COMSATS University Islamabad, Lahore Campus

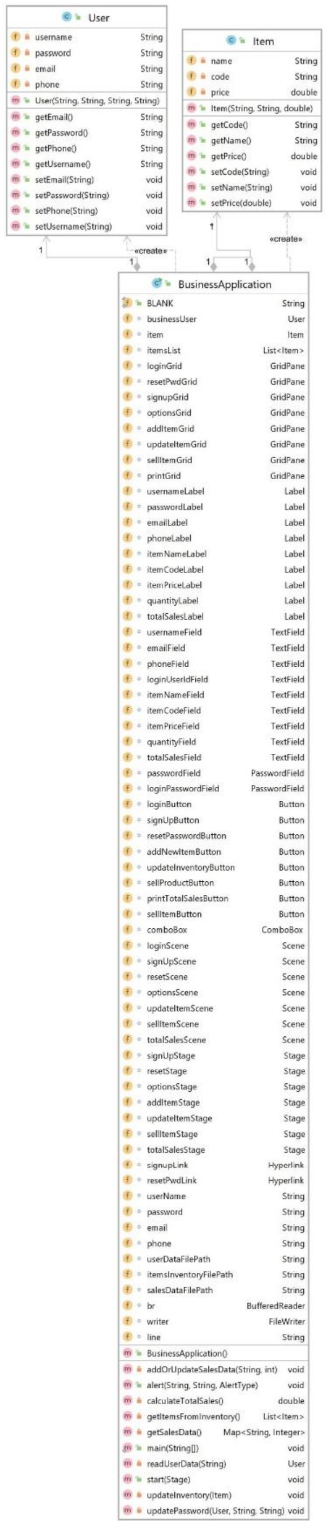
Object Oriented Programming CSC241

Assignment 02 & 03

Ahad Anwar | SP20-BCS-054

6-16-2021

UML Diagram:

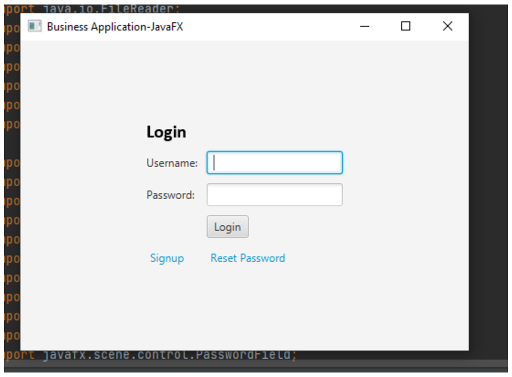

User
username String
password String
email String
phone String
User(String, String, String, String)
getEmail() String
getPassword() String
getPhone() String
getUsername() String
setEmail(String) void
setPassword(String) void
setPhone(String) void
setUsername(String) void
Item
name String
code String
price double
Item(String, String, double)
getCode() String
getName() String
getPrice() double
setCode(String) void
setName(String) void
setPrice(double) void
«create»
«create»
1
1
1
1
1
BusinessApplication
BLANK String
businessUser User
item Item
itemsList List<Item>
loginGrid GridPane
resetPwdGrid GridPane
signupGrid GridPane
optionsGrid GridPane
addItemGrid GridPane
updateItemGrid GridPane
sellItemGrid GridPane
printGrid GridPane
usernameLabel Label
passwordLabel Label
emailLabel Label
phoneLabel Label
itemNameLabel Label
itemCodeLabel Label
itemPriceLabel Label
quantityLabel Label
totalSalesLabel Label
usernameField TextField
emailField TextField
phoneField TextField
loginUserIdField TextField
itemNameField TextField
itemCodeField TextField
itemPriceField TextField
quantityField TextField
totalSalesField TextField
passwordField PasswordField
loginPasswordField PasswordField
loginButton Button
signUpButton Button
resetPasswordButton Button
addNewItemButton Button
updateInventoryButton Button
sellProductButton Button
printTotalSalesButton Button
sellItemButton Button
comboBox ComboBox
loginScene Scene
signUpScene Scene
resetScene Scene
optionsScene Scene
updateItemScene Scene
sellItemScene Scene
totalSalesScene Scene
signUpStage Stage
resetStage Stage
optionsStage Stage
addItemStage Stage
updateItemStage Stage
sellItemStage Stage
totalSalesStage Stage
signupLink Hyperlink
resetPwdLink Hyperlink
userName String
password String
email String
phone String
userDataFilePath String
itemsInventoryFilePath String
salesDataFilePath String
br BufferedReader
writer FileWriter
line String
BusinessApplication()
addOrUpdateSalesData(String, int) void
alert(String, String, AlertType) void
calculateTotalSales() double
getItemsFromInventory() List<Item>
getSalesData() Map<String, Integer>
main(String[]) void
readUserData(String) User
start(Stage) void
updateInventory(Item) void
updatePassword(User, String, String) void

