Criterion C: Development

Techniques Used:

1. Graphical User interface (GUI)- increases usability and attractiveness of program.

2. Methods and Variables- used for creation and reuse of a method through a single label. The variable can be used to store data values within the program.

4. New Reservation (Algorithmic thinking)-

5. Double Booking Prevention(algorithmic thinking)

6. Modify Reservation (algorithmic thinking)

- Main Page.java
- CustRec.java
- PriceCalculations.java

7. Cancel Reservation (algorithmic thinking)

- Main Page.java
- CustRec.java
- PriceCalculations.java

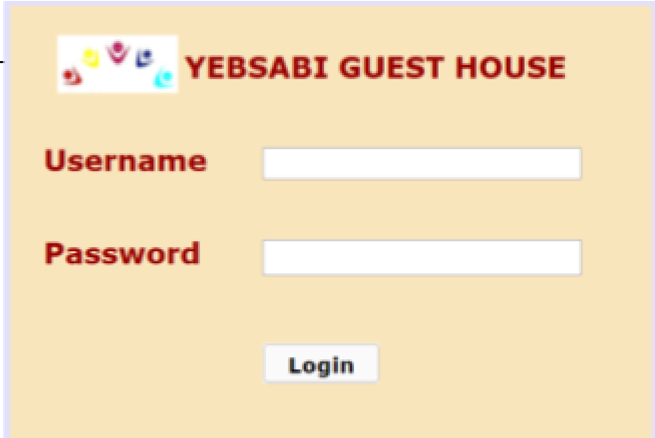
8. Search Function (algorithmic thinking- utilizes while loop)

9. Price Calculations

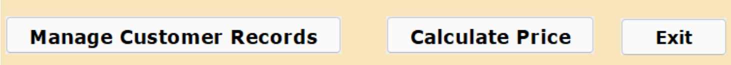
Graphical User Interface (GUI)

The GUI on each form is designed to continuously follow after the other. Each form has buttons such as ‘Log In’, ‘Back’, ‘Exit’ that increase the usability and attractiveness for the user.

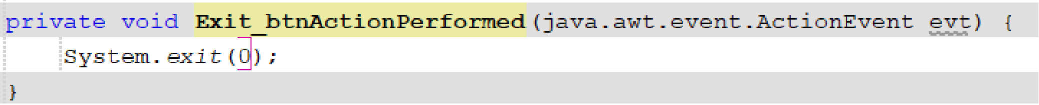
Each form has the company’s logo in the corner to increase the system’s appeal.



The Login form has a ‘Login’ button to transfer the user to the next screen. The inputted login details are compared with the saved login detail to see if they match. If incorrect password or username is chosen, user is denied access.

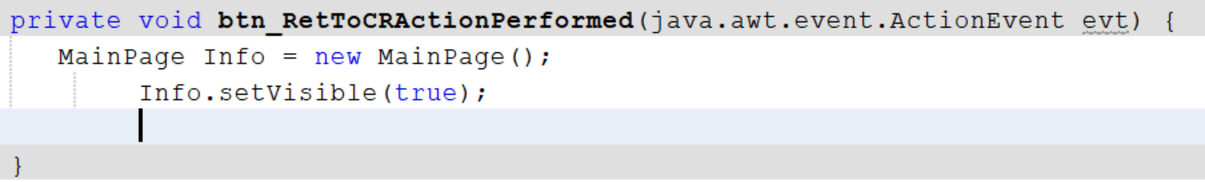


This is the code for the exit button on all the forms other than the log in form.



```
private void Exit_btnActionPerformed(java.awt.event.ActionEvent evt) {
    System.exit(0);
}
```

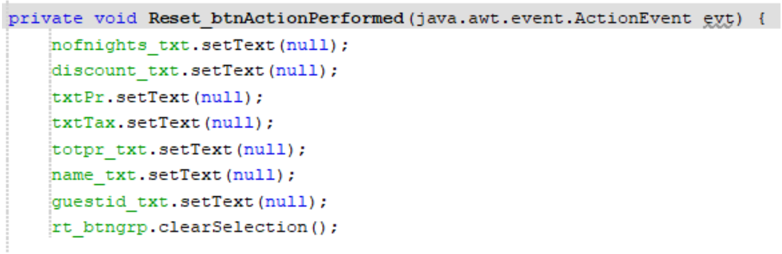
This is the code for the return button in both the forms, CustMain and PriceCalculator_1.



```
private void btn_RetToCRActionPerformed(java.awt.event.ActionEvent evt) {
    MainPage Info = new MainPage();
        Info.setVisible(true);

}
```

The buttons above are included on the main room reservations page.

The program also uses date choosers so the user can select which date they want from a calendar, rather than manually type it out.



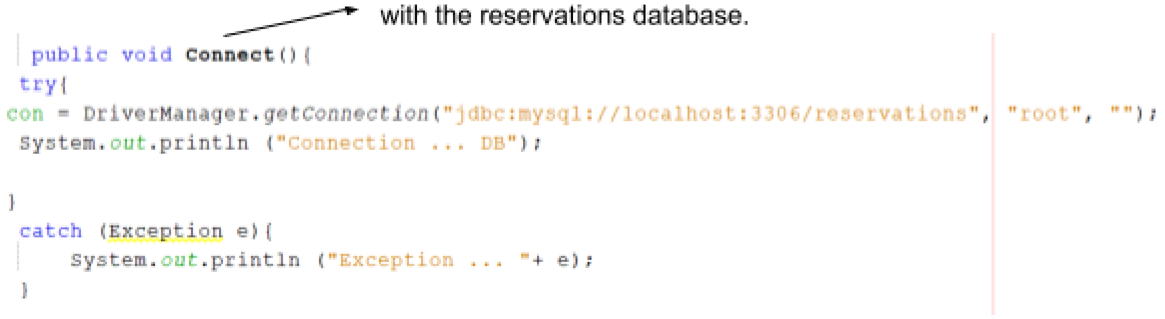
```
private void Reset_btnActionPerformed(java.awt.event.ActionEvent evt) {
    nofnights_txt.setText(null);
    discount_txt.setText(null);
    txtPr.setText(null);
    txtTax.setText(null);
    totpr_txt.setText(null);
    name_txt.setText(null);
    guestid_txt.setText(null);
    rt_btngrp.clearSelection();
```

The ‘Reset’ GUI is responsible for making all text fields blank within the PriceCalculator.java form. This increases efficiency for the user as they wouldn’t need to delete every text field before inserting different values.

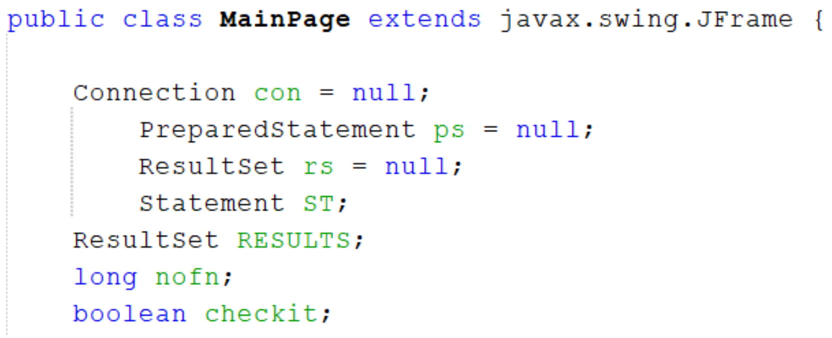
Methods and Variables

In the Room Reservations form multiple methods and variables exist.

The method 'Connect' is used to connect java with the reservations database.

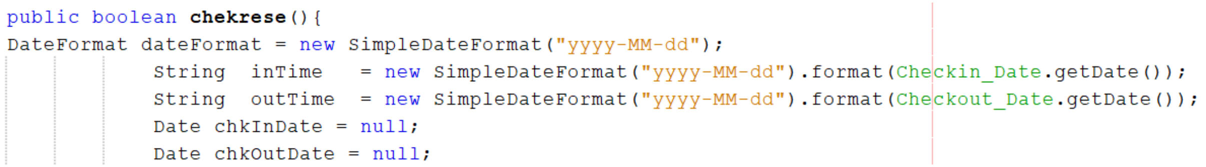


```
public void Connect(){
try{
con = DriverManager.getConnection("jdbc:mysql://localhost:3306/reservations", "root", "");
System.out.println ("Connection ... DB");

}
catch (Exception e){
    System.out.println ("Exception ... "+ e);
}
```



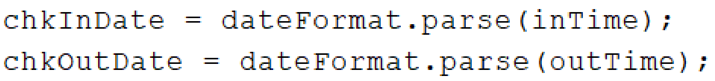
```
public class MainPage extends javax.swing.JFrame {

    Connection con = null;
        PreparedStatement ps = null;
        ResultSet rs = null;
        Statement ST;
    ResultSet RESULTS;
    long nofn;
    boolean checkit;
```

I initialized the following variables in my class so it can be used as I go along. The following variables were declared within the 'chekrese', 'checkdate', 'SaveRes' and 'ModifyRes' methods:



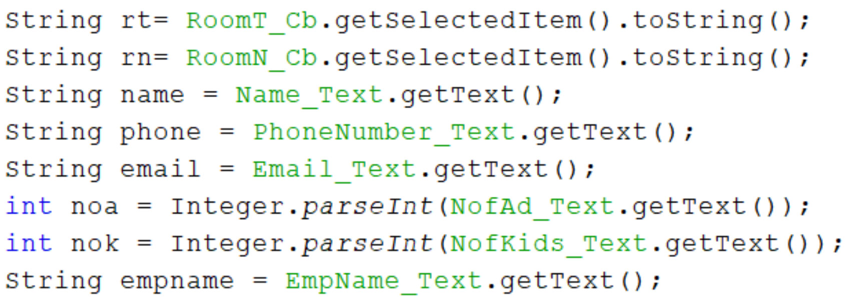
```
public boolean chekrese(){
DateFormat dateFormat = new SimpleDateFormat("yyyy-MM-dd");
            String  inTime    = new SimpleDateFormat("yyyy-MM-dd").format(Checkin_Date.getDate());
            String  outTime   = new SimpleDateFormat("yyyy-MM-dd").format(Checkout_Date.getDate());
            Date chkInDate = null;
            Date chkOutDate = null;
```

In this case, the following code would reference the values set to the variables using the new variable name.



```
chkInDate = dateFormat.parse(inTime);
chkOutDate = dateFormat.parse(outTime);
```

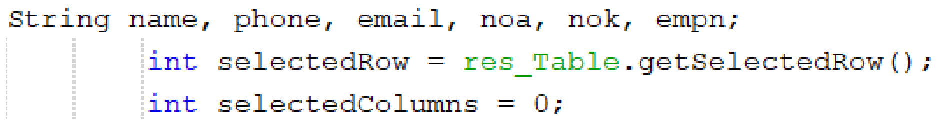
In the method 'SaveRes', the following variables are declared:



```
String rt= RoomT_Cb.getSelectedItem().toString();
String rn= RoomN_Cb.getSelectedItem().toString();
String name = Name_Text.getText();
String phone = PhoneNumber_Text.getText();
String email = Email_Text.getText();
int noa = Integer.parseInt(NofAd_Text.getText());
int nok = Integer.parseInt(NofKids_Text.getText());
String empname = EmpName_Text.getText();
```

This declaration of variables was done so the following components of the form will save in the table. The variables ensure that the intended use of these components are always apparent.

