1 AssEx1

```
import javax.swing.JOptionPane;
public class AssEx1 {
  /*Single lined style in main removes unnecessary pointers which would exist in main method only
      to pass objects to ultimately the GUI.
   *Obtaining name and balance by calling methods with String and double returns respectively
      directly in the Customer Account constructor
   *saves use of two unneeded name and balance pointers in the main.
   *Likewise, creating the customer account within the constructor of the GUI saves on a redundant
      pointer to the Customer Account object,
   *as this object is only accessed in the LWMGUI class, and its only purpose in the main is to be
      passed to the GUI, thus a pointer within the GUI
   *is sufficient, and one here would be redundant */
  public static void main(String[] args) {
  new LWMGUI(new CustomerAccount(promptName(), promptBalance()));
  }

  public static String promptName() {
    //prompts user for name with dialog box.
    //handles quit options for first dialog box as per specification
    String name = JOptionPane.showInputDialog("Please enter the name on the account:");
    if (name.equals(JOptionPane.CLOSED_OPTION) || name.equals(JOptionPane.CANCEL_OPTION) ||
        name.equals("")) {
      System.exit(0);
      return null;
    }
    return name; //return name ends method, gives name String as result of method.
  }

  public static double promptBalance() {
    //Prompts user for balance with dialog box.
    //handles quit options for second dialog box as per specification.

    double balance;
    for(;;) {//Infinite for loop will continue to prompt user until a valid input, or quit, is
        received.
      //showInputDialog returns a String, double value must be extracted from this input, use
          parseDouble.
      String balinput = JOptionPane.showInputDialog("Enter inital credit balance of account: ");
      if (balinput.equals(JOptionPane.CLOSED_OPTION) || balinput.equals(JOptionPane.CANCEL_OPTION))
          {
        System.exit(0);
        return 0;
      }
      try {
        balance = Double.parseDouble(balinput);
        //Customer enters initial value as credit +ve, need to invert for specified debit +ve
            system.
        //The IF is just to prevent -0, which is bad maths.
        if (balance != 0) {
          balance *= -1;
        }
        return balance;//return ends the method and breaks infinite loop, so break; statement
            redundant.
        //gives value of balance as result of method.
      }
      catch (NumberFormatException nfx) {//message displayed when an invalid input received, will
          repeat indefinitely until input valid.
        JOptionPane.showMessageDialog(null, "Enter a double value", "Error Report",
            JOptionPane.ERROR_MESSAGE);
      }
    }
  }

}
```

2 LWMGUI

```
import java.awt.*;
import java.awt.event.*;
import javax.swing.*;
public class LWMGUI implements ActionListener {
  /* LWMGUI method creates the GUI of the main window, not initial dialog boxes - handled by
      AssEx1 class.
   * Handles events for sale or return button presses. Transaction calculations in CustomerAccount
       class.
   * CustomerAccount object passed from AssEx1 in this class's constructor.
   * Wine object created in object, passed to CustomerAccount in actionPerformed for both sale and
       return.
   * Class has a constructor which creates GUI, adds starting balance.
   * Three methods set'X'Panel create the three main JPanels and add their constituent parts. Done
       for ease of problem solving and readability.
   * Event handling method processes sale or return buttons being pressed
   * clearInputs and update methods involved with processing inputs.*/

  //Declare and initialise GUI elements which change in program operation, ie. need to be passed
      between methods.
  private JFrame backFrame = new JFrame(); //backFrame
  private JButton returnButton = new JButton("Process Return");
  private JButton saleButton = new JButton("Process Sale");

  private TextField wineInput = new TextField();
  private TextField quantityInput = new TextField();
  private TextField priceInput = new TextField();

  private TextField lastWine = new TextField();
  private TextField lastCost = new TextField();
  private TextField balanceRemaining = new TextField();

  //Declare pointers to the Customer Account and Wine objects initialised later on.
  public CustomerAccount user;
  public Wine wine;


  public LWMGUI(CustomerAccount user) { //Constructor Method

    this.user = user; //Initialise the customer account as the one passed from main
    String username = this.user.getName(); //access username
    backFrame.setSize(640, 200);
    backFrame.setResizable(false);
    backFrame.setTitle("Lilybank Wine Merchants: "+username); //put username into title bar.
    backFrame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
    /*  Instructions in these methods could easily be in constructor.
     *  Parcelled off for ease of error finding in creating GUI.
     *  Helps with readability of how GUI is constructed also. */
    setInputPanel();
    setButtonPanel();
    setInfoPanel();

    balanceRemaining.setText(formatBalance()); //Displays initial balance immediately
    backFrame.setVisible(true);
  }


  @Override
  public void actionPerformed(ActionEvent e) { //handles pressing of the sale and return buttons.

    if (this.checkInput()) {//checkInput returns true if all inputs valid. Passes inputs to new
        wine object.

