Object Oriented Programming

Electronic Voting System
Restaurant Management System

Bsc (Hons) Computer Science - Jan 2019

Mohamed Ziaan Ahmed | S1900125
Mohamed Isaam Rameez | S1900343
Mariyam Yasmeen | S1800367

1. Abstract

Object-Oriented Programming utilises the concept of building a program using data in the form of objects which leads to enabling the user to re-use code as well as make changes and maintain the code in a much easier way. We have applied OOP concepts to develop two systems; an Electronic Voting system and a Restaurant Management system. Each system utilises different functions such as inheritance and relies on objects, classes and constructs to fulfil the goals required. Each system is explained by starting with the objectives, the UML Class Diagram / Pseudocode, screenshots of the program explaining the functions and important code snippets. This is followed by an overview of object-oriented programming as well as a breakdown of objects, classes, constructors, inheritance, polymorphism, overloading and overriding. The conclusion states the knowledge that we have achieved during the process of completing the project as well as future enhancements that can improve the systems.

Table of Contents

2. Introduction

2. 1 Chapter One - Electronic Voting System

We are creating an Electronic Voting system that can be used to manage all the complexities of the voting process in one simple program. We are eliminating the need for the large teams that are required to count votes as well as making the process more efficient by bringing down the time required in order to get results along with human error that would exist in the lengthy process of counting votes. The Electronic Voting system would give all the candidates a fair chance at winning. The process is easy to implement into a society such as ours here in the Maldives where our communities are tech savvy and the program is simple enough that workshops held before the voting sessions can help the population familiarise themselves with the process.

2.2 Chapter Two - Restaurant Management System

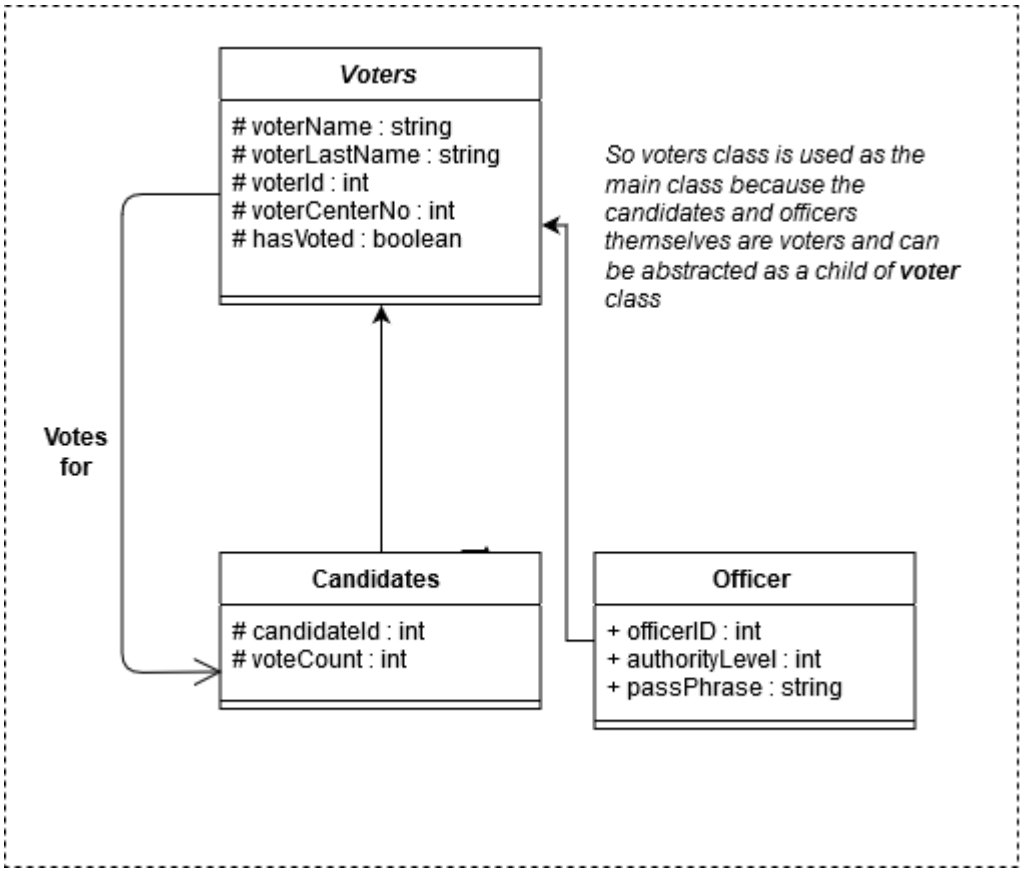
We built our restaurant order system for the convenience of both the customer and the vendor. The program is capable of registering users, letting the customer order from a menu which also displays the most popular items for the day, allowing the customer to opt to receive their invoice via email, change the menu items and also display a daily sales report. The process is fairly straightforward and we are targeting on simplifying the process as much as possible, while also opting for environment friendly solutions such as mailing invoices instead of printing it out.

3. Chapter One

4. Objectives

The Electronic Voting system will allow three types of users. These are civilian voters, officials that oversee the voting process as well as candidates who are running in the elections. While all three different users are able to vote, different users have different levels of access and permissions, such as the official's ability to add and remove voters. Additionally the program holds the information of the candidates, manage the list of voters assigned to the particular system, accept votes from the voters as well as tally and provide the results and statistics of the election.

