**Fr. CONCEICAO RODRIGUES COLLEGE OF ENGG.**

Fr. Agnel Ashram, Bandstand, Bandra(w) Mumbai 400 050

Deprtment of Computer Engineering

Subject-OOPM (SEM-III)

Academic year 2016-2017

**MINI PROJECT REPORT**

1. Edwin Clement - (Roll No. 7624)

2. Samson Anto Paul - (Roll No. 7669)

**Project Name: Shop Point-of-Service System with Inventory**

**1. Problem Statement:**

Write a program to handle a shops Sale Counter. The interface should be fast and intuitive. The program must have data persistence so all the inventory data is stored permanently. Also, there must be a system of Premium Customers such that the Premium customers will be given preferences. There must be provisions to quickly edit the shopping cart.

**2. ABSTRACT**

The POS System is meant to be a very fast interface so that te sales person can quickly serve the customers efficiently. This system implemented by has instant auto search completion, both for premium customers and items. Also, The search for items may be done both via Bar Code and Item Name. These Features add more to flexibility and speed. Since Tab-jumping is carefully setup, the focus of the user input flows through each step in a logical manner. Thus for an experienced user, this application will serve to be a performance enhancer; exactly what a program should be.

Names Of The Classes AND METHODS USED:

**// List Functions and variables**

|  |
| --- |
| **Object**  «final» edwinclement08::IMS:: **ItemsList** |
| Fields  ~ debug : boolean - input : ObjectInputStream ~ itemFile : File ~ itemList : Vector<Item> - output : ObjectOutputStream |
| Constructors  ~ ItemsList( ) : void |
| Methods  ~ addItem( Item ) : void ~ getItemListFromBarCode( String ) : Vector<Item> ~ getItemListFromName( String ) : Vector<Item> ~ getList( ) : Vector<Item> ~ loadData( ) : boolean ~ printList( ) : void ~ removeItem( Item ) : boolean ~ saveData( ) : boolean |

|  |
| --- |
| **Object**  edwinclement08::IMS:: **Item** |
| Constants  ~ serialVersionUID : long |
| Fields  - barCodeId : BarCode - name : String - price : double    «transient» ~ sc : Scanner |
| Constructors  ~ Item( String, double ) : void |
| Methods  ~ getBarCodeStringId( ) : String ~ getConcise( ) : String ~ getName( ) : String ~ getPrice( ) : double ~ inputData( ) : void ~ showConcise( ) : void |

|  |
| --- |
| **Object**  edwinclement08::IMS:: ***Customer*** |
| Fields  # name : String |
| Constructors  ~ Customer( ) : void ~ Customer( String ) : void |
| Methods  *~ buy( Bill ) : void* |

|  |
| --- |
| **Object**  «final» edwinclement08::IMS:: **CustomerList** |
| Fields  ~ customerFile : File ~ customerList : Vector<PremiumCustomer> - debug : boolean - fin : ObjectInputStream - fout : ObjectOutputStream |
| Constructors  ~ CustomerList( ) : void |
| Methods  ~ addCustomer( PremiumCustomer ) : void ~ getCustomerDetailsFromName( String ) : PremiumCustomer ~ getCustomerListFromName( String ) : Vector<PremiumCustomer> ~ getList( ) : Vector<PremiumCustomer> ~ loadData( ) : boolean + main( String[] ) : void ~ saveData( ) : boolean ~ setCardNumberCounter( ) : void |

|  |
| --- |
| **Object**  edwinclement08::IMS:: **Bill** |
| Constants  ~ serialVersionUID : long |
| Fields  ~ counterForId : long ~ date : Date ~ id : long ~ items : Vector<Item> |
| Constructors  ~ Bill( ) : void |
| Methods  ~ addItem( Item ) : void ~ getTotalCost( ) : int ~ listItems( ) : void |

|  |
| --- |
| **Object**  edwinclement08::IMS:: **BarCode** |
| Constants  ~ serialVersionUID : long |
| Properties     «readOnly» + id : String |
| Constructors  + BarCode( String ) : void |
| Methods  - generateId( String ) : void + verifyId( String ) : boolean |

|  |
| --- |
| ***Customer***  edwinclement08::IMS:: **CasualCustomer** |
| Fields  - bill : Bill - idCounter : long - prefix : String |
| Constructors  ~ CasualCustomer( ) : void |
| Methods  ~ buy( Bill ) : void ~ getName( ) : String ~ upgrade( String ) : PremiumCustomer |

