



ETES-204

Practical File

Submitted By:

Pandey Mohiya-Asu2013010100136

Chetan Gupta-Asu2013010100018

Prateek Verma-Asu2013010100052

Submitted to:

Mrs.Harsimran Kaur

CONTENT:

- **Introduction:**
 - JDBC
 - SWING
 - AWT
 - EVENTS
- **Socket Program.**
- **RMI Program.**
- **Project Program:[Designing Cafeteria System]**

INTRODUCTION.

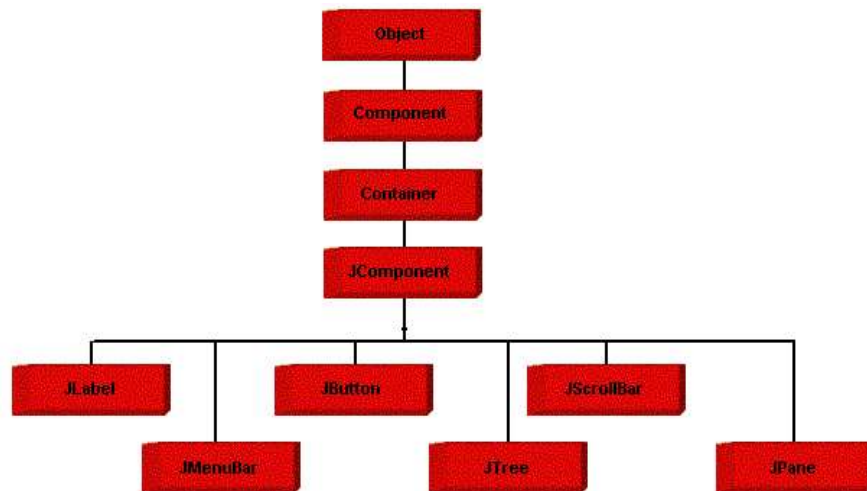
- *Jdbc :*



- The JDBC API is a Java API that can access any kind of tabular data, especially data stored in a Relational Database.
- JDBC helps you to write Java applications that manage these three programming activities:
 1. Connect to a data source, like a database
 2. Send queries and update statements to the database
 3. Retrieve and process the results received from the database in answer to your query
- The following simple code fragment gives a simple example of these three steps:
 1. `public void connectToAndQueryDatabase(String username, String password)`
 2. `{`
 3. `Connection con = DriverManager.getConnection(`
 4. `"jdbc:myDriver:myDatabase",`
 5. `username,`
 6. `password);`
 7. `Statement stmt = con.createStatement();`
 8. `ResultSet rs = stmt.executeQuery("SELECT a, b, c FROM Table1");`
 9. `while (rs.next()) {`
 10. `int x = rs.getInt("a");`
 11. `String s = rs.getString("b");`
 12. `float f = rs.getFloat("c");`
 13. `}`
 14. `}`
- This short code fragment instantiates a `DriverManager` object to connect to a database driver and log into the database, instantiates a `Statement` object that carries your SQL language query to the database; instantiates a `ResultSet` object that retrieves the results of your query, and executes a simple while loop, which retrieves and displays those results. It's that simple.

- **Swings :**

- Swing consists of elements or constituents that a developer will be making use of to come up with a Graphical User Interface. Let us take a look at some of the most commonly used elements or components of the Swing. The programming know-how of AWT is not at all required to equip with these swing programs.



- **Awt:**



- The AWT provides four container classes. They are class Window and its two subtypes --class Frame and class Dialog -- as well as the Panel class. In addition to the containers provided by the AWT, the Applet class is a container -- it is a subtype of the Panel class and can therefore hold components. Brief descriptions of each container class provided by the AWT are provided below.

A top-level display surface (a window). An instance of the Window class is not attached to nor embedded within another container. An instance of the Window class has no border and no title.

A top-level display surface (a window) with a border and title. An instance of the Frame class may have a menu bar. It is otherwise very much like an instance of the Window class.

