditCard.getErrorMessage();
        }
    }

    public static boolean isDouble(TextField text){ //method to see if a
Double was entered
        try {
            MoneyEnter = Double.parseDouble(text.getText());
            return true;
        }
        catch (NumberFormatException e) {
            getErrorMessage();
            throw new NumberFormatException("Invalid Input!");
        }
    }
}

```



```

import javafx.scene.control.Alert;
import javafx.scene.control.Button;
import javafx.scene.control.Label;
import javafx.scene.control.Alert.AlertType;
import javafx.scene.layout.BorderPane;
import javafx.scene.layout.HBox;
import javafx.scene.layout.VBox;
import javafx.scene.text.Font;
import javafx.stage.Modality;
import javafx.stage.Stage;

public class NoChange {

    static boolean answer;

    public static boolean display (){

        Stage window = new Stage();

        window.initModality(Modality.APPLICATION_MODAL);

        window.setTitle("Payment");

        window.setMinWidth(250);

        Label warning = new Label(); //Creating Label

        warning.setText("This Machine does NOT dispense Change");

        warning.setFont(new Font("Arial", 15));

        HBox h = new HBox();

        h.getChildren().add(warning);
    }
}

```

```
h.setAlignment(Pos.CENTER);  
h.setPadding(new Insets(13,18,10,10));
```

```
Label doYouWish = new Label();  
doYouWish.setText("Do you Wish to complete the Purchase");
```

```
//Making my Yes and No Buttons
```

```
Button yesButton = new Button("Yes");
```

```
Button noButton = new Button("No");
```

```
yesButton.setOnAction(e -> { //yes button that will return true when clicked  
    answer=true;  
    window.close();  
});
```

```
noButton.setOnAction(e -> { //no button that will return false when clicked  
    answer=false;  
    getHelp();  
    window.close();  
});
```

```
VBox middle = new VBox(); //Creating vbox
```

```
middle.getChildren().addAll(doYouWish,yesButton,noButton);  
  
middle.setSpacing(12);  
  
middle.setAlignment(Pos.CENTER);
```

```
Button help = new Button("Help"); //Creating help Button  
help.setOnAction(e -> {  
    getHelp();  
});
```

```
Insets insets = new Insets(10);
```

```
BorderPane borderpane = new BorderPane();//Creating BorderPane  
borderpane.setTop(warning);  
BorderPane.setAlignment(warning, Pos.CENTER);  
borderpane.setCenter(middle);  
BorderPane.setMargin(middle, insets);  
borderpane.setBottom(help);  
BorderPane.setMargin(help, insets);  
  
borderpane.setPadding(new Insets(6));
```

```
Scene scene = new Scene(borderpane , 400, 250);  
window.setScene(scene);  
scene.getStylesheets().add("style.css");  
window.showAndWait();
```

```

        return answer;

    }

    public static Alert getHelp() { //Get help alert

        Alert alert = new Alert(AlertType.INFORMATION);

        alert.setTitle("Help Box");

        alert.setHeaderText("Hints and Tips");

        alert.setContentText("If you have entered money it cannot be return you can now
either\n"

                                + "1. Press No close the program and forfeit your money\n"
                                + "2.Press Yes to complete the order and get your drinks"
                                + "Thank you for Shopping with us"

                                );

        alert.showAndWait();

        return alert;

    }

```

```

}

```

```

import javafx.geometry.Insets;
import javafx.geometry.Pos;
import javafx.scene.Scene;

```

```

import javafx.scene.control.Alert;
import javafx.scene.control.Button;
import javafx.scene.control.Label;
import javafx.scene.control.Alert.AlertType;
import javafx.scene.layout.BorderPane;
import javafx.scene.layout.HBox;
import javafx.scene.layout.VBox;
import javafx.scene.text.Font;
import javafx.stage.Modality;
import javafx.stage.Stage;

public class Manager {

    public static void display() {

        Stage window = new Stage();
        window.setTitle("Manger Menu");
        window.initModality(Modality.APPLICATION_MODAL);