The following variables are also declared for res_TableMouseClicked:

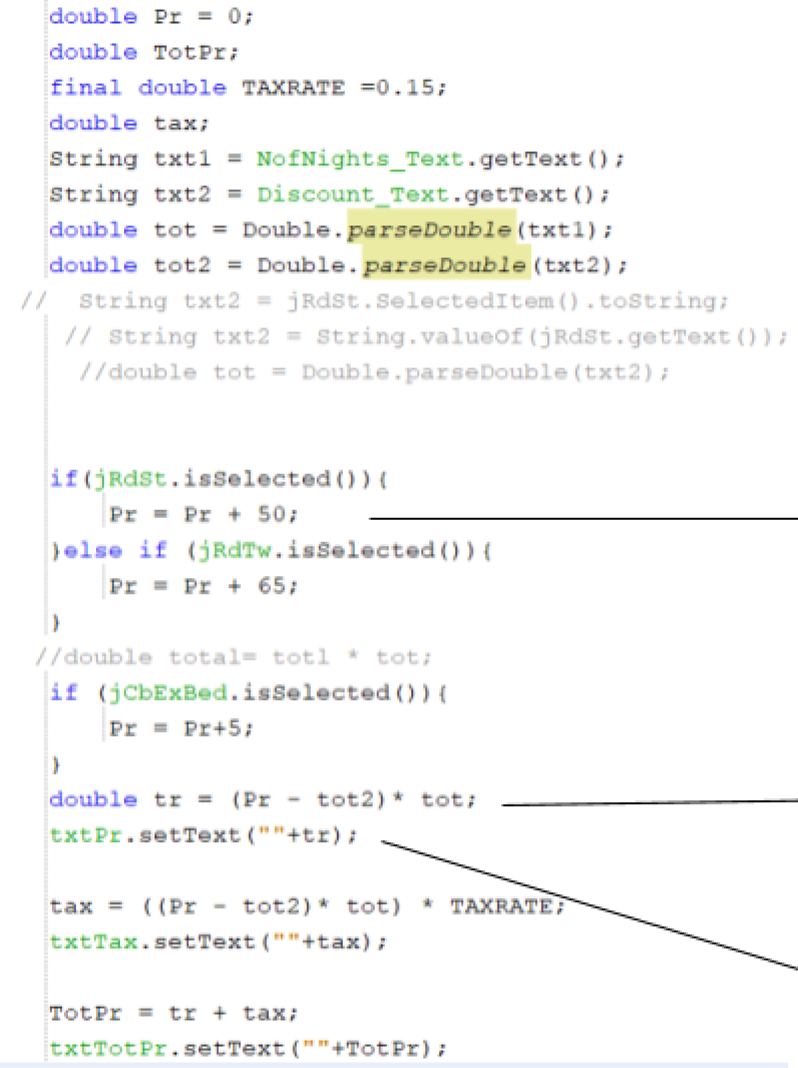


```
String name, phone, email, noa, nok, empn;
        int selectedRow = res_Table.getSelectedRow();
        int selectedColumns = 0;
```

These variables identify which components of the table will be selected and displayed on the text boxes.

Similar variables as the ones mentioned above are declared within the CustMain form which saves customer information

Variables are also declared within the PriceCalculator form within the calculations.



```
    double Pr = 0;
    double TotPr;
    final double TAXRATE =0.15;
    double tax;
    String txt1 = NofNights_Text.getText();
    String txt2 = Discount_Text.getText();
    double tot = Double.parseDouble(txt1);
    double tot2 = Double.parseDouble(txt2);
//   String txt2 = jRdSt.SelectedItem().toString;
     // String txt2 = String.valueOf(jRdSt.getText());
      //double tot = Double.parseDouble(txt2);


    if(jRdSt.isSelected()){
        Pr = Pr + 50;
    }else if (jRdTw.isSelected()){
        Pr = Pr + 65;
    }
  //double total= totl * tot;
    if (jCbExBed.isSelected()){
        Pr = Pr+5;
    }
    double tr = (Pr - tot2)* tot;
    txtPr.setText(""+tr);

    tax = ((Pr - tot2)* tot) * TAXRATE;
    txtTax.setText(""+tax);

    TotPr = tr + tax;
    txtTotPr.setText(""+TotPr);
```

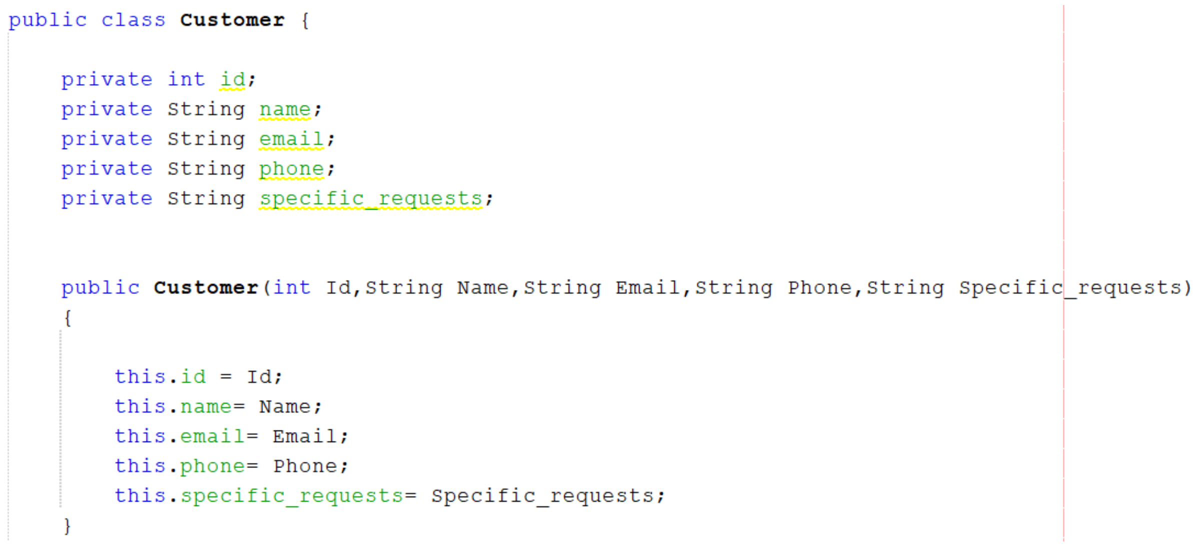
The following shows the initialization and value declaration of variables that will be recurrently used within the calculations button code. The type of data is also declared (double, String,...).

Pr is used within different conditions and is meant to be displayed within the Pr_Text text box

The user's chosen amount of discount is deducted from the price per night as shown by given calculation and declaration of new variable tr

The values within the new variables (tr, tax, TotPr) are displayed within the three text boxes in the form.

The search button on the Customer.java class also declares multiple variables.

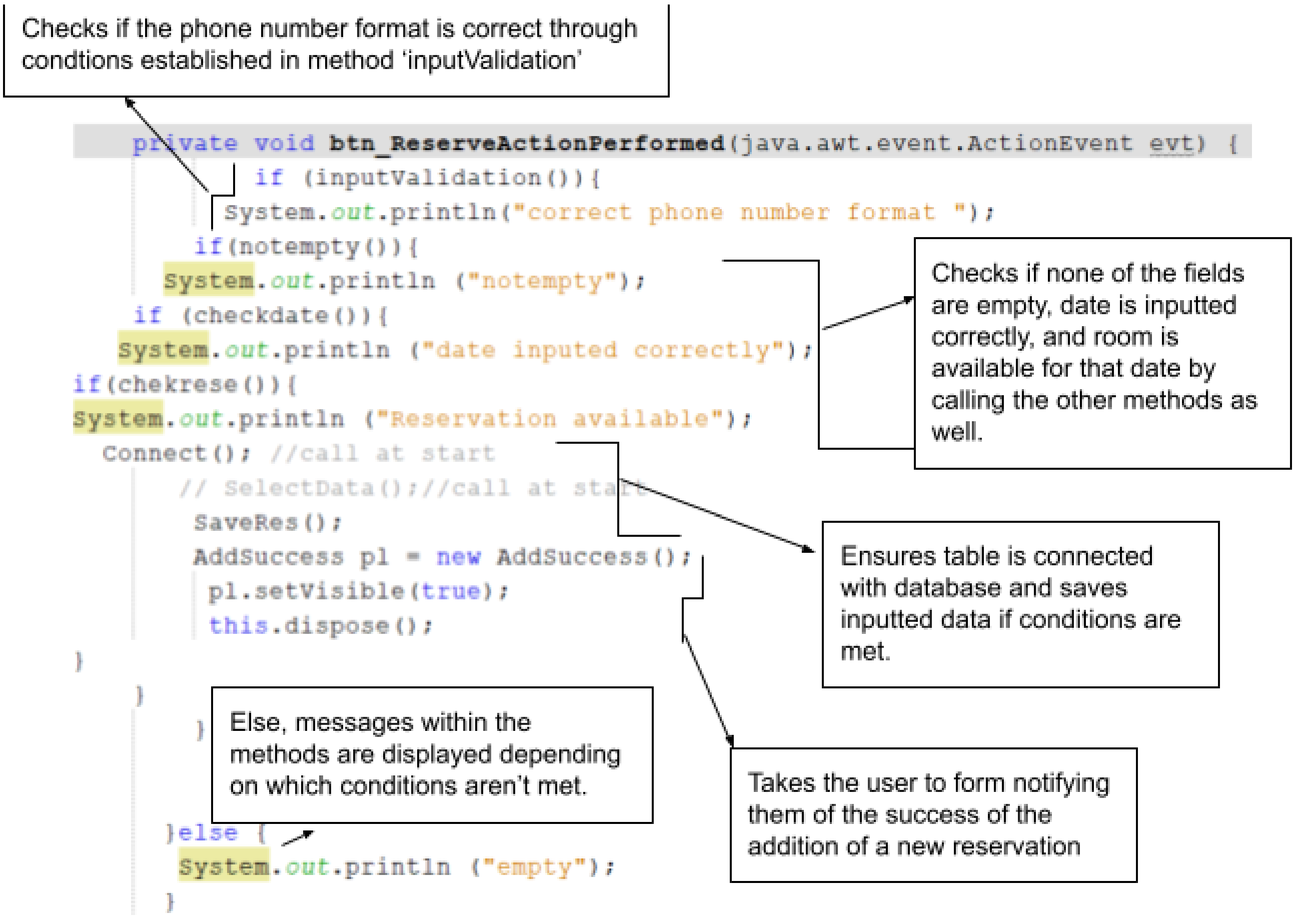


```
public class Customer {

    private int id;
    private String name;
    private String email;
    private String phone;
    private String specific_requests;


    public Customer(int Id,String Name,String Email,String Phone,String Specific_requests)
    {

        this.id = Id;
        this.name= Name;
        this.email= Email;
        this.phone= Phone;
        this.specific_requests= Specific_requests;
    }
```

These variables are used to store the text fields of the tblcust jTable.

New Reservation

The following shows the code behind the ‘Reserve’ button.



Checks if the phone number format is correct through condtions established in method ‘inputValidation’

```
    private void btn_ReserveActionPerformed(java.awt.event.ActionEvent evt) {
              if (inputValidation()){
             System.out.println("correct phone number format ");
          if(notempty()){
        System.out.println ("notempty");
    if (checkdate()){
  System.out.println ("date inputed correctly");
if(chekrese()){
System.out.println ("Reservation available");
   Connect(); //call at start
         // SelectData();//call at start
          SaveRes();
          AddSuccess pl = new AddSuccess();
           pl.setVisible(true);
           this.dispose();
}
    }
          }
      }else {
      System.out.println ("empty");
      }
```

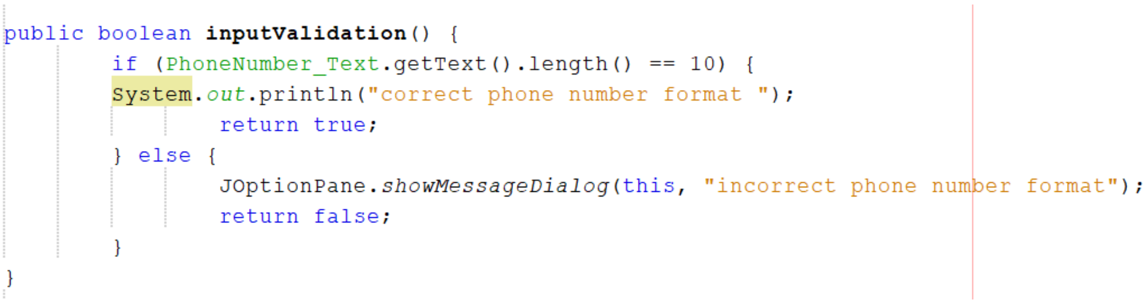
Checks if none of the fields are empty, date is inputted correctly, and room is available for that date by calling the other methods as well.