A top-level display surface (a window) with a border and title. An instance of the Dialog class cannot exist without an associated instance of the Frame class.

A generic container for holding components. An instance of the Panel class provides a container to which to add components.

- **Events:**

Web pages are all about interaction. Users perform a countless number of actions such as moving their mice over the page, clicking on elements, and typing in textboxes — all of these are examples of events. In addition to these user events, there are a slew of others that occur, like when the page is loaded, when video begins playing or is paused, etc. Whenever something interesting occurs on the page, an event is fired, meaning that the browser basically announces that something has happened. It's this announcement that allows developers to "listen" for events and react to them appropriately.

Socket program.

Socket

A socket is one of the most fundamental technologies of computer networking. Sockets allow applications to communicate using standard mechanisms built into network hardware and operating systems. Although network software may seem to be a relatively new "Web" phenomenon, socket technology actually has been employed for roughly two decades.

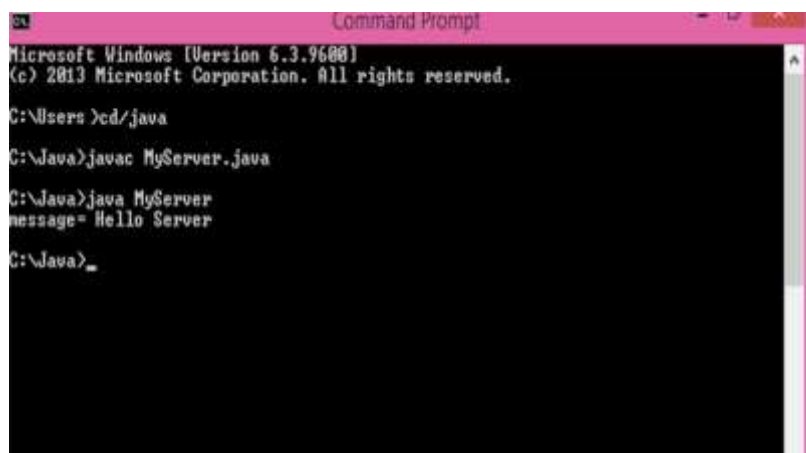
- **My server**

```
1. import java.io.*;
2. import java.net.*;
3. public class MyServer {
4.     public static void main(String[] args){
5.     try{
6.         ServerSocket ss=new ServerSocket(1008);
7.         Socket s=ss.accept();//establishes connection
8.         DataInputStreamdis=new
           DataInputStream(s.getInputStream())
```

```
9. m());
10. String str=(String)dis.readUTF();
11. System.out.println("message= "+str);
12. ss.close();
13. }catch(Exception e){System.out.println(e);}
14. }
15. }
```

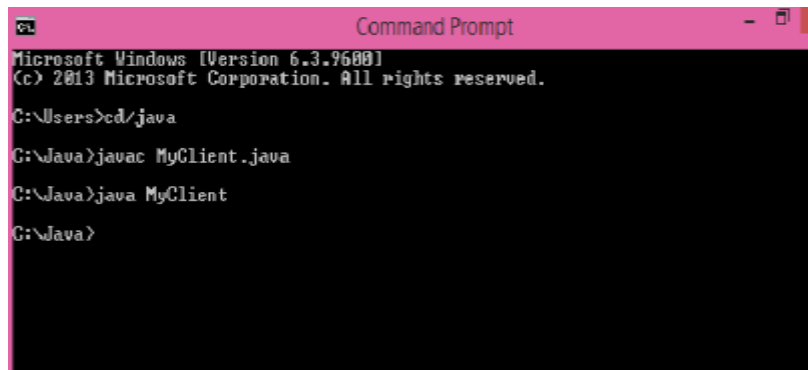
- My client

```
1. import java.io.*;
2. import java.net.*;
3. public class MyClient {
4. public static void main(String[] args) {
5. try{
6. Socket s=new Socket("USER-PC",1008);
7. DataOutputStream dout=new DataOutputStream(s.getOutputStream());
8. dout.writeUTF("Hello Server");
9. dout.flush();
10. dout.close();
11. s.close();
12. }catch(Exception e){System.out.println(e);}
13. }
14. }
```



```
Command Prompt
Microsoft Windows [Version 6.3.9600]
(c) 2013 Microsoft Corporation. All rights reserved.

C:\Users>cd /java
C:\Java>javac MyServer.java
C:\Java>java MyServer
message= Hello Server
C:\Java>_
```



```
Command Prompt
Microsoft Windows [Version 6.3.9600]
(c) 2013 Microsoft Corporation. All rights reserved.

C:\Users>cd java
C:\Java>javac MyClient.java
C:\Java>java MyClient
C:\Java>
```