        Label Managerlbl = new Label();
        Managerlbl.setText("Managers Menu");
        Managerlbl.setFont(new Font("Arial", 16));
        Managerlbl.setPadding(new Insets(50,10,10,10));

        Button help = new Button("Help");
        help.setOnAction(e -> {
            getHelp();
        });

        Button cancel = new Button("Cancel");
        cancel.setOnAction(e -> {
            window.close();
        });

        Button report = new Button("Report");
        report.setOnAction(e ->{
            Report.display();
        });
        report.setPrefWidth(150);

        Button maintain = new Button("Maintenance");
        maintain.setOnAction(e -> {
            Maintenance.display();
        });
        maintain.setPrefWidth(150);
    }
}

```

```

Button Money = new Button("Retrieve Revenue");
Money.setOnAction(e -> {
    money.display();
});
Money.setPrefWidth(150);

Button repairLog = new Button("Repair Log");
repairLog.setOnAction(e -> {
    RepairLog.display();
});
repairLog.setPrefWidth(150);

VBox v = new VBox();
v.getChildren().addAll(report, maintain, Money, repairLog);
v.setAlignment(Pos.CENTER);
v.setPadding(new Insets(10, 10, 10, 10));
v.setSpacing(15);

HBox bottom = new HBox();
bottom.getChildren().addAll(cancel, help);
bottom.setAlignment(Pos.BOTTOM_LEFT);
bottom.setSpacing(6);

Insets insets = new Insets(10);

BorderPane borderpane = new BorderPane();
borderpane.setTop(Managerlbl);
BorderPane.setAlignment(Managerlbl, Pos.CENTER);
borderpane.setCenter(v);
BorderPane.setMargin(v, insets);
borderpane.setBottom(bottom);
borderpane.setPadding(new Insets(6));

Scene sceneD = new Scene(borderpane, 400, 400);
sceneD.getStylesheets().add("style.css");
window.setTitle("Drinks Order");
window.setScene(sceneD);
window.show();

}

public static Alert getHelp() {

```



```

        Alert alert = new Alert(AlertType.INFORMATION);
        alert.setTitle("Help Box");
        alert.setHeaderText("Hints and Tips");
        alert.setContentText("Please Press one the buttons Available
add\n"
                                + "1. Press report to read any compaints made by
Customers\n"
                                + "2. Press the Add Drinks button to add drinks
into the vending machine\n"
                                + "3. Press Retrieve Money to take Revenue out of
the machine\n"
                                + "4. Press Cancel to return to the login
screen\n"
                                );
        alert.showAndWait();

        return alert;
    }
}

```

```

import javafx.geometry.Insets;
import javafx.geometry.Pos;
import javafx.scene.Scene;
import javafx.scene.control.Alert;

```

```

import javafx.scene.control.Button;
import javafx.scene.control.Label;
import javafx.scene.control.Alert.AlertType;
import javafx.scene.layout.BorderPane;
import javafx.scene.layout.HBox;
import javafx.scene.layout.VBox;
import javafx.scene.text.Font;
import javafx.stage.Modality;
import javafx.stage.Stage;

public class Report {

    public static void display() {

        Stage window6 = new Stage();
        window6.setTitle "Report Menu";
        window6.initModality Modality.APPLICATION_MODAL;

        ///Top label
        Label drinksSoldlbl = new Label(); //Creating Label
        drinksSoldlbl.setText "Each Drink Sold";
        drinksSoldlbl.setFont new Font("Arial", 19));

        HBox mainlbl = new HBox(); //creating HBox
        mainlbl.getChildren().addAll(drinksSoldlbl);
        mainlbl.setAlignment Pos.CENTER);

