You will need to put have more specific import statements for each class used, not just the whole packages

```
import java.awt.*;

import java.awt.event.*;

import java.sql.*;

import java.text.*;

import java.util.*;

import javax.swing.*;
```

In class you will need all of the following declarations as instance variables :

```
  // JLabel for Restaurant

  private JLabel restaurantJLabel;

  // JPanel for Waiter Information

  private JPanel waiterJPanel;

  // JLabel and JComboBox for Table Number

  private JLabel tableNumberJLabel;

  private JComboBox tableNumberJComboBox;

  // JLabel and JTextField for Waiter Name

  private JLabel waiterNameJLabel;

  private JTextField waiterNameJTextField;

  // JPanel for Menu Items

  private JPanel menuItemsJPanel;

  // JLabel and JComboBox for Beverage

  private JLabel beverageJLabel;

  private JComboBox beverageJComboBox;

  // JLabel and JComboBox for appetizer

  private JLabel appetizerJLabel;

  private JComboBox appetizerJComboBox;
```

```
// JLabel and JComboBox for Main Course

private JLabel mainCourseJLabel;

private JComboBox mainCourseJComboBox;

// JLabel and JComboBox for Dessert

private JLabel dessertJLabel;

private JComboBox dessertJComboBox;

// JLabel and JTextField for Subtotal

private JLabel subtotalJLabel;

private JTextField subtotalJTextField;

// JLabel and JTextField for Tax

private JLabel taxJLabel;

private JTextField taxJTextField;

// JLabel and JTextField for Total

private JLabel totalJLabel;

private JTextField totalJTextField;

// JButton for Save Table

private JButton saveTableJButton;

// JButton for Calculate Bill

private JButton calculateBillJButton;

// JButton for Pay Bill

private JButton payBillJButton;

// constant for tax rate

private final static double TAX_RATE = 0.05;

// declare instance variables for database processing

private Connection myConnection;

private Statement myStatement;

private ResultSet myResultSet;
```

```
  // other instance variables

  private ArrayList billItems = new ArrayList();

  private double subtotal;
```

Constructor will need two String parameters to receive the database username and password which will be provided as command line arguments (module 7)

It should also call method CreateUserInterface

```
  :

  public RestaurantBillCalculator(

    String databaseUser, String databasePassword )

    {

    // make database connection

    try

    {

    String url = "jdbc:mysql://localhost:3306/restaurant";


   String driver = "com.mysql.jdbc.Driver";

 try

{

   Class.forName(driver).newInstance();

   Connection conn = DriverManager.getConnection(url,databaseUser,  databasePassword)


      myStatement = myConnection.createStatement();

    }

    catch ( SQLException exception )

    {

      exception.printStackTrace();
   }

   catch ( ClassNotFoundException exception )

   {

     exception.printStackTrace();

   }


   // set up GUI

   createUserInterface();
```