5. UML Class Diagram



6. Screenshots and Explanations

6.1 Login Screen For Officials

The login screen prompts for the username and password assigned to the election officials assigned to the particular Electronic Voting station. If the credentials are correct they are notified with “Login Accepted”.

```
WELCOME TO THE 2019 ELECTIONS!
Please sign in using your assigned username and password
------------------------------------------------------------

[!] Enter Username :
71c82fd5d2

[!] Enter Password :
IsItWelcome123

[+] Login Accepted!
```

If the official enters either their username or password incorrectly, their login is declined and they are prompted to seek assistance from their supervisor. The login screen is also presented at this point so that the official can attempt to login again.

```
WELCOME TO THE 2019 ELECTIONS!
Please sign in using your assigned username and password
------------------------------------------------------------

[!] Enter Username :
71c82fd5d2

[!] Enter Password :
ItsNotWelcome123

[-] Login Declined!

Your Username and/or Password is incorrect
Please contact your supervisor for assistance or try again


WELCOME TO THE 2019 ELECTIONS!
Please sign in using your assigned username and password
------------------------------------------------------------

[!] Enter Username :
```

6.2 Main Menu For Officials

Once the correct username and password has been entered, the official is then presented with the main menu.

```
Welcome to the main menu
Logged in as : Tribore Menendez
---------------------

[1]     Add Voter
[2]     Add Officer
[3]     Remove User
[4]     Start Vote

[+] Enter a choice :
```

Since they oversee the voting process, they are able to add voters who are not on the voter list, for example in the event that the voter is registered at the incorrect Electronic Voting station and also add an officer if required. They are also authorized to remove users which include candidates in the event of a candidate being disqualified or withdrawing their name from the elections, voters who may have been registered at the incorrect Electronic Voting station or officers who no longer hold the status. The fourth option is for the officer to start the voting process at which point all three user types will then be able to cast their vote.

6.2.1 Add Voter

If it is required to add a voter to the station, officials can select [1] from the Main Menu and enter the full name of the voter along with their password and ID card number so that the voter can now vote. They have to login with the randomly generated user ID and their assigned password.

```
[+] Enter a choice :
1

Enter voter full name :
Rick Astley

Enter a password
NeverGonnaGiveYouUp

Enter ID Card Number
A137099

A voter has been added with a user ID of 5b379bdc
```

6.2.2 Add Officer

Similarly to adding voters, officials can select [2] from the Main Menu and enter the full name of the officer along with their password and ID card number so that the officer can now have full access granted to all officer level users. They have to login with the randomly generated user ID and their assigned password.

```
[+] Enter a choice :
2

Enter officer full name :
Amy Santiago

Enter a password
BindersAreBae

Enter ID Card Number
A145314

A officer has been added with a user ID of c2862648
```

6.2.3 Remove User

Removing a user is pretty straightforward and all you need is the user ID of the user that you would like to remove. Select [3] from the main menu and enter the user ID of either a voter, candidate or even an officer and you would immediately delete the user from the system.

```
[+] Enter a choice :
3

Enter the user ID you'd like to delete :
c8763442b5

Removing matching line...
User has been removed
```

6.2.4 Start Vote

Once the Electronic Voting station is prepared to start the voting process, an official can select option [4] on the main menu which takes you to the voting screen where voters, officials or candidates can log in to vote. If you are the official that started the voting process you are required to login here again in order to cast a vote.

```
WELCOME TO THE 2019 ELECTIONS!
Please enter your credentials

[+] Enter a username :
```

6.3 Voter's Vote menu

```
WELCOME TO THE 2019 ELECTIONS!
Please enter your credentials

[+] Enter a username :
f62eb340d4

[+] Enter a password :
ImNotBatman

[+] Validating login...

Welcome Bruce Wayne

                1.Wilson Higgsbury
                2.Captain Gary
                3.Jake Peralta

Please choose a candidate :
3

[ Thank you for voting! You'll be taken to the login screen in 5 seconds ]
```

We propose that voters will be validated by officials on-site to ensure that the voter is present at the correct Electronic Voting station. Their identity card can be used for verification and each voter can be presented with a sealed username and password to ensure that the system has an added layer of security. The voter (normal civilians, candidates and officials) can log in with their username and password and make a choice between the candidates (in this instance between 1 and 3). Once they enter their choice they will be taken to the login screen within 5 seconds.

Once a voter has voted, if they log in to the system they are notified that they have already voted. They are then taken back to the login screen within 5 seconds.

```
WELCOME TO THE 2019 ELECTIONS!
Please enter your credentials

[+] Enter a username :
f62eb340d4

[+] Enter a password :
ImNotBatman

[+] Validating login...

Welcome Bruce Wayne

You have already voted. Please leave.

[ Thank you for voting! You'll be taken to the login screen in 5 seconds ]
```

6.4 Candidate's Vote Menu

```
WELCOME TO THE 2019 ELECTIONS!
Please enter your credentials

[+] Enter a username :
8863d33502

[+] Enter a password :
GiveMeCookies

[+] Validating login...

Welcome candidate Captain Gary
You currently have 0 votes.

Seems like you haven't voted yet. So please make a selection.

                1.Wilson Higgsbury
                2.Captain Gary
                3.Jake Peralta

Please make a selection.
1

[ Thank you for voting! You'll be taken to the login screen in 5 seconds ]
```

Candidates will additionally see the number of votes that they currently have and if they haven't voted yet they will get to choose a candidate.

```
WELCOME TO THE 2019 ELECTIONS!
Please enter your credentials

[+] Enter a username :
8863d33502

[+] Enter a password :
GiveMeCookies

[+] Validating login...

Welcome candidate Captain Gary
You currently have 0 votes.

You have already voted.

[ Thank you for voting! You'll be taken to the login screen in 5 seconds ]
```

If the candidate has already voted, they see the number of votes as well as a notification saying they have already voted and that they will be taken to the login screen within 5 seconds.