|  |
| --- |
| **Object**  edwinclement08::IMS:: ***Customer*** |
| Fields  # name : String |
| Constructors  ~ Customer( ) : void ~ Customer( String ) : void |
| Methods  *~ buy( Bill ) : void* |

|  |
| --- |
| **Object**  «final» edwinclement08::IMS:: **CustomerList** |
| Fields  ~ customerFile : File ~ customerList : Vector<PremiumCustomer> - debug : boolean - fin : ObjectInputStream - fout : ObjectOutputStream |
| Constructors  ~ CustomerList( ) : void |
| Methods  ~ addCustomer( PremiumCustomer ) : void ~ getCustomerDetailsFromName( String ) : PremiumCustomer ~ getCustomerListFromName( String ) : Vector<PremiumCustomer> ~ getList( ) : Vector<PremiumCustomer> ~ loadData( ) : boolean + main( String[] ) : void ~ saveData( ) : boolean ~ setCardNumberCounter( ) : void |

|  |
| --- |
| «interface» edwinclement08::IMS:: ***IdentificationCode*** |
| Properties     «readOnly» *+ id : String* |
| Methods  *+ verifyId( String ) : boolean* |

|  |
| --- |
| **JFrame**  edwinclement08::IMS:: **MainInterface** |
| Fields  - AddToList : JButton - BalanceForCash : JLabel - BalanceForPoints : JLabel - BarCodeInput : JTextField - BecomePremium : JButton - CashInputForCash : JTextField - CashInputForPoints : JTextField - CashStillRequiredForPoints : JLabel - Confirmation : JLabel - CustomerEntryPanel : JPanel - CustomerNameInput : JTextField - CustomerNameOptions : JList<String> - DeleteItem : JButton - ForCash1 : JPanel - ItemNameInput : JTextField - ItemPrediction : JList<String> - ItemQuantityInput : JTextField - ItemSelectionPanel : JPanel - ItemsBought : JTable - MainWindow : JFrame - NextCustomer : JButton - Paymentselect : ButtonGroup - PremiumCustomerInfo : JTextPane - TotalPriceForCash : JLabel ~ casual : String ~ customerList : CustomerList ~ itemList : ItemsList - jButton1 : JButton - jButton2 : JButton - jButton3 : JButton - jLabel1 : JLabel - jLabel10 : JLabel - jLabel11 : JLabel - jLabel14 : JLabel - jLabel16 : JLabel - jLabel18 : JLabel - jLabel19 : JLabel - jLabel2 : JLabel - jLabel20 : JLabel - jLabel21 : JLabel - jLabel22 : JLabel - jLabel23 : JLabel - jLabel3 : JLabel - jLabel5 : JLabel - jLabel6 : JLabel - jLabel7 : JLabel - jLabel8 : JLabel - jLabel9 : JLabel - jPanel1 : JPanel - jPanel4 : JPanel - jPanel5 : JPanel - jScrollPane2 : JScrollPane - jScrollPane3 : JScrollPane - jScrollPane4 : JScrollPane - jScrollPane5 : JScrollPane - jTabbedPane1 : JTabbedPane ~ tempCustomer : Customer ~ totalPrice : double |
| Constructors  + MainInterface( ) : void |
| Methods  - AddToListActionPerformed( ActionEvent ) : void - BarCodeInputKeyReleased( KeyEvent ) : void - BecomePremiumActionPerformed( ActionEvent ) : void - CashInputForCashKeyReleased( KeyEvent ) : void - CustomerNameInputKeyReleased( KeyEvent ) : void - CustomerNameOptionsMouseClicked( MouseEvent ) : void - CustomerNameOptionsMouseExited( MouseEvent ) : void - DeleteItemActionPerformed( ActionEvent ) : void - ItemNameInputKeyReleased( KeyEvent ) : void - ItemQuantityInputFocusGained( FocusEvent ) : void - NextCustomerActionPerformed( ActionEvent ) : void    «synthetic» ~ access$000( MainInterface, KeyEvent ) : void    «synthetic» ~ access$100( MainInterface, FocusEvent ) : void    «synthetic» ~ access$1000( MainInterface, KeyEvent ) : void    «synthetic» ~ access$1100( MainInterface ) : void    «synthetic» ~ access$200( MainInterface, ActionEvent ) : void    «synthetic» ~ access$300( MainInterface, KeyEvent ) : void    «synthetic» ~ access$400( MainInterface, ActionEvent ) : void    «synthetic» ~ access$500( MainInterface, KeyEvent ) : void    «synthetic» ~ access$600( MainInterface, ActionEvent ) : void    «synthetic» ~ access$700( MainInterface, MouseEvent ) : void    «synthetic» ~ access$800( MainInterface, MouseEvent ) : void    «synthetic» ~ access$900( MainInterface, ActionEvent ) : void - blankOutForNewCustomer( ) : void - initComponents( ) : void + main( String[] ) : void - updateSinceTotalAmountChanged( ) : void |