Ensures table is connected with database and saves inputted data if conditions are met.

Takes the user to form notifying them of the success of the addition of a new reservation

Else, messages within the methods are displayed depending on which conditions aren’t met.

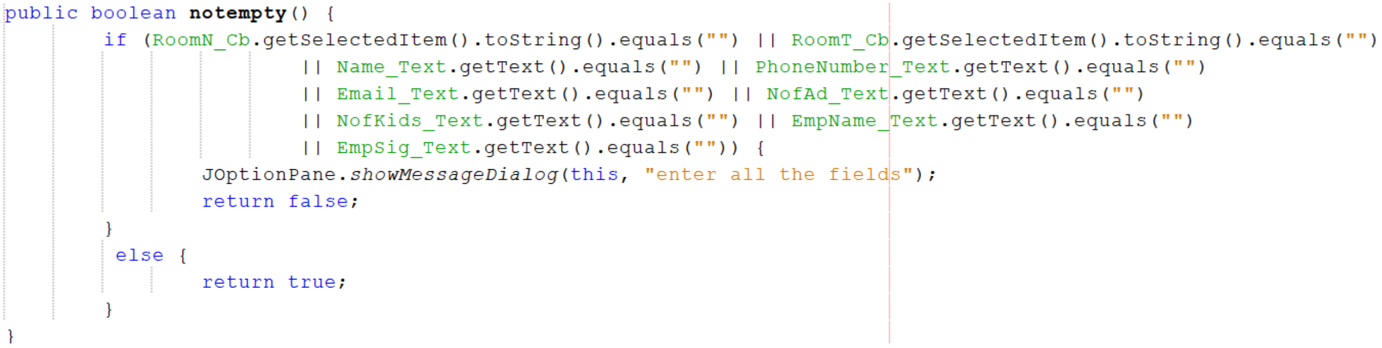
The inputValidation includes the following code. It ensures that the length of phone numbers are ten digits, as all phone numbers worldwide have to be ten digits. There was no condition put on what the phone number starts with, even though all Ethiopian phone numbers have to start with 0. This is because, after talking with the client, they notified me of frequent visits from international guests with different phone numbers starting with different numbers.



```
public boolean inputValidation() {
        if (PhoneNumber_Text.getText().length() == 10) {
        System.out.println("correct phone number format ");
                return true;
        } else {
                JOptionPane.showMessageDialog(this, "incorrect phone number format");
                return false;
        }
}
```

If phone number length is below or exceeding 10, reservation is not added and error message “incorrect phone number format” is displayed to the user.

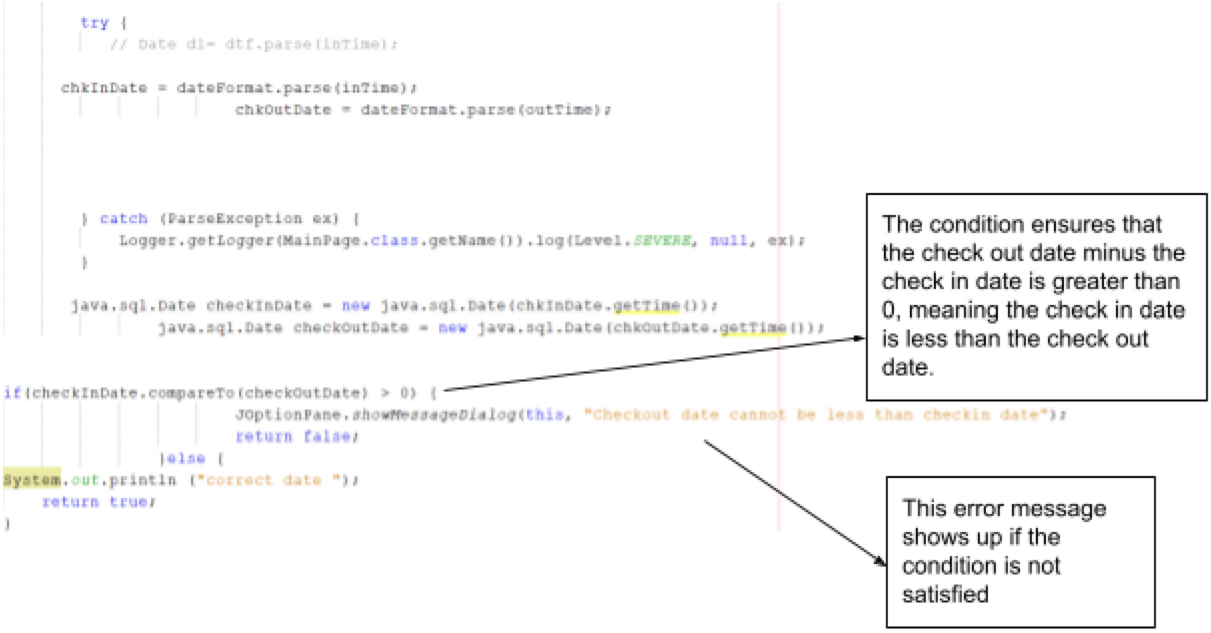
The notempty method ensures that none of the fields are empty, ensuring that all required information about the client is saved and no issues arise with the interaction between the client and the guest.



```
public boolean notempty() {
        if (RoomN_Cb.getSelectedItem().toString().equals("") || RoomT_Cb.getSelectedItem().toString().equals("")
                        || Name_Text.getText().equals("") || PhoneNumber_Text.getText().equals("")
                        || Email_Text.getText().equals("") || NofAd_Text.getText().equals("")
                        || NofKids_Text.getText().equals("") || EmpName_Text.getText().equals("")
                        || EmpSig_Text.getText().equals("")) {
                JOptionPane.showMessageDialog(this, "enter all the fields");
                return false;
        }
         else {
                return true;
        }
}
```

The if condition checks if any of the fields are empty (“”). If so, the error message “enter all the fields” is displayed.

The checkdate method includes a condition that ensures the check in date is before the check out date.



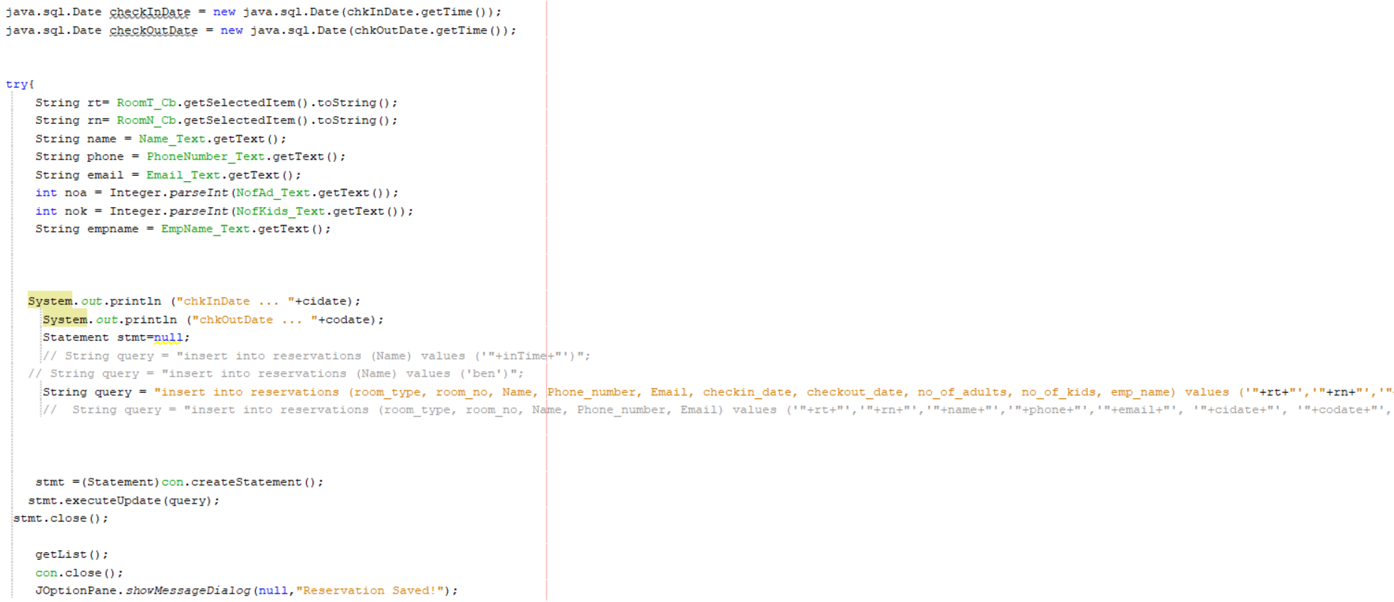
```
            try {
            |   // Date d1= dtf.parse(inTime);

          chkInDate = dateFormat.parse(inTime);
                                chkOutDate = dateFormat.parse(outTime);



            } catch (ParseException ex) {
                Logger.getLogger(MainPage.class.getName()).log(Level.SEVERE, null, ex);
            }

           java.sql.Date checkInDate = new java.sql.Date(chkInDate.getTime());
                     java.sql.Date checkOutDate = new java.sql.Date(chkOutDate.getTime());

if(checkInDate.compareTo(checkOutDate) > 0) {
                                JOptionPane.showMessageDialog(this, "Checkout date cannot be less than checkin date");
                                return false;
                        }else {
System.out.println ("correct date ");
        return true;
}
```

The condition ensures that the check out date minus the check in date is greater than 0, meaning the check in date is less than the check out date.

This error message shows up if the condition is not satisfied

The checkrese method will be explored in the next section.

The SaveRes method is responsible for the saving of the inputted data onto the table in the database.

