```
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-
   output="true">
 4     <exclude-output />
 5     <content url="file://$MODULE_DIR$">
 6       <sourceFolder url="file://$MODULE_DIR$/src"
   isTestSource="false" />
 7       <sourceFolder url="file://$MODULE_DIR$/Tests"
   isTestSource="true" />
 8     </content>
 9     <orderEntry type="inheritedJdk" />
10     <orderEntry type="sourceFolder" forTests="false" />
11     <orderEntry type="library" name="lib" level="project"
    />
12     <orderEntry type="library" name="log4j" level="project
   " />
13     <orderEntry type="library" exported="" name="junit-4.
   12" level="project" />
14     <orderEntry type="module-library" scope="TEST">
15       <library name="JUnit5.4">
16         <CLASSES>
17           <root url="jar://$MAVEN_REPOSITORY$/org/junit/
   jupiter/junit-jupiter/5.4.2/junit-jupiter-5.4.2.jar!/" />
18           <root url="jar://$MAVEN_REPOSITORY$/org/junit/
   jupiter/junit-jupiter-api/5.4.2/junit-jupiter-api-5.4.2.
   jar!/" />
19           <root url="jar://$MAVEN_REPOSITORY$/org/
   apiguardian/apiguardian-api/1.0.0/apiguardian-api-1.0.0.
   jar!/" />
20           <root url="jar://$MAVEN_REPOSITORY$/org/
   opentest4j/opentest4j/1.1.1/opentest4j-1.1.1.jar!/" />
21           <root url="jar://$MAVEN_REPOSITORY$/org/junit/
   platform/junit-platform-commons/1.4.2/junit-platform-
   commons-1.4.2.jar!/" />
22           <root url="jar://$MAVEN_REPOSITORY$/org/junit/
   jupiter/junit-jupiter-params/5.4.2/junit-jupiter-params-5.
   4.2.jar!/" />
23           <root url="jar://$MAVEN_REPOSITORY$/org/junit/
   jupiter/junit-jupiter-engine/5.4.2/junit-jupiter-engine-5.
   4.2.jar!/" />
24           <root url="jar://$MAVEN_REPOSITORY$/org/junit/
   platform/junit-platform-engine/1.4.2/junit-platform-engine
   -1.4.2.jar!/" />
25         </CLASSES>
26         <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>
```

```java
//
// Source code recreated from a .class file by IntelliJ IDEA
// (powered by Fernflower decompiler)
//

package com.RomTal.java;

import java.sql.ResultSet;
import java.sql.SQLException;
import org.junit.jupiter.api.AfterEach;
import org.junit.jupiter.api.BeforeEach;
import org.junit.jupiter.api.Test;

class DerbyDBTest {
    public DerbyDB database;
    public String Key = "KeyChecking";
    public ResultSet res;

    DerbyDBTest() {
    }

    @BeforeEach
    void setUp() {
        this.database = new DerbyDB();
    }

    @AfterEach
    void tearDown() {
        this.database = null;
    }

    @Test
    void closeConnection() throws SQLException {
        this.database.closeConnection();
    }

    @Test
    void displayUsers() throws SQLException {
        this.database.displayUsers();
    }

    @Test
    void displayInfo() throws SQLException {
        this.database.displayInfo(this.Key);
    }
```

```
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"
    , "Password", "Key");
55          this.database.addUser("NewAccount", "UserName", "
    URL", "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
```

```java
//
// Source code recreated from a .class file by IntelliJ IDEA
// (powered by Fernflower decompiler)
//

package com.RomTal.java;

import org.junit.jupiter.api.AfterEach;
import org.junit.jupiter.api.BeforeEach;
import org.junit.jupiter.api.Test;

class PasswordTest {
    public static Password password;
    int test = 20;
    char test1 = 'R';
    String Pass = "PassChecking";

    PasswordTest() {
    }

    @BeforeEach
    void setUp() {
        password = new Password();
    }

    @AfterEach
    void tearDown() {
        password = null;
    }

    @Test
    void generatePassword() {
        password.generatePassword();
    }

    @Test
    void testGeneratePassword() {
        password.generatePassword(this.test);
    }

    @Test
    void testGeneratePassword1() {
        password.generatePassword(this.test1);
    }