- **RMI program:**

RMI (Remote Method Invocation) is a Java API for manipulating remote objects (e.g an object instantiated on another virtual machine, possibly on another machine on the network) in a transparent way, that is to say in the same way as if the object was located in the virtual machine (JVM) of the local machine.

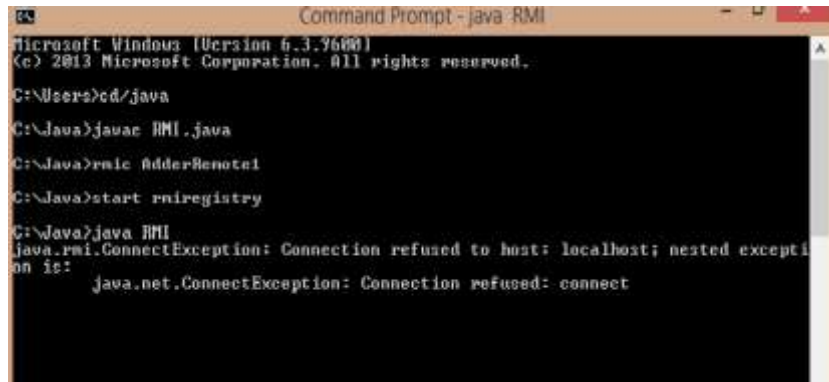
- **For Server:**

1. *import java.rmi.*;*
2. *import java.rmi.registry.*;*
3. *public class RMI*
4. *{*
5. *public static void main (String args[])*
6. *{*
7. *try{*
8. *AdderRemote1 stub=new AdderRemote1();*
9. *Naming.rebind("rmi://localhost:1008/USER-PC",stub);*
10. *}catch(Exception e){System.out.println(e);}*
11. *}}*

12. *For client:*

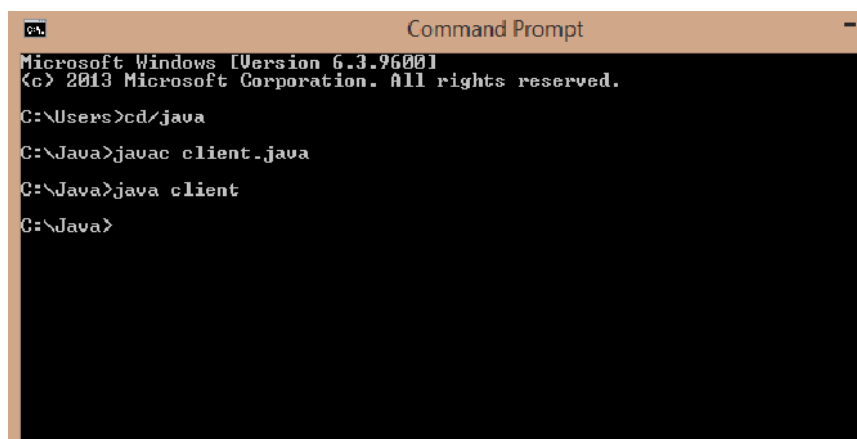
13. *import java .rmi.*;*
14. *import java.rmi.registry.*;*
15. *public class client*
16. *{*
17. *public static void main(String args[])*
18. *{*
19. *try*
20. *{*

```
21. AdderRemote stub=(AdderRemote)Naming.lookup("rmi://localhost:1008/USER-PC");
22. System.out.println(stub.add(34,4));
23. }catch(Exception e){}
24.}}
```



```
Microsoft Windows [Version 6.3.9600]
(c) 2013 Microsoft Corporation. All rights reserved.

C:\Users>cd /java
C:\Java>javac RMI.java
C:\Java>rmic AdderRemote1
C:\Java>start rmiregistry
C:\Java>java RMI
java.rmi.ConnectException: Connection refused to host: localhost; nested exception is:
    java.net.ConnectException: Connection refused: connect
```



```
Microsoft Windows [Version 6.3.9600]
(c) 2013 Microsoft Corporation. All rights reserved.

C:\Users>cd /java
C:\Java>javac client.java
C:\Java>java client
C:\Java>
```


