

```
1 INFO [AWT-EventQueue-0]: com.RomTal.java.  
PassGeneratorDemo$AddPassListener Account: 5 Added  
Successfully
```

2

```
1 <?xml version="1.0" encoding="UTF-8"?>
2 <module type="JAVA_MODULE" version="4">
3   <component name="NewModuleRootManager" inherit-compiler-
4     <exclude-output />
5     <content url="file://$MODULE_DIR$">
6       <sourceFolder url="file://$MODULE_DIR$/src"
7         isTestSource="false" />
8       <sourceFolder url="file://$MODULE_DIR$/Tests"
9         isTestSource="true" />
10      </content>
11      <orderEntry type="inheritedJdk" />
12      <orderEntry type="sourceFolder" forTests="false" />
13      <orderEntry type="library" name="lib" level="project"
14        />
15      <orderEntry type="library" name="log4j" level="project"
16        />
17      <orderEntry type="library" exported="" name="junit-4.
18        level="project" />
19      <orderEntry type="module-library" scope="TEST">
20        <library name="JUnit5.4">
21          <CLASSES>
22            <root url="jar://$MAVEN_REPOSITORY$/org/junit/
23              jupiter/junit-jupiter/5.4.2/junit-jupiter-5.4.2.jar!/" />
24            <root url="jar://$MAVEN_REPOSITORY$/org/junit/
25              jupiter/junit-jupiter-api/5.4.2/junit-jupiter-api-5.4.2.
26              jar!/" />
27            <root url="jar://$MAVEN_REPOSITORY$/org/
28              apiguardian/apiguardian-api/1.0.0/apiguardian-api-1.0.0.
29              jar!/" />
30            <root url="jar://$MAVEN_REPOSITORY$/org/
31              opentest4j/opentest4j/1.1.1/opentest4j-1.1.1.jar!/" />
32            <root url="jar://$MAVEN_REPOSITORY$/org/junit/
33              platform/junit-platform-commons/1.4.2/junit-platform-
34              commons-1.4.2.jar!/" />
35            <root url="jar://$MAVEN_REPOSITORY$/org/junit/
36              jupiter/junit-jupiter-params/5.4.2/junit-jupiter-params-5.
37              4.2.jar!/" />
38            <root url="jar://$MAVEN_REPOSITORY$/org/junit/
39              jupiter/junit-jupiter-engine/5.4.2/junit-jupiter-engine-5.
40              4.2.jar!/" />
41            <root url="jar://$MAVEN_REPOSITORY$/org/junit/
42              platform/junit-platform-engine/1.4.2/junit-platform-engine
43              -1.4.2.jar!/" />
44          </CLASSES>
45          <JAVADOC />
```

```
27      <SOURCES />
28      </library>
29  </orderEntry>
30  <orderEntry type="module-library" scope="TEST">
31      <library name="JUnit5.4">
32          <CLASSES>
33              <root url="jar://$MAVEN_REPOSITORY$/org/junit/
jupiter/junit-jupiter/5.4.2/junit-jupiter-5.4.2.jar!/" />
34              <root url="jar://$MAVEN_REPOSITORY$/org/junit/
jupiter/junit-jupiter-api/5.4.2/junit-jupiter-api-5.4.2.
jar!/" />
35              <root url="jar://$MAVEN_REPOSITORY$/org/
apiguardian/apiguardian-api/1.0.0/apiguardian-api-1.0.0.
jar!/" />
36              <root url="jar://$MAVEN_REPOSITORY$/org/
opentest4j/opentest4j/1.1.1/opentest4j-1.1.1.jar!/" />
37              <root url="jar://$MAVEN_REPOSITORY$/org/junit/
platform/junit-platform-commons/1.4.2/junit-platform-
commons-1.4.2.jar!/" />
38              <root url="jar://$MAVEN_REPOSITORY$/org/junit/
jupiter/junit-jupiter-params/5.4.2/junit-jupiter-params-5.
4.2.jar!/" />
39              <root url="jar://$MAVEN_REPOSITORY$/org/junit/
jupiter/junit-jupiter-engine/5.4.2/junit-jupiter-engine-5.
4.2.jar!/" />
40              <root url="jar://$MAVEN_REPOSITORY$/org/junit/
platform/junit-platform-engine/1.4.2/junit-platform-engine
-1.4.2.jar!/" />
41          </CLASSES>
42          <JAVADOC />
43          <SOURCES />
44      </library>
45  </orderEntry>
46  </component>
47 </module>
```

```
1 //
2 // Source code recreated from a .class file by IntelliJ
3 // (powered by Fernflower decompiler)
4 //
5
6 package com.RomTal.java;
7
8 import java.sql.ResultSet;
9 import java.sql.SQLException;
10 import org.junit.jupiter.api.AfterEach;
11 import org.junit.jupiter.api.BeforeEach;
12 import org.junit.jupiter.api.Test;
13
14 class DerbyDBTest {
15     public DerbyDB database;
16     public String Key = "KeyChecking";
17     public ResultSet res;
18
19     DerbyDBTest() {
20     }
21
22     @BeforeEach
23     void setUp() {
24         this.database = new DerbyDB();
25     }
26
27     @AfterEach
28     void tearDown() {
29         this.database = null;
30     }
31
32     @Test
33     void closeConnection() throws SQLException {
34         this.database.closeConnection();
35     }
36
37     @Test
38     void displayUsers() throws SQLException {
39         this.database.displayUsers();
40     }
41
42     @Test
43     void displayInfo() throws SQLException {
44         this.database.displayInfo(this.Key);
45     }
```

```
46
47     @Test
48     void displayAmount() throws SQLException {
49         this.database.displayAmount();
50     }
51
52     @Test
53     void addUser() {
54         this.database.addUser("Account", "UserName", "URL"
55 , "Password", "Key");
55         this.database.addUser("NewAccount", "UserName", "URL",
56 "Password", this.Key);
56     }
57
58     @Test
59     void changeUser() throws SQLException {
60         this.res = this.database.displayInfo(this.Key);
61         this.res.next();
62         this.database.changeUser(this.Key, "NewUserName");
63     }
64
65     @Test
66     void changePass() throws SQLException {
67         this.res = this.database.displayInfo(this.Key);
68         this.res.next();
69         this.database.changePass(this.Key, "NewPassword");
70     }
71
72     @Test
73     void changeSite() throws SQLException {
74         this.res = this.database.displayInfo(this.Key);
75         this.res.next();
76         this.database.changeSite(this.Key, "NewUrl");
77     }
78
79     @Test
80     void deleteUser() {
81         this.database.deleteUser(this.Key);
82     }
83 }
84
```

```
1 //
2 // Source code recreated from a .class file by IntelliJ
3 // (powered by Fernflower decompiler)
4 //
5
6 package com.RomTal.java;
7
8 import org.junit.jupiter.api.AfterEach;
9 import org.junit.jupiter.api.BeforeEach;
10 import org.junit.jupiter.api.Test;
11
12 class PasswordTest {
13     public static Password password;
14     int test = 20;
15     char test1 = 'R';
16     String Pass = "PassChecking";
17
18     PasswordTest() {
19     }
20
21     @BeforeEach
22     void setUp() {
23         password = new Password();
24     }
25
26     @AfterEach
27     void tearDown() {
28         password = null;
29     }
30
31     @Test
32     void generatePassword() {
33         password.generatePassword();
34     }
35
36     @Test
37     void testGeneratePassword() {
38         password.generatePassword(this.test);
39     }
40
41     @Test
42     void testGeneratePassword1() {
43         password.generatePassword(this.test1);
44     }
45
```

```
46     @Test
47     void testEquals() {
48         password.equals(this.Pass);
49     }
50
51     @Test
52     void hasSymbols() {
53         password.hasSymbols();
54     }
55
56     @Test
57     void testToString() {
58         password.toString();
59     }
60 }
61
```

```
1 //
2 // Source code recreated from a .class file by IntelliJ
3 // IDEA
4 // (powered by Fernflower decompiler)
5
6 package com.RomTal.java;
7
8 import org.junit.jupiter.api.AfterEach;
9 import org.junit.jupiter.api.BeforeEach;
10 import org.junit.jupiter.api.Test;
11
12 class TablePanelTest {
13     public TablePanel table;
14     String Pass = "PassChecking";
15
16     TablePanelTest() {
17     }
18
19     @BeforeEach
20     void setUp() {
21         this.table = new TablePanel();
22     }
23
24     @AfterEach
25     void tearDown() {
26         this.table = null;
27     }
28
29     @Test
30     void testRebuildPanel() {
31         this.table.rebuildPanel();
32     }
33
34     @Test
35     void testGetSelection() {
36         this.table.getSelection();
37     }
38
39     @Test
40     void testGetSelectionKey() {
41         this.table.getSelectionKey();
42     }
43
44     @Test
45     void testSetSelection() {
```

```
46         this.table.setSelection();
47     }
48
49     @Test
50     void testDisplayCopy() {
51         this.table.displayCopy(this.Pass);
52     }
53 }
54
```

```
1 //
2 // Source code recreated from a .class file by IntelliJ
3 // (powered by Fernflower decompiler)
4 //
5
6 package com.RomTal.java;
7
8 import org.junit.jupiter.api.AfterEach;
9 import org.junit.jupiter.api.BeforeEach;
10 import org.junit.jupiter.api.Test;
11
12 class PassGeneratorDemoTest {
13     public PassGeneratorDemo Application;
14
15     PassGeneratorDemoTest() {
16     }
17
18     @BeforeEach
19     void setUp() {
20         this.Application = new PassGeneratorDemo();
21     }
22
23     @AfterEach
24     void tearDown() {
25         this.Application = null;
26     }
27
28     @Test
29     void main() {
30     }
31 }
32
```

```
1 ## Root Location Option !!
2 #Log4j.rootCategory=debug,console
3 #
4 ## Package Based Logging
5 #Log4j.Logger.com.jcg.Log4j.console.appendер=debug,console
6 #Log4j.additivity.com.jcg.Log4j.console.appendер=false
7 #
8 ## Redirect Log Messages To Console !!
9 #log4j.appendер.console=org.apache.log4j.ConsoleAppender
10 #log4j.appendер.console.target=System.out
11 #log4j.appendер.console.immediateFlush=true
12 #log4j.appendер.console.encoding=UTF-8
13 #log4j.appendер.console.Layout=org.apache.log4j.
    PatternLayout
14 #log4j.appendер.console.Layout.conversionPattern=%d{yyyy-
    MM-dd HH:mm:ss} %-5p %c{1}:%L - %m%n
15 #public static Logger logger2=Logger.getLogger(Class.class
    .getName());
16
```

```
1 //
2 // Source code recreated from a .class file by IntelliJ
3 // (powered by Fernflower decompiler)
4 //
5
6 package com.RomTal.java;
7
8 public interface IView {
9     void rebuildPanel();
10
11    String getSelection();
12
13    String getSelectionKey();
14
15    void setSelection();
16
17    void displayCopy(String var1);
18
19    void displayWarning();
20 }
21
```

```
1 //
2 // Source code recreated from a .class file by IntelliJ
3 // (powered by Fernflower decompiler)
4 //
5
6 package com.RomTal.java;
7
8 public interface IModel {
9     void generatePassword();
10    void generatePassword(char var1);
11    void generatePassword(int var1);
12
13    boolean equals(String var1);
14
15    boolean hasSymbols();
16
17    String toString();
18
19 }
20
21
```

```
1 //
2 // Source code recreated from a .class file by IntelliJ
3 // (powered by Fernflower decompiler)
4 //
5
6 package com.RomTal.java;
7
8 import java.awt.Component;
9 import java.sql.Connection;
10 import java.sql.DatabaseMetaData;
11 import java.sql.DriverManager;
12 import java.sql.PreparedStatement;
13 import java.sql.ResultSet;
14 import java.sql.SQLException;
15 import java.sql.Statement;
16 import javax.swing.JOptionPane;
17
18 public class DerbyDB {
19     private Connection conn;
20     public static int flag = 0;
21     private boolean hasData = false;
22     public static String driver = "org.apache.derby.jdbc.