6.5 Officer's Vote Menu

Officer's will additionally have the ability to end the vote process. An initial log in will provide them with the ability to vote and either exit the screen by typing DONE or end the voting process by typing END.

```
WELCOME TO THE 2019 ELECTIONS!
Please enter your credentials

[+] Enter a username :
71c82fd5d2

[+] Enter a password :
IsItWelcome123

[+] Validating login...

Welcome officer Tribore Menendez
Seems like you haven't voted yet. So please make a selection.

                1.Wilson Higgsbury
                2.Captain Gary
                3.Jake Peralta

Please make a selection.
2
Please type in END to end the voting process or DONE if you would like to exit
DONE

[ Thank you for voting! You'll be taken to the login screen in 5 seconds ]
```

Once an officer has voted, if they login again, they are notified that they have already voted and they are asked to either exit to the login menu by typing DONE or they can also type END to stop the voting process.

```
WELCOME TO THE 2019 ELECTIONS!
Please enter your credentials

[+] Enter a username :
71c82fd5d2

[+] Enter a password :
IsItWelcome123

[+] Validating login...

Welcome officer Tribore Menendez
You have already voted.

Please type in END to end the voting process or DONE if you would like to exit
```

6.6 Voting Completed with Results

Once the voting process is over, an official can login and type END to complete the voting process. The results are tallied immediately and a report shows how many votes each candidate got as well as the winning candidate.

```
<< Voting Completed >>
----------------------
Invader Zim      3
Captain Gary     2
Jake Peralta     5

Candidate Jake Peralta has won with 5 votes.
```

7. Cope Snippets

Validating that the correct input is given by the user:

```
388     public static String getValidInput(String message, Integer min){
389
390         Scanner VariableInput = new Scanner(System.in);
391         boolean invalid_input = true;
392         System.out.println(message);
393         while(invalid_input){
394             String returnString = VariableInput.nextLine();
395             if (!returnString.contains(",") & returnString.length() >= min){
396                invalid_input = false;
397                return(returnString);
398             } else {
399                 System.out.println("[-] The input is invalid. Try again.");
400                 System.out.println("[-] Input should be longer than " + min + " characters and should not contain commas");
401             }
402         }
403         return("Error");
404     }
```

Creating an array of “Officer” objects. The same logic applies for “Candidates” and “Voters” and the type of user in the source file does not matter:

```
366     public static Officer[] classifyAdmins(ArrayList<String[]> Input){
367         ArrayList<Officer> ReturnList = new ArrayList<Officer>();
368         for (int i = 0; i < Input.size(); i++){
369             if (Input.get(i)[0].equals("0x03")){
370                 Officer Current = new Officer(Input.get(i)[3], Input.get(i)[1], Input.get(i)[2], Input.get(1)[3]);
371                 ReturnList.add(Current);
372             }
373         }
```

Creating a user that can then vote using the system:

```
415     public static void CreateUser(String UserTypeName, String TypeCodeIn) {
416
417         String TypeCode = TypeCodeIn;
418         String UserID = trivial.getRandomHexString(8);
419         String VoterName = getValidInput("\nEnter " + UserTypeName + " full name : ", 8);
420                 String UserPass = getValidInput("\nEnter a password", 8);
421                 String IDCard = getValidInput("\nEnter ID Card Number", 4);
422         read_write.WriteLine(AddUser(TypeCode, VoterName, UserPass, UserID, IDCard));
423         System.out.println("\nA " + UserTypeName + " has been added with a user ID of " + UserID+"\n");
424     }
```

Validating the credentials of a user that logged in to vote:

```
137         while (stillVoting){
138             if (totalVoters <= voteCount){
139                 stillVoting = false;
140             }
141             System.out.println("WELCOME TO THE 2019 ELECTIONS!");
142             System.out.println("Please enter your credentials");
143             String AttemptedUsername = getValidInput("\n[+] Enter a username : ", 8);
144             String AttemptedPassword = getValidInput("\n[+] Enter a password : ", 8);
145             System.out.println("\n[+] Validating login...");
146
147             for (int i = 0; i < voterList.length; i++){
148                 if (voterList[i].Identity.equals(AttemptedUsername) & voterList[i].Passphrase.equals(AttemptedPassword)){
149                     loginSuccessful = true;
150                     showMenu("0x01", i);
151                     trivial.clearConsole();
152                 }}
```

Letting the user vote for a candidate or if they have voted letting them know. This also takes the user back to the login screen:

```
202             if (!voterList[index].Voted){
203                                 // ******************************************************
204                                 for (int i = 0; i < candidateList.length; i++){
205                                         System.out.println("\t\t" + (i + 1) + "." + candidateList[i].UserName);
206                                 }
207                                 System.out.println("");
208                                 System.out.println("Please choose a candidate : ");
209                                 int selectedIndex = rangedInput(1, candidateList.length);
210                                 candidateList[selectedIndex - 1].votecount ++;
211                                 voterList[index].Voted = true;
212                                 // ******************************************************
213             } else {
214                 System.out.println("You have already voted. Please leave.");
215                         }
216                         try {
217                         System.out.println("\n[ Thank you for voting! You'll be taken to the login screen in 5 seconds ]");
218                         TimeUnit.SECONDS.sleep(5);
219                         } catch (InterruptedException e){}
```

This tallies the votes and prints the result of the election as well as a report on how many votes each candidate received:

```
174                 System.out.println("<< Voting Completed >>");
175                 System.out.println("-----------------------");
176                 for (int i = 0; i < candidateList.length; i++){
177                         System.out.println(candidateList[i].UserName + " \t\t\t " + candidateList[i].votecount);
178                 }
179
180                 int greatestIndex = 0;
181                 for (int i = 0; i < candidateList.length; i++){
182                         if (candidateList[i].votecount > greatestIndex){
183                                 greatestIndex = i;
184                         }}
185
186                 if (greatestIndex == 0){
187                         System.out.println("No one came to vote");
188                 } else {
189                         System.out.println("Candidate " + candidateList[greatestIndex].UserName + " has won with " + candidateL
190                 }
191                 exit(0);
192     }
```

8. Chapter Two

9. Objectives

The Restaurant Program is a simple one that allows the vendor to register their customers. Additionally they are able to display their menu, take the order from a customer and also bill using the system which accounts for 6% goods and services tax. Additionally, upon the request of the customer, the invoice is then emailed to them for future reference. This is an environmentally friendly option that additionally saves the cost of printed bills for the vendor. It is also a far more efficient way for the customer to be able to check their expenses when required since a printed bill can easily be misplaced. The final function of the system includes the ability to generate a sales report which can help the vendor keep track of their day-to-day sales.