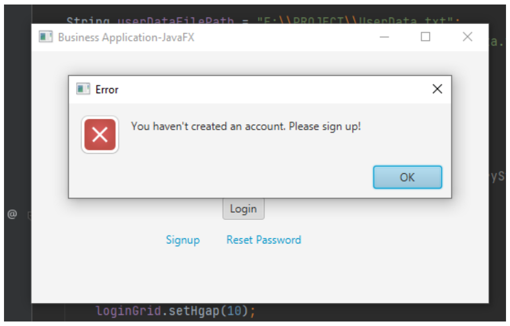
Program Description:

Following is a designed program for a Coffee Shop. This application enables Employees (referred to as Users) to automate the task of Adding Products (referred to as Items), Updating Inventory, selling products and calculating Total Sales. It also enables the Users to Sign Up multiple profiles using the Sign-Up Form and enables them to reset their password through Password Reset Form.

Login Page (Sign Up, Reset and Login Functionality)

When the Program is run, the Primary Stage i.e Login Page is opened. User (if already signed up) can sign in using their credentials and access the complete functionality. A User that has not signed up will get a prompt saying “You Haven’t Created an Account. Please SignUp”.

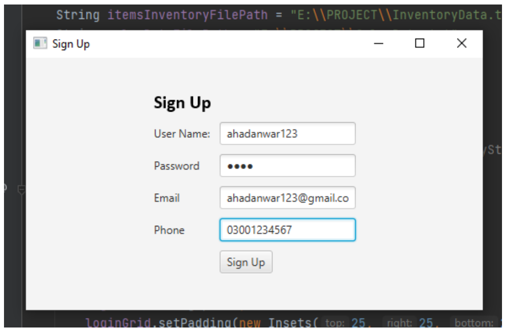




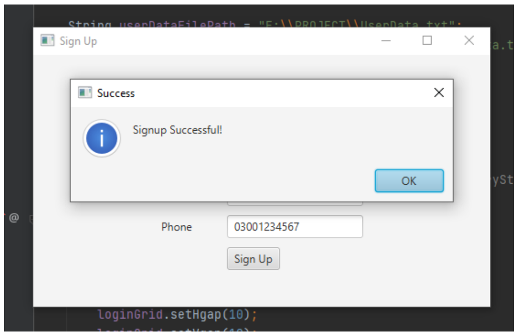
You haven't created an account. Please sign up!

Sign Up:

User can Sign Up using the Hyper Link “Signup”. The User Data is saved in a txt file named UserData.

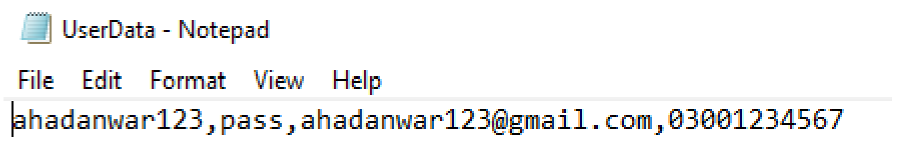


User Name: ahadanwar123
Password ••••
Email ahadanwar123@gmail.co
Phone 03001234567



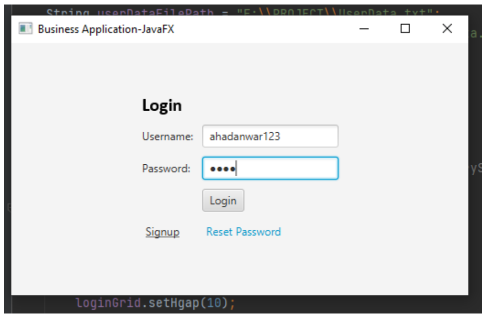
Signup Successful!