        Label cokeSold = new Label(); //Creating Label
        cokeSold.setText "Coke Sold: " +
(AccountdrinksMenu.CokeBought+GuestDrinksMenu.CokesBought));
        cokeSold.setFont new Font("Arial", 16));
        HBox h1 =new HBox();
        h1.getChildren().add(cokeSold);
        h1.setSpacing 6);
        h1.setAlignment Pos.CENTER);

        Label fantaSold = new Label(); //Creating Label
        fantaSold.setText "Fanta Sold: " +
(AccountdrinksMenu.FantaBought+GuestDrinksMenu.FantasBought));
        fantaSold.setFont new Font("Arial", 16));
        HBox h2 =new HBox();
        h2.getChildren().add(fantaSold);
        h2.setSpacing 6);
        h2.setAlignment Pos.CENTER);

        Label spriteSold = new Label(); //Creating Label
        spriteSold.setText "Sprite Sold: " +
(AccountdrinksMenu.SpriteBought+GuestDrinksMenu.SpritesBought));

```

```

spriteSold.setFont(new Font("Arial", 16));
HBox h3 = new HBox();
h3.getChildren().add(spriteSold);
h3.setSpacing(6);
h3.setAlignment(Pos.CENTER);

Label pepsiSold = new Label();//Creating Label
pepsiSold.setText("Pepsi Sold: " +
(AccountdrinksMenu.PepsiBought+GuestDrinksMenu.PepsisBought));
pepsiSold.setFont(new Font("Arial", 16));
HBox h4 = new HBox();
h4.getChildren().add(pepsiSold);
h4.setSpacing(6);
h4.setAlignment(Pos.CENTER);

//Creating Label
Label Rev = new Label();
Rev.setText("Revenue: " + AccountdrinksMenu.Revenue);
Rev.setFont(new Font("Arial", 16));
HBox h5 = new HBox();
h5.setSpacing(6);
h5.getChildren().add(Rev);
h5.setAlignment(Pos.CENTER);

Label Change = new Label();//Creating Label
Change.setText("Change Left: " + GuestDrinksMenu.ChangeLeft);
Change.setFont(new Font("Arial", 16));
HBox h6 = new HBox();
h6.setSpacing(6);
h6.getChildren().add(Change);
h6.setAlignment(Pos.CENTER);

Label moneyRev = new Label();//Creating Label
moneyRev.setText("Revenue Removed: " + money.RevenueRemoved );
moneyRev.setFont(new Font("Arial", 16));
HBox h7 = new HBox();
h7.setSpacing(6);
h7.getChildren().add(moneyRev);
h7.setAlignment(Pos.CENTER);

Label NoUsers = new Label();//Creating Label
NoUsers.setText("Number of Users: 1");
NoUsers.setFont(new Font("Arial", 16));
HBox h8 = new HBox();
h8.setSpacing(6);
h8.getChildren().add(NoUsers);
h8.setAlignment(Pos.CENTER);

VBox middle = new VBox();//Creating Label
middle.getChildren().addAll(h1, h2, h3, h4, h5, h6, h7, h8);

```

```

        middle.setSpacing(10);

        Button help = new Button("Help");
        help.setOnAction(e -> {
            getHelp();
        });

        Button back = new Button("Back");
        back.setOnAction(e -> {
            window6.close();
        });

        HBox bottom = new HBox();
        bottom.getChildren().addAll(back, help);
        bottom.setSpacing(6);

        Insets insets = new Insets(10);

        BorderPane borderpane = new BorderPane();
        borderpane.setTop(mainlbl);
        BorderPane.setMargin(mainlbl, insets);
        borderpane.setCenter(middle);
        BorderPane.setMargin(middle, insets);
        borderpane.setBottom(bottom);
        borderpane.setPadding(new Insets(6));

        Scene sceneE = new Scene(borderpane, 400, 400);
        sceneE.getStylesheets().add("style.css");
        window6.setScene(sceneE);
        window6.show();
    }
}

```