10. Pseudo Code

```
VAR
  int: input_choice
  double: total_cost, sumTemp, temporaryFoodPrice
  String: inputStringData, customerName, email_addr, temporaryFoodName, customer_name_input, inputFromUser
  text file:
        items.txt: will hold the "name", "price", "rating", "order count" for food items on the menu
        sales.txt: will hold the "ID", "revenue", "customer name" attributes for past sales
        customer.txt: will hold the "ID", "name", "average spending", "visits", "email address" attributes for customers
  list:
        Arraylist<String> history = new
        Arraylist<String> order_list = new
        Arraylist<String> foodList = new
        Arraylist<String> temporarySales = new
        Arraylist<String> temporaryFood = new

**********************************************************************************************************************

--START PROGRAM--

PROMPT Welcome to the main menu
PROMPT [1] New Order
PROMPT [2] Daily Sales
PROMPT [3] Edit Menu Items
PROMPT [4] Register Customer
PROMPT Please make a selection

INPUT input_choice

SWITCH CASE (input_choice)
  CASE 1
        PROMPT Enter customer name
        INPUT customerName

        i = 0
        LOOP while (customer.txt has next line)
              ADD id, name, average spending, and visits from text file to the list[history]
              i++
        END LOOP

        IF customerName is not in list[history]
```

```
            PROMPT New Customer!
        ELSE IF customer is in list[history] but has zero "visits"
            PROMPT Found customer
            PROMPT This customer is registered but has not purchased anything yet (if customer is registered but hasn't
ordered yet.
        ELSE when customer is in list[history] and has previous "visits"
            PROMPT Found customer
            PROMPT Print out number of "visits" this customer has placed
        ENDIF

        i = 0
        LOOP while(items.txt has next line)
            SET name, price, rating, order count into a temporary variables
            ADD temporary variables to list[foodList]
            i++
        END LOOP

        PROMPT Please select foods

        LOOP for(i = 0; i < size of list[foodList]; i++)
            PROMPT [foodList].get(i), name, price, rating, order count
        END LOOP

        PROMPT Please enter items to order and quantities, and please input 'done' when ordering is finished

        LOOP while (keep the user in a loop until 'done' is input AND keep user in loop if he inputs invalid inputs)
            PROMPT Enter item and quantity, please type 'done' keyword to finish ordering
            INPUT inputFromUser (valid only for two integers separated by a space) (item by food index number, and quantity)

                IF (food item selected does not exist on the menu)
                    PROMPT Invalid input. Try again.

                IF (inputFromUser == 'done')
                    PROMPT Order Completed
                    BREAK out of loop
                ELSE
                    ADD inputFromUser list[order_list]
                    use the first input to increment "order count" attribute of selected food item in items.txt
                ENDIF

        END LOOP

        LOOP for(i = 0; i < size of list[order_list]; i++)
            PROMPT name of item ordered
            PROMPT quantity of item ordered
            PROMPT price of item
            PROMPT price * quantity
            CALC sumTemp+= price * quantity
        END LOOP

        PROMPT sumTemp
        PROMPT sumTemp * 6/100
        PROMPT sumTemp + (sumTemp * 6/100)

        PROMPT the user if he would like to email the bill, please enter yes or no
        INPUT inputStringData
            IF (inputStringData == 'yes')
                IF customer exists and an email is already registered for him, send email
                ELSE prompt customer to input email, and use input to send email
            ELSE
                do nothing and continue
            ENDIF
```

```
        ADD the order details to sales.txt
        PROMPT Enter any key
        INPUT any key
                IF (any key is pressed)
                        CALL main, and go back to the main menu
                ENDIF
END CASE 1

CASE 2
        LOAD everything in sales.txt into a list[temporarySales]
        LOOP for(i = 0; i < size of list[temporarySales]; i++)
                PROMPT customer name, total amount of order, email address bill
                CALC total_cost += total for each order
        END LOOP
        PROMPT total_cost
        PROMPT enter any key to continue
        INPUT any key
                IF (any key is pressed)
                        CALL main, and go back to main menu
                ENDIF
END CASE 2

CASE 3
        PROMPT Edit items on menu
        PROMPT [0] Return
        PROMPT [1] Add food item
        PROMPT [2] Delete food item
        PROMPT [3] Edit price
        PROMPT Please make a selection

        INPUT input_choice
        INNER SWITCH CASE (input_choice)

                INNER CASE 0
                        PROMPT Going back
                        CALL main, and go back to main menu
                END INNER SWITCH CASE 0

                INNER CASE 1
                        LOOP while(items.txt has next line)
                                SET name, price, rating, order count into a temporary variables
                                ADD temporary variables to list[temporaryFood]
                        END LOOP

                        PROMPT Enter the name of food to add
                        INPUT temporaryFoodName
                        PROMPT Enter price of food
                        INPUT temporaryFoodPrice
                        PROMPT Confirmation (yes or no)
                        INPUT inputStringData
                                IF inputStringData == 'no'
                                        CALL main, and go back to main menu
                                ELSE
                                        ADD to temporaryFoodName, temporaryFoodPrice, 4.5 default rating, and zero
order count to list [temporaryFood]
                                ENDIF

                        LOOP for(i = 0; i < size of list[temporaryFood]; i++)
                                ADD item i in list[temporaryFood] to items.txt overwriting the file
                        END LOOP
                END INNER CASE 1
```

```
INNER CASE 2
                    i = 0
                    LOOP while(items.txt has next line)
                        SET name, price, rating, order count into a temporary variables
                        SET variables to list [temporaryFood].set(i)
                        i++
                    END LOOP

                    LOOP for(i = 0; i < size of list [temporaryFood]; i++)
                        PROMPT item index number, name, price, rating, order count from temporaryFood.get(i)
                    END LOOP

                    PROMPT Enter the name of the food item you would like to delete from the menu
                    INPUT inputStringData

                    LOOP for(i = 0; i < size of list [temporaryFood]; i++)
                        IF inputStringData is found in list [temporaryFood].get(i)
                            PROMPT Food item found
                            PROMPT Confirm deletion. Yes or No.
                                INPUT inputStringData
                                IF (inputStringData == 'yes')
                                    DELETE item i from list [temporaryFood]
                                    PROMPT Food item deleted!
                                ELSE
                                    PROMPT Acton aborted
                                ENDIF
                        ELSE do nothing
                    END LOOP

                    CALL main, and go back to main menu
                END INNDER CASE 2

                INNER CASE 3
                    LOOP while(items.txt has next line)
                        SET name, price, rating, order count into a temporary variables
                        ADD temporary variables to list [temporaryFood]
                    END LOOP

                    LOOP for(i = 0; i < size of list [temporaryFood]; i++)
                        PROMPT item index number, name, price, rating, order count
                    END LOOP

                    PROMPT Enter the name of the food item you would like to re-price
                    INPUT inputStringData

                    LOOP for(i = 0; i < size of list [temporaryFood]; i++)
                        IF inputStringData is found in list [temporaryFood].get(i)
                            PROMPT Food item found
                                INPUT temporaryFoodPrice
                                PROMPT You are about to change the price of an item. Type please
confirm with yes or no.
                                INPUT inputStringData
                                    IF (inputStringData == 'yes')
                                        [temporaryFood].set(i) price attribute =
temporaryFoodPrice
                                        LOOP for(i = 0; i < size of list[temporaryFood]; i+
+)
                                            ADD item [temporaryFood].get(i) to
items.txt overwriting the file
                                        END LOOP
                                    ELSE (inputStringData == 'no')
                                        PROMPT price change cancelled
                                    ENDIF
```

```
                                        ELSE do nothing
                                END LOOP

                                CALL main, and go back to main menu

                        END INNER CASE 3

                        INNER CASE DEFAULT
                                PROMPT Missing function
                                CALL main, and go back to main menu
                        END INNER CASE DEFAULT
                END INNER SWITCH CASE

        END CASE 3

        CASE 4
                PROMPT Enter the name of the customer
                INPUT customer_name_input
                PROMPT Would you like to add an email for the customer
                INPUT inputStringData
                IF (inputStringData == 'yes')
                        PROMPT add an email address
                        INPUT email_addr
                ELSE
                        SET email_addr = "none"

                ADD customer_name_input and email_addr to customer.txt
                PROMPT Customer has been added to the system
                PROMPT Press any key to continue
                INPUT any key
                        IF (any key is pressed)
                                CALL main, and go back to main menu

        END CASE 4

        CASE DEFAULT
                PROMPT The input have to be between 1 and 4
                CALL main, and go back to main menu
        END CASE DEFAULT

END SWITCH CASE

--END PROGRAM--
```