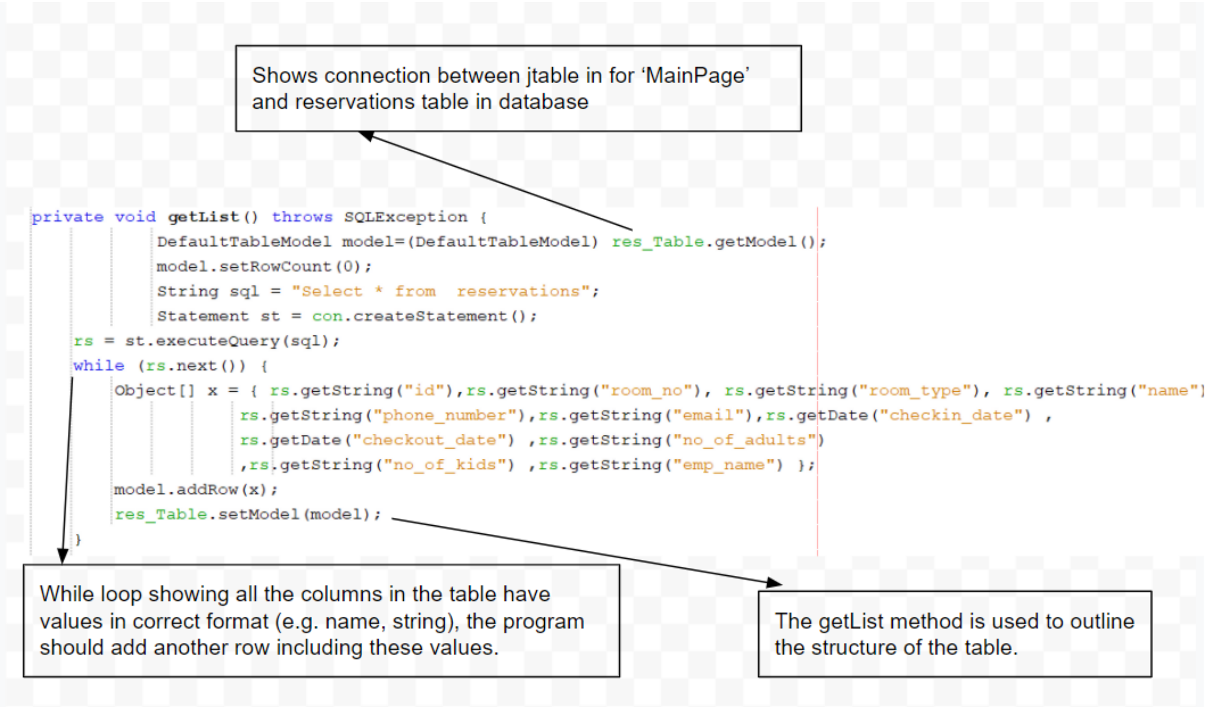


```
java.sql.Date checkInDate = new java.sql.Date(chkInDate.getTime());
java.sql.Date checkOutDate = new java.sql.Date(chkOutDate.getTime());


try{
    String rt= RoomT_Cb.getSelectedItem().toString();
    String rn= RoomN_Cb.getSelectedItem().toString();
    String name = Name_Text.getText();
    String phone = PhoneNumber_Text.getText();
    String email = Email_Text.getText();
    int noa = Integer.parseInt(NofAd_Text.getText());
    int nok = Integer.parseInt(NofKids_Text.getText());
    String empname = EmpName_Text.getText();



   System.out.println ("chkInDate ... "+cidate);
     System.out.println ("chkOutDate ... "+codate);
     Statement stmt=null;
     // String query = "insert into reservations (Name) values ('"+inTime+"')";
   // String query = "insert into reservations (Name) values ('ben')";
     String query = "insert into reservations (room_type, room_no, Name, Phone_number, Email, checkin_date, checkout_date, no_of_adults, no_of_kids, emp_name) values ('"+rt+"','"+rn+"','"
     //  String query = "insert into reservations (room_type, room_no, Name, Phone_number, Email) values ('"+rt+"','"+rn+"','"+name+"','"+phone+"','"+email+"', '"+cidate+"', '"+codate+"',



    stmt =(Statement)con.createStatement();
   stmt.executeUpdate(query);
 stmt.close();

    getList();
    con.close();
    JOptionPane.showMessageDialog(null,"Reservation Saved!");
```

('"+rt+"', '"+rn+"',... goes on until empname as displayed above within the variable declarations)

As shown in the image above, each column in the table is given a variable name and inserted into the table values in the database. Once a reservation has been saved successfully the message "Reservation saved" is displayed. "Statement stmt=con.createStatement(); the statement is a interface and it is used to sending a SQL query to the database and con is a variable of connection interface"[1] The getlist method is used to show the values saved in the database in the jtable 'res_Table' in the form as well.

[1] (www.javatpoint.com, n.d.)



Shows connection between jtable in for ‘MainPage’ and reservations table in database

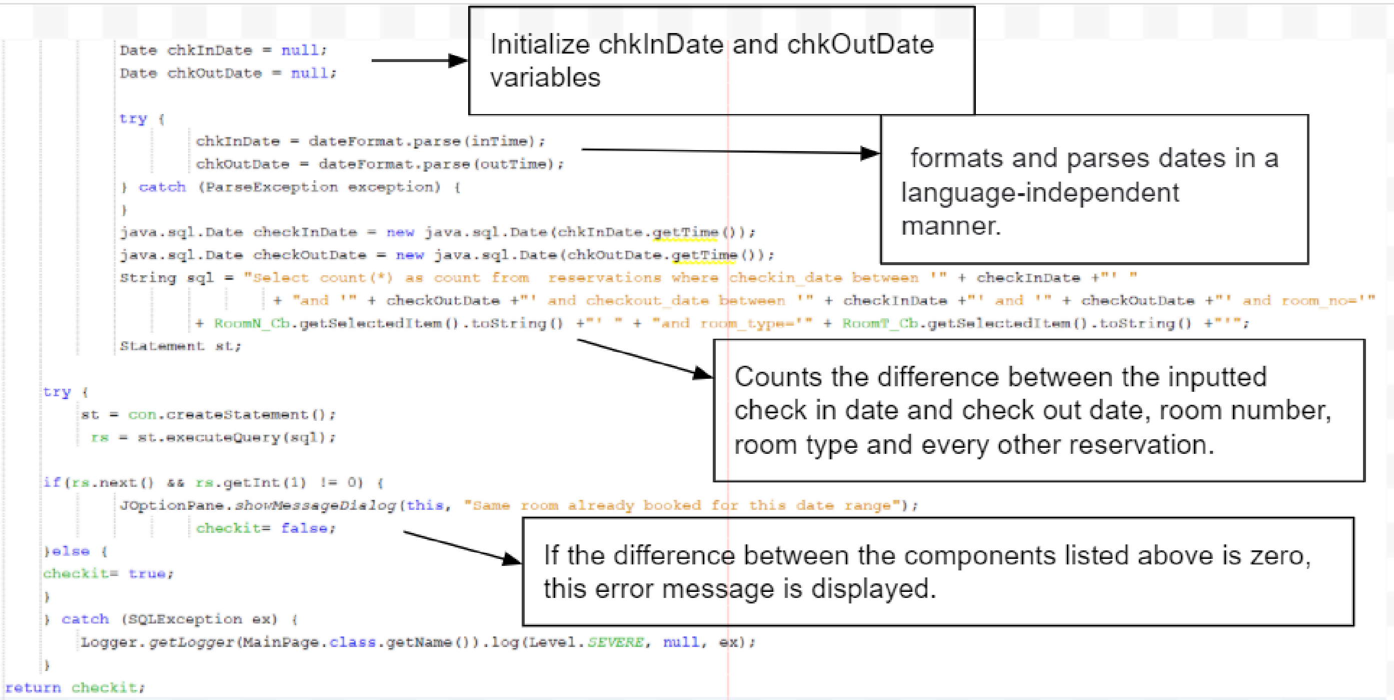
```java
private void getList() throws SQLException {
            DefaultTableModel model=(DefaultTableModel) res_Table.getModel();
            model.setRowCount(0);
            String sql = "Select * from  reservations";
            Statement st = con.createStatement();
    rs = st.executeQuery(sql);
    while (rs.next()) {
        Object[] x = { rs.getString("id"),rs.getString("room_no"), rs.getString("room_type"), rs.getString("name"
                    rs.getString("phone_number"),rs.getString("email"),rs.getDate("checkin_date") ,
                    rs.getDate("checkout_date") ,rs.getString("no_of_adults")
                    ,rs.getString("no_of_kids") ,rs.getString("emp_name") };
        model.addRow(x);
        res_Table.setModel(model);
    }
```

While loop showing all the columns in the table have values in correct format (e.g. name, string), the program should add another row including these values.

The getList method is used to outline the structure of the table.

Double Booking Prevention

The following code prevents the double-booking that could lead to conflict with guests.



```
        Date chkInDate = null;
        Date chkOutDate = null;

        try {
                chkInDate = dateFormat.parse(inTime);
                chkOutDate = dateFormat.parse(outTime);
        } catch (ParseException exception) {
        }
        java.sql.Date checkInDate = new java.sql.Date(chkInDate.getTime());
        java.sql.Date checkOutDate = new java.sql.Date(chkOutDate.getTime());
        String sql = "Select count(*) as count from  reservations where checkin_date between '" + checkInDate +"' "
                        + "and '" + checkOutDate +"' and checkout_date between '" + checkInDate +"' and '" + checkOutDate +"' and room_no='"
                + RoomN_Cb.getSelectedItem().toString() +"' " + "and room_type='" + RoomT_Cb.getSelectedItem().toString() +"'";
        Statement st;

    try {
        st = con.createStatement();
         rs = st.executeQuery(sql);

    if(rs.next() && rs.getInt(1) != 0) {
        JOptionPane.showMessageDialog(this, "Same room already booked for this date range");
                checkit= false;
    }else {
    checkit= true;
    }
    } catch (SQLException ex) {
        Logger.getLogger(MainPage.class.getName()).log(Level.SEVERE, null, ex);
    }
return checkit;
```

Initialize chkInDate and chkOutDate variables

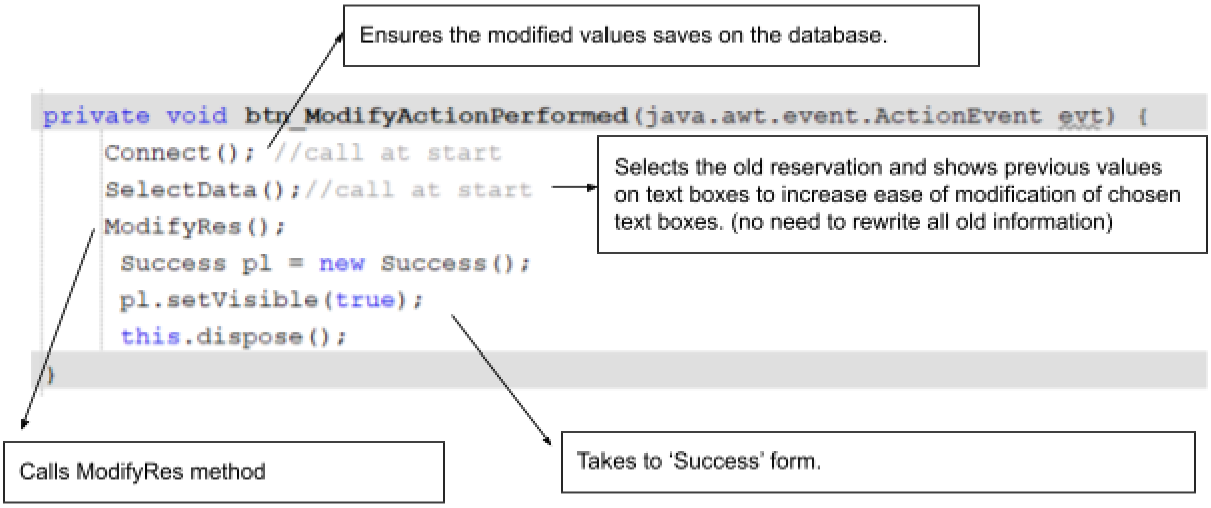
formats and parses dates in a language-independent manner.

Counts the difference between the inputted check in date and check out date, room number, room type and every other reservation.

If the difference between the components listed above is zero, this error message is displayed.

Modify Reservation

The Modify Reservation button is responsible for editing already saved reservations.



Ensures the modified values saves on the database.