method createUserInterface should create and position GUI components; register event handlers

```
 private void createUserInterface()

 {

   // get content pane for attaching GUI components

   Container contentPane = getContentPane();


   // enable explicit positioning of GUI components

   contentPane.setLayout( null );


   // set up restaurantJLabel

   restaurantJLabel = new JLabel();

   restaurantJLabel.setBounds( 80, 8, 128, 24 );

   restaurantJLabel.setText( "Restaurant" );

   restaurantJLabel.setFont(

     new Font( "SansSerif", Font.BOLD, 16 ) );

   contentPane.add( restaurantJLabel );


   // set up waiterJPanel
```

```
createWaiterJPanel();

contentPane.add( waiterJPanel );


// set up menuItemsJPanel

createMenuItemsJPanel();

contentPane.add( menuItemsJPanel );


// set up subtotalJLabel

subtotalJLabel = new JLabel();

subtotalJLabel.setBounds( 15, 340, 56, 16 );

subtotalJLabel.setText( "Subtotal:" );

contentPane.add( subtotalJLabel );


// set up subtotalJTextField

subtotalJTextField = new JTextField();

subtotalJTextField.setBounds( 70, 340, 80, 20 );

subtotalJTextField.setEditable( false );

subtotalJTextField.setBorder(

  BorderFactory.createLoweredBevelBorder() );

subtotalJTextField.setHorizontalAlignment( JTextField.RIGHT );

contentPane.add( subtotalJTextField );


Set up saveTableJButton

saveTableJButton = new JButton();

saveTableJButton.setBounds( 167, 328, 90, 24 );

saveTableJButton.setText( "Save Table" );

saveTableJButton.setBorder(
  BorderFactory.createRaisedBevelBorder() );
saveTableJButton.setEnabled( false );
contentPane.add( saveTableJButton );
saveTableJButton.addActionListener(

  new ActionListener()  // anonymous inner class
  {
    // event handler called when saveTableJButton is clicked
    public void actionPerformed( ActionEvent event )
    {
      saveTableJButtonActionPerformed( event );
    }

  } // end anonymous inner class

); // end addActionListener



  } // end anonymous inner class

); // end addActionListener

// set properties of application's window
setTitle( "Restaurant Bill Calculator" ); // set window title
setSize( 280, 500 ); // set window size
setVisible( true );  // display window
```

```
    // ensure database connection is closed
    // when user quits application
    addWindowListener(

      new WindowAdapter()  // anonymous inner class
      {
        // event handler called when close button is clicked
        public void windowClosing( WindowEvent event )
        {
          frameWindowClosing( event );
        }

      } // end anonymous inner class

    ); // end addWindowListener

  } // end method createUserInterface
```

Create method set up waiterJPanel

```
  private void createWaiterJPanel()
  {
    waiterJPanel = new JPanel();
    waiterJPanel.setBounds( 20, 48, 232, 88 );
    waiterJPanel.setBorder( BorderFactory.createTitledBorder(
      BorderFactory.createEtchedBorder(),
      "Waiter Information" ) );
  waiterJPanel.setLayout( null );
```

create method set up menuItemsJPanel

```
private void createMenuItemsJPanel()
{
  menuItemsJPanel = new JPanel();
  menuItemsJPanel.setBounds( 20, 152, 232, 152 );
  menuItemsJPanel.setEnabled( false );
  menuItemsJPanel.setBorder( BorderFactory.createTitledBorder(
    BorderFactory.createEtchedBorder(), "Menu Items" ) );
  menuItemsJPanel.setLayout( null );

  // set up beverageJLabel
  beverageJLabel = new JLabel();
  beverageJLabel.setBounds( 8, 24, 80, 24 );
  beverageJLabel.setText( "Beverage:" );
  menuItemsJPanel.add( beverageJLabel );

  // set up beverageJComboBox
  beverageJComboBox = new JComboBox();
  beverageJComboBox.setBounds( 88, 24, 128, 25 );
  beverageJComboBox.setEnabled( false );
  menuItemsJPanel.add( beverageJComboBox );
  beverageJComboBox.addItemListener(

    new ItemListener()  // anonymous inner class
    {
      // event handler called when item in beverageJComboBox
    // is selected
    public void itemStateChanged( ItemEvent event )
    {
      beverageJComboBoxItemStateChanged( event );
    }

  } // end anonymous inner class

); // end addItemListener

// add items to beverageJComboBox
beverageJComboBox.addItem( "" );
loadCategory( "Beverage", beverageJComboBox );

    }

  } // end anonymous inner class

); // end addItemListener
```