|  |
| --- |
| ***Customer***  edwinclement08::IMS:: **PremiumCustomer** |
| Constants  ~ serialVersionUID : long |
| Fields  ~ billHistory : Vector<Bill> - cardNumber : long ~ cardNumberCounter : long ~ customerList : Vector<PremiumCustomer> - pointMultiplier : double - points : int - prefix : String |
| Constructors  ~ PremiumCustomer( ) : void ~ PremiumCustomer( String ) : void |
| Methods  ~ addPoints( int ) : void ~ buy( Bill ) : void ~ deductPoints( int ) : boolean ~ formatedOutput( ) : String ~ getCardNumber( ) : long ~ getFormatedCardNumber( ) : String ~ getFormatedDetail( ) : String ~ getFormatedName( ) : String ~ getFormatedPoints( ) : String ~ getName( ) : String ~ getPoints( ) : int |

**3. Java Features Achieved:**

1. **Data Abstraction and Encapsulation:**

The wrapping up of data and methods into a single unit is known as encapsulation. Data encapsulation is the most striking feature of a class. The data is not accessible to the outside world and only those method which are wrapped in the class ,can access it.

In our program, we have extensively utilized this. From Bill to Item. All items have internal variables that can’t be accessed from the outside of the class. When required to expose the Variables; like in an Item’s Barcode, we used the getter method *getBarCodeStringId().* Similarly, the item name and price are also hidden away, exposed only by the getter functions *getName()* and *getPrice().*

1. **Polymorphism:**

Polymorphism is the ability to take more than one form. Polymorphism plays an important role in allowing object having different internal structures to share the same external interface.

For Example: In our program we have a base *Customer* class which have *PremiumCustomer* and *CasualCustomer* as its children. The Parent class has a method *buy()* which is inherited by both. The implementation varies between them.

By Utilizing Dynamic Method Dispatch, we can call buy on a parent reference resting easy that the correct method will be executed based on the object type.

1. **Interactive:**

We have achieve the interactive feature by implement the project in graphical application. We have created a GUI frame so that the user can interact with the system which makes it easy for an user understand the program. Also, the Intuitiveness makes it easy for users to catch up.