      if (e.getSource() == saleButton) {//procedure for sale button pressed
        lastCost.setText(String.format("%9.02f", this.user.updateBalanceSale(wine)));
        /*  calling either updateBalance methods in this manner processes the sale/return of the
            wine object passed into it
         *  wine object is updated prior by checkInput.
         *  method also returns the total cost of that sale/return, which is formatted and set in
             the lastCost box */
      }
      else if(e.getSource() == returnButton) {
        lastCost.setText(String.format("%9.02f", this.user.updateBalanceReturn(wine)));

      }
      purchaseFeedback();
      /* Irrespective of which button is pressed, wine name and current balance (latter handled by
          CustomerAccount class).
       * Only done if input determined to be valid, therefore within bounds of if statement on
           checkInput.*/
    }
    this.clearInputs(); //clear inputs irrespective of which of the two buttons is pressed and
        whether input is valid or not

  }

  private void setButtonPanel() { //Sets up the layout of central panel on window, which contains
      the two buttons
    JPanel buttonPanel = new JPanel();
    returnButton.setSize(10, 10);
    saleButton.setSize(10, 10);
    returnButton.addActionListener(this);
    saleButton.addActionListener(this);
    buttonPanel.setLayout(new GridBagLayout());
    buttonPanel.add(saleButton);
    buttonPanel.add(returnButton);
    backFrame.add(buttonPanel, BorderLayout.CENTER);
  }
  private void setInputPanel() { //Sets up the layout of top panel, which contains the three input
      boxes and their labels
    JPanel inputPanel = new JPanel();
    JLabel wineLabel = new JLabel("Wine Name: ");
    wineLabel.setHorizontalAlignment(SwingConstants.RIGHT);
```

```
    JLabel quantityLabel = new JLabel("Quantity: ");
    quantityLabel.setHorizontalAlignment(SwingConstants.RIGHT);
    JLabel priceLabel = new JLabel("Price: ");
    priceLabel.setHorizontalAlignment(SwingConstants.RIGHT);

    inputPanel.setLayout(new GridLayout(1, 6, 0, 0));
    inputPanel.add(wineLabel);
    inputPanel.add(wineInput);
    inputPanel.add(quantityLabel);
    inputPanel.add(quantityInput);
    inputPanel.add(priceLabel);
    inputPanel.add(priceInput);
    backFrame.add(inputPanel, BorderLayout.NORTH);
  }
  private void setInfoPanel() {//Sets up the layout of bottom panel, which contains the user
      feedback on balance and last purchase.
    JPanel infoPanel = new JPanel();
    JLabel lastWineLabel = new JLabel("Last Wine Purchased: ");
    lastWineLabel.setHorizontalAlignment(SwingConstants.RIGHT);
    JLabel lastCostLabel = new JLabel("Last Purchase Cost: ");
    lastCostLabel.setHorizontalAlignment(SwingConstants.RIGHT);
    JLabel balanceLabel = new JLabel("Balance(Debit +ve): ");
    balanceLabel.setHorizontalAlignment(SwingConstants.RIGHT);

    JPanel top = new JPanel();
    top.setLayout(new GridLayout(1,2, 0, 0));
    JPanel bottom = new JPanel();
    bottom.setLayout(new GridLayout(1,4, 0, 0));
    JPanel middle = new JPanel();
    middle.setSize(0, 5);
    GridBagLayout infoLayout = new GridBagLayout();
    GridBagConstraints con = new GridBagConstraints();
    infoPanel.setLayout(infoLayout);

    top.add(lastWineLabel);
    top.add(lastWine);
    lastWine.setEditable(false);
    lastWine.setBackground(Color.lightGray);
    bottom.add(lastCostLabel);
    bottom.add(lastCost);
    lastCost.setEditable(false);
    lastCost.setBackground(Color.lightGray);
    bottom.add(balanceLabel);
    bottom.add(balanceRemaining);
    balanceRemaining.setEditable(false);
    balanceRemaining.setBackground(Color.lightGray);
    con.gridx = 0;
    con.gridy = 0;
    infoLayout.setConstraints(top, con);
    infoPanel.add(top);
    con.gridx = 0;
    con.gridy = 1;
    infoLayout.setConstraints(middle, con);
    infoPanel.add(middle);
    con.gridx = 0;
    con.gridy = 2;
    infoLayout.setConstraints(bottom, con);
    infoPanel.add(bottom);
    backFrame.add(infoPanel, BorderLayout.SOUTH);
```

```
}

private boolean checkInput() {
  /* Method takes the inputs from each text field.
   * It checks the validity as per the specification of all three
   * If all three are valid it creates a creates a new wine object with the given input
       parameters
   * directs the class global wine pointer to this new wine, so that it maybe used for other
       methods.
   * This is fine as program only ever needs to keep track of a single wine at a time, as it can
       only process sales item by item.
   * Method returns a boolean variable to tell actionPerfomed method if inputs were valid at
       time of button press.