Method loadTableNumbers is used to load numbers to tableNumberJComboBox - this info will come from the database

```
private void loadTableNumbers()
{
  // read all table numbers from database
  try
  {
    // obtain all table numbers
    myResultSet = myStatement.executeQuery(
      "SELECT tableNumber FROM restaurantTables" );
    // add numbers to tableNumberJComboBox
    while ( myResultSet.next() == true )
    {
      tableNumberJComboBox.addItem(
        String.valueOf( myResultSet.getInt(
              "tableNumber" ) ) );
    }
    myResultSet.close(); // close myResultSet
  } // end try

  // catch SQLException
  catch ( SQLException exception )
  {
    exception.printStackTrace();
  }

} // end method loadTableNumbers
```

Method loadTableNumbers is used to menu items to the various combo boxes - this info will come from the database

```
 private void loadCategory(

   String category, JComboBox categoryJComboBox )

 {

   // read all items from database for specified category

   try

   {

     // obtain all items in specified category

     myResultSet = myStatement.executeQuery( "SELECT name FROM "

       + "menu WHERE category = '" + category + "'" );


     // add items to JComboBox

     while ( myResultSet.next() == true )

     {

       categoryJComboBox.addItem(

         myResultSet.getString( "name" ) );

     }

     myResultSet.close(); // close myResultSet

  } // end try


   // catch SQLException

   catch ( SQLException exception )

   {

     exception.printStackTrace();

   }

 } // end method loadCategory
```

This method is called when the user selects an item on the tableNumberJComboBox, the ItemStateChangedEvent for that JCombo will be called

```
private void tableNumberJComboBoxItemStateChanged(  ItemEvent event )

{

  String selectedTableNumber = ( String ) event.getItem();

  // select a number

  if ( !selectedTableNumber.equals( "" ) &&

    event.getStateChange() == ItemEvent.SELECTED )

  {

    // load table data

    try

    {

      // get table data

      myResultSet = myStatement.executeQuery( "SELECT * FROM "

        + "restaurantTables WHERE tableNumber = " +

        Integer.parseInt( selectedTableNumber ) );


      // if myResultSet not empty

      if ( myResultSet.next() == true )

      {

        waiterNameJTextField.setText(

          myResultSet.getString( "waiterName" ) );

        subtotal = myResultSet.getDouble( "subtotal" );

        displayTotal( subtotal );

      }

      myResultSet.close(); // close myResultSet

    } // end try
```

```
      // catch SQLException
      catch ( SQLException exception )
      {
        exception.printStackTrace();
      }

      // enable JComboBoxes in menuItemsJPanel
      // disable JComboBox in waiterJPanel
      menuItemsJPanel.setEnabled( true );

      waiterJPanel.setEnabled( false );
      tableNumberJComboBox.setEnabled( false );
      saveTableJButton.setEnabled( true );
      calculateBillJButton.setEnabled( true );
      payBillJButton.setEnabled( true );

    } // end if

  } // end method tableNumberJComboBoxItemStateChanged
```

T

This method will be called when an item is selected on the beverage Combo - similar methods will be required for the other menu combos (appetizer, main, desert)

```
private void beverageJComboBoxItemStateChanged( ItemEvent event )

{

  // select an item

  if ( event.getStateChange() == ItemEvent.SELECTED )

  {

    billItems.add(

      ( String ) beverageJComboBox.getSelectedItem() );

  }


} // end method beverageJComboBoxItemStateChanged
```

Clicking the Save table button will cause the table total to be calculated and then save to the database

```
// user click saveTableJButton

private void saveTableJButtonActionPerformed( ActionEvent event )

{

  // calculate subtotal

  subtotal = calculateSubtotal();

  // update subtotal in database

  updateTable();

  // reset JFrame

  resetJFrame();

} // end method saveTableJButtonActionPerformed
```

method payBillJButtonActionPerformed calls the same methods as saveTableJButtonActionPerformed, but the subtotal is set to 0

Method updateTable creates and updates statement to save the subtotal for each table to the database

```
private void updateTable()
{
  // update subtotal for table number in database
  try
  {
    myStatement.executeUpdate( "UPDATE restaurantTables SET " +
      "subtotal = " + subtotal + " WHERE tableNumber = " +
      Integer.parseInt(
        ( String ) tableNumberJComboBox.getSelectedItem() ) );
  }
  catch ( SQLException exception )
  {
    exception.printStackTrace();
  }

} // end method updateTable
```