PROJECT-[DESIGNING A CAFETERIA SYSTEM]

AIM:

TO DEVELOP AN INTERACTIVE APPLICATION OF CAFETERIA MANAGEMENT SYSYTEM.

Components:

- FRAME1[LOGIN]



- BUTTON1[LOGIN]-
- BUTTON2[SIGNUP]
- BUTTON3[UPDATE]
- BUTTON4[VIEW]
- BUTTON5[LOGOUT]
- FRAME2[SIGNUP]

APEEJAY STYA UNIVERSITY SIGNUP

NAME:

ENROLLMENTNO.:

PASSWORD:

CREDITS:

APEEJAY STYA UNIVERSITY SIGNUP

NAME:

ENROLLMENTNO.:

PASSWORD:

CREDITS:

Message

You Missed a Field



- BUTTON1[STORE]
- BUTTON2[DELETE]
- BUTTON3[OK]
- BUTTON[VIEW]
- FRAME3[UPDATE]



- BUTTON1[UPDATE]
- BUTTON2[OK]
- BUTTON[VIEW]
- FRAME4[VIEW]

A screenshot of a Java Swing window titled "List of user". It contains a table with 5 columns: enroll, user_name, user_type, user_pass, and credit. The table has 6 rows of data. Below the table is a large empty rectangular area. At the bottom right of the window is a green button labeled "OK".

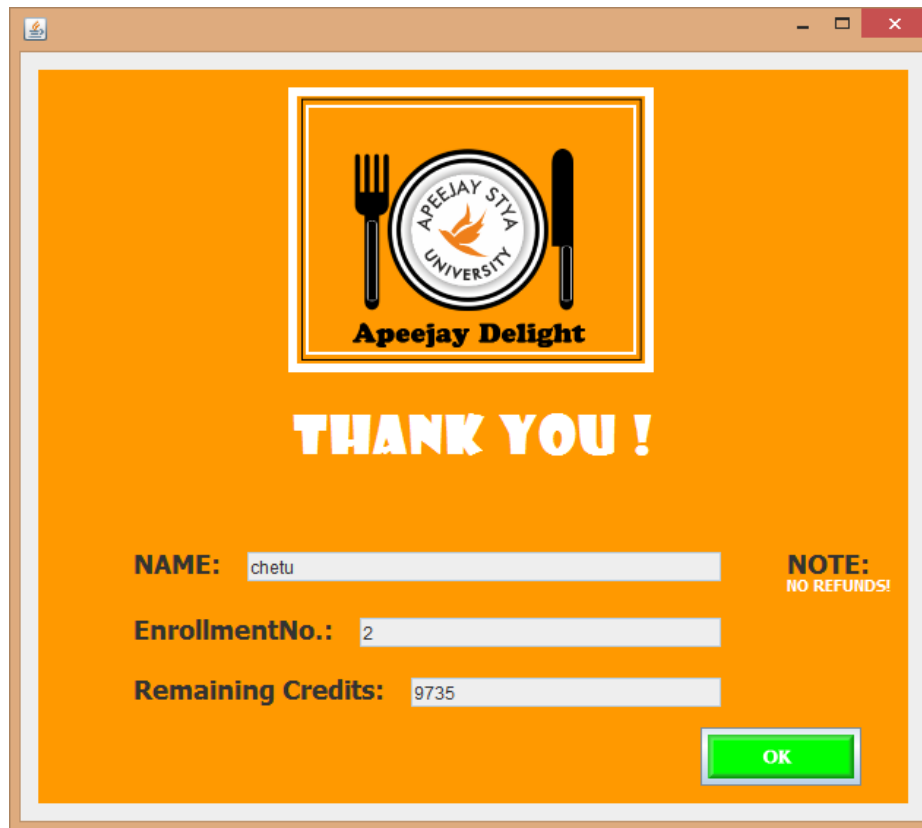
enroll	user_name	user_type	user_pass	credit
1	marc	student	1234	999999910
2	chetu		1234	9940
12	abhinav		123	0
51	prateek		12345	0
123	anshuman		123	430