1. **Distributed:**

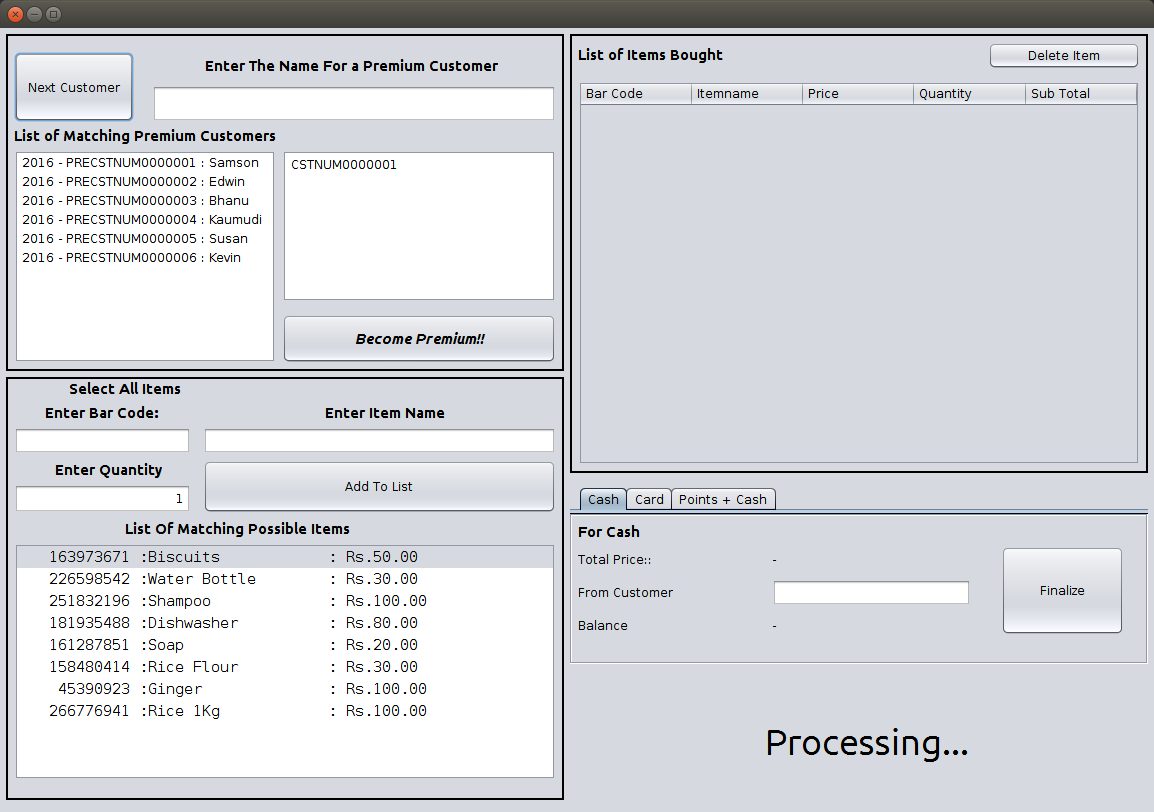
Java is designed as a Distributed language for creating applications on networks. It has the ability to share both data and programs. Java can open and access remote objects on Internet as easily as they can do it in a local system.

For Example: In our program we have store our data file on the local host machine so any communication done with the data file would be done by the local host.

1. **Dynamic and extensible:**

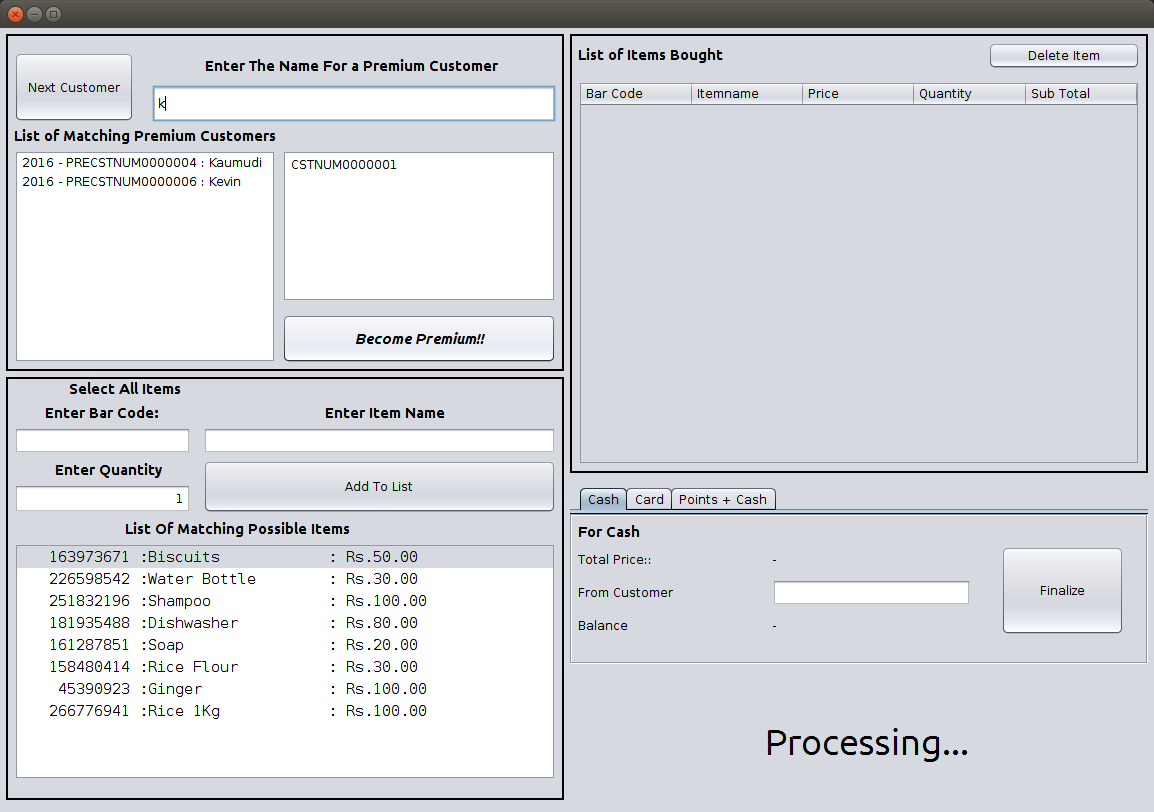
Java is a dynamic language. Java is capable of dynamically linking in new class libraries, methods and objects. We have used an Serialization Concept to easily store data and retrieve it in a quick and easy manner which can be extended dynamically.