```

    public static Alert getHelp() { //get Help Alert
        Alert alert = new Alert(AlertType.INFORMATION);
        alert.setTitle("Help Box");
        alert.setHeaderText("Hints and Tips");
        alert.setContentText("If you wish to exit this report press
the back button");
        alert.showAndWait();
        return alert;
    }
}
import javafx.geometry.Insets;
import javafx.geometry.Pos;
import javafx.scene.Scene;
import javafx.scene.control.Alert;

```

```

import javafx.scene.control.Button;
import javafx.scene.control.Label;
import javafx.scene.control.TextField;
import javafx.scene.control.Alert.AlertType;
import javafx.scene.layout.BorderPane;
import javafx.scene.layout.HBox;
import javafx.scene.layout.VBox;
import javafx.scene.text.Font;
import javafx.stage.Modality;
import javafx.stage.Stage;

public class Maintenance {
    static int CokeAdded;
    static int FantaAdded;
    static int SpriteAdded;
    static int PepsiAdded;

    public static void display () {

        Stage window = new Stage();
        window.setTitle "Maintenance";
        window.initModality Modality.APPLICATION_MODAL;

        Label drinkStock = new Label();//creating a label
        drinkStock.setText("Please Enter the Quantity of drinks you
wish to add");
        drinkStock.setFont new Font("Arial", 16));

        //For adding Coke
        Label addcoke = new Label("Coke In Stock: " +
AccountdrinksMenu.coke + " ");
        TextField amountCoke = new TextField();
        amountCoke.setPrefWidth 150;
        amountCoke.setPromptText "Add Coke";
        Button addCoke = new Button("Add");
        addCoke.setOnAction(e -> {
            CokeAdded =
Integer.parseInt(amountCoke.getText()); //Changing the textfield entry
into an Int
            if( ((CokeAdded+AccountdrinksMenu.coke)<=20) ) {
//An if statement to see if the amount added and in stock are not over 20

```

```

        AccountdrinksMenu.coke=AccountdrinksMenu.coke+CokeAdded; //Adding
the Coke to the vending machine
    }
    else {
        getErrorMessage();
    }
}

});
HBox h1 = new HBox();
h1.getChildren().addAll(addcoke, amountCoke, addCoke);
h1.setSpacing(4);

//For adding Fanta
Label addfanta = new Label("Fanta In Stock: " +
AccountdrinksMenu.fanta + " ");
TextField amountFanta = new TextField();
amountFanta.setPrefWidth(150);
amountFanta.setPromptText("Add Fanta");
Button addFanta = new Button("Add");
addFanta.setOnAction(e -> {

        FantaAdded =
Integer.parseInt(amountFanta.getText());
        if( ((FantaAdded+AccountdrinksMenu.fanta)<=20) ) {

            AccountdrinksMenu.fanta=AccountdrinksMenu.fanta+FantaAdded; //Same as
Coke

        }
        else {
            getErrorMessage();
        }
    }
});
HBox h2 = new HBox();
h2.getChildren().addAll(addfanta, amountFanta, addFanta);
h2.setSpacing(4);

//For adding Sprite
Label addsprite = new Label("Sprite In Stock: " +
AccountdrinksMenu.sprite + " ");
TextField amountSprite = new TextField();
amountSprite.setPrefWidth(150);
amountSprite.setPromptText("Add Sprite");
Button addSprite = new Button("Add");
addSprite.setOnAction(e -> {

        SpriteAdded = Integer.parseInt(amountSprite.getText());
        if( ((SpriteAdded+AccountdrinksMenu.sprite)<=20) ) {

            AccountdrinksMenu.sprite=AccountdrinksMenu.sprite+SpriteAdded;
//Same as Coke

        }
    }
});

```

```

        else {
            getErrorMessage();
        }
    });
    HBox h3 = new HBox();
    h3.getChildren().addAll(addsprite, amountSprite, addSprite);
    h3.setSpacing(4);

