Appendix B Buyer Program for JEEDI System

```java
import java.awt.*;
import java.awt.event.*;
import java.sql.*;
import java.text.*;

/*This class builds a frame that has the ability to close*/
class BuyerFrame extends Frame implements WindowListener
{
    /*constructor, has title of frame, width, length, x and y as a parameters*/
    public BuyerFrame()
    {
        setTitle("buyer");
        setSize(500, 180);
        setLocation(100, 50);
        addWindowListener(this);
    }

    /* lets the window close when the close button is pressed*/
    public void windowClosing(WindowEvent e)
    {
        System.exit(0);
    }

    /* methods not need to be implemented for this class*/
    public void windowClosed(WindowEvent e)
    {
    }
    public void windowIconified(WindowEvent e)
    {
    }
    public void windowOpened(WindowEvent e)
    {
    }
    public void windowDeiconified(WindowEvent e)
    {
    }
    public void windowActivated(WindowEvent e)
    {
    }
    public void windowDeactivated(WindowEvent e)
    {
    }
}

/*A frame with the properties of a BuyerPlaceOrder object*/
class BuyerPlaceOrder extends BuyerFrame
{
    BPO_GUI gui = new BPO_GUI(this);
    Connection connect, connectSel;
    String _orderNumber;

    /* constructor calls the super constructor */
    public BuyerPlaceOrder()
    {
        super();
        setLayout(new FlowLayout(FlowLayout.CENTER));
        gui.make();
        add(gui._frame);

        String buyUrl = "jdbc:odbc:Buyer Database";
        String selUrl = "jdbc:odbc: Seller Database";
        try
        {
        }
    }
```
Class.forName("sun.jdbc.odbc.JdbcOdbcDriver");
}
catch(ClassNotFoundException e)
{
    System.out.println(e.getMessage());
}
try
{
    connect = DriverManager.getConnection(buyUrl);
    connectSel = DriverManager.getConnection(selUrl);
}
catch(SQLException e)
{
    System.out.println(e.getMessage());
} listUnsent(); listUnac();

/*list the unsent orders in the buyer database*/
public void listUnsent()
{
    gui._area.setText(""y; try
    {
        Statement buyState = connect.createStatement();
        ResultSet rs = buyState.executeQuery("SELECT * FROM [Purchase Order]");
        while(rs.next())
        {
            String sent = rs.getString("Order_Sent");
            if(Integer.parseInt(sent) == 0)
                gui.add_text(gui._area, rs.getString("OrderNum"));
        }
    }
catch(SQLException e)
    {
        System.out.println(e.getMessage());
    } gui.add_text(gui._area, "--------------"); gui.add_text(gui._area, "no more orders");
}

/*lists the unconfirmed orders in the buyer database*/
public void listUnac()
{
    gui._area2.setText(""y; try
    {
        Statement buyState = connect.createStatement();
        ResultSet rs = buyState.executeQuery("SELECT * FROM [Purchase Order]");
        while(rs.next())
        {
            String sent = rs.getString("Confirmation_Rec");
            if(Integer.parseInt(sent) == 0)
                gui.add_text(gui._area2, rs.getString("OrderNum"));
        }
    }

92
catch(SQLException e) {
    System.out.println(e.getMessage());
}
gui.add_text(gui._area2, "---------");
gui.add_text(gui._area2, "no more orders");

public void sendOrderToServer() {
    _orderNumber = gui.get_orderNumber();
    if(_orderNumber.length() > 0)
    {
        gui.update_label(gui._sentLabel, "trying-to-send order: " + _orderNumber);
        pause(50000000);
        if(OrderInDataBase())
            updateDatabase();
        else
            cannotSend();
    }
    else
    {
        gui.update_label(gui._sentLabel, "No order number");
        pause(40000000);
    }
    reset();
}