**4. OUTPUT SCREEN SHOTS OF EACH STAGE**



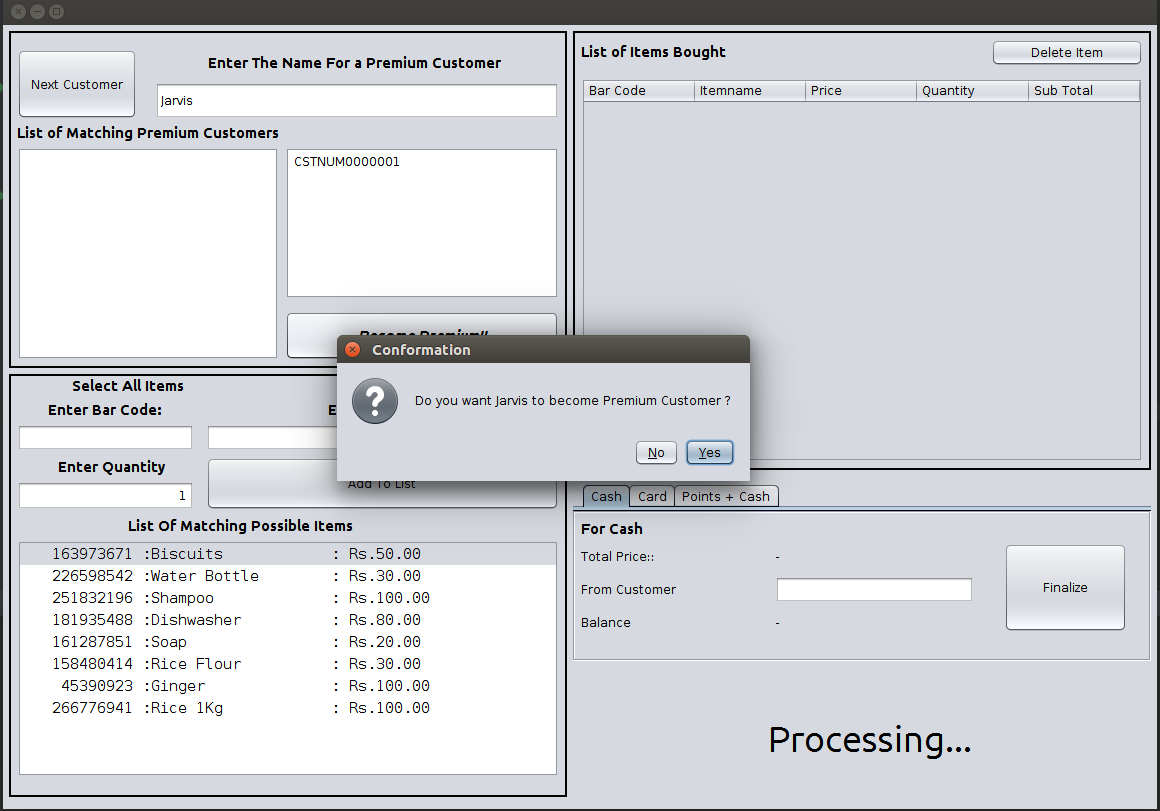
Start up Screen

Here The Customer Can be Selected.