- BUTTON1[OK]
- FRAME5[CAFE]

A screenshot of a Java Swing window titled "WE SERVE YOU:". It features a menu board on the left with the word "Menu" on it. The main area displays three columns of food and drink items, each with a checkbox and a price. At the bottom, there is a "BILL:" label followed by a text box containing the number "110", and a green "PURCHASE" button.

Snacks	Beverages	Meals
<input checked="" type="checkbox"/> Sandwich-Rs.20/-	<input type="checkbox"/> Cold Drinks-Rs.15/-	<input type="checkbox"/> Veg.Nod-Rs.20/-
<input checked="" type="checkbox"/> Pastry-Rs.20/-	<input type="checkbox"/> Tea-Rs.7/-	<input type="checkbox"/> Egg-Nod-Rs.25/-
<input type="checkbox"/> Chips-Rs.20/-	<input type="checkbox"/> Coffee-Rs.10/-	<input type="checkbox"/> Omllette-Rs....
<input type="checkbox"/> Biscuits-Rs.10/-	<input type="checkbox"/> Juice-Rs.20/-	<input checked="" type="checkbox"/> Kulcha-Rs.20/-
<input type="checkbox"/> Namkeen-Rs.10/-	<input checked="" type="checkbox"/> Milkshake-Rs.20/-	<input checked="" type="checkbox"/> ChillyPot-Rs.30/-
<input type="checkbox"/> Muffins-Rs.8/-	<input type="checkbox"/> Smoothy-Rs.20/-	<input type="checkbox"/> PawBhaji-Rs.35/-

- TEXTBOX[BILL]
- BUTTON1[PURCHASE]
- FRAME6[THANKU]



THANK YOU !

NAME:

EnrollmentNo.:

Remaining Credits:

NOTE:
NO REFUNDS!

OK

- TEXTBOX1[NAME]
- TEXTBOX2[ENROLSMENT NO]
- TEXTBOX3[REMAINING CREDITS]
- BUTTON[OK]

Codings:

Components:

- FRAME1[LOGIN]
 - BUTTON1[LOGIN]

```

225
226 private void b1ActionPerformed(java.awt.event.ActionEvent evt) {
227     // TODO add your handling code here:
228
229     String userName = getUsername();
230     String password = getPassword();
231     getUsername();
232     getPassword();
233
234
235     if(r1.isSelected())
236     {
237         String query = "SELECT *FROM java_progl where user_name = '"+userName+"'";
238         String que="insert into wholog (user_name,user_pass) values ('"+userName+"','"+password+"')";
239         Connect con = new Connect();
240         PreparedStatement ps,ps2 ;
241         ResultSet rs;
242         try
243         {
244             ps = con.getConnection().prepareStatement(query);
245             ps2=con.getConnection().prepareStatement(que);
246
247             rs = ps.executeQuery(query);
248             ps2.executeUpdate(que);
249             while(rs.next()){
250                 if(userName.equals(rs.getString("user_name")) && password.equals(rs.getString("user_pass")))
251                 {
252                     cafe_frame m = new cafe_frame() ;
253                     m.setVisible(true);
254                     dispose();
255                 }
256             }
257         }
258     }