ClientDriver";
23     public static String protocol = "jdbc:derby://
localhost:1527/gagamoDB;create=true";
24
25     public DerbyDB() {
26     }
27
28     private void getConnection() {
29         try {
30             Class.forName(driver);
31             this.conn = DriverManager.getConnection(
32                 protocol);
33             System.out.println("Established Connection to
DB");
34             this.initialize();
35         } catch (SQLException | ClassNotFoundException
var2) {
36             JOptionPane.showMessageDialog((Component)null
            , var2.getMessage());
37         }
38     }
39 }
```

```
40     private void initialize() {
41         if (!this.hasData) {
42             this.hasData = true;
43
44             try {
45                 Statement state = this.conn.
46                     createStatement();
47                 Statement statement = this.conn.
48                     createStatement();
49                 if (flag == 0) {
50                     System.out.println("Checking database
51                         for table");
52                     DatabaseMetaData databaseMetadata =
53                         this.conn.getMetaData();
54                     ResultSet resultSet = databaseMetadata
55                         .getTables((String)null, (String)null, "DATA", (String[])
56                         null);
57                     if (resultSet.next()) {
58                         System.out.println("TABLE ALREADY
59                         EXISTS");
60                     } else {
61                         state.execute("create table data(
62                             account varchar(60) ,username varchar(60) ,url varchar(120)
63                             ,password varchar(60),KeyVal varchar(60))");
64                     }
65                     ++flag;
66                 }
67             }
68         }
69
70     }
71
72     public void closeConnection() throws SQLException {
```

```
73         if (this.conn == null) {
74             this.getConnection();
75         }
76
77         this.conn.close();
78         this.conn = null;
79         System.out.println("Layout panel successfully
close connection.");
80     }
81
82     public ResultSet displayUsers() throws SQLException {
83         if (this.conn == null) {
84             this.getConnection();
85         }
86
87         Statement state = this.conn.createStatement();
88         ResultSet res = state.executeQuery("SELECT
account, username, url, password, KeyVal FROM data");
89         return res;
90     }
91
92     public ResultSet displayInfo(String KeyVal) throws
SQLException {
93         if (this.conn == null) {
94             this.getConnection();
95         }
96
97         Statement state = this.conn.createStatement();
98         ResultSet res = state.executeQuery("SELECT
account, username, url, password FROM data WHERE KeyVal
= '" + KeyVal + "'");
99         return res;
100    }
101
102    public int displayAmount() throws SQLException {
103        if (this.conn == null) {
104            this.getConnection();
105        }
106
107        int count = 0;
108        PreparedStatement state = this.conn.
prepareStatement("SELECT * FROM data");
109
110        for(ResultSet res = state.executeQuery(); res.
next(); ++count) {
111        }
```

```
112
113         return count;
114     }
115
116     public void addUser(String account, String username,
117                           String url, String password, String KeyVal) {
117         if (this.conn == null) {
118             this.getConnection();
119         }
120
121         try {
122             PreparedStatement prep = this.conn.
123             prepareStatement("INSERT INTO data VALUES(?, ?, ?, ?, ?, ?)");
123             prep.setString(1, account);
124             prep.setString(2, username);
125             prep.setString(3, url);
126             prep.setString(4, password);
127             prep.setString(5, KeyVal);
128             prep.execute();
129         } catch (SQLException var7) {
130             JOptionPane.showMessageDialog((Component)null
130             , var7.getMessage());
131         }
132
133     }
134
135     public void changeUser(String KeyVal, String
136                           newUsername) {
136         if (this.conn == null) {
137             this.getConnection();
138         }
139
140         try {
141             PreparedStatement prep = this.conn.
141             prepareStatement("UPDATE data SET username = '" +
142                           newUsername + "' WHERE KeyVal = '" + KeyVal + "'");
142             prep.execute();
143         } catch (SQLException var4) {
144             JOptionPane.showMessageDialog((Component)null
144             , var4.getMessage());
145         }
146
147     }
148
149     public void changePass(String KeyVal, String
149                           newPassword) {
```

```
150         if (this.conn == null) {
151             this.getConnection();
152         }
153
154         try {
155             PreparedStatement prep = this.conn.
156             prepareStatement("UPDATE data SET password = '" +
157                 newPassword + "' WHERE KeyVal = '" + KeyVal + "'");
158             prep.execute();
159         } catch (SQLException var4) {
160             JOptionPane.showMessageDialog((Component)null
161             , var4.getMessage());
162         }
163     public void changeSite(String KeyVal, String newUrl
164     ) {
165         if (this.conn == null) {
166             this.getConnection();
167         }
168         try {
169             PreparedStatement prep = this.conn.
170             prepareStatement("UPDATE data SET url = '" + newUrl + "'"
171                 + " WHERE KeyVal = '" + KeyVal + "'");
172             prep.execute();
173         } catch (SQLException var4) {
174             JOptionPane.showMessageDialog((Component)null
175             , var4.getMessage());
176         }
177     public void deleteUser(String KeyVal) {
178         if (this.conn == null) {
179             this.getConnection();
180         }
181
182         try {
183             PreparedStatement prep = this.conn.
184             prepareStatement("DELETE FROM data WHERE KeyVal = '" +
185                 KeyVal + "'");
186             prep.execute();
187         } catch (SQLException var3) {
188             JOptionPane.showMessageDialog((Component)null
```

```
186 , var3.getMessage());  
187 }  
188  
189 }  
190 }  
191
```

```
1 //
2 // Source code recreated from a .class file by IntelliJ
3 // (powered by FernFlower decompiler)
4 //
5
6 package com.RomTal.java;
7
8 import java.security.SecureRandom;
9
10 public class Password implements IModel {
11     private final byte PASSWORD_LENGTH = 16;
12     private final String CHARACTER_DICTIONARY =
13         abcdefghijklmnopqrstuvwxyzABCDEFGHIJKLMNOPQRSTUVWXYZ123456
14         7890!?@#$%";;
15     private String password;
16     private SecureRandom secureGenerator = new
17     SecureRandom();
18
19
20     public Password() {
21         this.password = "";
22     }
23
24     public Password(String existingPass) {
25         this.password = existingPass;
26     }
27
28     public void generatePassword() {
29         for(String[] arrString = new String[16]; !this.
30         hasSymbols(); this.password = String.join("", arrString
31 )) {
32             for(int i = 0; i < 16; ++i) {
33                 arrString[i] = Character.toString(
34                     abcdefghijklmnopqrstuvwxyzABCDEFGHIJKLMNOPQRSTUVWXYZ123456
35                     7890!?@#$%.charAt(this.secureGenerator.nextInt(
36                     abcdefghijklmnopqrstuvwxyzABCDEFGHIJKLMNOPQRSTUVWXYZ123456
37                     7890!?@#$%.length())));
38             }
39         }
40     }
41 }
```

```
37     public void generatePassword(char startingChar) {
38         for(String[] arrString = new String[16]; !this.
39             hasSymbols(); this.password = String.join("", arrString
40             )) {
41             arrString[0] = Character.toString(startingChar
42             );
43             for(int i = 1; i < 16; ++i) {
44                 arrString[i] = Character.toString(
45                     "abcdefghijklmnopqrstuvwxyzABCDEFGHIJKLMNOPQRSTUVWXYZ123456
46                     7890!?@#$%".charAt(this.secureGenerator.nextInt(
47                     "abcdefghijklmnopqrstuvwxyzABCDEFGHIJKLMNOPQRSTUVWXYZ123456
48                     7890!?@#$%".length())));
49             }
50         }
51     }
52
53     public void generatePassword(int length) {
54         for(String[] arrString = new String[length]; !this.
55             .hasSymbols(); this.password = String.join("", arrString
56             )) {
57             for(int i = 0; i < 16; ++i) {
58                 arrString[i] = Character.toString(
59                     "abcdefghijklmnopqrstuvwxyzABCDEFGHIJKLMNOPQRSTUVWXYZ123456
60                     7890!?@#$%".charAt(this.secureGenerator.nextInt(
61                     "abcdefghijklmnopqrstuvwxyzABCDEFGHIJKLMNOPQRSTUVWXYZ123456
62                     7890!?@#$%".length())));
63             }
64         }
65     }
66     public boolean equals(String compareString) {
67         return this.password == compareString;
68     }
69
70     public boolean hasSymbols() {
71         return this.password.contains("?") || this.
72             password.contains("!") || this.password.contains("@") ||
73             this.password.contains("#") || this.password.contains("$"
74             ) || this.password.contains("%");
75     }
76
77     public String toString() {
78         return this.password;
```

```
67      }
68 }
69
```

```
1 //
2 // Source code recreated from a .class file by IntelliJ
3 // (powered by Fernflower decompiler)
4 //
5
6 package com.RomTal.java;
7
8 public interface IViewModel {
9     static void main(String[] args) throws Exception {
10         new PassGeneratorDemo();
11     }
12 }
13
```

```
1 //
2 // Source code recreated from a .class file by IntelliJ
3 // (powered by Fernflower decompiler)
4 //
5
6 package com.RomTal.java;
7
8 import com.RomTal.java.PassGeneratorDemo.AddPassListener;
9 import com.RomTal.java.PassGeneratorDemo.AltPassListener;
10 import com.RomTal.java.PassGeneratorDemo.AltSiteListener;
11 import com.RomTal.java.PassGeneratorDemo.AltUserListener;
12 import com.RomTal.java.PassGeneratorDemo.CopyPassListener;
13 import com.RomTal.java.PassGeneratorDemo.CopyUrlListener;
14 import com.RomTal.java.PassGeneratorDemo.CopyUserListener;
15 import com.RomTal.java.PassGeneratorDemo.DelPassListener;
16 import com.RomTal.java.PassGeneratorDemo.SeePassListener;
17 import java.awt.BorderLayout;
18 import java.awt.Component;
19 import java.awt.GridLayout;
20 import java.sql.ResultSet;
21 import java.sql.SQLException;
22 import javax.swing.BorderFactory;
23 import javax.swing.JButton;
24 import javax.swing.JFrame;
25 import javax.swing.JLabel;
26 import javax.swing.JOptionPane;
27 import javax.swing.JPanel;
28 import javax.swing.JScrollPane;
29 import javax.swing.JTable;
30 import javax.swing.ListSelectionModel;
31 import javax.swing.event.ListSelectionEvent;
32 import javax.swing.event.ListSelectionListener;
33 import javax.swing.table.AbstractTableModel;
34
35 public class TablePanel extends JFrame implements IView {
36     private DerbyDB database = new DerbyDB();
37     private String[] columnNames = new String[]{"Account
Name", "Username", "URL", "Password", "Key"};
38     private String[][] data;
39     private int amountAccounts;
40     private static String selectedAccount;
41     private static String selectedKey;
42     private JTable table;
43     private JPanel buttonPanel;
44     private JPanel mainPanel;
```

```
45
46     public TablePanel() {
47         this.setTitle("PassGenerator");
48         this.setDefaultCloseOperation(3);
49         this.setLayout(new BorderLayout());
50         this.buildButtonPanel();
51         this.buildPanel();
52         this.add(this.buttonPanel, "West");
53         this.add(this.mainPanel, "Center");
54         this.pack();
55         this.setSize(800, 600);
56         this.setVisible(true);
57     }
58
59     private void buildPanel() {
60         this.builddataArray();
61         this.mainPanel = new JPanel();
62         this.table = new JTable(new TablePanel.TableModel
63 ());
63         this.mainPanel.add(this.table);
64         this.table.setSelectionMode(0);
65         ListSelectionModel listSelectionModel = this.table
66             .getSelectionModel();
66         listSelectionModel.addListSelectionListener(new
67             TablePanel.TableListener());
67         JScrollPane scrollPane = new JScrollPane(this.
68             table);
68         this.table.getColumnModel().getColumn(4).