    //For adding Pepsi
    Label addpepsi = new Label("Pepsi In Stock: " +
AccountdrinksMenu.coke + " ");
    TextField amountPepsi = new TextField();
    amountPepsi.setPrefWidth(150);
    amountPepsi.setPromptText("Add Pepsi");
    Button addPepsi = new Button("Add");

    addPepsi.setOnAction(e -> {

        PepsiAdded = Integer.parseInt(amountPepsi.getText());
        if( ((PepsiAdded+AccountdrinksMenu.pepsi)<=20) ) {

            AccountdrinksMenu.pepsi=AccountdrinksMenu.pepsi+PepsiAdded; //Same
as Coke

        }
        else {
            getErrorMessage();
        }
    });
    HBox h4 = new HBox();
    h4.getChildren().addAll(addpepsi, amountPepsi, addPepsi);
    h4.setSpacing(4);

    Button help = new Button("Help"); //Help Button
    help.setOnAction(e -> {
        getHelp();
    });

    HBox h = new HBox();
    h.setAlignment(Pos.BOTTOM_RIGHT);
    h.getChildren().addAll(help);
    h.setSpacing(5);

    Button cancel = new Button("Cancel"); //Cancel Button
    cancel.setOnAction(e -> {
        window.close();
    });

```

```

        Button complaint = new Button("Complaints");
        complaint.setOnAction(e ->{
            getComLaint();
        });

        HBox bottom = new HBox();
        bottom.getChildren().addAll(cancel, help, complaint);
        bottom.setAlignment(Pos.BOTTOM_LEFT);
        bottom.setSpacing(6);

        Button submit = new Button();
        submit.setText("Submit");
        submit.setOnAction(e -> {
            Manager.display();
        });

        VBox v = new VBox(10);
        v.setPadding(new Insets(10,10,10,10));
        v.getChildren().addAll(h1, h2, h3, h4, submit);

        Insets insets = new Insets(10);

        BorderPane borderpane = new BorderPane();
        borderpane.setTop(drinkStock);
        BorderPane.setAlignment(drinkStock, Pos.CENTER);
        BorderPane.setMargin(drinkStock, insets);
        borderpane.setCenter(v);
        BorderPane.setMargin(v, insets);
        BorderPane.setAlignment(v, Pos.CENTER);
        borderpane.setBottom(bottom);
        borderpane.setPadding(new Insets(6));

        Scene sceneD = new Scene(borderpane, 500, 400);

        window.setTitle("Drinks Order");
        sceneD.getStylesheets().add("style.css");
        window.setScene(sceneD);
        window.show();
    }

    public static void getHelp() { //getHelp

        Alert alert = new Alert(AlertType.INFORMATION);
        alert.setTitle("Help Box");
        alert.setHeaderText("Hints and Tips");
    }
}

```



```

        alert.setContentText("Please Enter the quantity of drinks you
want to add in the textfield beside it and press add\n"
+ "1. Once you are done press enter to go back to
the Manager screen\n"
);
        alert.showAndWait();
    }

```

```

    public static void getErrorMessage() { //Error message alert
        Alert alert1 = new Alert(AlertType.ERROR);
        alert1.setTitle("Validation Box");
        alert1.setHeaderText("incorrect input!");
        alert1.setContentText("This Machine can hold only 20 of each
Drink ");

        alert1.showAndWait();
    }

```

```

    public static Alert getReport() { //Report alert
        Alert alert = new Alert(AlertType.INFORMATION);
        alert.setTitle("Report Box");
        alert.setHeaderText("The Report");
        alert.setContentText(Complaints.getComplaint());
        alert.showAndWait();

        return alert;
    }

```

```

    public static Alert getComlaint() {

        Alert alert = new Alert(AlertType.INFORMATION);
        alert.setTitle("Complaint Box");
        //alert.setHeaderText("Complaints");
        alert.setContentText("Cusomter complaints: " +
Complaints.getComplaint());
        alert.showAndWait();

        return alert;
    }

```

```

import javafx.geometry.Insets;
import javafx.geometry.Pos;
import javafx.scene.Scene;

```