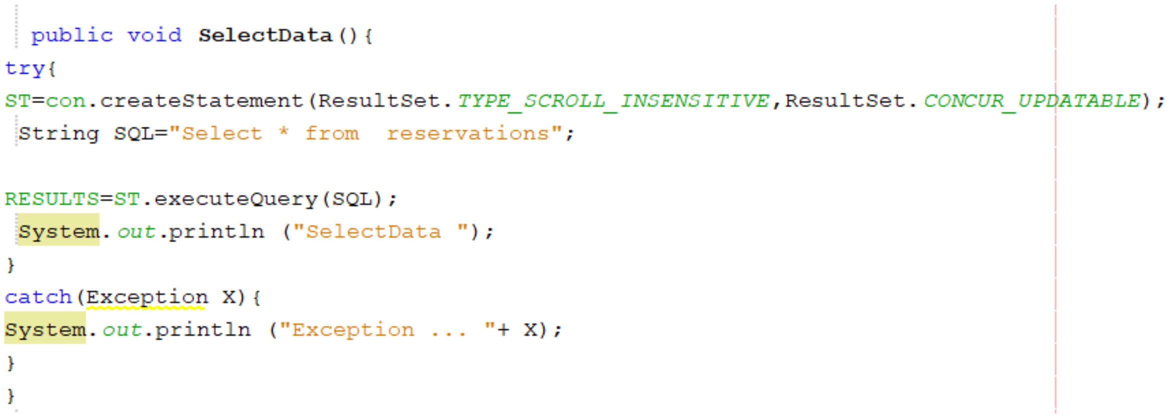
```
private void btn_ModifyActionPerformed(java.awt.event.ActionEvent evt) {
        Connect(); //call at start
        SelectData();//call at start
        ModifyRes();
         Success pl = new Success();
         pl.setVisible(true);
         this.dispose();
}
```

Selects the old reservation and shows previous values on text boxes to increase ease of modification of chosen text boxes. (no need to rewrite all old information)

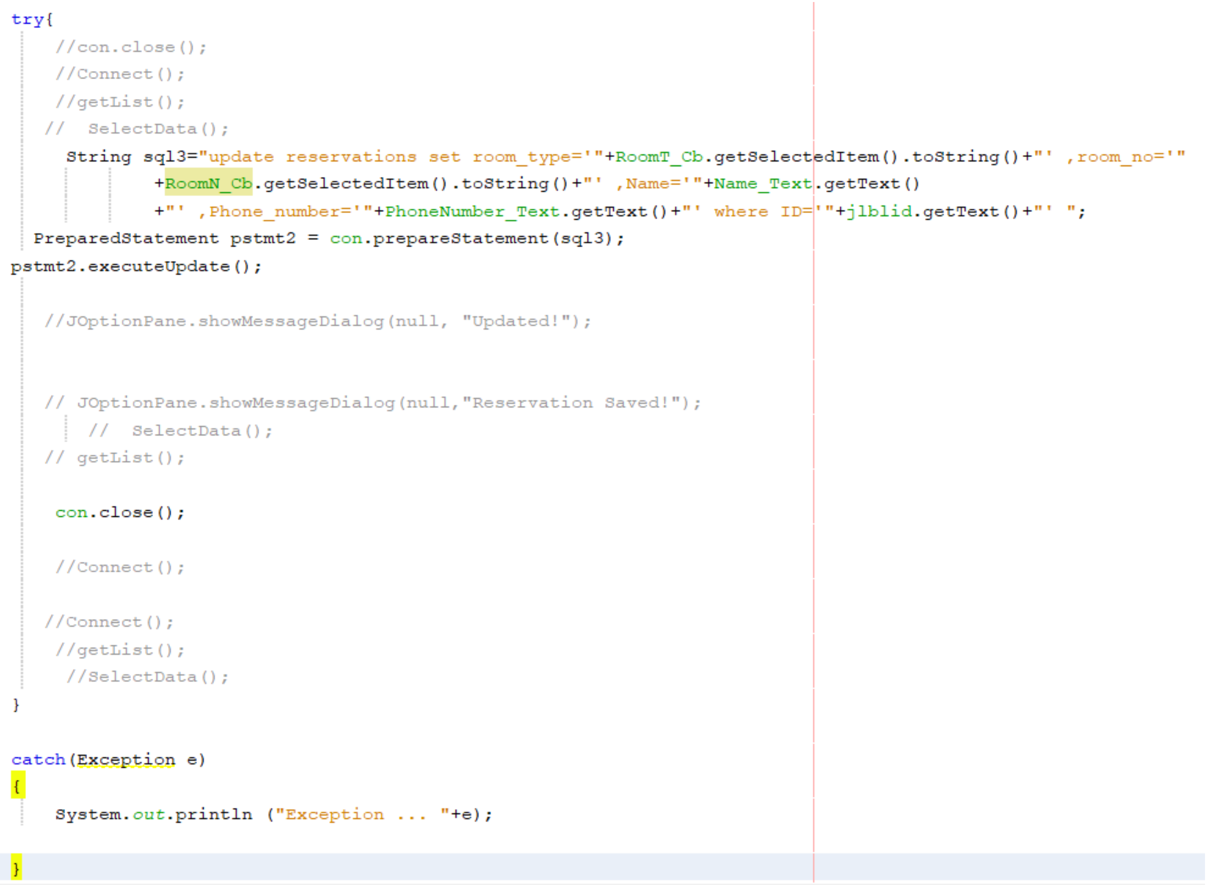
Calls ModifyRes method

Takes to ‘Success’ form.

The following displays the SelectData method which is responsible for selecting a reservation and displaying the reservation information on the textboxes.



```
  public void SelectData(){
try{
ST=con.createStatement(ResultSet.TYPE_SCROLL_INSENSITIVE,ResultSet.CONCUR_UPDATABLE);
 String SQL="Select * from  reservations";

RESULTS=ST.executeQuery(SQL);
 System. out.println ("SelectData ");
}
catch(Exception X){
System. out.println ("Exception ... "+ X);
}
}
```

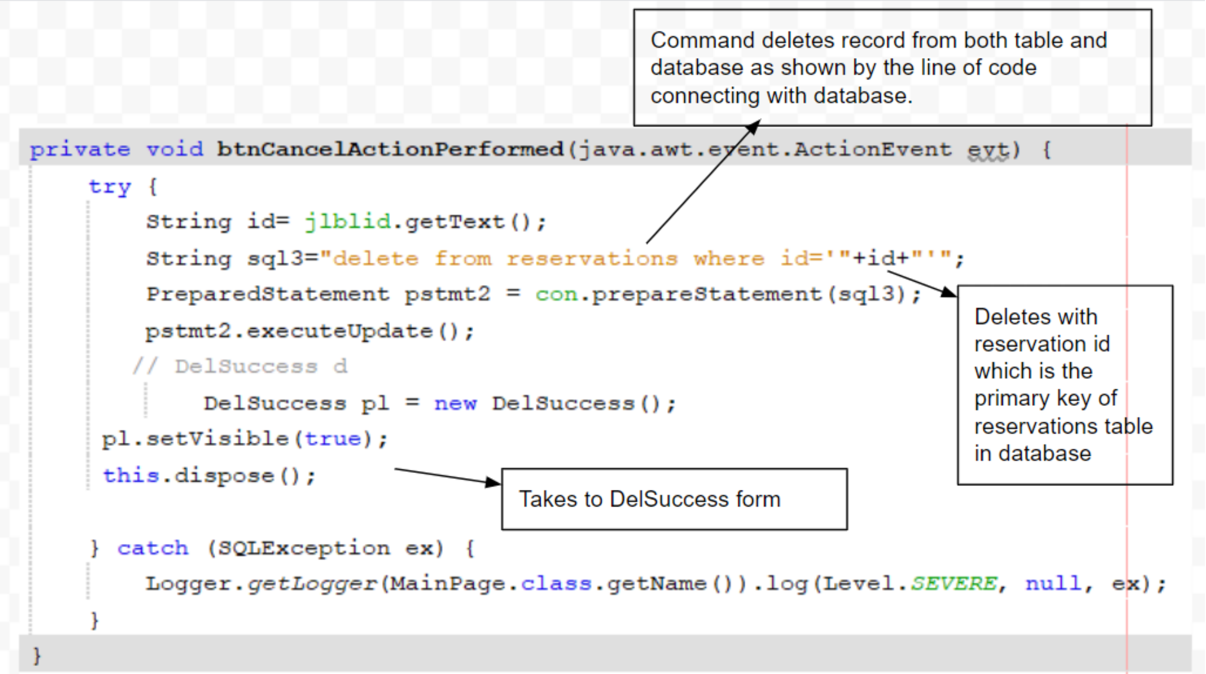


```
try{
    //con.close();
    //Connect();
    //getList();
   //  SelectData();
     String sql3="update reservations set room_type='"+RoomT_Cb.getSelectedItem().toString()+"' ,room_no='"
             +RoomN_Cb.getSelectedItem().toString()+"' ,Name='"+Name_Text.getText()
             +"' ,Phone_number='"+PhoneNumber_Text.getText()+"' where ID='"+jlblid.getText()+"' ";
  PreparedStatement pstmt2 = con.prepareStatement(sql3);
pstmt2.executeUpdate();

   //JOptionPane.showMessageDialog(null, "Updated!");


   // JOptionPane.showMessageDialog(null,"Reservation Saved!");
       //  SelectData();
   // getList();

    con.close();

    //Connect();

   //Connect();
    //getList();
     //SelectData();
}

catch(Exception e)
{
    System.out.println ("Exception ... "+e);

}
```

The update reservations command allows the user to modify each record.

Cancel Reservations

The cancel button uses the delete command to delete the selected record from the table and the database.



```
private void btnCancelActionPerformed(java.awt.event.ActionEvent evt) {
        try {
            String id= jlblid.getText();
            String sql3="delete from reservations where id='"+id+"'";
            PreparedStatement pstmt2 = con.prepareStatement(sql3);
            pstmt2.executeUpdate();
           // DelSuccess d
                DelSuccess pl = new DelSuccess();
      pl.setVisible(true);
      this.dispose();

        } catch (SQLException ex) {
            Logger.getLogger(MainPage.class.getName()).log(Level.SEVERE, null, ex);
        }
}
```

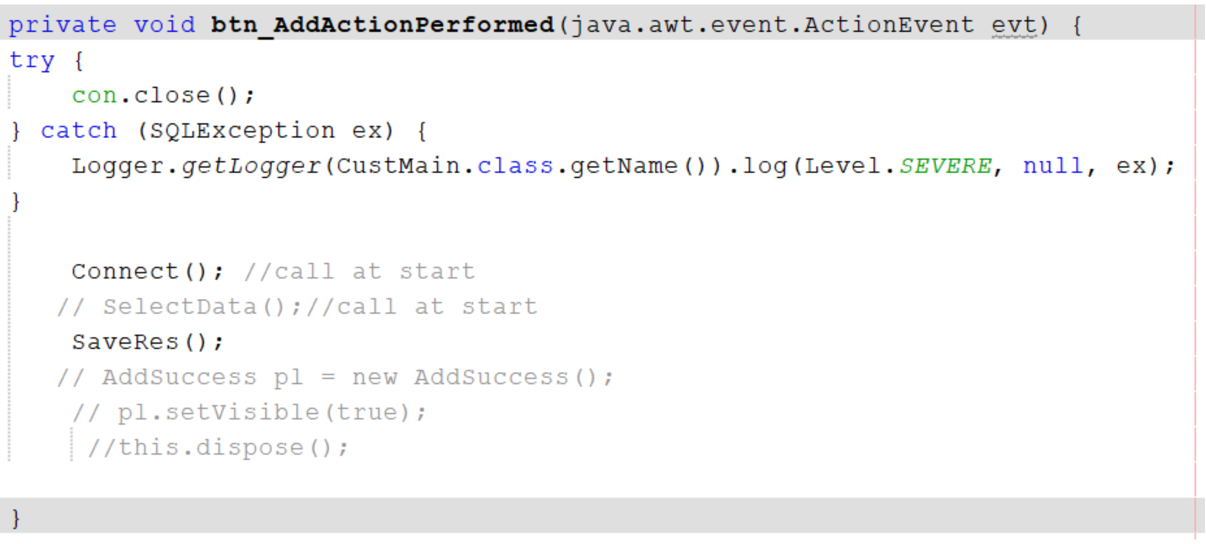
Command deletes record from both table and database as shown by the line of code connecting with database.