```

```
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
   IDEA
 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 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
   IDEA
 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.appender=debug,console
6 #log4j.additivity.com.jcg.log4j.console.appender=false
7 #
8 ## Redirect Log Messages To Console !!
9 #log4j.appender.console=org.apache.log4j.ConsoleAppender
10 #log4j.appender.console.target=System.out
11 #log4j.appender.console.immediateFlush=true
12 #log4j.appender.console.encoding=UTF-8
13 #log4j.appender.console.layout=org.apache.log4j.
   PatternLayout
14 #log4j.appender.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
   IDEA
 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
```

```java
//
// Source code recreated from a .class file by IntelliJ IDEA
// (powered by Fernflower decompiler)
//

package com.RomTal.java;

public interface IModel {
    void generatePassword();

    void generatePassword(char var1);

    void generatePassword(int var1);

    boolean equals(String var1);

    boolean hasSymbols();

    String toString();
}

```

```
 1 //
 2 // Source code recreated from a .class file by IntelliJ
   IDEA
 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(
   protocol);
32             System.out.println("Established Connection to
   DB");
33             this.initialize();
34         } catch (SQLException | ClassNotFoundException
   var2) {
35             JOptionPane.showMessageDialog((Component)null
   , var2.getMessage());
36         }
37
38     }
39
```

```
40      private void initialize() {
41          if (!this.hasData) {
42              this.hasData = true;
43
44              try {
45                  Statement state = this.conn.
   createStatement();
46                  Statement statement = this.conn.
   createStatement();
47                  if (flag == 0) {
48                      System.out.println("Checking database
   for table");
49                      DatabaseMetaData databaseMetadata =
   this.conn.getMetaData();
50                      ResultSet resultSet = databaseMetadata
   .getTables((String)null, (String)null, "DATA", (String[])
   null);
51                      if (resultSet.next()) {
52                          System.out.println("TABLE ALREADY
   EXISTS");
53                      } else {
54                          state.execute("create table data(
   account varchar(60) ,username varchar(60) ,url varchar(120
   ) ,password varchar(60),KeyVal varchar(60))");
55                      }
56
57                      ++flag;
58                  }
59
60                  ResultSet res = state.executeQuery("SELECT
    account FROM data");
61                  if (!res.next()) {
62                      System.out.println("Building data
   table.");
63                      PassGeneratorDemo.logger.info(this.
   getClass().getName() + " Building data table");
64                  }
65              } catch (SQLException var5) {
66                  JOptionPane.showMessageDialog((Component)
   null, var5.getMessage());
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,
    String url, String password, String KeyVal) {
117         if (this.conn == null) {
118             this.getConnection();
119         }
120
121         try {
122             PreparedStatement prep = this.conn.
    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
    , var7.getMessage());
131         }
132
133     }
134
135     public void changeUser(String KeyVal, String
    newUsername) {
136         if (this.conn == null) {
137             this.getConnection();
138         }
139
140         try {
141             PreparedStatement prep = this.conn.
    prepareStatement("UPDATE data SET username = '" +
    newUsername + "' WHERE KeyVal = '" + KeyVal + "'");
142             prep.execute();
143         } catch (SQLException var4) {
144             JOptionPane.showMessageDialog((Component)null
    , var4.getMessage());
145         }
146
147     }
148
149     public void changePass(String KeyVal, String
    newPassword) {
```

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

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

```
 1 //
 2 // Source code recreated from a .class file by IntelliJ
   IDEA
 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 = "
   abcdefghijklmnopqrstuvwxyzABCDEFGHIJKLMNOPQRSTUVWXYZ123456
   7890!?@#$%";
13     private String password;
14     private SecureRandom secureGenerator = new
   SecureRandom();
15
16     public Password() {
17         this.password = "";
18     }
19
20     public Password(String existingPass) {
21         this.password = existingPass;
22     }
23
24     public Password(Password object2) {
25         this.password = object2.password;
26     }
27
28     public void generatePassword() {
29         for(String[] arrString = new String[16]; !this.
   hasSymbols(); this.password = String.join("", arrString
   )) {
30             for(int i = 0; i < 16; ++i) {
31                 arrString[i] = Character.toString("
   abcdefghijklmnopqrstuvwxyzABCDEFGHIJKLMNOPQRSTUVWXYZ123456
   7890!?@#$%".charAt(this.secureGenerator.nextInt("
   abcdefghijklmnopqrstuvwxyzABCDEFGHIJKLMNOPQRSTUVWXYZ123456
   7890!?@#$%".length()));
32             }
33         }
34
35     }
36
```