```

import javafx.scene.control.Alert;
import javafx.scene.control.Button;
import javafx.scene.control.Label;
import javafx.scene.control.TextField;
import javafx.scene.control.Alert.AlertType;
import javafx.scene.layout.BorderPane;
import javafx.scene.layout.HBox;
import javafx.scene.layout.VBox;
import javafx.scene.text.Font;
import javafx.stage.Modality;
import javafx.stage.Stage;

public class money {

    static double RevenueRemoved; //The amount of money removed
    static TextField AmountR;
    double InMachineRev; // The amount of revenue in the machine

    public static void display() {

        Stage window7 = new Stage();
        window7.setTitle "Report Menu";
        window7.initModality Modality.APPLICATION_MODAL);

        Label RevenueLbl = new Label(); //Main Label
        RevenueLbl.setText "Revenue Removal";
        RevenueLbl.setFont new Font "Arial", 19));
        RevenueLbl.setPadding new Insets 23,13,5,5));
        RevenueLbl.setAlignment Pos.CENTER);

        Label amountRemove = new Label();
        amountRemove.setText "Enter the Amount you Wish to Remove: ";
        amountRemove.setFont new Font "Arial", 16));
        AmountR = new TextField(); //TextField to enter the
amount desired to be removed
        AmountR.setPromptText "Enter Quantity";
        Button Remove = new Button "Remove";
        Remove.setOnAction e -> {

            if (isInt(AmountR)) { //If statement to see if an int has
been entered
                RevenueRemoved =
Double.parseDouble(AmountR.getText());
            } else {
                CreditCard.getErrorMessage(); //Error message if
an Int has not been entered
            }
        }
    }
}

```

```

    }

    if (RevenueRemoved <= AccountdrinksMenu.Revenue) { //if
statement to see if the amount wanting to be taken out is less than or
equal to what's in the machine
        AccountdrinksMenu.Revenue -= RevenueRemoved;
//Deducting the revenue removed from the machine Revenue

    }
    else {
        ErrorMessage();
    }
});
HBox middle = new HBox(); //Creating HBox
middle.getChildren().addAll(amountRemove, AmountR, Remove);
middle.setAlignment(Pos.CENTER);
middle.setSpacing(6);

Button showBalance = new Button("Show Balance");
showBalance.setOnAction(e -> {
    RevInMachine(); //Button showing how much money is in
the machine
});

VBox center = new VBox();
center.getChildren().addAll(middle, showBalance);
center.setSpacing(16);
center.setAlignment(Pos.CENTER);

Button help = new Button("Help");
help.setOnAction(e -> {
    getHelp();
});

Button back = new Button("Back");
back.setOnAction(e -> {
    window7.close();
});

HBox bottom = new HBox();
bottom.getChildren().addAll(back, help);
bottom.setSpacing(6);

Insets insets = new Insets(10);

BorderPane borderpane = new BorderPane();
borderpane.setTop(Revenuelbl);
BorderPane.setAlignment(Revenuelbl, Pos.CENTER);

```

```

borderpane.setCenter(center);
BorderPane.setMargin(center, insets);
borderpane.setBottom(bottom);
borderpane.setPadding(new Insets(6));

Scene sceneF = new Scene(borderpane, 640, 400);
sceneF.getStylesheets().add("style.css");
window7.setScene(sceneF);
window7.show();

}

public static boolean isInt(TextField text){ //method checking if
its an int

    try {
        RevenueRemoved = Double.parseDouble(text.getText());
        TopUp.getConfirmationBox();
        return true;
    }
    catch (NumberFormatException e) {
        throw new NumberFormatException("Invalid Input!");
    }

}

public static void RevInMachine() { //method showing how much money
is in the machine
    double InMachineRev = AccountdrinksMenu.Revenue;
    Alert alert = new Alert(Alert.AlertType.INFORMATION);
    alert.setContentText("The Money in the Machine is : " +
InMachineRev);
    alert.show();