Deletes with reservation id which is the primary key of reservations table in database

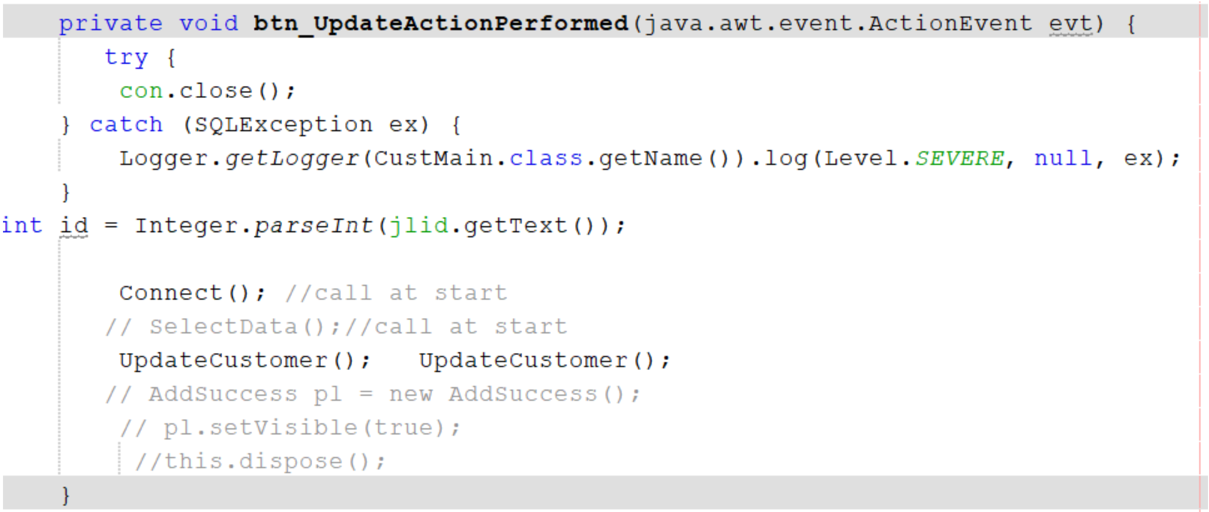
Takes to DelSuccess form

Manage Customer Records

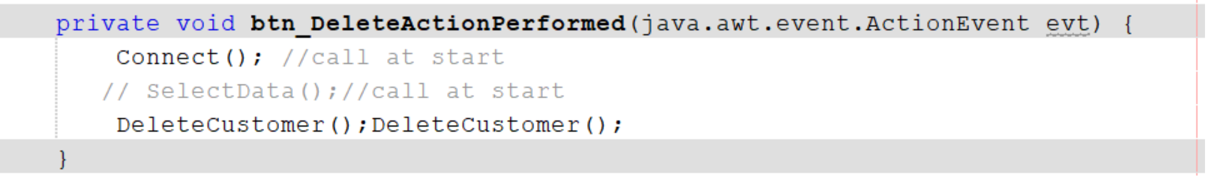
The Manage Customer Records form saves client information to keep track of returning customers and reward discounts accordingly.



```
private void btn_AddActionPerformed(java.awt.event.ActionEvent evt) {
try {
    con.close();
} catch (SQLException ex) {
    Logger.getLogger(CustMain.class.getName()).log(Level.SEVERE, null, ex);
}


    Connect(); //call at start
  // SelectData();//call at start
    SaveRes();
  // AddSuccess pl = new AddSuccess();
   // pl.setVisible(true);
    //this.dispose();

}
```

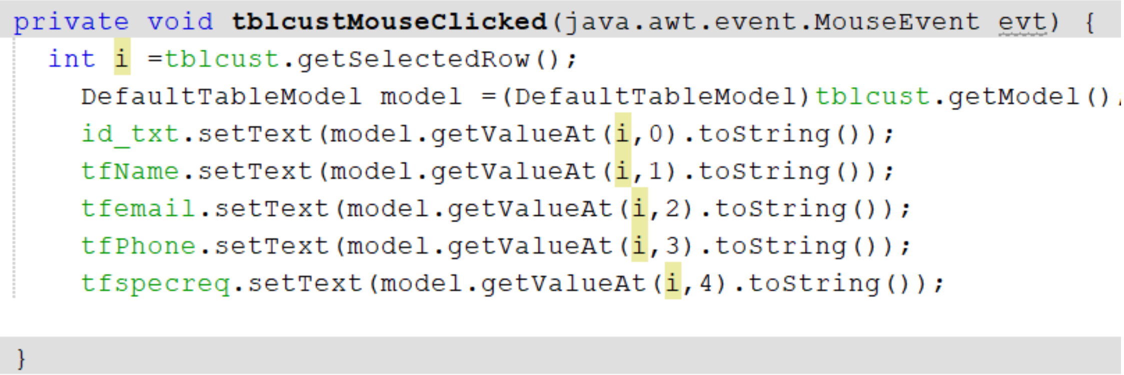


```
    private void btn_UpdateActionPerformed(java.awt.event.ActionEvent evt) {
       try {
        con.close();
    } catch (SQLException ex) {
        Logger.getLogger(CustMain.class.getName()).log(Level.SEVERE, null, ex);
    }
int id = Integer.parseInt(jlid.getText());

        Connect(); //call at start
      // SelectData();//call at start
        UpdateCustomer();    UpdateCustomer();
      // AddSuccess pl = new AddSuccess();
        // pl.setVisible(true);
         //this.dispose();
    }
```



```
    private void btn_DeleteActionPerformed(java.awt.event.ActionEvent evt) {
        Connect(); //call at start
      // SelectData();//call at start
        DeleteCustomer();DeleteCustomer();
    }
```

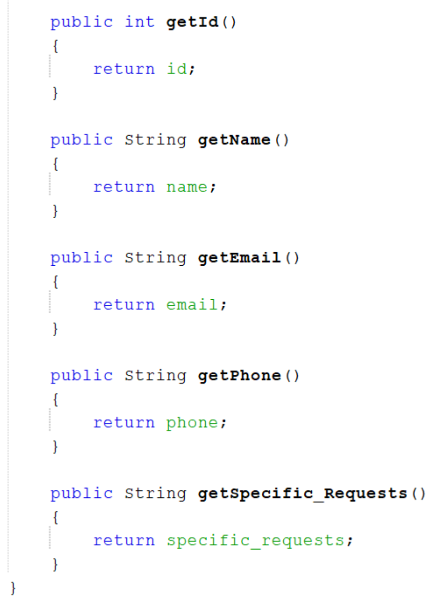
The code is similar to the ones on the main page for the ‘Reserve’, ‘Modify’, and ‘Delete buttons. It uses the connect method to ensure all inputted, modified, or deleted information saves on the database. It also refers to other methods with code including similar commands from mysql to the ones referred to on the ‘Main Page’ form.



```
private void tblcustMouseClicked(java.awt.event.MouseEvent evt) {
  int i =tblcust.getSelectedRow();
    DefaultTableModel model =(DefaultTableModel)tblcust.getModel();
    id_txt.setText(model.getValueAt(i,0).toString());
    tfName.setText(model.getValueAt(i,1).toString());
    tfemail.setText(model.getValueAt(i,2).toString());
    tfPhone.setText(model.getValueAt(i,3).toString());
    tfspecreq.setText(model.getValueAt(i,4).toString());

}
```

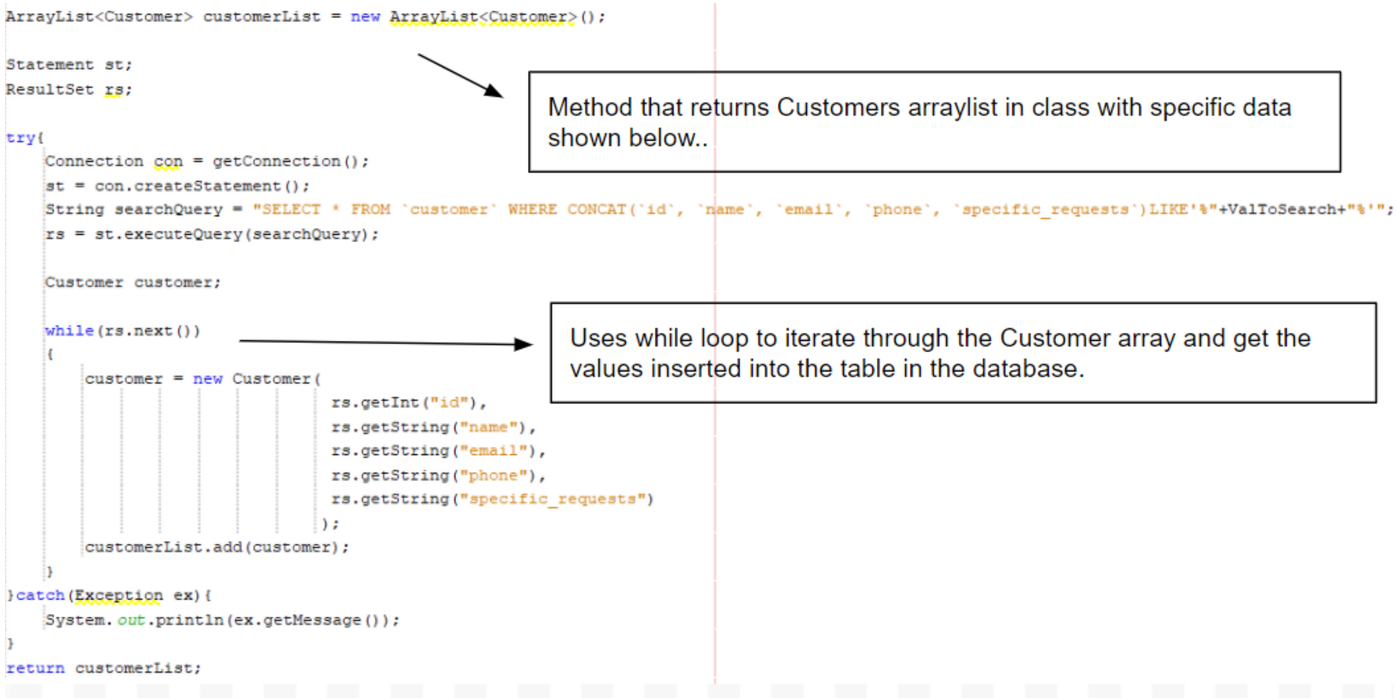
The tblcustMouseClicked allows the user to select a record in the table and it displays the values in the text boxes.

Search Function



```
    public int getId()
    {
        return id;
    }

    public String getName()
    {
        return name;
    }

    public String getEmail()
    {
        return email;
    }

    public String getPhone()
    {
        return phone;
    }

    public String getSpecific_Requests()
    {
        return specific_requests;
    }
}
```

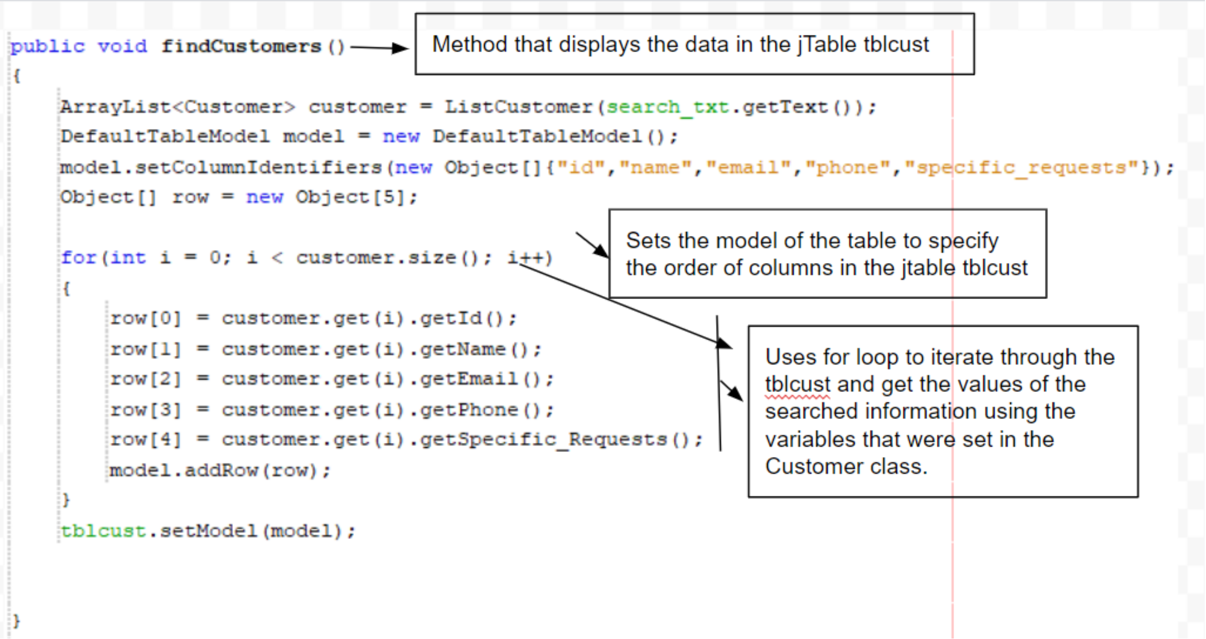
-> returned values of variables in tblcust.



```
ArrayList<Customer> customerList = new ArrayList<Customer>();

Statement st;
ResultSet rs;

try{
    Connection con = getConnection();
    st = con.createStatement();
    String searchQuery = "SELECT * FROM `customer` WHERE CONCAT(`id`, `name`, `email`, `phone`, `specific_requests`)LIKE'%"+ValToSearch+"%'";
    rs = st.executeQuery(searchQuery);

    Customer customer;

    while(rs.next())
    {
        customer = new Customer(
                                 rs.getInt("id"),
                                 rs.getString("name"),
                                 rs.getString("email"),
                                 rs.getString("phone"),
                                 rs.getString("specific_requests")
                                );
        customerList.add(customer);
    }
}catch(Exception ex){
    System.out.println(ex.getMessage());
}
return customerList;
```

Method that returns Customers arraylist in class with specific data shown below..