public void updateDatabase() {
    String _orderNumber = gui.get_orderNumber();
    try
    {
        Statement buyState = connect.createStatement();
        ResultSet rs = buyState.executeQuery("SELECT * FROM [Purchase Order]");
        while(rs.next())
        {
            String orderNum = rs.getString("Order Num");
            if(Integer.parseInt(orderNum) == Integer.parseInt(_orderNumber))
            {
                String sent = rs.getString("Order_Sent");
                int day = rs.getInt("R_Day");
                if(Integer.parseInt(sent) == 0)
                {
                    gui.update_label(gui._sentLabel, "Order:  " + _orderNumber + " ... now sending!");
                    String updateStatement = "UPDATE [Purchase Order] SET Order_Sent = 1 WHERE OrderNum = " + _orderNumber + " ORDER = " + Integer.parseInt(_orderNumber) + " WHERE OrderSent = 0"
                    buyState.executeUpdate(updateStatement);
                    sendOrder(Integer.parseInt(_orderNumber), day);
                } else
                {
                    gui.update_label(gui._sentLabel, "Order:  " + _orderNumber + " is already sent.");
                    pause(50000000);
                }
            } else
            {
                gui.update_label(gui._sentLabel, "Order:  " + _orderNumber + " is already sent.");
                pause(50000000);
            }
        }
    } catch(SQLException e) {
        System.out.println(e.getMessage());
    }
}
reset();
    }
    break;/***get out of while loop*/
    }
    }
    catch(SQLException e)
    {
        System.out.println(e.getMessage());
    }
    }
    */sends r_d_date to seller database*/
    public void sendOrder(int seller_order_num, int rd_day)
    {
        String url = "jdbc:odbc:Seller Database";
        try
        {
            Class.forName("sun.jdbc.odbc.JdbcOdbcDriver");
        }
        catch(ClassNotFoundException e)
        {
            System.out.println(e.getMessage());
        }
        try
        {
            connectSel = DriverManager.getConnection(url);
            Statement st = connectSel.createStatement();
            //System.out.println(r_d_date);
            String updateStatement = "UPDATE [Seller_Order] SET RD_Day = " + rd_day + " WHERE CustomerPO = " + seller_order_num;
            st.executeUpdate(updateStatement);
        }
        catch(SQLException e)
        {
            System.out.println(e.getMessage());
        }
    }
    /*after order sent, change label to read 'order sent' or 'cannot send*/
    public void cannotSend()
    {
        _orderNumber = gui.get_orderNumber();
        gui.update_label(gui._sentLabel, "Cannot Send");
        pause(50000000);
        gui.update_label(gui._sentLabel, "order:" + _orderNumber + " not in database");
    }
    public void pause(int speed)
    {
        for(int i = 0; i < speed; i ++);
    }
    /*returns true if order is in server database*/
    public boolean OrderInServer()
    {
String _orderNumber = gui.get_orderNumber();
try
{
    Statement selState = connectSel.createStatement();
    ResultSet result = selState.executeQuery("SELECT * FROM Seller_Order");
    while(result.next())
    {
        if(result.getInt("CustomerPO") == Integer.parseInt(_orderNumber))
            return true;
    }
} catch(SQLException e)
{
    System.out.println(e.getMessage());
    return false;
}
/*returns true if order is in database*/
public boolean OrderInDataBase()
{
    try
    {
        Statement buyState = connect.createStatement();
        ResultSet rs = buyState.executeQuery("SELECT * FROM [Purchase Order]");
        while(rs.next())
        {
            String orderNum = rs.getString("OrderNum");
            if(Integer.parseInt(orderNum) == Integer.parseInt(_orderNumber))
                return true;
        }
    } catch(SQLException e)
    {
        System.out.println(e.getMessage());
    }
    return false;
}
/*resets all fields of applet*/
public void reset()
{
    gui.update_label(gui._sentLabel, "Place Order");
    gui.reset_field();
}
}

class BPO_GUI
{
    Color _grey = new Color(155, 165, 143);
    Label _itemLabel = new Label("Order #");
    Label _sentLabel = new Label("Place Order", _sentLabel.CENTER);
    TextField _itemField = new TextField(15);
    Button _unsentButton = new Button("Unsent Orders");
    Panel _unsentBPanel = makePanel(new BorderLayout(), _grey);
    TextArea _area = new TextArea(6, 15);
}
Button _unacOrders = new Button("Unconfirmed");
TextArea _area2 = new TextArea(6, 15);
Button _placeOrderButton = new Button("Place Order");
Panel _unacPanel = makePanel(new BorderLayout(), _grey);
Panel _unsentPanel = makePanel(new BorderLayout(), _grey);
Panel _frame = makePanel(new FlowLayout(), _grey);
Panel _appletPanel = makePanel(new BorderLayout(), _grey);
Panel _itemPanel = makePanel(new FlowLayout(FlowLayout.CENTER), _grey);
Panel _buttonPanel = makePanel(new FlowLayout(FlowLayout.CENTER), _grey);
BuyerPlaceOrder _applet;
BPO_Action_Listener listener;

public BPO_GUI(BuyerPlaceOrder BPO_applet)
{
    _applet = BPO_applet;
    listener = new BPO_Action_Listener(_applet);
}

private Panel makePanel(LayoutManager layout, Color color)
{
    Panel panel = new Panel();
    panel.setLayout(layout);
    panel.setBackground(color);
    return panel;
}

public void make()
{
    _unacPanel.add("North", _unacOrders);
    _unacPanel.add("South", _area2);
    _unsentPanel.add(_unsentButton);
    _unsentPanel.add("North", _unsentBPanel);
    _unsentPanel.add("South", _area);
    _itemPanel.add(_itemLabel);
    _itemPanel.add(_itemField);
    _buttonPanel.add(_placeOrderButton);
    _appletPanel.add("North", _sentLabel);
    _appletPanel.add("Center", _itemPanel);
    _appletPanel.add("South", _buttonPanel);
    _frame.add(_unsentPanel);
    _frame.add(_appletPanel);
    _frame.add(_unacPanel);
    _placeOrderButton.addActionListener(listener);
    _unsentButton.addActionListener(listener);
    _unacOrders.addActionListener(listener);
}

public String get_orderNumber()
{
    return _itemField.getText();
}

public void update_label(Label label, String message)
{
    label.setText(message);
}
public void reset_field()
{
    _itemField.setText("\n");
}

public void add_text(TextArea ar, String str)
{
    ar.append(str + "\n");
}

class BPO_Action_Listener implements ActionListener
{
    private BuyerPlaceOrder _frame;

    public BPO_Action_Listener(BuyerPlaceOrder listening_frame)
    {
        _frame = listening_frame;
    }

    public void actionPerformed(ActionEvent event)
    {
        String command = (String)event.getActionCommand();
        if(command.equals("Place Order"))
        {
            _frame.sendOrderToServer();
            _frame.listUnsent();
            _frame.listUnac();
        }
        else if(command.equals("Unsent Orders"))
        {
            _frame.listUnsent();
        }
        else if(command.equals("Unconfirmed"))
        {
            _frame.listUnac();
        }
        else
        {
            System.out.println("ERROR");
        }
    }
}

public class BuyerProg
{
    public static void main(String args[])
    {
        BuyerPlaceOrder buyer = new BuyerPlaceOrder();
        buyer.show();
    }
}