11. Screenshots and Explanations

11.1 Main Menu

The main menu gives the user four options. They can take a new order, see the sales for the day, edit the menu items and register a customer. Taking a new order also enables the user to quickly create a customer, or if the customer already exists this will be detected by the system.

```
Welcome to the main menu
========================

         [1] Take a new order
         [2] Show daily sales
         [3] Edit menu items
         [4] Register customer

[+] Please make a selection
```

11.2 Register Customer

To add a customer, the user must select 4 from the main menu. They must then input the name and email to add a new customer to the database. If the customer does not want emailed receipts and does not wish to disclose their email address, the user can type ‘n’ when prompted if they want to proceed with adding a customer email. If the user chooses no then the customer is added without an email. The user may then press any key to proceed.

```
Enter the name of the customer :
Ice Bear

You are about to add an email for this customer. Would you like to proceed? Type 'y' for YES and 'n' for NO.
y

Enter the email address :
ice@gmail.com

Customer has been added to the system.

Press any key to continue...
```

11.3 Make an Order

When the user selects 1 from the main menu they can then take an order from the customer. If the customer is new they can use the customer name field as a quick shortcut to add a new customer as well without going through the process of creating a new customer from the main menu. If the customer exists then the user is notified along with their purchase history and the user is presented with the restaurant menu. The food item, price, rating of the item as well as the popularity so that the user can make recommendations if required. The item is ordered by typing the item number + space + quantity required. The user can keep adding items to the bill this way and once the customer is finished the user has to type ‘done’ to end the current transaction.

```
Enter customer name :
Ice Bear

Found customer

This customer is registered but has not purchased anything yet

Please select your items
-------------------------

1. Chicken burger
         > $12.99
         > 4.5/5 stars
         > Ordered 10 times

2. Steak burger
         > $13.99
         > 5.0/5 stars
         > Ordered 35 times

3. Double Decker Taco
         > $10.0
         > 4.5/5 stars
         > Ordered 7 times

4. Dark Chocolate Cake
         > $5.0
         > 4.6/5 stars
         > Ordered 12 times

5. Pink Salmon Mousse w/ fries
         > $2.44
         > 4.7/5 stars
         > Ordered 18 times

6. Tuna Noodles
         > $2.5
         > 4.5/5 stars
         > Ordered 25 times


Enter an item and quantity
Type 'done' when finished
6 2
Enter an item and quantity
Type 'done' when finished
done
```

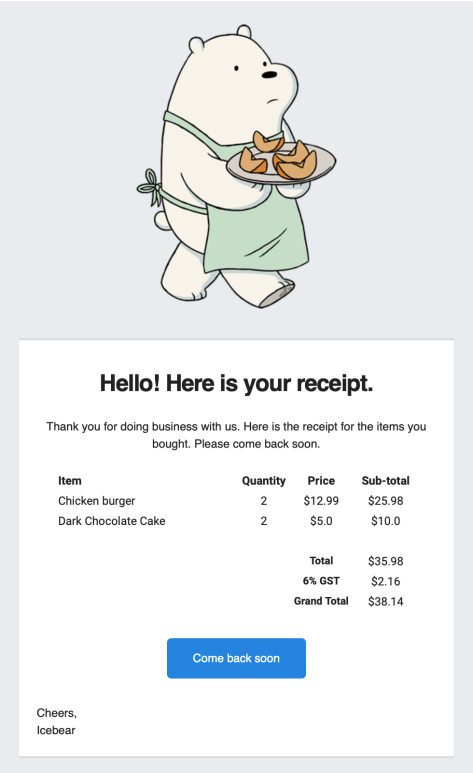
Once the user types ‘done’ they can see the bill with the order details, along with the GST and total amount that the customer needs to pay. At this point the customer can choose to get their receipt via email in which case the user has to type ‘y’ and the system sends the email. If the customer does not need a receipt the user can type ‘n’ and end the transaction.

```
Order Completed

Name                                        Quantity    Price       Final
----                                        --------    -----       -----
Tuna Noodles  ..............................  2         $2.5        $5.0
                                                        --------------------
                                                        Sub-total   $5.00
                                                        6% GST      $0.30
                                                        Total       $5.30
                                                        --------------------

You are about to send an email receipt to the customer. Would you like to proceed? Type 'y' for YES and 'n' for NO.
y
Sending email to ice@gmail.com. Please wait...
```

If you are looking for something amazing in this project, just check out the design of the email that is sent.