69             setMinWidth(0);
69         this.table.getColumnModel().getColumn(4).
70             setMaxWidth(0);
70         this.table.getColumnModel().getColumn(4).setWidth(
71             0);
71         this.mainPanel.add(scrollPane);
72         this.mainPanel.setLayout(new GridLayout(1, 0, 0, 0
73 ));
73         this.mainPanel.setBorder(BorderFactory.
74             createTitledBorder("Data"));
74     }
75
76     public void rebuildPanel() {
77         this.remove(this.mainPanel);
78         this.buildPanel();
79         this.add(this.mainPanel, "Center");
80         this.revalidate();
81         this.repaint();
```

```
82         System.gc();
83     }
84
85     private void builddataArray() {
86         try {
87             this.amountAccounts = this.database.
88             displayAmount();
89         } catch (SQLException var4) {
90             JOptionPane.showMessageDialog((Component)null
91             , var4.getMessage());
92         }
93
94         this.data = new String[this.amountAccounts][5];
95         int row = 0;
96
97         try {
98             ResultSet res;
99             for(res = this.database.displayUsers(); res.
100             next(); ++row) {
101                 this.data[row][0] = res.getString("
102                   Account");
103                 this.data[row][1] = res.getString("
104                   Username");
105                 this.data[row][2] = res.getString("URL");
106                 this.data[row][3] = "*****";
107                 this.data[row][4] = res.getString("KeyVal
108               ");
109             }
110         }
111
112         private void buildButtonPanel() {
113             this.buttonPanel = new JPanel();
114             this.buttonPanel.setLayout(new GridLayout(6, 1, 0
115             , 10));
116             JButton addPass = new JButton("Add Account");
117             JButton delPass = new JButton("Delete Account");
118             JButton seePass = new JButton("Copy/View
119               Credentials");
120             JButton altUser = new JButton("Change Username");
```

```
119         JButton altPass = new JButton("Change Password");
120         JButton altSite = new JButton("Change Website");
121         addPass.addActionListener(new AddPassListener());
122         delPass.addActionListener(new DelPassListener());
123         seePass.addActionListener(new SeePassListener());
124         altUser.addActionListener(new AltUserListener());
125         altPass.addActionListener(new AltPassListener());
126         altSite.addActionListener(new AltSiteListener());
127         this.buttonPanel.setBorder(BorderFactory.
128             createTitledBorder("Operation "));
129         this.buttonPanel.add(addPass);
130         this.buttonPanel.add(delPass);
131         this.buttonPanel.add(seePass);
132         this.buttonPanel.add(altUser);
133         this.buttonPanel.add(altPass);
134         this.buttonPanel.add(altSite);
135     }
136 
137     public String getSelection() {
138         return selectedAccount;
139     }
140 
141     public String getSelectionKey() {
142         return selectedKey;
143     }
144 
145     public void setSelection() {
146         selectedAccount = null;
147     }
148 
149     public void displayCopy(String password) {
150         JPanel display = new JPanel();
151         JButton copyUsername = new JButton("Copy Username");
152         JButton copyPassword = new JButton("Copy Password");
153         JButton copyUrl = new JButton("Copy Url");
154         JButton goSite = new JButton("Go to Site");
155         copyPassword.addActionListener(new
156             CopyPassListener());
157         copyUsername.addActionListener(new
158             CopyUserListener());
159         copyUrl.addActionListener(new CopyUrlListener());
160         JLabel passText = new JLabel("Password:");
161         display.add(passText);
162         passText = new JLabel(password);
```

```
160         display.add(passText);
161         display.add(copyUsername);
162         display.add(copyPassword);
163         display.add(copyUrl);
164         JOptionPane.showMessageDialog(this, display);
165     }
166
167     public void displayWarning() {
168         JOptionPane.showMessageDialog(this, "Warning no
account was selected, retry after selecting an account");
169     }
170
171     private class TableModel extends AbstractTableModel {
172         private TableModel() {
173             }
174
175         public int getColumnCount() {
176             return TablePanel.this.columnNames.length;
177         }
178
179         public int getRowCount() {
180             return TablePanel.this.data.length;
181         }
182
183         public String getColumnName(int col) {
184             return TablePanel.this.columnNames[col];
185         }
186
187         public String getValueAt(int rowNum, int colNum
188 ) {
189             return TablePanel.this.data[rowNum][colNum];
190         }
191
192         private class TableListener implements
193             ListSelectionListener {
194             private TableListener() {
195             }
196
197             public void valueChanged(ListSelectionEvent event
198 ) {
199                 TablePanel.selectedAccount = (String)
TablePanel.this.table.getValueAt(TablePanel.this.table.
getSelectedRow(), 0);
200                 TablePanel.selectedKey = (String)TablePanel.
this.table.getValueAt(TablePanel.this.table.
```

```
198    setSelectedRow(), 4);  
199    }  
200    }  
201 }  
202
```

```
1 //
2 // Source code recreated from a .class file by IntelliJ
3 // (powered by Fernflower decompiler)
4 //
5
6 package com.RomTal.java;
7
8 import java.awt.Component;
9 import java.awt.Toolkit;
10 import java.awt.datatransfer.Clipboard;
11 import java.awt.datatransfer.ClipboardOwner;
12 import java.awt.datatransfer.StringSelection;
13 import java.awt.event.ActionEvent;
14 import java.awt.event.ActionListener;
15 import java.io.IOException;
16 import java.io.PrintStream;
17 import java.io.PrintWriter;
18 import java.sql.ResultSet;
19 import java.sql.SQLException;
20 import javax.swing.JOptionPane;
21 import org.apache.log4j.FileAppender;
22 import org.apache.log4j.Logger;
23 import org.apache.log4j.PatternLayout;
24
25 public class PassGeneratorDemo implements IViewModel {
26     public static Logger logger;
27     private static DerbyDB database;
28     private static TablePanel table;
29     private static Clipboard clipboard;
30     private static Password password;
31     private static Password key;
32
33     public PassGeneratorDemo() {
34 }
35
36     public static void main(String[] args) throws
Exception {
37         new PassGeneratorDemo();
38     }
39
40     static {
41         String file = "logs.txt";
42         logger = Logger.getLogger(Class.class.getName());
43
44         try {
```

```
45             PrintWriter writer = new PrintWriter(file);
46             writer.print("");
47             writer.close();
48             logger.addAppender(new FileAppender(new
49             PatternLayout("%-5p [%t]: %m%n"), file));
50         } catch (IOException var2) {
51             var2.printStackTrace();
52         }
53
54         database = new DerbyDB();
55         table = new TablePanel();
56         clipboard = Toolkit.getDefaultToolkit().
57             getSystemClipboard();
58     }
59
60     public static class AltSiteListener implements
61         ActionListener {
62         public AltSiteListener() {
63     }
64
65         public void actionPerformed(ActionEvent event) {
66             try {
67                 if (PassGeneratorDemo.table.getSelection
68                     () != null) {
69                     String newSite = JOptionPane.
70                     showInputDialog(PassGeneratorDemo.table, "Enter the URL of
71                     the new website.");
72                     if (!newSite.equals("")) {
73                         ResultSet res = PassGeneratorDemo.
74                         database.displayInfo(PassGeneratorDemo.table.
75                         getSelectionKey());
76                         res.next();
77                         PassGeneratorDemo.database.
78                         changeSite(PassGeneratorDemo.table.getSelectionKey(),
79                         newSite);
80                         PrintStream var10000 = System.out;
81                         Class var10001 = this.getClass();
82                         var10000.println(var10001 + " URL
83 : " + res.getString("URL") + " Change To New Username: "
84 + newSite.toString());
85                     Logger var5 = PassGeneratorDemo.
86                     logger;
87                     var10001 = this.getClass();
88                     var5.info(var10001 + " URL: " +
89                     res.getString("URL") + " Change To New Username: " +
90                     newSite.toString());
91                 }
92             }
93         }
94     }
95 }
```

```
76                                PassGeneratorDemo.table.  
77          rebuildPanel();  
78      } else {  
79          JOptionPane.showMessageDialog(  
80          PassGeneratorDemo.table, "Warning no replacement was  
81          entered, retry after entering a replacement.");  
82      }  
83  } catch (Exception var4) {  
84      System.out.println("Cancel Pressed");  
85      PassGeneratorDemo.logger.info("Cancel  
86      Pressed");  
87  }  
88 }  
89 }  
90  
91 public static class AltPassListener implements  
92     ActionListener {  
93     public AltPassListener() {  
94     }  
95     public void actionPerformed(ActionEvent event) {  
96         PassGeneratorDemo.password = new Password();  
97         PassGeneratorDemo.password.generatePassword  
98     };  
99     if (PassGeneratorDemo.table.getSelection  
100    () == null) {  
101         PassGeneratorDemo.table.displayWarning();  
102     } else {  
103         int answer = JOptionPane.  
showConfirmDialog(PassGeneratorDemo.table, "Are you sure  
you would like to regenerate this account's password?", "  
PassGenerator", 0);  
104         if (answer == 0) {  
105             int answer2 = JOptionPane.  
showConfirmDialog(PassGeneratorDemo.table, "Would you  
like to enter your own password?", "PassGenerator", 0);  