```

```

259     }
260 }
261 public String getUsername()
262 {
263     String userName = t1.getText();
264     return userName;
265 }
266
267 public String getPassword()
268 {
269     String password = new String(pw1.getPassword());
270     return password;
271 }
272
273 private void b2ActionPerformed(java.awt.event.ActionEvent evt) {
274     // TODO add your handling code here:
275     signup_frame m2=new signup_frame();
276     m2.setVisible(true);
277     dispose();
278 }
279
280 private void b3ActionPerformed(java.awt.event.ActionEvent evt) {
281     // TODO add your handling code here:
282     update_frame m3=new update_frame();
283     m3.setVisible(true);
284     dispose();
285 }
286
287 private void b4ActionPerformed(java.awt.event.ActionEvent evt) {
288     // TODO add your handling code here:
289     View m4 = new View();
290     m4.setVisible(true);

```

```

297     catch(SQLException e)
298     {
299         System.out.println(e);
300     }
301 }
302
303 if(r2.isSelected())
304 {
305     String query = "SELECT *FROM java_progl where user_name = '"+ userName+"'";
306     String que="insert into wholog (user_name,user_pass) values ('"+userName+"','"+password+"')";
307     Connect con = new Connect();
308     PreparedStatement ps,ps2 ;
309     ResultSet rs;
310     try
311     {
312         ps = con.getConnection().prepareStatement(query);
313         ps2=con.getConnection().prepareStatement(que);
314         ps2.executeUpdate(que);
315         rs = ps.executeQuery(query);
316
317         while(rs.next()){
318             if(userName.equals(rs.getString("user_name")) && password.equals(rs.getString("user_pass")))
319             {
320                 cafe_frame m1 = new cafe_frame() ;
321                 m1.setVisible(true);
322                 dispose();
323             }
324         }
325     }
326     catch(SQLException e)
327     {
328         System.out.println(e);
329     }
330 }

```

```

289     }
290 }
291 public String getUsername()
292 {
293     String userName = t1.getText();
294     return userName;
295 }
296
297 public String getPassword()
298 {
299
300     String password = new String(pw1.getPassword());
301     return password;
302 }

```

○ BUTTON2[SIGNUP]

```

303 private void b2ActionPerformed(java.awt.event.ActionEvent evt) {
304     // TODO add your handling code here:
305     signup_frame m2=new signup_frame();
306     m2.setVisible(true);
307     dispose();
308 }

```

○ BUTTON3[UPDATE]

```

310 private void b3ActionPerformed(java.awt.event.ActionEvent evt) {
311     // TODO add your handling code here:
312     update_frame m3=new update_frame();
313     m3.setVisible(true);
314     dispose();
315 }

```

○ BUTTON4[VIEW]

```

317 private void b4ActionPerformed(java.awt.event.ActionEvent evt) {
318     // TODO add your handling code here:
319     View m4 = new View();
320     m4.setVisible(true);
321     dispose();
322 }

```

○ BUTTON5[LOGOUT]

```

332 private void jButton1ActionPerformed(java.awt.event.ActionEvent evt) {
333     // TODO add your handling code here:
334     String query = "delete FROM wholog; ";
335     Connect con = new Connect();
336     PreparedStatement ps ;
337
338     try
339     {
340         ps = con.getConnection().prepareStatement(query);
341
342         ps.executeUpdate(query);
343     }
344
345     catch(SQLException e)
346     {
347         System.out.println(e);
348     }
349
350 }
351