}

public static void getHelp() { //Help Alert

```

```

        Alert alert = new Alert(AlertType.INFORMATION);
        alert.setTitle("Help Box");
        alert.setHeaderText("Hints and Tips");
        alert.setContentText("Enter the amount you wish to retrieve
and press the remove button\n"
            + "1. Make sure you are not taking out more money
than is present by pressing the show balance button\n"
            + "2. To go back press the back Button"
        );
        alert.showAndWait();
    }
}

```

```

    public static void ErrorMessage() { //error message alert
        Alert alert1 = new Alert(AlertType.ERROR);
        alert1.setTitle("Validation Box");
        alert1.setHeaderText("incorrect input!");
        alert1.setContentText("You are trying to remove more money
than is in the machine ");
        alert1.showAndWait();
    }
}
}

```

```

import javafx.geometry.Insets;
import javafx.geometry.Pos;
import javafx.scene.Scene;

```

```

import javafx.scene.control.Alert;
import javafx.scene.control.Button;
import javafx.scene.control.Label;
import javafx.scene.control.Alert.AlertType;
import javafx.scene.layout.BorderPane;
import javafx.scene.layout.HBox;
import javafx.scene.layout.VBox;
import javafx.scene.text.Font;
import javafx.stage.Modality;
import javafx.stage.Stage;

public class RepairLog {

    public static void display() {
        Stage window8 = new Stage();
        window8.setTitle("Report Menu");
        window8.initModality(Modality.APPLICATION_MODAL);

        Label drinksAdded = new Label();
        drinksAdded.setText("Repair Log");
        drinksAdded.setFont(new Font("Arial", 19));

        HBox top = new HBox();
        top.getChildren().addAll(drinksAdded);
        top.setAlignment(Pos.CENTER);

        Label cokeAdd = new Label();

        cokeAdd.setText("Coke Added to the Machine: " +
Maintenance.CokeAdded); //How many Coke the Manager Added
        cokeAdd.setFont(new Font("Arial", 16));
        HBox h1 =new HBox();
        h1.getChildren().add(cokeAdd);
        h1.setSpacing(6);
        h1.setAlignment(Pos.CENTER);

        Label fantaAdd = new Label();
        fantaAdd.setText("Fanta Added to the Machine: " +
Maintenance.FantaAdded); //How many Fanta the Manager Added
        fantaAdd.setFont(new Font("Arial", 16));
        HBox h2 =new HBox();
        h2.getChildren().add(fantaAdd);
        h2.setSpacing(6);
        h2.setAlignment(Pos.CENTER);

        Label spriteAdd = new Label();
        spriteAdd.setText("Sprite Added to the Machine: " +
Maintenance.SpriteAdded); //How many Sprite the Manager Added

```

```

        spriteAdd.setFont(new Font("Arial", 16));
        HBox h3 =new HBox();
        h3.getChildren().add(spriteAdd);
        h3.setSpacing(6);
        h3.setAlignment(Pos.CENTER);

        Label pepsiAdd = new Label();
        pepsiAdd.setText("Pepsi Added to the Machine: " +
Maintenance.PepsiAdded); //How many Pepsi the Manager Added
        pepsiAdd.setFont(new Font("Arial", 16));
        HBox h4 =new HBox();
        h4.getChildren().add(pepsiAdd);
        h4.setSpacing(6);
        h4.setAlignment(Pos.CENTER);

        Label revrem = new Label();
        revrem.setText("Amount of Revenue Removed: " +
money.RevenueRemoved);
        revrem.setFont(new Font("Arial", 16));
        HBox h5 =new HBox();
        h5.getChildren().add(revrem);
        h5.setSpacing(6);
        h5.setAlignment(Pos.CENTER);

        VBox Middle = new VBox(10);
        Middle.setPadding(new Insets(10,10,10,10));
        Middle.getChildren().addAll(h1, h2 ,h3, h4, h5);