Uses while loop to iterate through the Customer array and get the values inserted into the table in the database.

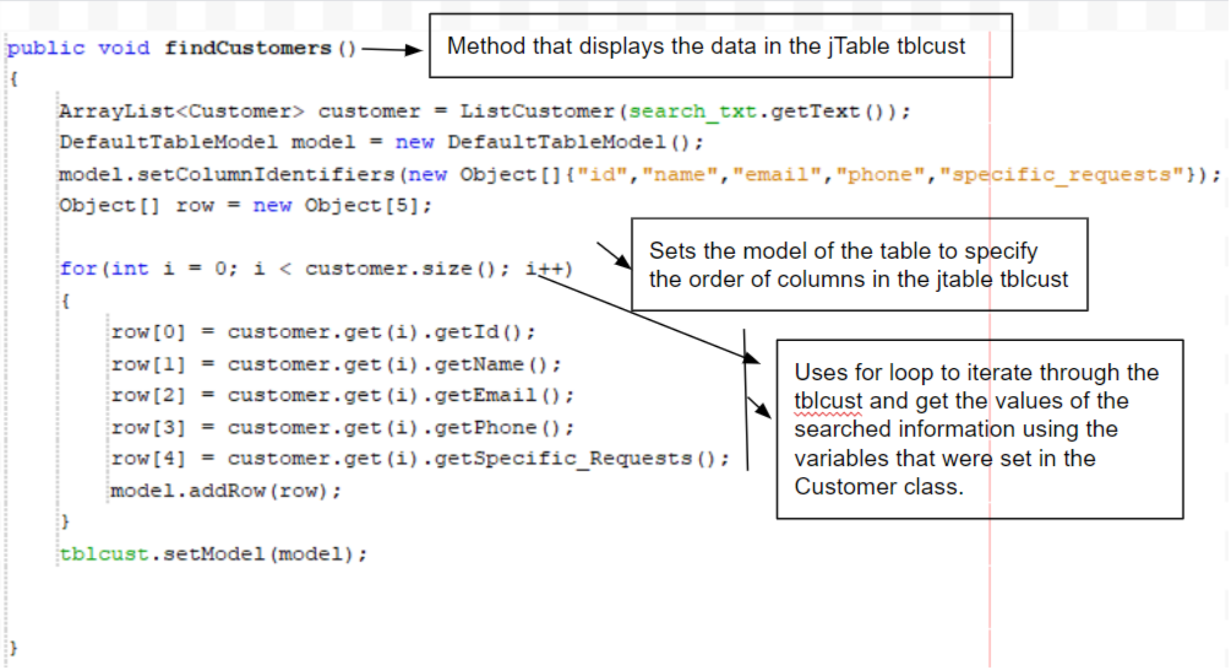


```
public void findCustomers()
{
    ArrayList<Customer> customer = ListCustomer(search_txt.getText());
    DefaultTableModel model = new DefaultTableModel();
    model.setColumnIdentifiers(new Object[]{"id","name","email","phone","specific_requests"});
    Object[] row = new Object[5];

    for(int i = 0; i < customer.size(); i++)
    {
        row[0] = customer.get(i).getId();
        row[1] = customer.get(i).getName();
        row[2] = customer.get(i).getEmail();
        row[3] = customer.get(i).getPhone();
        row[4] = customer.get(i).getSpecific_Requests();
        model.addRow(row);
    }
    tblcust.setModel(model);


}
```

Method that displays the data in the jTable tblcust

Sets the model of the table to specify the order of columns in the jtable tblcust

Uses for loop to iterate through the tblcust and get the values of the searched information using the variables that were set in the Customer class.



```
public void findCustomers()
{
    ArrayList<Customer> customer = ListCustomer(search_txt.getText());
    DefaultTableModel model = new DefaultTableModel();
    model.setColumnIdentifiers(new Object[]{"id","name","email","phone","specific_requests"});
    Object[] row = new Object[5];

    for(int i = 0; i < customer.size(); i++)
    {
        row[0] = customer.get(i).getId();
        row[1] = customer.get(i).getName();
        row[2] = customer.get(i).getEmail();
        row[3] = customer.get(i).getPhone();
        row[4] = customer.get(i).getSpecific_Requests();
        model.addRow(row);
    }
    tblcust.setModel(model);


}
```

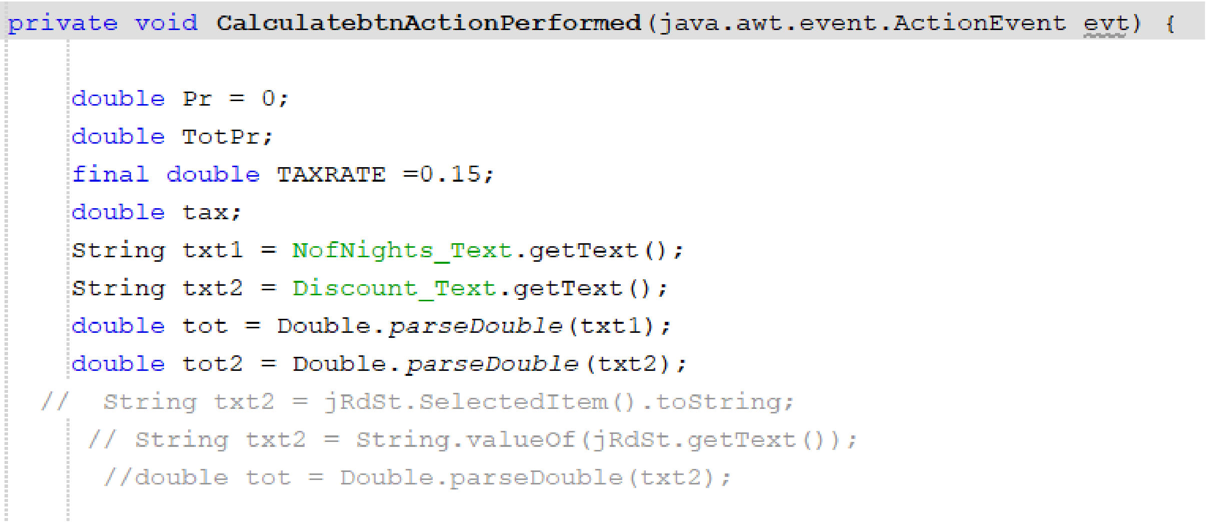
Method that displays the data in the jTable tblcust

Sets the model of the table to specify the order of columns in the jtable tblcust

Uses for loop to iterate through the tblcust and get the values of the searched information using the variables that were set in the Customer class.

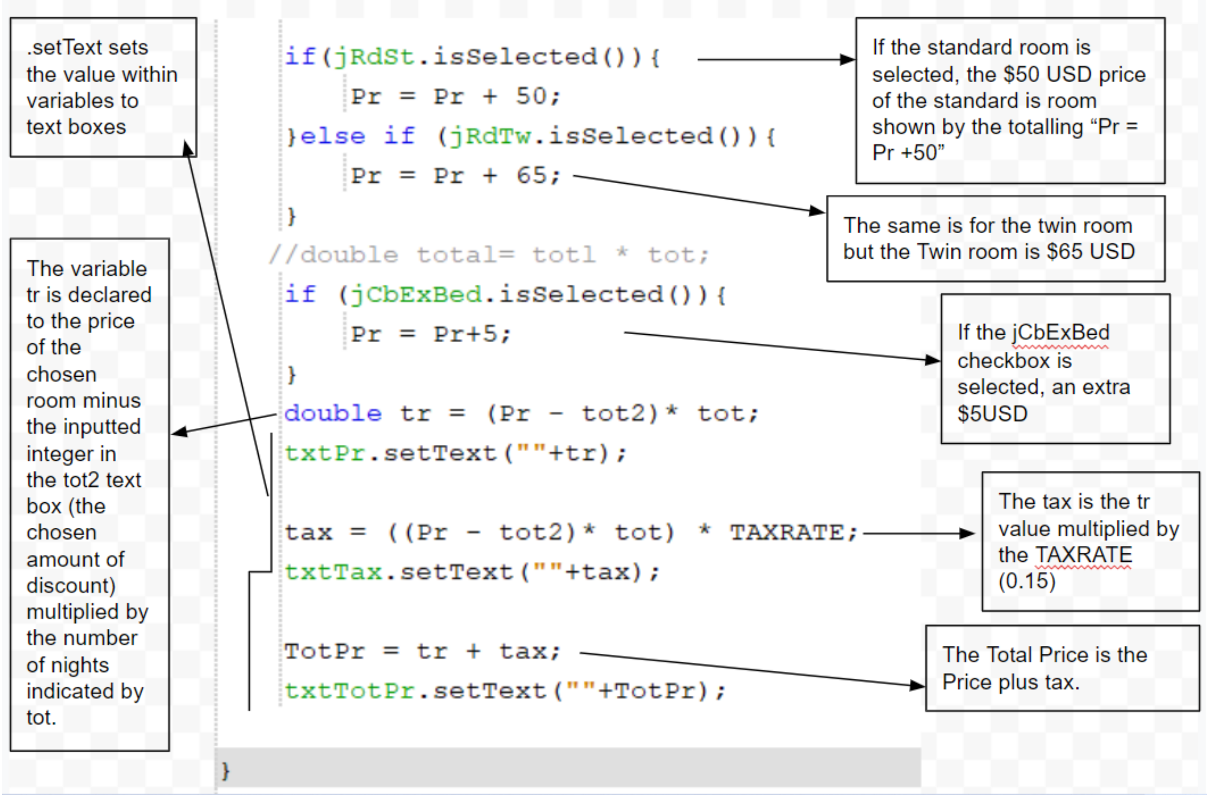
Price Calculations

The Price Calculators allows ease of price calculation and notification to guests.



```
private void CalculatebtnActionPerformed(java.awt.event.ActionEvent evt) {

        double Pr = 0;
        double TotPr;
        final double TAXRATE =0.15;
        double tax;
        String txt1 = NofNights_Text.getText();
        String txt2 = Discount_Text.getText();
        double tot = Double.parseDouble(txt1);
        double tot2 = Double.parseDouble(txt2);
    //  String txt2 = jRdSt.SelectedItem().toString;
         // String txt2 = String.valueOf(jRdSt.getText());
          //double tot = Double.parseDouble(txt2);
```

These are all the variables initialized and declared under the Calculatebtn button.



```
if(jRdSt.isSelected()){
    Pr = Pr + 50;
}else if (jRdTw.isSelected()){
    Pr = Pr + 65;
}
//double total= totl * tot;
if (jCbExBed.isSelected()){
    Pr = Pr+5;
}
double tr = (Pr - tot2)* tot;
txtPr.setText(""+tr);

tax = ((Pr - tot2)* tot) * TAXRATE;
txtTax.setText(""+tax);

TotPr = tr + tax;
txtTotPr.setText(""+TotPr);

}
```

.setText sets the value within variables to text boxes

The variable tr is declared to the price of the chosen room minus the inputted integer in the tot2 text box (the chosen amount of discount) multiplied by the number of nights indicated by tot.