```

- **FRAME2[SIGNUP]**

- **BUTTON1[STORE]**

```
209 private void b1ActionPerformed(java.awt.event.ActionEvent evt) {
210     // TODO add your handling code here:
211     String name = t1.getText();
212     String roll = t2.getText();
213     String pass = new String(pw1.getPassword());
214     String credit = t4.getText();
215
216
217     String query = "insert into java_progl (enroll,user_name, user_pass, credit)"
218         + "values('"+roll+"', '"+name+"', '"+pass+"', '"+credit+"')";
219     if(roll.equals("") || name.equals("") || pass.equals("") || credit.equals(""))
220     {
221         JOptionPane.showMessageDialog(this,"You Missed a Field");
222     }
223     else
224     {
225         Connect con = new Connect();
226         PreparedStatement ps ;
227         ResultSet rs;
228         try
229         {
230             ps = con.getConnection().prepareStatement(query);
231
232             ps.executeUpdate(query);
233
234         }
235         catch(SQLException e)
236         {
237             System.out.println(e);
238         }
239
240
241         JOptionPane.showMessageDialog(this,"Added Successfully");
242     }
}
```

- **BUTTON2[DELETE]**


```

251 private void b2ActionPerformed(java.awt.event.ActionEvent evt) {
252     // TODO add your handling code here:
253     String roll = t2.getText();
254     String password = new String(pw1.getPassword());
255
256
257     String query = " select *FROM java_prog1 where enroll = '"+roll+"'";
258     Connect con = new Connect();
259     PreparedStatement ps ,ps2;
260     ResultSet rs;
261     try
262     {
263         ps = con.getConnection().prepareStatement(query);
264
265         rs =ps.executeQuery(query);
266         while(rs.next())
267         {
268             if(roll.equals(rs.getString("enroll")) && password.equals(rs.getString("user_pass")))
269             {
270
271                 String en = t2.getText();
272                 String pass = new String(pw1.getPassword());
273                 String quer = " delete from java_prog1 where enroll = '"+en+"' && user_pass = '"+pass+"'";
274
275                 ps2 = con.getConnection().prepareStatement(quer);
276
277                 ps2.executeUpdate(quer);
278
279                 JOptionPane.showMessageDialog(this,"delete Successfully");
280
281             }
282
283         }
284     } catch (SQLException e)
285     {
286         System.out.println(e);
287     }
288 }

```

- **BUTTON3[OK]**

```

244 private void b3ActionPerformed(java.awt.event.ActionEvent evt) {
245     // TODO add your handling code here:
246     login_frame m = new login_frame();
247     m.setVisible(true);
248     dispose();
249 }

```

- **BUTTON4[VIEW]::SAME AS BUTTON4 IN FRAME[LOGIN].**
- **FRAME3[UPDATE]**
 - **BUTTON1[UPDATE]**

```

170
171 private void biActionPerformed(java.awt.event.ActionEvent evt) {
172     // TODO add your handling code here:
173     String roll = t1.getText();
174     String password = new String(t2.getPassword());
175
176
177     String query = "SELECT *FROM java_progl where enroll = '"+roll+"'";
178     Connect con = new Connect();
179     PreparedStatement ps,ps2;
180     ResultSet rs;
181     try
182     {
183         ps = con.getConnection().prepareStatement(query);
184
185         rs = ps.executeQuery(query);
186         while(rs.next()){
187             if(roll.equals(rs.getString("enroll")) && password.equals(rs.getString("user_pass"))){
188
189                 String cre = t3.getText();
190                 int credit=Integer.parseInt(cre);
191                 String en = t1.getText();
192                 String pass = new String(t2.getPassword());
193                 String quer = " update java_progl set credit = '"+credit+"' where enroll = '"+en+"'&&user_pass='"+pass+"'";
194                 ps2 = con.getConnection().prepareStatement(quer);
195
196                 ps2.executeUpdate(quer);
197                 JOptionPane.showMessageDialog(this,"Update Successful");
198             }
199         }