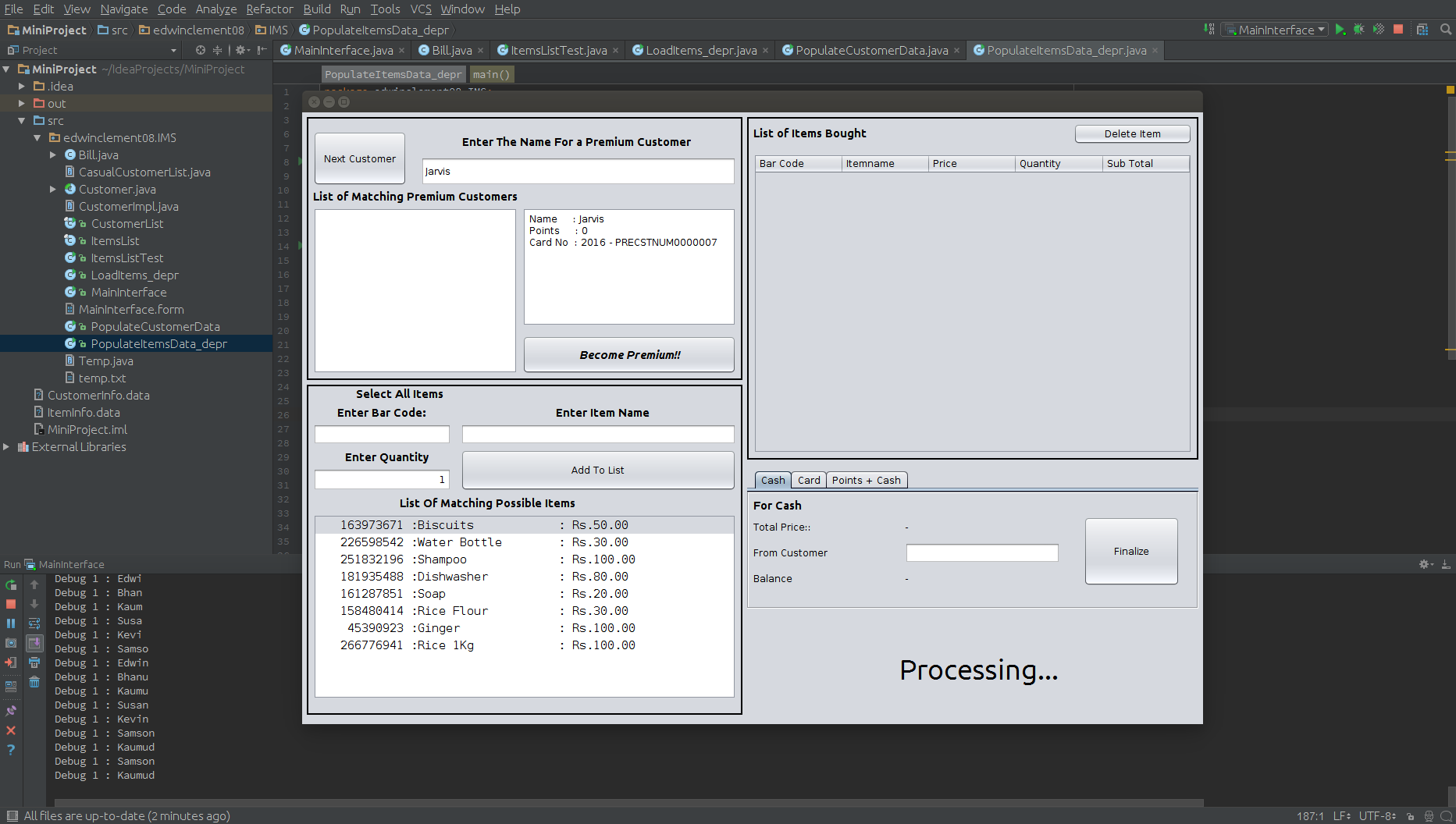


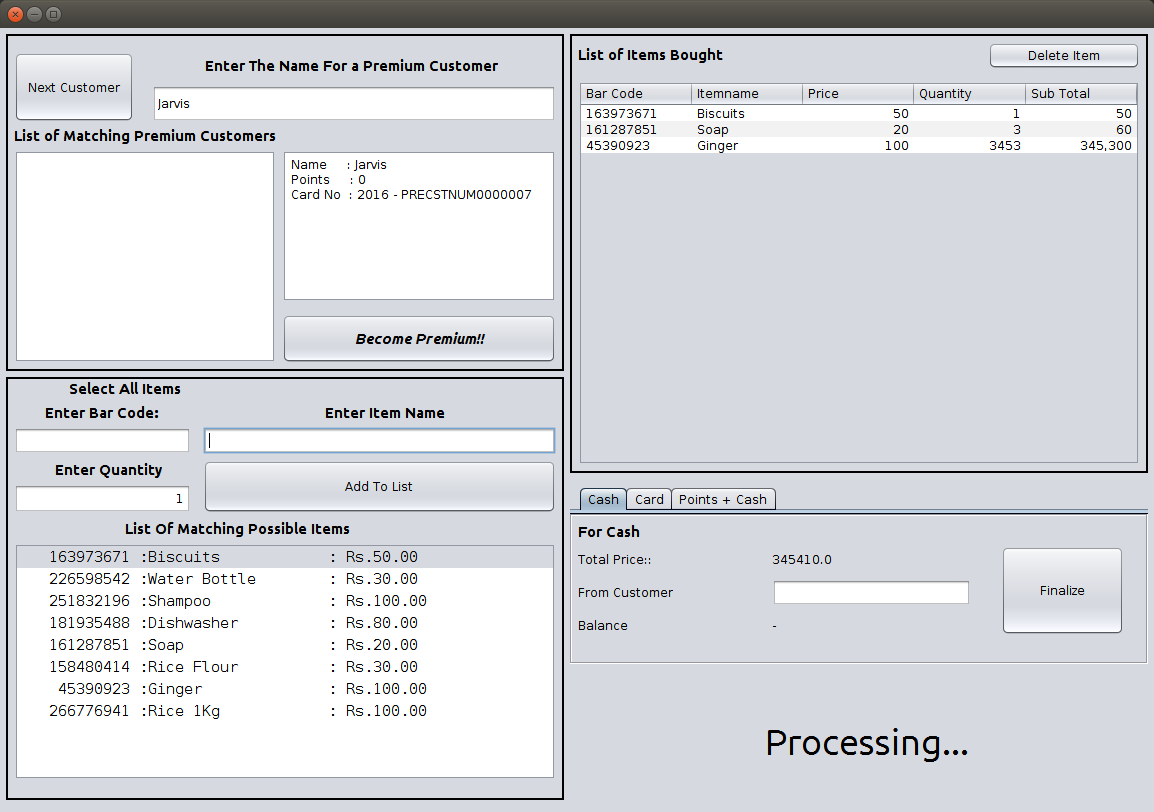
AutoSearch

Here, The Auto Search Feature has been demonstrated.