106             if (answer2 == 0) {  
107                 String newPassword = JOptionPane.  
showInputDialog(PassGeneratorDemo.table, "Enter new  
password:");  
108                 PassGeneratorDemo.password = new  
109             }  
110         }  
111     }  
112 }  
113 }  
114 }  
115 }  
116 }  
117 }  
118 }  
119 }  
120 }  
121 }  
122 }  
123 }  
124 }  
125 }  
126 }  
127 }  
128 }  
129 }  
130 }  
131 }  
132 }  
133 }  
134 }  
135 }  
136 }  
137 }  
138 }  
139 }  
140 }  
141 }  
142 }  
143 }  
144 }  
145 }  
146 }  
147 }  
148 }  
149 }  
150 }  
151 }  
152 }  
153 }  
154 }  
155 }  
156 }  
157 }  
158 }  
159 }  
160 }  
161 }  
162 }  
163 }  
164 }  
165 }  
166 }  
167 }  
168 }  
169 }  
170 }  
171 }  
172 }  
173 }  
174 }  
175 }  
176 }  
177 }  
178 }  
179 }  
180 }  
181 }  
182 }  
183 }  
184 }  
185 }  
186 }  
187 }  
188 }  
189 }  
190 }  
191 }  
192 }  
193 }  
194 }  
195 }  
196 }  
197 }  
198 }  
199 }  
200 }  
201 }  
202 }  
203 }  
204 }  
205 }  
206 }  
207 }  
208 }  
209 }  
210 }  
211 }  
212 }  
213 }  
214 }  
215 }  
216 }  
217 }  
218 }  
219 }  
220 }  
221 }  
222 }  
223 }  
224 }  
225 }  
226 }  
227 }  
228 }  
229 }  
230 }  
231 }  
232 }  
233 }  
234 }  
235 }  
236 }  
237 }  
238 }  
239 }  
240 }  
241 }  
242 }  
243 }  
244 }  
245 }  
246 }  
247 }  
248 }  
249 }  
250 }  
251 }  
252 }  
253 }  
254 }  
255 }  
256 }  
257 }  
258 }  
259 }  
260 }  
261 }  
262 }  
263 }  
264 }  
265 }  
266 }  
267 }  
268 }  
269 }  
270 }  
271 }  
272 }  
273 }  
274 }  
275 }  
276 }  
277 }  
278 }  
279 }  
280 }  
281 }  
282 }  
283 }  
284 }  
285 }  
286 }  
287 }  
288 }  
289 }  
290 }  
291 }  
292 }  
293 }  
294 }  
295 }  
296 }  
297 }  
298 }  
299 }  
300 }  
301 }  
302 }  
303 }  
304 }  
305 }  
306 }  
307 }  
308 }  
309 }  
310 }  
311 }  
312 }  
313 }  
314 }  
315 }  
316 }  
317 }  
318 }  
319 }  
320 }  
321 }  
322 }  
323 }  
324 }  
325 }  
326 }  
327 }  
328 }  
329 }  
330 }  
331 }  
332 }  
333 }  
334 }  
335 }  
336 }  
337 }  
338 }  
339 }  
340 }  
341 }  
342 }  
343 }  
344 }  
345 }  
346 }  
347 }  
348 }  
349 }  
350 }  
351 }  
352 }  
353 }  
354 }  
355 }  
356 }  
357 }  
358 }  
359 }  
360 }  
361 }  
362 }  
363 }  
364 }  
365 }  
366 }  
367 }  
368 }  
369 }  
370 }  
371 }  
372 }  
373 }  
374 }  
375 }  
376 }  
377 }  
378 }  
379 }  
380 }  
381 }  
382 }  
383 }  
384 }  
385 }  
386 }  
387 }  
388 }  
389 }  
390 }  
391 }  
392 }  
393 }  
394 }  
395 }  
396 }  
397 }  
398 }  
399 }  
400 }  
401 }  
402 }  
403 }  
404 }  
405 }  
406 }  
407 }  
408 }  
409 }  
410 }  
411 }  
412 }  
413 }  
414 }  
415 }  
416 }  
417 }  
418 }  
419 }  
420 }  
421 }  
422 }  
423 }  
424 }  
425 }  
426 }  
427 }  
428 }  
429 }  
430 }  
431 }  
432 }  
433 }  
434 }  
435 }  
436 }  
437 }  
438 }  
439 }  
440 }  
441 }  
442 }  
443 }  
444 }  
445 }  
446 }  
447 }  
448 }  
449 }  
450 }  
451 }  
452 }  
453 }  
454 }  
455 }  
456 }  
457 }  
458 }  
459 }  
460 }  
461 }  
462 }  
463 }  
464 }  
465 }  
466 }  
467 }  
468 }  
469 }  
470 }  
471 }  
472 }  
473 }  
474 }  
475 }  
476 }  
477 }  
478 }  
479 }  
480 }  
481 }  
482 }  
483 }  
484 }  
485 }  
486 }  
487 }  
488 }  
489 }  
490 }  
491 }  
492 }  
493 }  
494 }  
495 }  
496 }  
497 }  
498 }  
499 }  
500 }  
501 }  
502 }  
503 }  
504 }  
505 }  
506 }  
507 }  
508 }  
509 }  
510 }  
511 }  
512 }  
513 }  
514 }  
515 }  
516 }  
517 }  
518 }  
519 }  
520 }  
521 }  
522 }  
523 }  
524 }  
525 }  
526 }  
527 }  
528 }  
529 }  
530 }  
531 }  
532 }  
533 }  
534 }  
535 }  
536 }  
537 }  
538 }  
539 }  
540 }  
541 }  
542 }  
543 }  
544 }  
545 }  
546 }  
547 }  
548 }  
549 }  
550 }  
551 }  
552 }  
553 }  
554 }  
555 }  
556 }  
557 }  
558 }  
559 }  
560 }  
561 }  
562 }  
563 }  
564 }  
565 }  
566 }  
567 }  
568 }  
569 }  
570 }  
571 }  
572 }  
573 }  
574 }  
575 }  
576 }  
577 }  
578 }  
579 }  
580 }  
581 }  
582 }  
583 }  
584 }  
585 }  
586 }  
587 }  
588 }  
589 }  
590 }  
591 }  
592 }  
593 }  
594 }  
595 }  
596 }  
597 }  
598 }  
599 }  
600 }  
601 }  
602 }  
603 }  
604 }  
605 }  
606 }  
607 }  
608 }  
609 }  
610 }  
611 }  
612 }  
613 }  
614 }  
615 }  
616 }  
617 }  
618 }  
619 }  
620 }  
621 }  
622 }  
623 }  
624 }  
625 }  
626 }  
627 }  
628 }  
629 }  
630 }  
631 }  
632 }  
633 }  
634 }  
635 }  
636 }  
637 }  
638 }  
639 }  
640 }  
641 }  
642 }  
643 }  
644 }  
645 }  
646 }  
647 }  
648 }  
649 }  
650 }  
651 }  
652 }  
653 }  
654 }  
655 }  
656 }  
657 }  
658 }  
659 }  
660 }  
661 }  
662 }  
663 }  
664 }  
665 }  
666 }  
667 }  
668 }  
669 }  
670 }  
671 }  
672 }  
673 }  
674 }  
675 }  
676 }  
677 }  
678 }  
679 }  
680 }  
681 }  
682 }  
683 }  
684 }  
685 }  
686 }  
687 }  
688 }  
689 }  
690 }  
691 }  
692 }  
693 }  
694 }  
695 }  
696 }  
697 }  
698 }  
699 }  
700 }  
701 }  
702 }  
703 }  
704 }  
705 }  
706 }  
707 }  
708 }  
709 }  
710 }  
711 }  
712 }  
713 }  
714 }  
715 }  
716 }  
717 }  
718 }  
719 }  
720 }  
721 }  
722 }  
723 }  
724 }  
725 }  
726 }  
727 }  
728 }  
729 }  
730 }  
731 }  
732 }  
733 }  
734 }  
735 }  
736 }  
737 }  
738 }  
739 }  
740 }  
741 }  
742 }  
743 }  
744 }  
745 }  
746 }  
747 }  
748 }  
749 }  
750 }  
751 }  
752 }  
753 }  
754 }  
755 }  
756 }  
757 }  
758 }  
759 }  
760 }  
761 }  
762 }  
763 }  
764 }  
765 }  
766 }  
767 }  
768 }  
769 }  
770 }  
771 }  
772 }  
773 }  
774 }  
775 }  
776 }  
777 }  
778 }  
779 }  
780 }  
781 }  
782 }  
783 }  
784 }  
785 }  
786 }  
787 }  
788 }  
789 }  
790 }  
791 }  
792 }  
793 }  
794 }  
795 }  
796 }  
797 }  
798 }  
799 }  
800 }  
801 }  
802 }  
803 }  
804 }  
805 }  
806 }  
807 }  
808 }  
809 }  
810 }  
811 }  
812 }  
813 }  
814 }  
815 }  
816 }  
817 }  
818 }  
819 }  
820 }  
821 }  
822 }  
823 }  
824 }  
825 }  
826 }  
827 }  
828 }  
829 }  
830 }  
831 }  
832 }  
833 }  
834 }  
835 }  
836 }  
837 }  
838 }  
839 }  
840 }  
841 }  
842 }  
843 }  
844 }  
845 }  
846 }  
847 }  
848 }  
849 }  
850 }  
851 }  
852 }  
853 }  
854 }  
855 }  
856 }  
857 }  
858 }  
859 }  
860 }  
861 }  
862 }  
863 }  
864 }  
865 }  
866 }  
867 }  
868 }  
869 }  
870 }  
871 }  
872 }  
873 }  
874 }  
875 }  
876 }  
877 }  
878 }  
879 }  
880 }  
881 }  
882 }  
883 }  
884 }  
885 }  
886 }  
887 }  
888 }  
889 }  
890 }  
891 }  
892 }  
893 }  
894 }  
895 }  
896 }  
897 }  
898 }  
899 }  
900 }  
901 }  
902 }  
903 }  
904 }  
905 }  
906 }  
907 }  
908 }  
909 }  
910 }  
911 }  
912 }  
913 }  
914 }  
915 }  
916 }  
917 }  
918 }  
919 }  
920 }  
921 }  
922 }  
923 }  
924 }  
925 }  
926 }  
927 }  
928 }  
929 }  
930 }  
931 }  
932 }  
933 }  
934 }  
935 }  
936 }  
937 }  
938 }  
939 }  
940 }  
941 }  
942 }  
943 }  
944 }  
945 }  
946 }  
947 }  
948 }  
949 }  
950 }  
951 }  
952 }  
953 }  
954 }  
955 }  
956 }  
957 }  
958 }  
959 }  
960 }  
961 }  
962 }  
963 }  
964 }  
965 }  
966 }  
967 }  
968 }  
969 }  
970 }  
971 }  
972 }  
973 }  
974 }  
975 }  
976 }  
977 }  
978 }  
979 }  
980 }  
981 }  
982 }  
983 }  
984 }  
985 }  
986 }  
987 }  
988 }  
989 }  
990 }  
991 }  
992 }  
993 }  
994 }  
995 }  
996 }  
997 }  
998 }  
999 }  
1000 }
```

```
106 Password(newPassword);
107         } else {
108             System.out.println("New Password
109             has Generated");
110         PassGeneratorDemo.logger.info("New Password has Generated");
111     }
112     try {
113         ResultSet res = PassGeneratorDemo
114         .database.displayInfo(PassGeneratorDemo.table.
115         getSelectionKey());
116         res.next();
117         PassGeneratorDemo.database.
118         changePass(PassGeneratorDemo.table.getSelectionKey(),
119         PassGeneratorDemo.password.toString());
120         PrintStream var10000 = System.out
121     ;
122         Class var10001 = this.getClass();
123         var10000.println(var10001 + "
124             Password: " + res.getString("Password") + " Change To New
125             Password: " + PassGeneratorDemo.password.toString());
126         Logger var8 = PassGeneratorDemo.
127         logger;
128         var10001 = this.getClass();
129         var8.info(var10001 + " Password
130             : " + res.getString("Password") + " Change To New
131             Password: " + PassGeneratorDemo.password.toString());
132         PassGeneratorDemo.table.
133         rebuildPanel();
134     } catch (SQLException var7) {
135         var7.printStackTrace();
136     }
137     JOptionPane.showMessageDialog(
138         PassGeneratorDemo.table, "Password successfully changed.");
139 } else {
140     JOptionPane.showMessageDialog(
141         PassGeneratorDemo.table, "Password not changed");
142     PassGeneratorDemo.logger.info(this.
143     getClass().getName() + "Password not changed");
144     System.out.println(this.getClass().
145     getName() + "Password not changed");
146 }
147 }
```

```
134
135             PassGeneratorDemo.password = null;
136         }
137     }
138
139     public static class AltUserListener implements
140         ActionListener {
141         public AltUserListener() {
142         }
143
144         public void actionPerformed(ActionEvent event) {
145             try {
146                 if (PassGeneratorDemo.table.getSelection
147                     () != null) {
148                     String newUsername = JOptionPane.
149                     showInputDialog(PassGeneratorDemo.table, "Enter new
150                     username:");
151                     if (!newUsername.equals("")) {
152                         ResultSet res = PassGeneratorDemo.
153                         database.displayInfo(PassGeneratorDemo.table.