If the standard room is selected, the $50 USD price of the standard is room shown by the totalling “Pr = Pr +50”

The same is for the twin room but the Twin room is $65 USD

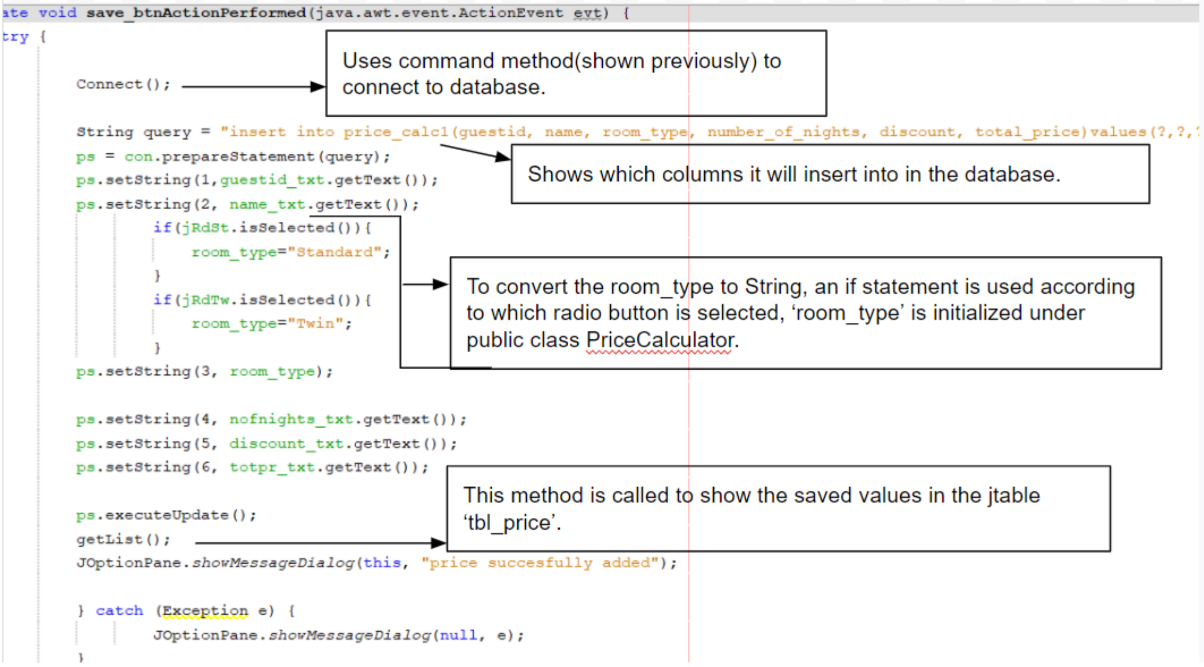
If the jCbExBed checkbox is selected, an extra $5USD

The tax is the tr value multiplied by the TAXRATE (0.15)

The Total Price is the Price plus tax.

Saving of Price Calculations

The following is the code under the save_btn. This button is responsible for saving the information onto the database as well as displaying it on the table on the form.



```
ate void save_btnActionPerformed(java.awt.event.ActionEvent evt) {
try {

        Connect();

        String query = "insert into price_calc1(guestid, name, room_type, number_of_nights, discount, total_price)values(?,?,
        ps = con.prepareStatement(query);
        ps.setString(1,guestid_txt.getText());
        ps.setString(2, name_txt.getText());
                if(jRdSt.isSelected()){
                    room_type="Standard";
                }
                if(jRdTw.isSelected()){
                    room_type="Twin";
                }
        ps.setString(3, room_type);

        ps.setString(4, nofnights_txt.getText());
        ps.setString(5, discount_txt.getText());
        ps.setString(6, totpr_txt.getText());

        ps.executeUpdate();
        getList();
        JOptionPane.showMessageDialog(this, "price succesfully added");

        } catch (Exception e) {
                JOptionPane.showMessageDialog(null, e);
        }
```

Uses command method(shown previously) to connect to database.

Shows which columns it will insert into in the database.

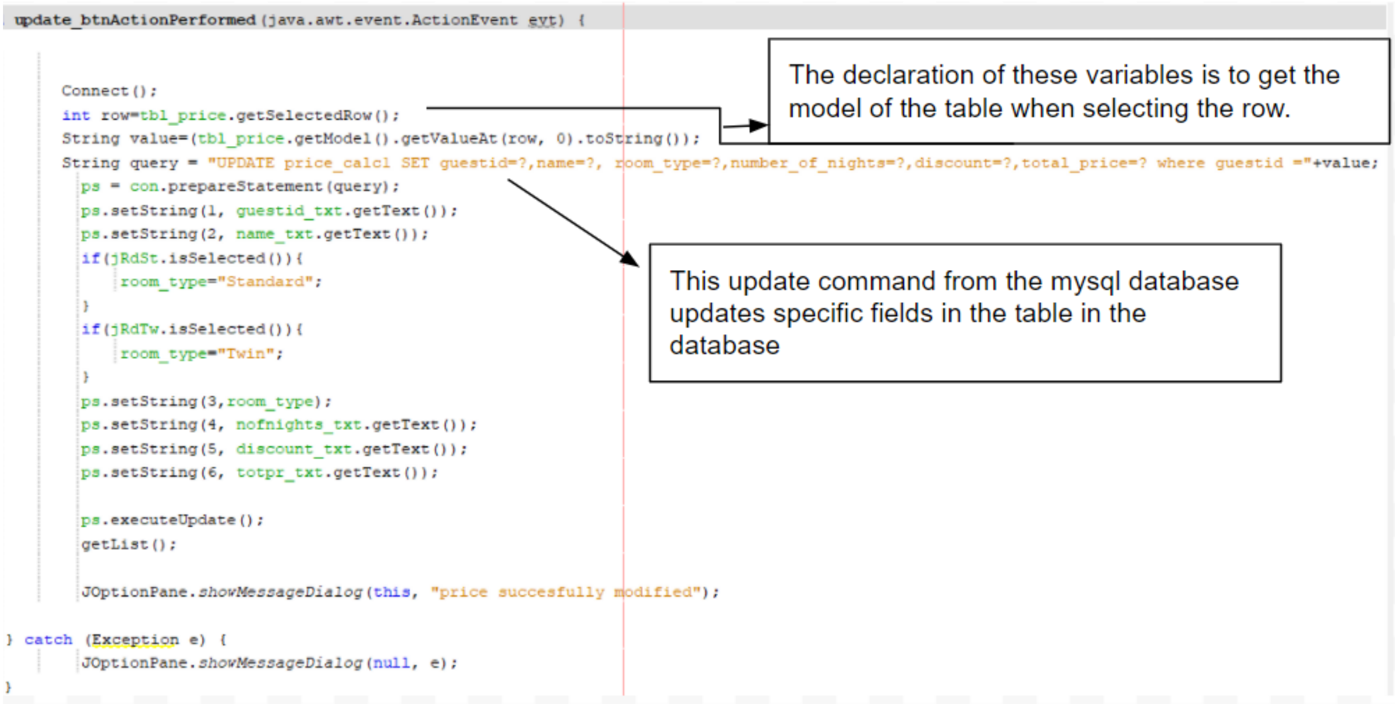
To convert the room_type to String, an if statement is used according to which radio button is selected, 'room_type' is initialized under public class PriceCalculator.

This method is called to show the saved values in the jtable 'tbl_price'.

Modify Price Calculations

The 'update_btn' is responsible for the editing of the values stored into the record, both in the table and the database. This code is responsible for that:

```
private void tbl_priceMouseClicked(java.awt.event.MouseEvent evt) {
    int i =tbl_price.getSelectedRow();
    TableModel model =tbl_price.getModel();
    guestid_txt.setText(model.getValueAt(i,1).toString());
    name_txt.setText(model.getValueAt(i,2).toString());
    String Room_type = model.getValueAt(i,3).toString();
     if(Room_type.equals("Standard")){
    jRdSt.setSelected(true);
}
else{
    jRdTw.setSelected(true);
}
    nofnights_txt.setText(model.getValueAt(i,4).toString());
    discount_txt.setText(model.getValueAt(i,5).toString());
    totpr_txt.setText(model.getValueAt(i,6).toString());


}
```

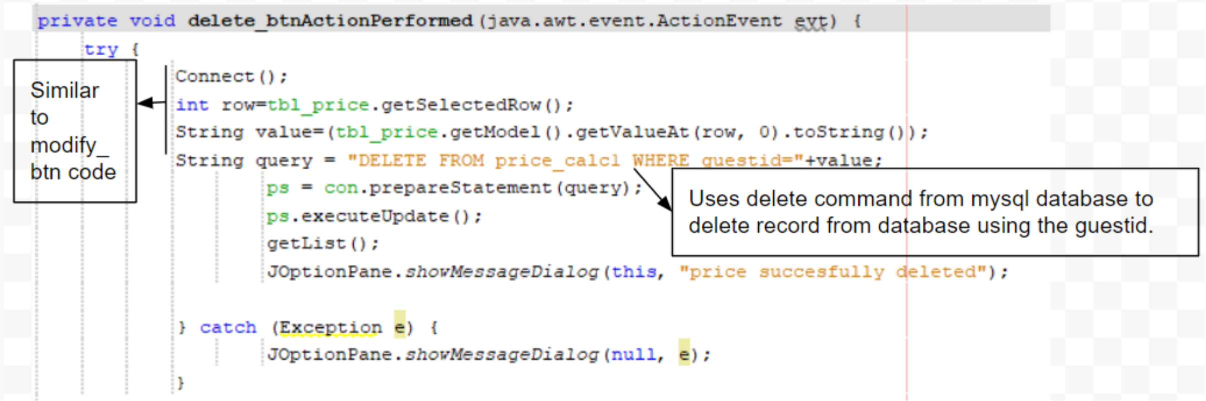


```
update_btnActionPerformed(java.awt.event.ActionEvent evt) {

    Connect();
    int row=tbl_price.getSelectedRow();
    String value=(tbl_price.getModel().getValueAt(row, 0).toString());
    String query = "UPDATE price_calcl SET guestid=?,name=?, room_type=?,number_of_nights=?,discount=?,total_price=? where guestid ="+value;
      ps = con.prepareStatement(query);
      ps.setString(1, guestid_txt.getText());
      ps.setString(2, name_txt.getText());
      if(jRdSt.isSelected()){
          room_type="Standard";
      }
      if(jRdTw.isSelected()){
          room_type="Twin";
      }
      ps.setString(3,room_type);
      ps.setString(4, nofnights_txt.getText());
      ps.setString(5, discount_txt.getText());
      ps.setString(6, totpr_txt.getText());

      ps.executeUpdate();
      getList();

      JOptionPane.showMessageDialog(this, "price succesfully modified");

} catch (Exception e) {
      JOptionPane.showMessageDialog(null, e);
}
```

The declaration of these variables is to get the model of the table when selecting the row.

This update command from the mysql database updates specific fields in the table in the database

Delete Price Calculations



```
private void delete_btnActionPerformed(java.awt.event.ActionEvent evt) {
    try {
        Connect();
        int row=tbl_price.getSelectedRow();
        String value=(tbl_price.getModel().getValueAt(row, 0).toString());
        String query = "DELETE FROM price_calcl WHERE guestid="+value;
                ps = con.prepareStatement(query);
                ps.executeUpdate();
                getList();
                JOptionPane.showMessageDialog(this, "price succesfully deleted");

        } catch (Exception e) {
                JOptionPane.showMessageDialog(null, e);
        }
```

Similar to modify_ btn code

Uses delete command from mysql database to delete record from database using the guestid.

Bibliography

www.javatpoint.com. (n.d.). *explain Statement stmt=con.createStatement(); | 7789 - javatpoint.com*. [online] Available at: https://www.javatpoint.com/q/7789/explain-statement-stmt=con-createstatement() [Accessed 23 Mar. 2022].