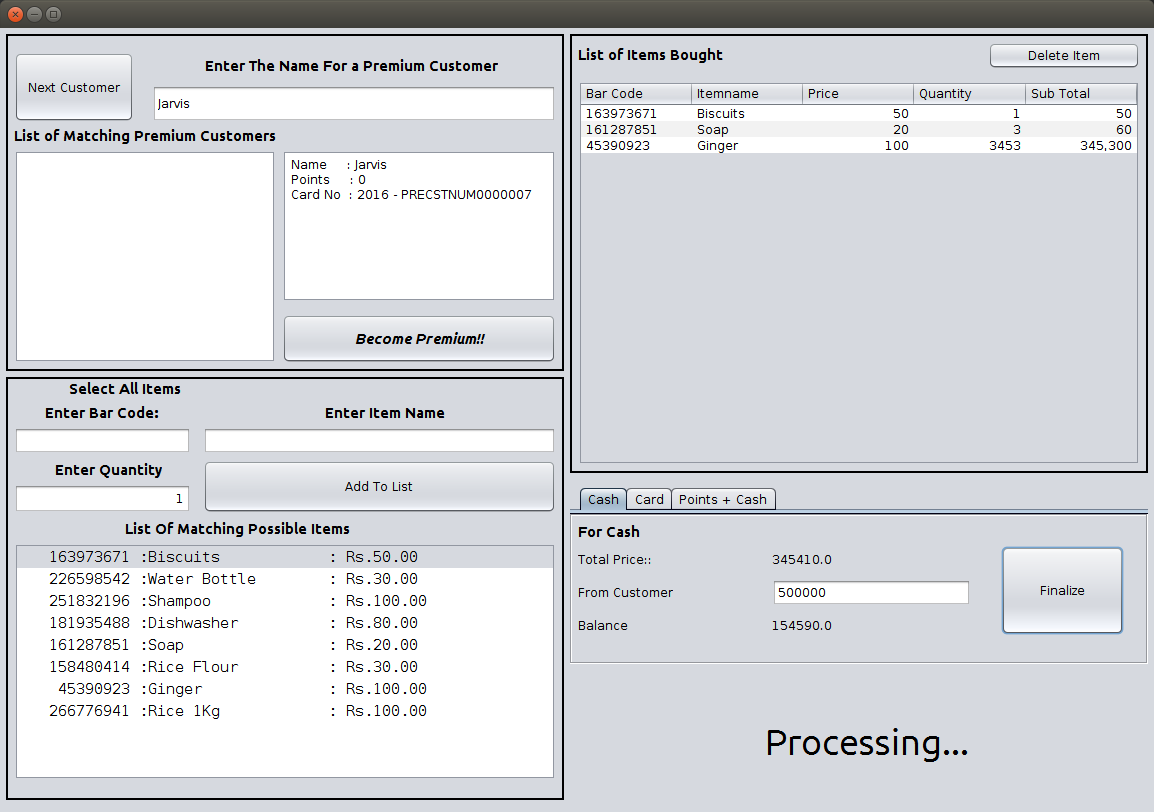


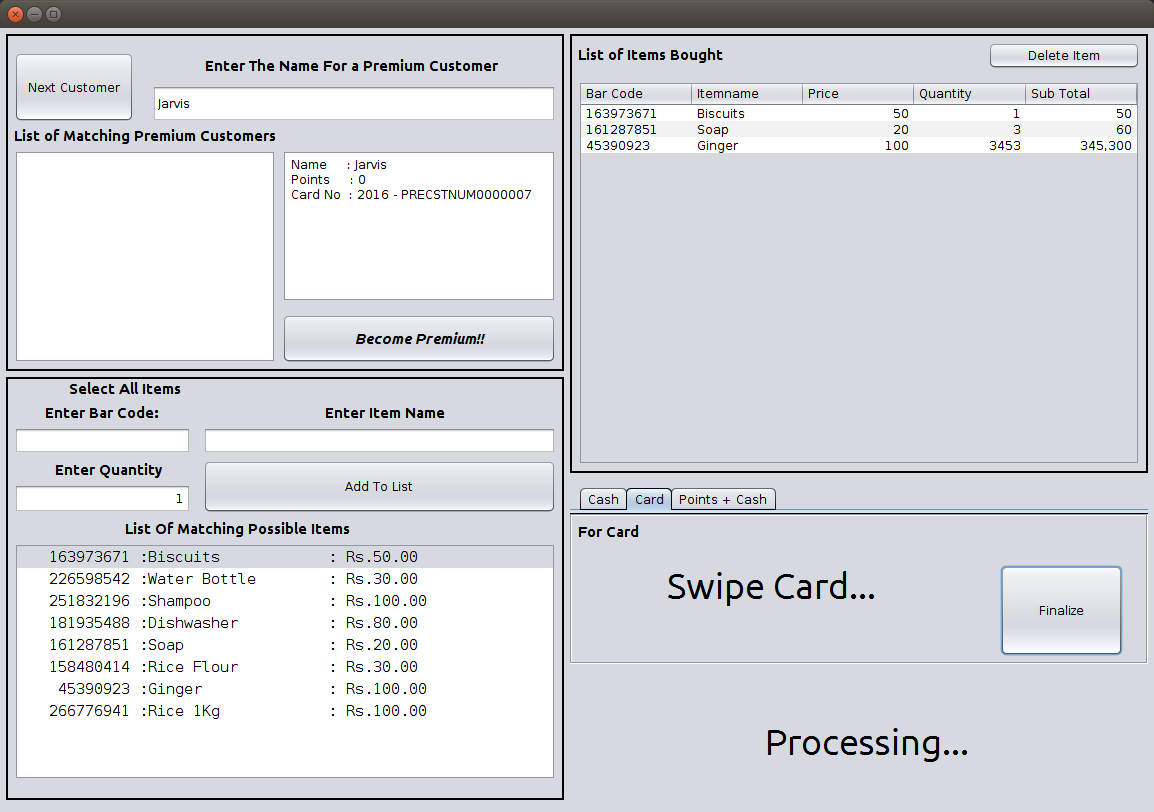
Here, A new Premium Customer is Created.





Here, The Item Basket is Illustrated.





Multiple Payment options shown.