154                         getSelectionKey());
155                         res.next();
156                         PrintStream var10000 = System.out
157                         ;
158                         Class var10001 = this.getClass();
159                         var10000.println(var10001 + "
160                         Username: " + res.getString("Username") + " Change To New
161                         Username: " + newUsername.toString());
162                         Logger var5 = PassGeneratorDemo.
163                         logger;
164                         var10001 = this.getClass();
165                         var5.info(var10001 + " Username
166                         : " + res.getString("Username") + " Change To New
167                         Username: " + newUsername.toString());
168                         PassGeneratorDemo.database.
169                         changeUser(PassGeneratorDemo.table.getSelectionKey(),
170                         newUsername);
171                         PassGeneratorDemo.table.
172                         rebuildPanel();
173                     } else {
174                         JOptionPane.showMessageDialog(
175                             PassGeneratorDemo.table, "Warning no replacement was
176                             entered, retry after entering a replacement.");
177                     }
178                 } else {
179                     PassGeneratorDemo.table.
```

```
162 displayWarning();
163         }
164     } catch (Exception var4) {
165         System.out.println("Cancel Pressed");
166         PassGeneratorDemo.logger.info("Cancel
167             Pressed");
168         }
169     }
170 }
171
172     public static class CopyUrlListener implements
173     ActionListener {
174         public CopyUrlListener() {
175
176             public void actionPerformed(ActionEvent event) {
177                 try {
178                     ResultSet res = PassGeneratorDemo.
179                     database.displayInfo(PassGeneratorDemo.table.
180                     getSelectionKey());
181                     res.next();
182                     StringSelection stringSelection = new
183                     StringSelection(res.getString("Url"));
184                     PassGeneratorDemo.clipboard.setContents(
185                     stringSelection, (ClipboardOwner)null);
186                     PrintStream var10000 = System.out;
187                     String var10001 = this.getClass().getName
188                     ();
189                     var10000.println(var10001 + " Url: " +
190                     res.getString("Url") + " Copied Successfully");
191                     Logger var6 = PassGeneratorDemo.logger;
192                     var10001 = this.getClass().getName();
193                     var6.info(var10001 + " Url: " + res.
194                     getString("Url") + " Copied Successfully");
195                 } catch (SQLException var5) {
196                     JOptionPane.showMessageDialog((Component)
197                     null, var5.getMessage());
198                 }
199             }
200         }
201
202         public static class CopyPassListener implements
203         ActionListener {
204             public CopyPassListener() {
```

```
197      }
198
199      public void actionPerformed(ActionEvent event) {
200          try {
201              ResultSet res = PassGeneratorDemo.
202                  database.displayInfo(PassGeneratorDemo.table.
203                      getSelectionKey());
204              res.next();
205              StringSelection stringSelection = new
206                  StringSelection(res.getString("Password"));
207              PassGeneratorDemo.clipboard.setContents(
208                  stringSelection, (ClipboardOwner)null);
209              PrintStream var10000 = System.out;
210              String var10001 = this.getClass().getName
211                  ();
212              var10000.println(var10001 + " Password: "
213                  + res.getString("Password") + " Copped Successfully");
214              Logger var6 = PassGeneratorDemo.logger;
215              var10001 = this.getClass().getName();
216              var6.info(var10001 + " Password: " + res.
217                  getString("Password") + " Copped Successfully");
218          } catch (SQLException var5) {
219              JOptionPane.showMessageDialog((Component)
220                  null, var5.getMessage());
221          }
222      }
223
224      public static class CopyUserListener implements
225          ActionListener {
226          public CopyUserListener() {
227
228          }
229
229      public void actionPerformed(ActionEvent event) {
230          try {
231              ResultSet res = PassGeneratorDemo.
232                  database.displayInfo(PassGeneratorDemo.table.
233                      getSelectionKey());
234              res.next();
235              StringSelection stringSelection = new
236                  StringSelection(res.getString("Username"));
237              PassGeneratorDemo.clipboard.setContents(
238                  stringSelection, (ClipboardOwner)null);
239              PrintStream var10000 = System.out;
240              String var10001 = this.getClass().getName
```

```
229 ();
230             var1000.println(var1001 + " Username: "
+ res.getString("Username") + " Copped Successfully");
231             Logger var6 = PassGeneratorDemo.logger;
232             var1001 = this.getClass().getName();
233             var6.info(var1001 + " Username: " + res.
getString("Username") + " Copped Successfully");
234         } catch (SQLException var5) {
235             JOptionPane.showMessageDialog((Component)
null, var5.getMessage());
236         }
237     }
238 }
239 }
240
241     public static class SeePassListener implements
ActionListener {
242         public SeePassListener() {
243     }
244
245         public void actionPerformed(ActionEvent event) {
246             try {
247                 if (PassGeneratorDemo.table.getSelection
() == null) {
248                     PassGeneratorDemo.table.
displayWarning();
249                 } else {
250                     ResultSet res = PassGeneratorDemo.
database.displayInfo(PassGeneratorDemo.table.
getSelectionKey());
251                     res.next();
252                     PassGeneratorDemo.table.displayCopy(
res.getString("Password"));
253                 }
254             } catch (SQLException var4) {
255                 JOptionPane.showMessageDialog((Component)
null, var4.getMessage());
256             }
257
258     }
259 }
260
261     public static class DelPassListener implements
ActionListener {
262         public DelPassListener() {
263     }
```

```
264
265     public void actionPerformed(ActionEvent event) {
266         if (PassGeneratorDemo.table.getSelection()
267             () == null) {
268             PassGeneratorDemo.table.displayWarning();
269         } else {
270             try {
271                 String confirmation = JOptionPane.
272                     showInputDialog(PassGeneratorDemo.table, "Confirm
273                     deletion by typing " + PassGeneratorDemo.table.
274                     getSelection() + ":");
275                 if (confirmation.equalsIgnoreCase(
276                     PassGeneratorDemo.table.getSelection())) {
277                     ResultSet res = PassGeneratorDemo
278                         .database.displayInfo(PassGeneratorDemo.table.
279                         getSelectionKey());
280                     res.next();
281                     PrintStream var10000 = System.out
282                         ;
283                     Class var10001 = this.getClass();
284                     var10000.println(var10001 + "
285                     Account: " + res.getString("Username") + " Deleted
286                     Successfully");
287                     Logger var5 = PassGeneratorDemo.
288                         logger;
289                     var10001 = this.getClass();
290                     var5.info(var10001 + " Account: "
291                         + res.getString("Username") + " Deleted Successfully");
292                     PassGeneratorDemo.database.
293                     deleteUser(PassGeneratorDemo.table.getSelectionKey());
294                     PassGeneratorDemo.table.
295                     setSelection();
296                     PassGeneratorDemo.table.
297                     rebuildPanel();
298                     } else {
299                         JOptionPane.showMessageDialog(
300                             PassGeneratorDemo.table, "Incorrect account name");
301                     }
302                     } catch (Exception var4) {
303                         System.out.println("Cancel Pressed");
304                         PassGeneratorDemo.logger.info("Cancel
305                         Pressed");
306                     }
307                     }
308                     }
309                     }
```

```
293     }
294
295     public static class AddPassListener implements
296         ActionListener {
297             public AddPassListener() {
298             }
299
300             public void actionPerformed(ActionEvent event) {
301                 try {
302                     String accountName;
303                     for(accountName = JOptionPane.
304                         showInputDialog(PassGeneratorDemo.table, "Enter Account
305                         Name: "); accountName.contains("") || accountName.equals(
306                         ""); accountName = JOptionPane.showInputDialog(
307                         PassGeneratorDemo.table, "Your account name empty, please
308                         re-enter account name: "));
309                     }
310
311                     String usernameName;
312                     for(usernameName = JOptionPane.
313                         showInputDialog(PassGeneratorDemo.table, "Enter Account
314                         Username: "); usernameName.contains("") || usernameName.equals(
315                         ""); usernameName = JOptionPane.showInputDialog(
316                         PassGeneratorDemo.table, "Your username name empty,
317                         please re-enter username name: "));
318
319                     String urlName = JOptionPane.
320                     showInputDialog(PassGeneratorDemo.table, "Enter Website
321                         Name: ");
322
323                     PassGeneratorDemo.password = new Password
324                         ();
325
326                     PassGeneratorDemo.password.
327                         generatePassword();
328
329                     PassGeneratorDemo.key = new Password();
330
331                     PassGeneratorDemo.key.generatePassword();
332
333                     PassGeneratorDemo.database.addUser(
334                         accountName, usernameName, urlName, PassGeneratorDemo.
335                         password.toString(), PassGeneratorDemo.key.toString());
336
337                     PrintStream var10000 = System.out;
338
339                     String var10001 = this.getClass().getName
340                         ();
341
342                     var10000.println(var10001 + " Account: "
343                         + accountName + " Added Successfully");
344
345                     Logger var6 = PassGeneratorDemo.logger;
346
347                     var10001 = this.getClass().getName();
```

```
320             var6.info(var10001 + " Account: " +
  accountName + " Added Successfully");
321             PassGeneratorDemo.table.rebuildPanel();
322         } catch (Exception var5) {
323             System.out.println("Cancel Pressed");
324             PassGeneratorDemo.logger.info("Cancel
  Pressed");
325         }
326
327         PassGeneratorDemo.password = null;
328     }
329 }
330 }
331
```

```
1 ## Root Location Option !!
2 #log4j.rootCategory=debug,console
3 #
4 ## Package Based Logging
5 #Log4j.Logger.com.jcg.Log4j.console.appendер=debug,console
6 #Log4j.additivity.com.jcg.Log4j.console.appendер=false
7 #
8 ## Redirect Log Messages To Console !!
9 #log4j.appendер.console=org.apache.log4j.ConsoleAppender
10 #log4j.appendер.console.target=System.out
11 #log4j.appendер.console.immediateFlush=true
12 #log4j.appendер.console.encoding=UTF-8
13 #log4j.appendер.console.Layout=org.apache.log4j.
    PatternLayout
14 #log4j.appendер.console.Layout.conversionPattern=%d{yyyy-
    MM-dd HH:mm:ss} %-5p %c{1}:%L - %m%n
15 #public static Logger logger2=Logger.getLogger(Class.class
    .getName());
16
```

```
1 package com.RomTal.java;
2
3 /**
4  * GUI Manager Interface
5  */
6 public interface IView {
7     /**
8      * Update table of data
9      */
10    public void rebuildPanel();
11
12    /**
13     * get name of account
14     * @return selected key of account
15     */
16    public String getSelection();
17
18    /**
19     * get key of account
20     * @return selected name of account
21     */
22    public String getSelectionKey();
23
24    /**
25     * set selection to null after deleting account
26     */
27    public void setSelection();
28
29    /**
30     * view/copy parameter of account
31     * @param password password of an account
32     */
33    public void displayCopy(String password);
34
35    /**
36     * popup warning if account not been selected
37     */
38    public void displayWarning();
39 }
```

```
1 package com.RomTal.java;
2
3 /**
4  * Password Generator Manger Interface
5  */
6 public interface IModel {
7     /**
8      * Generate a new Password
9      */
10    public void generatePassword();
11
12    /**
13     * Generate a new password from a specific characters
14     * @param startingChar Character dictionary
15     */
16    public void generatePassword(char startingChar);
17
18    /**
19     * Generate a new password in Length that specific
Length(minimum 16 characters)
20     * @param length Password Length (minimum 16
characters)
21     */
22    public void generatePassword(int length);
23
24    /**
25     * checking if password equal to string
26     * @param compareString password to compare
27     * @return
28     */
29    public boolean equals(String compareString);
30
31    /**
32     * checking if password include symbols
33     * @return
34     */
35    public boolean hasSymbols();
36
37    /**
38     * convert to string
39     * @return
40     */
41    public String toString();
42 }
43
```

```

1 package com.RomTal.java;
2 import javax.swing.*;
3 import java.sql.*;
4
5 import static com.RomTal.java.PassGeneratorDemo.logger;
6
7 /**
8 *Responsible Manage database
9 */
10 public class DerbyDB{
11     private Connection conn;
12     public static int flag=0;
13     private boolean hasData = false;
14     public static String driver = "org.apache.derby.jdbc.