```

```

198     }
199 }
200 catch(SQLException e)
201 {
202     System.out.println(e);
203 }
204
205 }

```

- **BUTTON2[OK]:: SAME AS BUTTON3 IN FRAME[SIGNUP].**
- **BUTTON[VIEW]:: SAME AS BUTTON4 IN FRAME[LOGIN].**
- **FRAME4[VIEW]**
 - **BUTTON1[OK]:: SAME AS BUTTON3 IN FRAME[SIGNUP].**
- **FRAME5[CAFE]**
 - **TEXTBOX[BILL]**

```

465 private void jCheckBox8ActionPerformed(java.awt.event.ActionEvent evt) {
466     // TODO add your handling code here:
467     bill = bill + 7 ;
468     t1.setText(""+bill+"");
469 }

```

EVERY CHECKBOX HAVE CODE TO UPDATE BILL AND SET IT TO TEXTBOX

- **BUTTON1[PURCHASE]**

```

477 private void biActionPerformed(java.awt.event.ActionEvent evt) {
478     // TODO add your handling code here:
479     String que = "SELECT *FROM wholog";
480     Connect con = new Connect();
481     PreparedStatement ps1 ;
482     ResultSet rs1;
483     try
484     {
485         ps1=con.getConnection().prepareStatement(que);
486         rs1 = ps1.executeQuery(que);
487         while(rs1.next())
488         {
489
490
491             String userName = rs1.getString("user_name");
492             String passWord = rs1.getString("user_pass");
493
494             String query = "SELECT *FROM java_progl where user_name = '"+userName+"' && user_pass = '"+passWord+"'";
495
496             PreparedStatement ps ;
497             ResultSet rs;
498             try
499             {
500                 ps = con.getConnection().prepareStatement(query);
501
502                 rs = ps.executeQuery(query);
503                 while(rs.next()){
504                     String line = rs.getString("credit");
505                     int store = Integer.parseInt(line);
506                     if(userName.equals(rs.getString("user_name")) && passWord.equals(rs.getString("user_pass"))){
507

```

```

507     {
508         if(bill < store)
509         {
510             store = store - bill;
511             String quer = "update java_progl set credit = '"+store+"' where user_name = '"+userName+"' &&
512
513             try
514             {
515                 ps = con.getConnection().prepareStatement(quer);
516
517                 ps.executeUpdate(quer);
518
519             }
520             catch(SQLException e)
521             {
522                 System.out.println(e);
523             }
524             thanks_frame k = new thanks_frame();
525             k.setVisible(true);
526             dispose();
527         }
528         else
529         {
530             JOptionPane.showMessageDialog(this, "you need to update your credits");
531             login_frame n = new login_frame();
532             n.setVisible(true);
533             dispose();
534         }
535     }
536 }
537 catch(SQLException e)

```

```

537     catch(SQLException e)
538     {
539         System.out.println(e);
540     }
541 }
542 catch(SQLException e)
543 {
544     System.out.println(e);
545 }
546 }

```

- **FRAME6[THANKU]**

○ **BUTTON[OK]::**

```
231 private void b1ActionPerformed(java.awt.event.ActionEvent evt) {  
232     // TODO add your handling code here:  
233     login_frame m=new login_frame();  
234     m.setVisible(true);  
235     dispose();  
236  
237     String query = "delete FROM wholog; ";  
238     Connect con = new Connect();  
239     PreparedStatement ps ;  
240  
241     try  
242     {  
243         ps = con.getConnection().prepareStatement(query);  
244  
245         ps.executeUpdate(query);  
246     }  
247  
248     catch(SQLException e)  
249     {  
250         System.out.println(e);  
251     }  
252  
253  
254 }  
255
```