Hello! Here is your receipt.

Thank you for doing business with us. Here is the receipt for the items you bought. Please come back soon.

Item	Quantity	Price	Sub-total
Chicken burger	2	$12.99	$25.98
Dark Chocolate Cake	2	$5.0	$10.0
		Total	$35.98
		6% GST	$2.16
		Grand Total	$38.14

Come back soon

Cheers,
Icebear

11.4 Sales Report

When the user chooses 2 from the Main Menu, the daily sales for the day is presented. This includes a breakdown of the sales as well as the total sales for the day.

```
fe2b0a | Phoebe Buffay bought items worth $15.0 (princessconsuela@gmail.com)
6864c0 | Lara Croft bought items worth $13.99 (None)
3c3afe | Julian Casablancas bought items worth $4.88 (None)
7a50b5 | Moon Cake bought items worth $20.0 (chookity@gmail.com)
9f972e | Serj Tankian bought items worth $2.44 (None)
---------------------------
Total revenue : $56.31

Press any key to continue
```

11.5 Edit Item

When the user chooses 3 in the Main Menu they are taken to an Edit Items Menu where they can proceed with adding a new item, deleting an item or editing the price of an item.

```
Edit Items Menu
===============

        [1] Add food item
        [2] Delete food item
        [3] Edit price
        [0] Return

[+] Please make a selection
```

Entering ‘0’ will take the user back to the Main Menu

11.5.1 Add food item

The user needs to enter the name of the item, price and confirm the addition by typing ‘y’

```
Enter the name of the food you would like to add :
Seafood Rice

Enter a price for the item
7.25

You are about to add a new food item to the menu. Would you like to proceed? Type 'y' for YES and 'n' for NO.
y
```

11.5.2 Delete food item

When the user chooses 2 to delete a food item, they get presented with the items currently in the menu. When the user writes the name of the item that needs to be deleted and confirms by typing ‘y’ the item is deleted.

```
1. Chicken burger
         > $12.99
         > 4.5/5 stars
         > Ordered 10 times

2. Steak burger
         > $13.99
         > 5.0/5 stars
         > Ordered 36 times

3. Double Decker Taco
         > $10.0
         > 4.5/5 stars
         > Ordered 7 times

4. Dark Chocolate Cake
         > $5.0
         > 4.6/5 stars
         > Ordered 14 times

5. Pink Salmon Mousse w/ fries
         > $2.44
         > 4.7/5 stars
         > Ordered 43 times

6. Tuna Noodles
         > $2.5
         > 4.5/5 stars
         > Ordered 26 times

7. Seafood Rice
         > $7.25
         > 4.5/5 stars
         > Ordered 0 times


Enter the name of the food item you would like to delete from the menu
Pink Salmon Mousse w/ fries

Food Item 'Pink Salmon Mousse w/ fries' found!

You are about to delete a food item. Would you like to proceed? Type 'y' for YES and 'n' for NO.
y

Food item deleted!
```

11.5.3 Edit price

When the user chooses 3 to edit the price, they are presented with the current menu and prices. They can then type the name of the item for which they want to change the price, input the new price and type ‘y’ to confirm the change.

```
1. Chicken burger
         > $12.99
         > 4.5/5 stars
         > Ordered 10 times

2. Steak burger
         > $13.99
         > 5.0/5 stars
         > Ordered 36 times

3. Double Decker Taco
         > $10.0
         > 4.5/5 stars
         > Ordered 7 times

4. Dark Chocolate Cake
         > $5.0
         > 4.6/5 stars
         > Ordered 14 times

5. Tuna Noodles
         > $2.5
         > 4.5/5 stars
         > Ordered 26 times

6. Seafood Rice
         > $7.25
         > 4.5/5 stars
         > Ordered 0 times


Enter a the name of the food item you'd like to re-price
Dark Chocolate Cake

Food Item 'Dark Chocolate Cake' found!

Please enter a new price for the item.
4.75

You are about to change the price of an item. Would you like to proceed? Type 'y' for YES and 'n' for NO.
y

Price for the food item was changed!
```

12. Cope Snippets

Takes the user input of a name and price for the new food item which will get added into the items database.

```
11          static void add_food(){
12                  App.clearConsole();
13                  ArrayList<FoodItem> raw_food_list = ReadWrite.LoadFoods();
14                  App.print("Enter the name of the food you would like to add : ");
15                  String food_name_in = App.input();
16                  App.print("\nEnter a price for the item");
17                  double food_price_in = adv_input.getDouble();
18                  FoodItem new_food_item = new FoodItem(food_name_in, food_price_in,4.5,0);
19                  raw_food_list.add(new_food_item);
20
21                  if (adv_input.confirmAction("add a new food item to the menu")){
22                          ReadWrite.writeToFoodFile(raw_food_list);
23                  } else {
24                          App.print("Action aborted");}}
```

This allows the user to edit the price of an item that is already listed on the menu.

```
52          static void edit_price(){
53                  App.clearConsole();
54                  App.listFood();
55                  ArrayList<FoodItem> raw_food_list = ReadWrite.LoadFoods();
56                  App.print("Enter a the name of the food item you'd like to re-price");
57                  String food_name = App.input();
58                  for (int i = 0; i < raw_food_list.size(); i++){
59                          if (raw_food_list.get(i).name.equals(food_name)){
60                                  App.print("\nFood Item '" + food_name + "' found!");
61                                  App.print("\nPlease enter a new price for the item.");
62                                  raw_food_list.get(i).price = adv_input.getDouble();
63                                  if (adv_input.confirmAction("change the price of an item")){
64                                          ReadWrite.writeToFoodFile(raw_food_list);
65                                  } else {
66                                          App.print("Action aborted");
67                                          return;
68                                  }}}
69                  App.print("\nPrice for the food item was changed!");}
```