Phone 03001234567



```
ahadanwar123,pass,ahadanwar123@gmail.com,03001234567
```

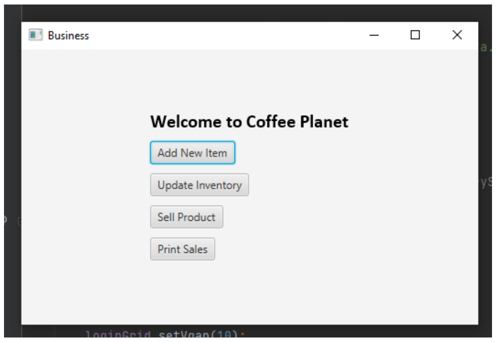
Exiting the prompt will take the user back to the Login Screen where they can now login using their credentials. If the credentials are wrong. User will get a Prompt saying their entered credentials are incorrect. If their entered credentials match with a record in UserData txt file, the user will be able to successfully login.



Username: ahadanwar123

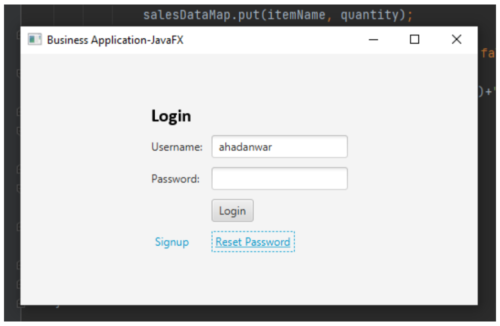
Password: ••••

```
loginGrid.setHgap(10);
```



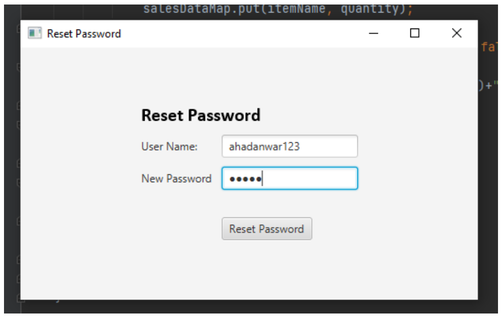
Reset Password:

User can also Reset their password by clicking on the Reset Password hyperlink that will take them to the following page where they can enter their correct username and update their password.



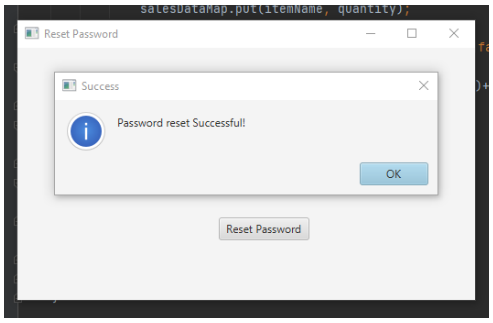
```
salesDataMap.put(itemName, quantity);
```

Username: ahadanwar



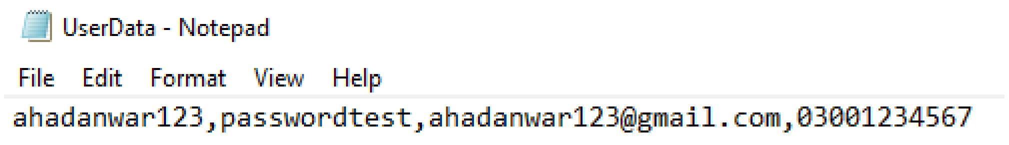
User Name: ahadanwar123

New Password: •••••



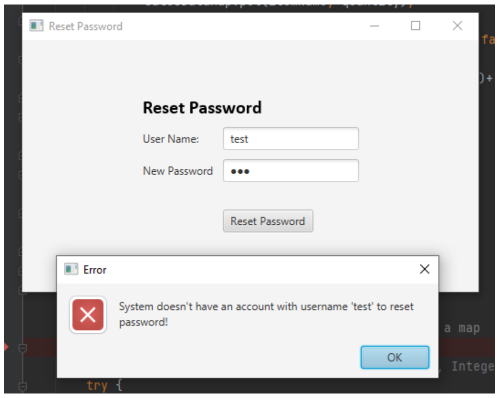
Password reset Successful!

These changed are reflected in the UserData txt file.



```
ahadanwar123,passwordtest,ahadanwar123@gmail.com,03001234567
```

If the User given username does not match with any usernames in the UserData txt file, the following error pop up will show.



Reset Password

User Name: test

New Password ●●●

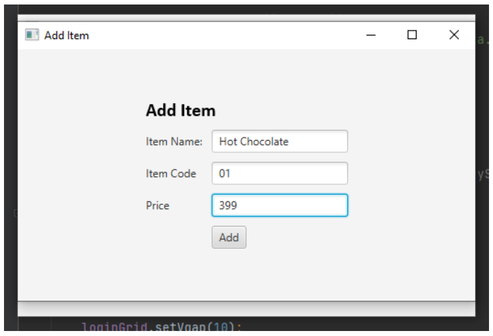
System doesn't have an account with username 'test' to reset password!