        Button help = new Button("Help");
        help.setOnAction(e -> {
            getHelp();
        });

        Button back = new Button("Back");
        back.setOnAction(e -> {
            window8.close();
        });

        HBox bottom = new HBox();
        bottom.setAlignment(Pos.BOTTOM_LEFT);
        bottom.getChildren().addAll(back, help);
        bottom.setSpacing(5);

        Insets insets = new Insets(10);
        BorderPane borderpane = new BorderPane();
        borderpane.setTop(top);
        BorderPane.setAlignment(top, Pos.CENTER);
        BorderPane.setMargin(top, insets);
        borderpane.setCenter(Middle);

```

```

        BorderPane.setMargin(Middle, insets);
        BorderPane.setAlignment(Middle, Pos.CENTER);
        borderpane.setBottom(bottom);
        borderpane.setPadding(new Insets(6));

        Scene scene = new Scene( borderpane, 500, 400);

        window8.setTitle("Drinks Order");
        scene.getStylesheets().add("style.css");
        window8.setScene(scene);
        window8.show();

    }

    public static void getHelp() { //Get Help Facility

        Alert alert = new Alert(AlertType.INFORMATION);
        alert.setTitle("Help Box");
        alert.setHeaderText("Hints and Tips");
        alert.setContentText("The Drink you added folow by the
quantity is Displayed\n"
                            + "1. To Go Back press the back Button\n"
);
        alert.showAndWait();

    }

}
}

```

```
import java.io.FileNotFoundException;
```



```

import java.io.IOException;
import java.io.File;
import java.io.PrintWriter;

public class pw {

    public static void write() { //Method which will write the user
input to the file
        File file = new File("C:\\Users\\milan\\OneDrive - Cork
College of
Commerce\\Fx\\VendingMachineAssignment\\src\\Registration.txt");
//Creating the new file
        System.out.println("File path is: " + new
File("C:\\Users\\milan\\OneDrive - Cork College of
Commerce\\Fx\\VendingMachineAssignment\\src\\Registration.txt").getAbsolu
tePath());

        if (file.exists()) { //if the file exist, write the message
            System.out.println("Found the file");

            try {
                file.createNewFile();
            } catch (IOException e) {
                e.printStackTrace();
            }

            PrintWriter pw = null; //Setting PrintWriter to null
            try {
                pw = new PrintWriter(file);
            } catch (FileNotFoundException e) {
                e.printStackTrace();
            }
            //Display all information in the file
            pw.print("\nUser name: ");

            pw.println(CreateAccount.FirstNameTextField().getText());

            pw.print("\nSurname: ");
            pw.println(CreateAccount.getLastName().getText());

            pw.print("Date of Birth: ");
            pw.println(CreateAccount.getDob().getText());

            pw.print("Gender: ");
            pw.println(CreateAccount.getGender().getValue());

            pw.print("Pin: ");
            pw.println(CreateAccount.getPin().getText());

            pw.print("Mobile Number: ");

```



```

import java.io.FileReader;

import java.io.IOException;

public class ReaderFile {

    //creating username and password instances

    public static String username;

    public static String pin;

    public static void readTheFile(String title) { //Method to read from the file

        BufferedReader br = null;

        try {

            br = new BufferedReader(new FileReader("C:\\Users\\milan\\OneDrive -
Cork College of Commerce\\Fx\\VendingMachineAssignment\\src\\Registration.txt")); //Create
BufferedReader in order to read the file

            String line;

            //Read through the file

            while ((line = br.readLine()) != null) {

                System.out.println(line);

                if(line.contains("User name")) username = line.split(":")[1].trim() ;

                if(line.contains("Pin")) pin = line.split(":")[1].trim() ;

            }

        } catch (IOException e) { //Throw exception if the username and password is not
found

            e.printStackTrace();

        } finally {

            try {

```

```
        br.close();
    } catch (IOException e) {
        // TODO Auto-generated catch block
        e.printStackTrace();
    }
}
}
}
```