ClientDriver";
15     public static String protocol = "jdbc:derby://
localhost:1527/gagamoDB;create=true";
16     private void getConnection() {
17
18         try
19         {
20             Class.forName(driver); //Loads class with this
name
21             conn = DriverManager.getConnection(protocol);
22             System.out.println("Established Connection to
DB");
23             initialize();
24         }
25         catch(ClassNotFoundException | SQLException e)
26         {
27             JOptionPane.showMessageDialog(null, e.
getMessage());
28         }
29     }
30     private void initialize() {
31         if (!hasData) {
32             hasData = true;
33             try
34             {
35                 Statement state = conn.createStatement();
36                 //state.execute("DROP TABLE data");//clean
data if needed
37                 Statement statement = conn.
createStatement();
38                 if(flag==0) {//check if its first
Login

```

```

39                     System.out.println("Checking
40                         database for table");
41                         DatabaseMetaData databaseMetadata
42                         = conn.getMetaData();
43                         ResultSet resultSet =
44                         databaseMetadata.getTables(null, null, "DATA", null); ////
45                         checking if table data exist
46                         if (resultSet.next()) {
47                             System.out.println("TABLE
48                             ALREADY EXISTS");
49                         } else {
50                             state.execute("create table
51                             data(account varchar(60) ,username varchar(60) ,url
52                             varchar(120) ,password varchar(60),KeyVal varchar(60))");
53                             }
54                             flag++;
55                         }
56                         ResultSet res = state.executeQuery("SELECT
57                         account FROM data");
58                         if(!res.next()) {
59                             System.out.println("Building data
60                             table.");
61                             logger.info(this.getClass().getName
62                             () + " Building data table");
63                             }
64                         }
65                         catch (SQLException e)
66                         {
67                             JOptionPane.showMessageDialog(null, e.
68                             getMessage());
69                         }
70                         }
71                         */
72                         /**
73                         * Close connection to database
74                         * @throws SQLException
75                         */
76                         public void closeConnection() throws SQLException {
77                             if (conn == null) {
78                                 getConnection();
79                             }
80                             conn.close();
81                             conn=null;
82                             System.out.println("Layout panel successfully
83                             close connection.");

```

```
73     }
74
75     /**
76      * get a reference to all account saved in database
77      * @return reference to all account saved in database
78      * @throws SQLException if connection fail
79     */
80     public ResultSet displayUsers() throws SQLException {
81         if (conn == null) {
82             getConnection();
83         }
84         Statement state = conn.createStatement();
85         ResultSet res = state.executeQuery("SELECT
86             account, username, url, password, KeyVal FROM data");
87         return res;
88     }
89
90     /**
91      * get a reference to account match key in database
92      * @param KeyVal key of account
93      * @return
94      * @throws SQLException
95     */
96     public ResultSet displayInfo(String KeyVal) throws
97         SQLException {
98         if (conn == null) {
99             getConnection();
100        }
101        Statement state = conn.createStatement();
102        ResultSet res = state.executeQuery("SELECT
103            account, username, url, password FROM data WHERE KeyVal
104            = '" + KeyVal + "'");
105        return res;
106    }
107
108    /**
109     * @return get amount of account saved in database
110     * @throws SQLException
111     */
112     public int displayAmount() throws SQLException {
113         if (conn == null) {
114             getConnection();
115         }
116         int count = 0;
117         PreparedStatement state = conn.prepareStatement("
```

```

114 SELECT * FROM data");
115     ResultSet res = state.executeQuery();
116     while (res.next()) {
117         count++;
118     }
119     return count;
120 }
121
122 /***
123     * Add a user to database
124     * @param account name of account
125     * @param username name of username
126     * @param url value of url
127     * @param password url of password
128     * @param KeyVal url of KeyVal
129 */
130     public void addUser(String account, String username,
131     String url, String password, String KeyVal) {
132         if (conn == null) {
133             getConnection();
134         }
135         try
136         {
137             PreparedStatement prep = conn.
138             prepareStatement("INSERT INTO data VALUES(?,?,?,?,?,?)");
139             prep.setString(1, account);
140             prep.setString(2, username);
141             prep.setString(3, url);
142             prep.setString(4, password);
143             prep.setString(5, KeyVal);
144             prep.execute();
145         }
146         catch (SQLException e)
147         {
148             JOptionPane.showMessageDialog(null, e.
149             getMessage());
150         }
151     }
152 /***
153     * Change username of account match key in database
154     * @param KeyVal key of account
155     * @param newUsername New username to change for
156     * /
157     public void changeUser(String KeyVal, String
158     newUsername) {

```

```

156         if (conn == null) {
157             getConnection();
158         }
159         try {
160             {
161                 PreparedStatement prep = conn.
162                     prepareStatement("UPDATE data SET username = '" +
163                         newUsername + "' WHERE KeyVal = '" + KeyVal + "'");
164                     prep.execute();
165             }
166             catch (SQLException e)
167             {
168                 JOptionPane.showMessageDialog(null, e.
169                     getMessage());
170             }
171         /**
172          * Change password of account match key in database
173          * param KeyVal key of account
174          * param newPassword New password to change for
175          */
176         public void changePass(String KeyVal, String
177             newPassword) {
178             if (conn == null) {
179                 getConnection();
180             }
181             try {
182                 PreparedStatement prep = conn.
183                     prepareStatement("UPDATE data SET password = '" +
184                         newPassword + "' WHERE KeyVal = '" + KeyVal + "'");
185                     prep.execute();
186             }
187             catch (SQLException e)
188             {
189                 JOptionPane.showMessageDialog(null, e.
190                     getMessage());
191             }
192         /**
193          * Change url of account match key in database
194          * param KeyVal key of account
195          * param newUrl New url to change for
196          */
197         public void changeSite(String KeyVal, String newUrl

```

```
194 ) {  
195     if (conn == null) {  
196         getConnection();  
197     }  
198     try  
199     {  
200         PreparedStatement prep = conn.  
prepareStatement("UPDATE data SET url = '" + newUrl + "'  
WHERE KeyVal = '" + KeyVal + "'");  
201         prep.execute();  
202     }  
203     catch (SQLException e)  
204     {  
205         JOptionPane.showMessageDialog(null, e.  
getMessage());  
206     }  
207 }  
208  
209 /**  
210 * deleting password of account match key in database  
211 * param KeyVal key of account  
212 */  
213 public void deleteUser(String KeyVal) {  
214     if (conn == null) {  
215         getConnection();  
216     }  
217     try  
218     {  
219         PreparedStatement prep = conn.  
prepareStatement("DELETE FROM data WHERE KeyVal = '" +  
KeyVal + "'");  
220         prep.execute();  
221     }  
222     catch (SQLException e)  
223     {  
224         JOptionPane.showMessageDialog(null, e.  
getMessage());  
225     }  
226 }  
227 }  
228 }  
229 }
```

```

1 package com.RomTal.java;
2
3 import java.security.SecureRandom;
4
5 /**
6  * Password Generator Manger
7 */
8 public class Password implements IModel {
9 // passwords Generator at Least 10 characters Long.
10    private final byte PASSWORD_LENGTH = 16;
11    //This string holds all the possible values that a
12    generated password might contain.
13    private final String CHARACTER_DICTIONARY = "
14        abcdefghijklmnopqrstuvwxyz"
15                                +
16        ABCDEFGHIJKLMNOPQRSTUVWXYZ"
17                                +
18        1234567890"
19                                +
20        "!?@#$%";+
21    private String password;
22    private SecureRandom secureGenerator = new
23    SecureRandom();
24    //Constructor that generates a 16 character string,
25    //that will be used to create a password.zB7yI6@quL5SS6pd
26    public Password() {
27        password = "";
28    }
29    public Password(String existingPass) {
30        password = existingPass;
31    }
32    public Password(Password object2) {
33        password = object2.password;
34    }
35    @Override
36    public void generatePassword() {
37        String[] arrString = new String[PASSWORD_LENGTH];
38        //array that will be used for the generation of the
39        password.
40        while (!(this.hasSymbols())) { //Loop that checks
41            for (int i = 0; i < PASSWORD_LENGTH; i++) {
42                arrString[i] = Character.toString(
43                    CHARACTER_DICTIONARY.charAt(secureGenerator.nextInt(
44                        CHARACTER_DICTIONARY.length())));
45            }
46            password = String.join("", arrString);

```

```
36         }
37     }
38     @Override
39     public void generatePassword(char startingChar) {
40         String[] arrString = new String[PASSWORD_LENGTH];
//array that will be used for the generation of the password.
41         while (!(this.hasSymbols())) { //Loop that checks
for the password containing a symbol.
42             arrString[0] = Character.toString(startingChar
43         );
44             for (int i = 1; i < PASSWORD_LENGTH; i++) {
45                 arrString[i] = Character.toString(
46                     CHARACTER_DICTIONARY.charAt(secureGenerator.nextInt(
47                         CHARACTER_DICTIONARY.length())));
48             }
49         }
50         @Override
51         public void generatePassword(int length) {
52             String[] arrString = new String[length]; //array
that will be used for the generation of the password.
53             while (!(this.hasSymbols())) { //Loop that checks
for the password containing a symbol.
54                 for (int i = 0; i < PASSWORD_LENGTH; i++) {
55                     arrString[i] = Character.toString(
56                         CHARACTER_DICTIONARY.charAt(secureGenerator.nextInt(
57                             CHARACTER_DICTIONARY.length())));
58                 }
59             }
60             @Override
61             public boolean equals(String compareString) {
62                 return password == compareString;
63             }
64             @Override
65             public boolean hasSymbols() {
66                 return (password.contains("?") || password.
67                     contains("!")) || password.contains("@") || password.
68                     contains("#") || password.contains("$") || password.
69                     contains("%"));
70             }
71             @Override
72             public String toString() {
```

```
69         return password;  
70     }  
71 }  
72
```

```
1 package com.RomTal.java;
2
3 /**
4  * Main Application Interface
5  */
6 public interface IViewModel {
7     public static void main(String[] args) throws
8         Exception{ new PassGeneratorDemo();}
9 }
```

```

1 package com.RomTal.java;
2
3 import javax.swing.*;
4 import javax.swing.event.ListSelectionEvent;
5 import javax.swing.event.ListSelectionListener;
6 import javax.swing.table.AbstractTableModel;
7 import java.awt.*;
8 import java.sql.ResultSet;
9 import java.sql.SQLException;
10
11 /**
12  * GUI Manager
13 */
14 public class TablePanel extends JFrame implements IView {
15     private DerbyDB database = new DerbyDB();
16     private String[] columnNames = {"Account Name", "Username", "URL", "Password", "Key"};
17     private String[][] data;
18     private int amountAccounts;
19     private static String selectedAccount, selectedKey;
20     private JTable table;
21     private JPanel buttonPanel; // To hold a button
22     private JPanel mainPanel;
23
24     public TablePanel() {
25         setTitle("PassGenerator");
26         setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
27         setLayout(new BorderLayout());
28         buildButtonPanel();           // Build the panels.