User Console (Business Activities)

Once the User has successfully logged in. They are now able to make business operations embedded in the program. These function include *Adding New Items, Updating Existing inventory, Selling Products* and *Printing Total Sales.*

Adding Items:

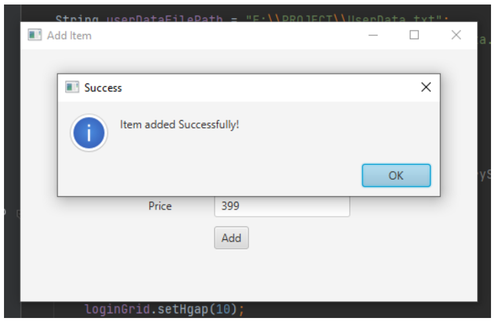
Following is the preview when User clicks on Add New Item and adds their desired item to the list.



Item Name: Hot Chocolate

Item Code: 01

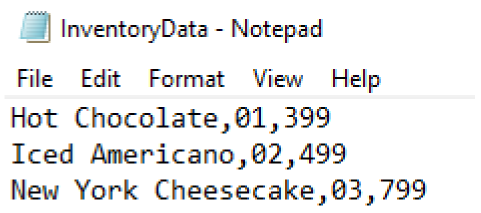
Price: 399



Item added Successfully!

Price: 399

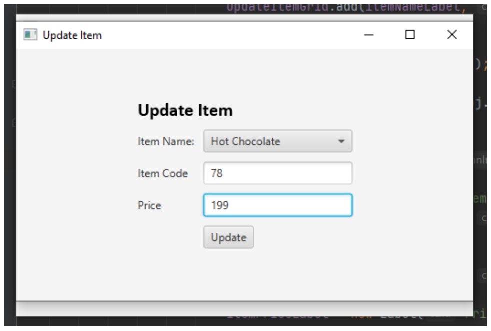
Similarly, the User can add as many Items as needed and they wil be saved in the InventoryData txt file.

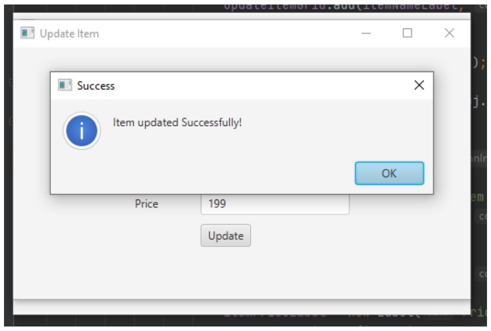


```
Hot Chocolate,01,399
Iced Americano,02,499
New York Cheesecake,03,799
```

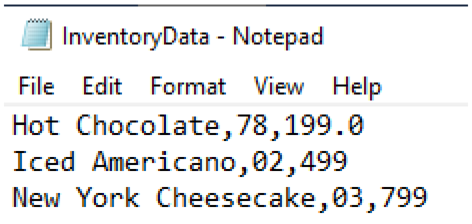
Updating Inventory:

The User can Update the existing Inventory by using the Update Inventory Functionality. This will a prompt an Updation screen with a drop down list of all available Items. The User can select their desired Item and change the details. This will update the record in the InventoryData txt file.





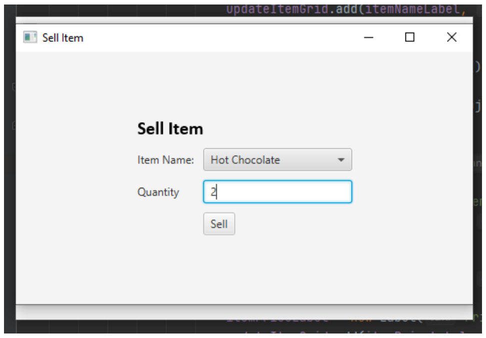
Item updated Successfully!