Add a customer to the database with their name and upon their approval, an email address.

```
95          static void addCustomer(){
96                  clearConsole();
97                  print("Enter the name of the customer : ");
98                  String customer_name_input = input();
99                  String email_addr;
100                 if (adv_input.confirmAction("add an email for this customer")){
101                         print("\nEnter the email address : ");
102                         email_addr = input();
103                 } else {
104                         email_addr = "None";
105                 }
106
107                 Customer new_customer = new Customer(adv_input.getRandomHexString(8), customer_name_input, 0.0, 0, email_addr);
108
109                 ArrayList<Customer> old_customers = ReadWrite.loadCustomers();
110                 old_customers.add(new_customer);
111                 ReadWrite.writeToCustomerFile(old_customers);
112                 print("\nCustomer has been added to the system.");
113                 print("\nPress any key to continue...");
114                 input();
115         }
```

Add all sales to the sales.txt file so that daily sales reports can be easily generated

```
184         static void WriteSales(ArrayList<Sale> saveData){
185                 try {
186                         PrintWriter salesFile = new PrintWriter(new FileWriter("sales.txt"));
187                         DecimalFormat df = new DecimalFormat("####0.00");
188                         for (int i = 0; i < saveData.size(); i++){
189                                 Sale currentSale = saveData.get(i);
190                                 String currentLine =    currentSale.saleId + "," +
191                                                                                 df.format(currentSale.revenue) + "," +
192                                                                                 currentSale.customername + ","+
193                                                                                 currentSale.emailaddress + "\n";
194                                 salesFile.write(currentLine);
195                         }
196                         salesFile.close();
197                 } catch (IOException e){App.print("IO Error : " + e);}
198         }
```

Send the details of the transaction to the customer via email.

```
29          public static void sendEmail(Session session, String toEmail, String subject, String body, String title, String replyTo){
30                  try
31              {
32                MimeMessage msg = new MimeMessage(session);
33                //set message headers
34                msg.addHeader("Content-type", "text/HTML; charset=UTF-8");
35                msg.addHeader("format", "flowed");
36                msg.addHeader("Content-Transfer-Encoding", "8bit");
37                msg.setFrom(new InternetAddress(replyTo, title));
38                msg.setReplyTo(InternetAddress.parse(replyTo, false));
39                msg.setSubject(subject, "UTF-8");
40                msg.setText(body, "UTF-8");
41                msg.setSentDate(new Date());
42                    msg.setRecipients(Message.RecipientType.TO, InternetAddress.parse(toEmail, false));
43                    msg.setContent(body, "text/html; charset=utf-8");
44              //   System.out.println("Message is ready");
45             Transport.send(msg);
46
47              //   System.out.println("EMail Sent Successfully!!");
48              }
49              catch (Exception e) {
50                e.printStackTrace();
51              }
52          }
```

13. Object-Oriented Programming

Object-oriented programming (OOP) utilises a model where a program is built using data in the form of objects instead of using logic and functions. Object-oriented programming originated with the Simula languages developed at the Norwegian Computing Center, Oslo, in the 1960s (Madsen, Møller-Pedersen & Nygaard, 1993). However, the first substantial interactive, display-based implementation was the SMALLTALK language (Goldberg & Robson, 1983).

Some of the benefits of object-oriented programming include the ability to reuse code and divide it into groups for collaboration purposes, the increased efficiency of the program and the method is extremely useful for complex programs that need to be regularly maintained or updated. When you're using an OO programming paradigm, you only need to change the data element in the object's definition and the other objects can keep on interacting with it like they did before, minimizing the maintenance (Baesens, Backiel & Broucke, 2015)

13. 1 Objects

An object in Java combines data and algorithms.The state of the object is defined by data and the behavior is defined by the functions or algorithm. A simple example would be the "voters" that we have used in our Electronic Voting system. The voters all have classifiable data, such as names and identity card numbers making them an object we can utilize in our system. The procedure for this example is the ability to allow the voter to cast a vote. All of the action in object-oriented programming comes from sending messages between objects (Stefik and Bobrow, 1985). The main difference between object-oriented programming and the earlier approaches to programming is the importance given to how the data is defined within the logic instead of the way that we write the logic.

We create an object in Java with "new" and the steps to creating an object include declaration, instantiation and initialization of the object. Declaration of an object is associating the object with a name. Instantiation is when memory has been allocated for the object and its constructor has been run meanwhile initializing is when an initial value has been assigned to the variable.

```
341     public static Candidate[] classifyCandidates(ArrayList<String[]> Input){
342         ArrayList<Candidate> ReturnList = new ArrayList<Candidate>();
343         for (int i = 0; i < Input.size(); i++){
344             if (Input.get(i)[0].equals("0x02")){
345                 Candidate Current = new Candidate(Input.get(i)[3], Input.get(i)[1], Input.get(i)[2], Input.get(1)[3]);
```

Here an array of the object "Candidate" is being built. Declaration, instantiation and initialisation can be carried out simultaneously in this way.

13.2 Classes

Every object in Java belongs to a class. You define a class in source code, which is compiled into a binary format (a class file with a .class extension)(Sharan, 2017). Classes are the blueprint with which you create objects. An easy way to understand the concept is, if you were asked “what is an apple?” your answer would likely be that it’s a fruit. This is because certain characteristics are shared among a variety of produce which we collectively call “fruits”. In this example you can understand that while other fruits also exist, an object “apple” is an instance of its class “fruit”.

```
class Voter {
        /** This String will store the Name of the user */
        String Identity = "None";
        /** This string will store the username of the user */
        String UserName = "None";
        /** This string will store the password of the user */
        String Passphrase = "None";
        /** This String will store the ID Card number of the user */
        String IDCard = "None";
        /** This boolean will determine if the user has voted or not. */
    Boolean Voted = false;
```

In our Electronic Voting system, one of the classes is “Voter” which has characteristics such as identity, username and passphrase. They are also initialized in this step with “None”.