29         buildPanel();
30         add(buttonPanel, BorderLayout.WEST);
31         add(mainPanel, BorderLayout.CENTER);
32         pack();
33         setSize(800,600);
34         setVisible(true);
35     }
36     private void buildPanel() {
37         builddataArray();
38         mainPanel = new JPanel();
39         table = new JTable(new TableModel());
40         mainPanel.add(table);
41         table.setSelectionMode(ListSelectionModel.
42             SINGLE_SELECTION);
42         ListSelectionModel listSelectionModel = table.
43             getSelectionModel();
43         listSelectionModel.addListSelectionListener(new

```

```

43 TableListener()));
44     JScrollPane scrollPane = new JScrollPane(table);
45     table.getColumnModel().getColumn(4).setMinWidth(0
46 );
47     table.getColumnModel().getColumn(4).setMaxWidth(0
48 );
49     table.getColumnModel().getColumn(4).setWidth(0);
50     mainPanel.add(scrollPane);
51     mainPanel.setLayout(new GridLayout(1,0,0,0));
52     mainPanel.setBorder(BorderFactory.
53         createTitledBorder("Data"));
54     }
55     @Override
56     public void rebuildPanel(){
57         remove(mainPanel);
58         buildPanel();
59         this.add(mainPanel, BorderLayout.CENTER);
60         revalidate();
61         repaint();
62         System.gc();
63     }
64     private void builddataArray() {
65         try {
66             amountAccounts = database.displayAmount();
67         } catch (SQLException e) {
68             JOptionPane.showMessageDialog(null, e.getMessage
());
69         }
70         data = new String[amountAccounts][5];
71         ResultSet res;
72         int row = 0;
73         try {
74             res = database.displayUsers();
75             while(res.next()) {
76                 data[row][0] = res.getString("Account");
77                 data[row][1] = res.getString("Username");
78                 data[row][2] = res.getString("URL");
79                 data[row][3] = "*****";
80                 //data[row][3] = res.getString("Password");
81                 data[row][4] = res.getString("KeyVal");
82                 row++;
83             }
84             res.close();
85         } catch (SQLException e) {
86             JOptionPane.showMessageDialog(null, e.getMessage
)
87         }
88     }
89 
```

```
84 () );
85     }
86 }
87     private void buildButtonPanel() {
88         buttonPanel = new JPanel();
89         buttonPanel.setLayout(new GridLayout(6,1,0,10));
90         //finalize buttons
91         JButton addPass = new JButton("Add Account");
92         JButton delPass = new JButton("Delete Account");
93         JButton seePass = new JButton("Copy/View
94             Credentials");
95         JButton altUser = new JButton("Change Username");
96         JButton altPass = new JButton("Change Password");
97         JButton altSite = new JButton("Change Website");
98         addPass.addActionListener(new PassGeneratorDemo.
99             AddPassListener());
100        delPass.addActionListener(new PassGeneratorDemo.
101            DelPassListener());
102        seePass.addActionListener(new PassGeneratorDemo.
103            SeePassListener());
104        altUser.addActionListener(new PassGeneratorDemo.
105            AltUserListener());
106        altPass.addActionListener(new PassGeneratorDemo.
107            AltPassListener());
108        altSite.addActionListener(new PassGeneratorDemo.
109            AltSiteListener());
110        //set border around buttons
111        buttonPanel.setBorder(BorderFactory.
112            createTitledBorder("Operation "));
113        //adds buttons to the panel
114        buttonPanel.add(addPass);
115        buttonPanel.add(delPass);
116        buttonPanel.add(seePass);
117        buttonPanel.add(altUser);
118        buttonPanel.add(altPass);
119        buttonPanel.add(altSite);
120    }
121    @Override
122    public String getSelection() {
123        return selectedAccount;
124    }
125    @Override
126    public String getSelectionKey() {
127        return selectedKey;
128    }
129    public void setSelection(){
```

```
122         selectedAccount=null;
123     }
124     @Override
125     public void displayCopy(String password){
126         JPanel display = new JPanel();
127         JLabel passText;
128         JButton copyUsername = new JButton("Copy Username");
129         JButton copyPassword = new JButton("Copy Password");
130         JButton copyUrl = new JButton("Copy Url");
131         JButton toWebsite = new JButton("Go to Site");
132         copyPassword.addActionListener(new
133             PassGeneratorDemo.CopyPassListener());
134         copyUsername.addActionListener(new
135             PassGeneratorDemo.CopyUserListener());
136         copyUrl.addActionListener(new PassGeneratorDemo.
137             CopyUrlListener());
138         passText = new JLabel("Password:");
139         display.add(passText);
140         passText = new JLabel(password);
141         display.add(passText);
142         display.add(copyUsername);
143         display.add(copyPassword);
144         display.add(copyUrl);
145         JOptionPane.showMessageDialog(this, display);
146     };
147     @Override
148     public void displayWarning(){
149         JOptionPane.showMessageDialog(this, "Warning no
150 account was selected, retry after selecting an account");
151     };
152     private class TableModel extends AbstractTableModel {
153         public int getColumnCount() {
154             return columnNames.length;
155         }
156         public int getRowCount() {
157             return data.length;
158         }
159         public String getColumnName(int col) {
160             return columnNames[col];
161         }
162         public String getValueAt(int rowNum, int colNum
163         ) {
164             return data[rowNum][colNum];
165         }
166     }
167 }
```

```
161      }
162      private class TableListener implements
163          ListSelectionListener {
164              public void valueChanged(ListSelectionEvent event
165              ) {
166                  selectedAccount = (String) table.getValueAt(
167                      table.getSelectedRow(), 0);
168                  selectedKey = (String) table.getValueAt(table
169                      .getSelectedRow(), 4);
170              }
171      }
172  }
```

```
1 package com.RomTal.java;
2
3 import org.apache.log4j.FileAppender;
4 import org.apache.log4j.Logger;
5 import org.apache.log4j.PatternLayout;
6 import javax.swing.*;
7 import java.awt.*;
8 import java.awt.datatransfer.Clipboard;
9 import java.awt.datatransfer.StringSelection;
10 import java.awt.event.ActionEvent;
11 import java.awt.event.ActionListener;
12 import java.io.IOException;
13 import java.io.PrintWriter;
14 import java.sql.ResultSet;
15 import java.sql.SQLException;
16
17 /**
18 * Main Application
19 */
20 public class PassGeneratorDemo implements IViewModel{
21     public static Logger logger;
22     static {
23         String file = "logs.txt";
24         logger = Logger.getLogger(Class.class.getName());
25         try {
26             PrintWriter writer = new PrintWriter(file);
27             writer.print("");
28             writer.close();
29             logger.addAppender(new FileAppender(new
30             PatternLayout("%-5p [%t]: %m%n"), file));
31         } catch (IOException e) {
32             e.printStackTrace();
33         }
34     }
35     private static DerbyDB database = new DerbyDB();
36     private static TablePanel table= new TablePanel();
37     private static Clipboard clipboard = Toolkit.
38         getDefaultToolkit().getSystemClipboard();
39     private static Password password;
40     private static Password key;
41     /**
42      * Add new account (implements Listener in Add account
43      button)
44     */
45     public static class AddPassListener implements
```

```

43 ActionListener {
44     public void actionPerformed(ActionEvent event) {
45         try {
46             String accountName = JOptionPane.
47             showInputDialog(table, "Enter Account Name: ");
48             while (accountName.contains("") || 
49             accountName.equals("")) {
50                 accountName = JOptionPane.
51                 showInputDialog(table, "Your account name empty, please re-
52                 -enter account name: ");
53             }
54             String usernameName = JOptionPane.
55             showInputDialog(table, "Enter Account Username: ");
56             while (usernameName.contains("") || 
57             usernameName.equals("")) {
58                 usernameName = JOptionPane.
59                 showInputDialog(table, "Your username name empty, please re-enter 
56                 username name: ");
57             }
58             String urlName = JOptionPane.
59             showInputDialog(table, "Enter Website Name: ");
60             password = new Password();
61             password.generatePassword();
62             key = new Password();
63             key.generatePassword();
64             database.addUser(accountName, usernameName
65             , urlName, password.toString(), key.toString());
66             System.out.println(this.getClass().getName
67             () + " Account: " + accountName + " Added Successfully");
68             logger.info(this.getClass().getName() + "
69             Account: " + accountName + " Added Successfully");
70             table.rebuildPanel();
71         } catch (Exception e) {
72             System.out.println("Cancel Pressed");
73             logger.info("Cancel Pressed");
74         }
75         password = null;
76     }
77 }
78 /**
79 * deleting account (implements Listener in delete
80 account button)
81 */
82 public static class DelPassListener implements
83 ActionListener {

```

```

75      public void actionPerformed(ActionEvent event) {
76          ResultSet res;
77          if (table.getSelection() == null) {
78              table.displayWarning();
79          } else {
80              try {
81                  String confirmation = JOptionPane.
82                      showInputDialog(table, "Confirm deletion by typing " +
83                      table.getSelection() + ":");
84                  if (confirmation.equalsIgnoreCase(
85                      table.getSelection())) {
86                      res = database.displayInfo(table.
87                          getSelectionKey());
88                      res.next();
89                      System.out.println(this.getClass
89                         () + " Account: " + res.getString("Username") + " Deleted
90                         Successfully");
91                      logger.info(this.getClass() + " "
92                         Account: " + res.getString("Username") + " Deleted
93                         Successfully");
94                      database.deleteUser(table.
95                          getSelectionKey());
96                      table.setSelection();
97                      table.rebuildPanel();
98                  } else {
99                      JOptionPane.showMessageDialog(
100                         table, "Incorrect account name");
101                  }
102              } catch (Exception e) {
103                  System.out.println("Cancel Pressed");
104                  logger.info("Cancel Pressed");
105              }
106          }
107      /**
108       * view/copy account credentials (implements listener
109       * in copy/view credentials button)
110      */
111      public static class SeePassListener implements
112          ActionListener {
113          public void actionPerformed(ActionEvent event) {
114              ResultSet res;
115              try {
116                  if (table.getSelection() == null) {
117                      table.displayWarning();
118                  }
119              }
120          }
121      }
122  }

```

```

109             } else {
110                 res = database.displayInfo(table.
111                     getSelectionKey());
112                     res.next();
113                     table.displayCopy(res.getString("
114                         Password"));
115                     }
116                     } catch (SQLException e) {
117                         JOptionPane.showMessageDialog(null, e.
118                             getMessage());
119                     }
120                     /**
121                      * Copy username of selected account (implements
122                         listener in copy username button)
123                         */
124                     public static class CopyUserListener implements
125                         ActionListener {
126                             public void actionPerformed(ActionEvent event) {
127                                 ResultSet res;
128                                 StringSelection stringSelection;
129                                 try {
130                                     res = database.displayInfo(table.
131                                         getSelectionKey());
132                                     res.next();
133                                     stringSelection = new StringSelection(res
134                                         .getString("Username"));
135                                     clipboard.setContents(stringSelection,
136                                         null);
137                                     System.out.println(this.getClass().
138                                         getName() + " Username: " + res.getString("Username") +
139                                         " Copped Successfully");
140                                     logger.info(this.getClass().getName() +
141                                         " Username: " + res.getString("Username") + " Copped
Successfullly");
142                                     } catch (SQLException e) {
143                                         JOptionPane.showMessageDialog(null, e.
144                                         getMessage());
145                                     }
146                                 }
147                                 */
148                                 /**
149                      * Copy password of selected account (implements
150                         listener in password username button)
151                         */

```

```

141     public static class CopyPassListener implements
142         ActionListener {
143             public void actionPerformed(ActionEvent event) {
144                 ResultSet res;
145                 StringSelection stringSelection;
146                 try {
147                     res = database.displayInfo(table.