Method resetJFrame returns all controls to there start values, the method below is partly completed only

```
private void resetJFrame()

{

  // reset instance variable

  billItems = new ArrayList();


  // reset and disable menuItemsJPanel

  menuItemsJPanel.setEnabled( false );

  beverageJComboBox.setSelectedIndex( 0 );

  beverageJComboBox.setEnabled( false );

  // reset and enable waiterJPanel

  waiterJPanel.setEnabled( true );

  tableNumberJComboBox.setEnabled( true );

  tableNumberJComboBox.setSelectedIndex( 0 );

  waiterNameJTextField.setText( "" );

  // clear JTextFields

  subtotalJTextField.setText( "" );

  taxJTextField.setText( "" );

  totalJTextField.setText( "" );

  // disable JButtons

  saveTableJButton.setEnabled( false );

  calculateBillJButton.setEnabled( false );

  payBillJButton.setEnabled( false );

} // end method resetJFrame
```

Code to run when the user clicks the Calculate Bill button

```
 calculateBillJButtonActionPerformed(
   ActionEvent event )
{
   double total = calculateSubtotal();


   // display subtotal, tax and total
   displayTotal( total );


} // end method calculateBillJButtonActionPerformed
```

method displayTotal is used to display subtotal, tax and total

```
private void displayTotal( double total )
{
   // define display format
   DecimalFormat dollars = new DecimalFormat( "$0.00" );
   // display subtotal
   subtotalJTextField.setText( dollars.format( ? ) );
   // calculate and display tax
   double tax = total * TAX_RATE;
   taxJTextField.setText( dollars.format( ? ) );
   // display total
   totalJTextField.setText(
      dollars.format( total + tax ) );


} // end method displayTotal
```

Method calculateSubtotal obtains the price of each food item from the database and then totals the prices

```
private double calculateSubtotal()  {

  double total = subtotal;

  Object[] items = billItems.toArray();

  // get data from database

  try  {

    // get price for each item in items array

    for ( int i = 0; i < items.length; i++ )    {

      // execute query to get price

      myResultSet = myStatement.executeQuery( "SELECT price " +

        "FROM menu WHERE name = '" + ( String ) items[ i ] +

        "'" );

      // myResultSet not empty

      if ( myResultSet.next() == true )

      {

        total += myResultSet.getDouble( "price" );

      }

      myResultSet.close(); // close myResultSet

    } // end for

  } // end try

  // catch SQLException

  catch ( SQLException exception )  {

    exception.printStackTrace();

  }

  return total;

} // end method calculateSubtotal
```

The method frameWindowClosing will be called when the program window is closed.

```
// user close window
private void frameWindowClosing( WindowEvent event )
{
  // close myStatement and database connection
  try
  {
    myStatement.close();
    myConnection.close();
  }
  catch ( SQLException sqlException )
  {
    sqlException.printStackTrace();
  }
  finally
  {
    System.exit( 0 );
  }

}  // end method frameWindowClosing

// main method
```

The main method should accept 2 command line arguments representing the databaseUserName and databasePassword, it these are not correct the program display and error message

```
  public static void main( String[] args )

  {

    // check command-line arguments

    if ( args.length == 2 )

    {

      // get command-line arguments

      String databaseUserName= args[ 0 ];

      String databasePassword = args[ 1 ];


      // create new RestaurantBillCalculator

      RestaurantBillCalculator application =

        new RestaurantBillCalculator(

          databaseDriver, databaseURL );

    }

    else

    {

      System.out.println( "Usage: java " +

        "RestaurantBillCalculator databaseUser databasePassword" );

    }


  } // end method main


} // end class RestaurantBillCalculator
```