13.3 Constructors

All classes need a constructor and therefore, if you do not state one then the compiler in Java would build one for the class you created. A constructor usually gives the initial values to the defined variables in the class or may start other procedures which may be required to create the object. Constructors must have the same name as the class but a single class can have multiple constructors.

The constructor is an unusual type of method because it has no return value. This is distinctly different from a void return value, in which the method returns nothing but you still have the option to make it return something else (Eckel, 2003). The two types of constructors in Java are default constructors and parameterized constructors.

```
class Sale {
    String saleId = "None";
    double revenue = 0.0;
    Date datetime = new Date();
    String customername = "None";
    String emailaddress = "None";

    public Sale(String saleIdIn, double revenueIn, Date datetimeIn, String customernameIn, String emailaddressIn){
        saleId = saleIdIn;
        revenue = revenueIn;
        datetime = datetimeIn;
        customername = customernameIn;
        emailaddress = emailaddressIn;
    }
}
```

This is an example of a parameterised constructor used in our Restaurant Program which accepts parameters such as a customer name and an email address.

13.4 Inheritance

When a class acquires the properties of another it is called inheritance. This is a useful way to ensure that information is managed in a hierarchical order. The child class which inherits the properties of the parent class is called a subclass and the parent class from which these properties were inherited from is known as the superclass. The keyword used to inherit the properties of a class is “extends”. Another use of inheritance (some might argue “misuse”) is implementation inheritance. Here the only purpose of creating a parent class is to factor code that is needed by other subclasses (Wiener & Pinson, 2000)

```
698  class Officer extends Voter{
699
700  //    int OfficerVerificationID = 0;
701      public Officer(String IdentityIn, String UsernameIn, String PassphraseIn, String IDCardIn) {
702          super(IdentityIn, UsernameIn, PassphraseIn, IDCardIn);
703      }
704  }
```

In our Electronic Voting system, the subclass Officer inherits the properties of the superclass Voter.

13.5 Polymorphism

The ability of an object to take on various forms is known as Polymorphism. In Java all objects are considered polymorphic due to the fact that they pass several is-a or instanceof tests. Is-a depends on class inheritance or interface inheritance and instanceof tests whether the object is an instance of the class, subclass or interface. Polymorphism helps reduce complexity by allowing the same interface to be used to specify a general class of action (Schildt, 2014). Java supports static polymorphism and dynamic polymorphism.

13.6 Overloading

Overloading in Java is when different functions have the same variable name but with unique parameter types or order allowing for them to be easily called. A method can be overloaded not only in the same class but also in a descendant (Fain, 2015).In order to overload a method, the number of parameters, data type of the parameter or the order in which the parameter data types appear must be different. Similarly to methods, constructors can also be overloaded so that each constructor executes a different task.

13.7 Overriding

Overriding just means that a subclass redefines one of its inherited methods when it needs to change or extend the behaviour of that method (Sierra & Bates, 2005). When overriding a method, the argument list and the return type should be identical and the access level of the overriding method cannot be more restrictive than the access level of the superclass method. A method cannot be overridden in instances where the method cannot be inherited, if the method was declared final or on a method that is declared static. However, a method that was declared static can be re-declared.

14. Conclusion & Recommendations

In conclusion, Object-Oriented Java can be utilised to build systems such as an Electronic Voting system or a Restaurant Management system with a variety of useful features. We were able to learn about different OOP concepts such as how to use objects and classes in order to make our code more efficient and easier to maintain or update. We were also able to learn useful collaboration skills as well as how to efficiently make a project timeline and resources such as Github. Additionally our developer has successfully learnt how to spell restaurant and receipt.

While the current systems fulfil the requirements of the assignment we can improve these in a number of ways. For example, the Electronic voting machine could utilise a finger print machine so that passwords are not required and we would have a system that is more secure at validating the identity of the person that is voting. We could also implement a text to speech function so that the system is more accessible for those with special needs. The Restaurant System can also include a timestamp so that the sales report for a certain date or time period can be generated, a customer loyalty programme so that frequent customers get a loyalty benefit and with their consent we can even send out ads for the vendor through the email service increasing revenue for the establishment. Overall both systems can be further improved and also expanded to provide its users with more useful functions.

15. Bibliography

1. Baesens, B.B., Backiel, A.B. and Broucke, SVB (2015) Beginning Java Programming The Object-Oriented Approach [online]. Indianapolis, Indiana, USA: John Wiley & Sons, Inc,. [Accessed 26 July 2019].

2. Eckel, BE (2003) Thinking in Java. 3rd ed. New Jersey, USA: Pearson Education, Inc.

3. Fain, YF (2015) Java Programming 24-Hour Trainer [online]. 2nd ed. Indianapolis, Indiana, USA: Wiley Publishing, Inc. [Accessed 28 July 2019].

4. Goldberg, A.G. and Robson, D.R. (1983) Smalltalk-80: The Language and Its Implementation. Boston, Ma, USA: Addison-wesley Longman Publishing Co, Inc.

5. Madsen, OLM, Møller-Pedersen, BMP and Nygaard, KN (1993) Object-Oriented Programming in the Beta Programming Language. USA: Addison-Wesley.

6. Schildt, HS (2014) Java: A Beginner's Guide [online]. 6th ed. New York, USA: McGraw-Hill Education. [Accessed 27 July 2019].

7. Sharan, KS (2017) Beginning Java 9 Fundamentals [online]. 2nd ed. Montgomery, Alabama, USA: Apress Media LLC. [Accessed 27 July 2019].

8. Sierra, KS and Bates, BB (2005) Head First Java. 2nd ed. Sebastopol, California, USA: O'Reilly Media, Inc.

9. Stefik, MS and Bobrow, D.G.B. (1985) Object-oriented Programming: Themes and Variations. Ai Magazine [online]. 6 (4), p. 41. [Accessed 26 July 2019].

10. Wiener, RW and Pinson, LJP (2000) Fundamentals of OOP and Data Structures in Java. Cambridge, United Kingdom: Cambridge University Press.