148                         getSelectionKey());
149                     res.next();
150                     stringSelection = new StringSelection(res
151                         .getString("Password"));
152                     clipboard.setContents(stringSelection,
153                         null);
154                     System.out.println(this.getClass().
155                         getName() + " Password: " + res.getString("Password") +
156                         " Copped Successfully");
157                     logger.info(this.getClass().getName() +
158                         " Password: " + res.getString("Password") + " Copped
159                         Successfully");
160                 } catch (SQLException e) {
161                     JOptionPane.showMessageDialog(null, e.
162                         getMessage());
163                 }
164             }
165             /**
166             * Copy url of selected account (implements Listener
167             * in url username button)
168             */
169             public static class CopyUrlListener implements
170                 ActionListener {
171                 public void actionPerformed(ActionEvent event) {
172                     ResultSet res;
173                     StringSelection stringSelection;
174                     try {
175                         res = database.displayInfo(table.
176                             getSelectionKey());
177                         res.next();
178                         stringSelection = new StringSelection(res
179                             .getString("Url"));
180                         clipboard.setContents(stringSelection,
181                             null);
182                         System.out.println(this.getClass().
183                             getName() + " Url: " + res.getString("Url") + " Copped
184                             Successfully");
185                     logger.info(this.getClass().getName() +

```

```

170 " Url: " + res.getString("Url") + " Copped Successfully"
    );
171     } catch (SQLException e) {
172         JOptionPane.showMessageDialog(null, e.
173             getMessage());
174     }
175 }
176 /**
177     * Change username of selected account to new
178     * username (implements Listener in copy username button)
179 */
180 public static class AltUserListener implements
181     ActionListener {
182     public void actionPerformed(ActionEvent event) {
183         ResultSet res;
184         try {
185             if (table.getSelection() != null) {
186                 String newUsername = JOptionPane.
187                     showInputDialog(table, "Enter new username:");
188                 if (!newUsername.equals("")) {
189                     res = database.displayInfo(table.
190                         getSelectionKey());
191                     res.next();
192                     System.out.println(this.getClass
193                         () + " Username: " + res.getString("Username") + " Change
194                         To New Username: " + newUsername.toString());
195                     logger.info(this.getClass() + "
196                         Username: " + res.getString("Username") + " Change To New
197                         Username: " + newUsername.toString());
198                     database.changeUser(table.
199                         getSelectionKey(), newUsername);
200                     table.rebuildPanel();
201                 } else {
202                     JOptionPane.showMessageDialog(
203                         table, "Warning no replacement was entered, retry after
204                         entering a replacement.");
205                 }
206             } else {
207                 table.displayWarning();
208             }
209         } catch (Exception e) {
210             System.out.println("Cancel Pressed");
211             logger.info("Cancel Pressed");
212         }
213     }

```

```
203     }
204     /**
205      * Change password of selected account to new
206      * username (implements Listener in copy username button)
207      */
208     public static class AltPassListener implements
209     ActionListener {
210         public void actionPerformed(ActionEvent event) {
211             ResultSet res;
212             password = new Password();
213             password.generatePassword();
214             String newPassword;
215
216             if (table.getSelection() == null) {
217                 table.displayWarning();
218             } else {
219                 int answer = JOptionPane.
220                 showConfirmDialog(table, "Are you sure you would like to
221                 regenerate this account's password?", "PassGenerator" ,
222                 JOptionPane.YES_NO_OPTION);
223                 if (answer == 0) {
224                     int answer2 = JOptionPane.
225                     showConfirmDialog(table, "Would you like to enter your
226                     own password?", "PassGenerator", JOptionPane.
227                     YES_NO_OPTION);
228                     if (answer2 == 0) {
229                         newPassword = JOptionPane.
230                         showInputDialog(table, "Enter new password:");
231                         password = new Password(
232                             newPassword);
233                     }
234                     else
235                     {
236                         System.out.println("New Password
237                             has Generated");
238                         logger.info("New Password has
239                             Generated");
240                     }
241                     try {
242                         res = database.displayInfo(table.
243                             getSelectionKey());
244                         res.next();
245                         database.changePass(table.
246                             getSelectionKey(), password.toString());
247                         System.out.println(this.getClass
248                             () + " Password: " + res.getString("Password") + " Change
249                             Password");
250                     }
251                 }
252             }
253         }
254     }
```

```

233 To New Password: "+ password.toString());
234                                     logger.info(this.getClass() + "
235                                         Password: " + res.getString("Password") + " Change To New
236                                         Password: "+ password.toString());
237                                         table.rebuildPanel();
238                                     } catch (SQLException e) {
239                                         e.printStackTrace();
240                                     }
241                                         JOptionPane.showMessageDialog(table,
242                                         "Password successfully changed.");
243                                     } else {
244                                         JOptionPane.showMessageDialog(table,
245                                         "Password not changed");
246                                         logger.info(this.getClass().getName
247                                         () + "Password not changed");
248                                         System.out.println(this.getClass().
249                                         getName() + "Password not changed");
250                                         }
251                                         }
252                                         password = null;
253                                     }
254                                     }
255                                     /**
256                                     * Change url of selected account to new username (
257                                     implements listener in copy username button)
258                                     */
259                                     public static class AltSiteListener implements
260                                     ActionListener {
261                                         public void actionPerformed(ActionEvent event) {
262                                             ResultSet res;
263                                             try {
264                                                 if (table.getSelection() != null) {
265                                                     String newSite = JOptionPane.
266                                                     showInputDialog(table, "Enter the URL of the new website
267 .");
268                                                     if (!newSite.equals("")) {
269                                                         res = database.displayInfo(table.
270                                                         getSelectionKey());
271                                                         res.next();
272                                                         database.changeSite(table.
273                                                         getSelectionKey(), newSite);
274                                                         System.out.println(this.getClass
275                                         () + " URL: " + res.getString("URL") + " Change To New
276                                         Username: " + newSite.toString());
277                                         logger.info(this.getClass() + "
278                                         URL: " + res.getString("URL") + " Change To New Username

```

```
263 : " + newSite.toString());
264                     table.rebuildPanel();
265                 } else {
266                     JOptionPane.showMessageDialog(
267                         table, "Warning no replacement was entered, retry after
268                         entering a replacement.");
269                     }
270                 }
271             } catch (Exception e) {
272                 System.out.println("Cancel Pressed");
273                 logger.info("Cancel Pressed");
274             }
275         }
276     }
277 }
278 /**
279 * Start application
280 * @param args
281 * @throws Exception
282 */
283 public static void main(String[] args) throws
284     Exception{ new PassGeneratorDemo();}
285 }
286
```

```
1 package com.RomTal.java;
2
3 import org.junit.jupiter.api.AfterEach;
4 import org.junit.jupiter.api.BeforeEach;
5 import org.junit.jupiter.api.Test;
6
7 import java.sql.ResultSet;
8 import java.sql.SQLException;
9
10 import static org.junit.jupiter.api.Assertions.*;
11
12 class DerbyDBTest {
13
14     public DerbyDB database;
15     public String Key="KeyChecking";
16     public ResultSet res;
17
18
19     @BeforeEach
20     void setUp() {
21         database = new DerbyDB();
22     }
23
24     @AfterEach
25     void tearDown() {
26         database = null;
27     }
28
29     @Test
30     void closeConnection() throws SQLException {
31         database.closeConnection();
32     }
33
34     @Test
35     void displayUsers() throws SQLException {
36         database.displayUsers();
37     }
38
39     @Test
40     void displayInfo() throws SQLException {
41         database.displayInfo(Key);
42     }
43
44     @Test
45     void displayAmount() throws SQLException {
46         database.displayAmount();
```

```
47    }
48
49    @Test
50    void addUser() {
51        database.addUser("Account", "UserName", "URL", "
52        Password", "Key");
53        database.addUser("NewAccount", "UserName", "URL", "
54        Password", Key);
55    }
56
57    @Test
58    void changeUser() throws SQLException {
59        res = database.displayInfo(Key);
60        res.next();
61        database.changeUser(Key, "NewUserName");
62    }
63
64    @Test
65    void changePass() throws SQLException {
66        res = database.displayInfo(Key);
67        res.next();
68        database.changePass(Key, "NewPassword");
69    }
70
71    @Test
72    void changeSite() throws SQLException {
73        res = database.displayInfo(Key);
74        res.next();
75        database.changeSite(Key, "NewUrl");
76    }
77
78    @Test
79    void deleteUser() {
80        database.deleteUser(Key);
81    }
82 }
```

```
1 package com.RomTal.java;
2
3 import org.junit.jupiter.api.AfterEach;
4 import org.junit.jupiter.api.BeforeEach;
5 import org.junit.jupiter.api.Test;
6
7 import static org.junit.jupiter.api.Assertions.*;
8
9 class PasswordTest {
10
11     public static Password password;
12     int test = 20;
13     char test1='R';
14     String Pass="PassChecking";
15     @BeforeEach
16     void setUp() {
17         password = new Password();
18     }
19
20     @AfterEach
21     void tearDown() {
22         password=null;
23     }
24
25     @Test
26     void generatePassword() {
27         password.generatePassword();
28     }
29
30     @Test
31     void testGeneratePassword() {
32         password.generatePassword(test);
33     }
34
35     @Test
36     void testGeneratePassword1() {
37         password.generatePassword(test1);
38     }
39
40     @Test
41     void testEquals() {
42         password.equals(Pass);
43     }
44
45     @Test
46     void hasSymbols() {
```

```
47         password.hasSymbols();  
48     }  
49  
50     @Test  
51     void testToString() {  
52         password.toString();  
53     }  
54 }
```

```
1 package com.RomTal.java;
2
3 import org.junit.jupiter.api.AfterEach;
4 import org.junit.jupiter.api.BeforeEach;
5 import org.junit.jupiter.api.Test;
6
7 class TablePanelTest {
8
9     public TablePanel table;
10    String Pass="PassChecking";
11    @BeforeEach
12    void setUp() {
13        table = new TablePanel();
14    }
15
16    @AfterEach
17    void tearDown() {
18        table= null;
19    }
20
21    @Test
22    void testRebuildPanel() {
23        table.rebuildPanel();
24    }
25
26    @Test
27    void testGetSelection() {
28        table.getSelection();
29    }
30
31    @Test
32    void testGetSelectionKey() {
33        table.getSelectionKey();
34    }
35
36    @Test
37    void testSetSelection() {
38        table.setSelection();
39    }
40
41    @Test
42    void testDisplayCopy() {
43        table.displayCopy(Pass);
44    }
45 }
```

```
1 package com.RomTal.java;
2
3 import org.junit.jupiter.api.AfterEach;
4 import org.junit.jupiter.api.BeforeEach;
5 import org.junit.jupiter.api.Test;
6
7 import static org.junit.jupiter.api.Assertions.*;
8
9 class PassGeneratorDemoTest {
10
11     public PassGeneratorDemo Application;
12     @BeforeEach
13     void setUp() {
14         Application = new PassGeneratorDemo();
15     }
16
17     @AfterEach
18     void tearDown() {
19         Application = null;
20     }
21
22     @Test
23     void main() {
24     }
25 }
```