   * True only if all inputs are valid. */

  int quantity = 0;
  double bottleCost = 0;
  String name = wineInput.getText();
  if (name.equals("")) {//Only criteria on name is to not be empty.
    JOptionPane.showMessageDialog(null, "Require a Wine Name input","Error Message",
        JOptionPane.ERROR_MESSAGE);
    return false;
  }
  try {
    quantity = Integer.parseInt(quantityInput.getText());
  }
  catch (NumberFormatException nfx) {//stops the method here if quantity not an integer, gives
      according error message
    JOptionPane.showMessageDialog(null, "Require an integer value for Quantity input", "Error
        Message", JOptionPane.ERROR_MESSAGE);
    return false;
  }
  try {
    bottleCost = Double.parseDouble(priceInput.getText());
  }
  catch (NumberFormatException nfx) {//stops method here if price not a double, gives according
      error message.
    JOptionPane.showMessageDialog(null, "Require a valid Price input", "Error Message",
        JOptionPane.ERROR_MESSAGE);
    return false;
  }


  if ( quantity > 0 && bottleCost > 0) {
    /* Only if input types are all valid; quantity and cost are positive valued, then wine object
        is created and method returns true
     * processing either a sale or return respectively in actionPerformed. */
    wine = new Wine(wineInput.getText(),bottleCost,quantity);
    return true;
  }
  else { //Shows an error method if types are valid but negative/zero entries present for
      quantity or cost
    JOptionPane.showMessageDialog(null, "Quantity and Price require positive valued, non-zero
        inputs.", "Error Message", JOptionPane.ERROR_MESSAGE);
    return false;
  }
}
```

```
  private void clearInputs() { //Clears the inputs
    wineInput.setText(" ");
    quantityInput.setText(" ");
    priceInput.setText(" ");
  }

  private void purchaseFeedback() {
    /* Updates text in wine name and balance remaining.
     * Both are independent of whether sale or return processed:
     *    wine name is not involved in calculations
     *    balance is an instance variable of CustomerAccount and can be accessed the same
         regardless of which transaction is processed
     * Last cost is not updated by this method as it is passed from the sale/return methods
         respectively in both cases */
    lastWine.setText(wine.getName());
    balanceRemaining.setText(formatBalance());
  }

  public String formatBalance() {//Formats balance display to two decimal places, and negative
       balances as positive with CR (credit)
    if (this.user.getBalance() < 0) {
      String output = String.format("%9.02f", -this.user.getBalance())+"CR";
      /*number will be negative, so invert to remove minus sign and add CR.
       * Inversion only part of string formatting so does not affect stored balance value for
           futher transactions.*/
      return output; //returns balance formatted as a strung
    }
    else {
      return String.format("%9.02f", this.user.getBalance()); //simply format to two dp for
          positive (debit) balanaces.
    }
  }


}
```

---

3 CustomerAccount

```
public class CustomerAccount {
  /* The CustomerAccount class contains the constructor, methods to return instance variables
   * and handles balance updating in both sales and returns.
   * Declaration of name and balance instance variables.
   * serviceCharge is final as value not expected to change during operation,
   * declared here for ease of later alteration.*/

  private final double serviceCharge = 0.8; //20% service charge on returns
  private String name;
  private double balance;

  public CustomerAccount(String name, double balance) {
    //Constructor initialises instance variables with the values passed into method.
    this.name = name;
    this.balance = balance;
  }

  public String getName() {//Accessor method for Customer Name
```

```
    return name;
  }

  public double getBalance() {//Accessor method for Account Balance
    return balance;
  }


  public double updateBalanceSale(Wine wine) { //Processes a sale based on wine object passed by
       LWMGUI class
    //updates the instance balance but also returns the salePrice for purposes of user feedback on
         infoPanel.
    double salePrice = wine.getBottleCost()*wine.getQuantity();
    this.balance += salePrice;
    return salePrice;
  }

  public double updateBalanceReturn(Wine wine) {//Processes a return based on wine object passed
       by LWMGUI class
    //updates the instance balance but also returns the returnPrice for purposes of user feedback
         on infoPanel.
    double returnPrice = wine.getBottleCost()*wine.getQuantity()*serviceCharge;
    this.balance -= returnPrice;
    return returnPrice;
  }

}
```

4 Wine

```
public class Wine {
  //The Wine class contains the constructor for the object and methods to return instance variables

  //Declare instance variable pointers.
  private String name;
  private double bottleCost;
  private int quantity;

  public Wine(String name, double bottleCost, int quantity) {
    //Constructor initialised instance variables with the values passed into it.
    this.name = name;
    this.bottleCost = bottleCost;
    this.quantity = quantity;
  }

  public String getName() { //accessor method for wine name
    return name;
  }

  public double getBottleCost() { //accessor method for cost per bottle
    return bottleCost;
  }

  public int getQuantity() {//accessor method for quantity
    return quantity;
  }
}
```