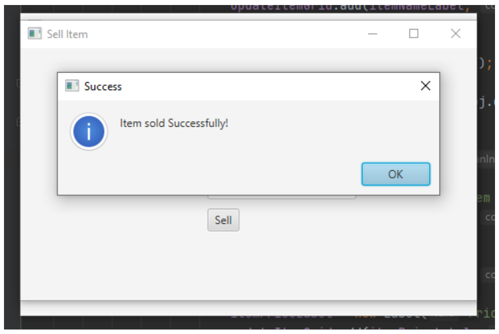


```
Hot Chocolate,78,199.0
Iced Americano,02,499
New York Cheesecake,03,799
```

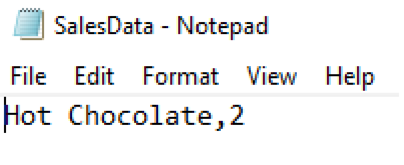
Selling Items:

User can make a transaction by selling the product. Item name and number of items sold are saved in a SalesData txt file. When another transaction for the same item takes place, it updates the number of items sold and retains the total number of items sold.





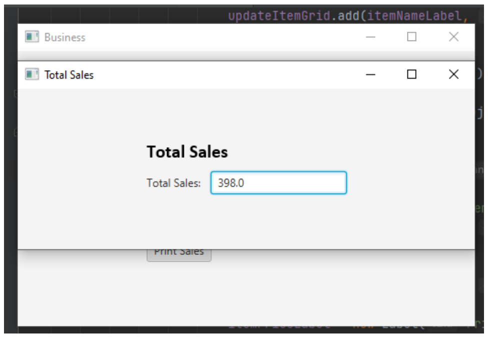
Item sold Successfully!



```
Hot Chocolate,2
```

Total Sales Calculator:

The User can find the total gross sale by clicking the Print Sales Button.



Total Sales

Total Sales: 398.0

Since we made a transaction for x2 Hot Chocolates at rate of 199, it shows us out total sales as 398.

Sample Code:

Following is a sample code for Adding or Updating Sales Data. It is followed by a getSalesData method that returns a Map to the addOrUpdateSalesData method.

```java
//Adds or Updates items sold and its quantity in a SalesData file
private void addOrUpdateSalesData(String itemName, int quantity) {
    FileWriter writer = null;
    try {
        Map<String, Integer> salesDataMap = getSalesData();
        if (salesDataMap.containsKey(itemName)) {
            salesDataMap.put(itemName, salesDataMap.get(itemName)+quantity);
        } else {
            salesDataMap.put(itemName, quantity);
        }
        writer = new FileWriter(salesDataFilePath, false);
        for (String key: salesDataMap.keySet()) {
            writer.write(key+","+ salesDataMap.get(key)+"\n");
        }
    } catch (Exception e) {
        e.printStackTrace();
    } finally {
        try {
            writer.close();
        } catch (IOException e) {
            e.printStackTrace();
        }
    }
}
```

```java
//Reads the data from Business data file and returns a map
private Map<String, Integer> getSalesData() {
    Map<String, Integer> result = new HashMap<String, Integer>();
    try {
        br = new BufferedReader(new FileReader(salesDataFilePath));
        String line = "";
        while ((line = br.readLine()) != null) {
            String[] recordArr = line.split(",");
            result.put(recordArr[0], Integer.parseInt(recordArr[1]));
        }
    } catch (Exception e) {
        e.printStackTrace();
    }
    return result;
}
```

The following code reads the data from Inventory data file and returns list of Items available. It reads the itemsInventoryFilePath and stores the data line by line into an array using while loop. The line is split by commas and an object of Item type is made from the data retrieved by each line and passing it to the Item Constructor. These objects are then added to the ArrayList which is later returned by the method.

```java
//Reads the data from Inventory data file and returns list of Items available
private List<Item> getItemsFromInventory() {
    List<Item> items = new ArrayList<>();
    try {
        br = new BufferedReader(new FileReader(itemsInventoryFilePath));
        String line = "";
        while ((line = br.readLine()) != null) {
            String[] itemArr = line.split(",");
            Item item = new Item(itemArr[0], itemArr[1],
Double.parseDouble(itemArr[2]));
            items.add(item);
        }
    } catch (Exception e) {
        e.printStackTrace();
    }
    return items;
}
```

Following is the code for Item Class:

```java
package sample;

public class Item {
    private String name;
    private String code;
    private double price;

    public Item(String name, String code, double price) {
        super();
        this.name = name;
        this.code = code;
        this.price = price;
    }

    public String getName() {
        return name;
    }
    public void setName(String name) {
        this.name = name;
    }
    public String getCode() {
        return code;
    }
    public void setCode(String code) {
        this.code = code;
    }
    public double getPrice() {
        return price;
    }
    public void setPrice(double price) {
        this.price = price;
     }

}
```