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

```
67      }
68  }
69
```

```
 1 //
 2 // Source code recreated from a .class file by IntelliJ
   IDEA
 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
   IDEA
 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          this.mainPanel.add(this.table);
64          this.table.setSelectionMode(0);
65          ListSelectionModel listSelectionModel = this.table
    .getSelectionModel();
66          listSelectionModel.addListSelectionListener(new
    TablePanel.TableListener());
67          JScrollPane scrollPane = new JScrollPane(this.
    table);
68          this.table.getColumnModel().getColumn(4).
    setMinWidth(0);
69          this.table.getColumnModel().getColumn(4).
    setMaxWidth(0);
70          this.table.getColumnModel().getColumn(4).setWidth(
    0);
71          this.mainPanel.add(scrollPane);
72          this.mainPanel.setLayout(new GridLayout(1, 0, 0, 0
    ));
73          this.mainPanel.setBorder(BorderFactory.
    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.
    displayAmount();
 88          } catch (SQLException var4) {
 89              JOptionPane.showMessageDialog((Component)null
    , var4.getMessage());
 90          }
 91
 92          this.data = new String[this.amountAccounts][5];
 93          int row = 0;
 94
 95          try {
 96              ResultSet res;
 97              for(res = this.database.displayUsers(); res.
    next(); ++row) {
 98                  this.data[row][0] = res.getString("
    Account");
 99                  this.data[row][1] = res.getString("
    Username");
100                  this.data[row][2] = res.getString("URL");
101                  this.data[row][3] = "****************";
102                  this.data[row][4] = res.getString("KeyVal
    ");
103              }
104
105              res.close();
106          } catch (SQLException var5) {
107              JOptionPane.showMessageDialog((Component)null
    , var5.getMessage());
108          }
109
110      }
111
112      private void buildButtonPanel() {
113          this.buttonPanel = new JPanel();
114          this.buttonPanel.setLayout(new GridLayout(6, 1, 0
    , 10));
115          JButton addPass = new JButton("Add Account");
116          JButton delPass = new JButton("Delete Account");
117          JButton seePass = new JButton("Copy/View
    Credentials");
118          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.
    createTitledBorder("Operation "));
128         this.buttonPanel.add(addPass);
129         this.buttonPanel.add(delPass);
130         this.buttonPanel.add(seePass);
131         this.buttonPanel.add(altUser);
132         this.buttonPanel.add(altPass);
133         this.buttonPanel.add(altSite);
134     }
135
136     public String getSelection() {
137         return selectedAccount;
138     }
139
140     public String getSelectionKey() {
141         return selectedKey;
142     }
143
144     public void setSelection() {
145         selectedAccount = null;
146     }
147
148     public void displayCopy(String password) {
149         JPanel display = new JPanel();
150         JButton copyUsername = new JButton("Copy Username
    ");
151         JButton copyPassword = new JButton("Copy Password
    ");
152         JButton copyUrl = new JButton("Copy Url");
153         new JButton("Go to Site");
154         copyPassword.addActionListener(new
    CopyPassListener());
155         copyUsername.addActionListener(new
    CopyUserListener());
156         copyUrl.addActionListener(new CopyUrlListener());
157         JLabel passText = new JLabel("Password:");
158         display.add(passText);
159         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             return TablePanel.this.data[rowNum][colNum];
189         }
190     }
191 
192     private class TableListener implements
    ListSelectionListener {
193         private TableListener() {
194         }
195 
196         public void valueChanged(ListSelectionEvent event
    ) {
197             TablePanel.selectedAccount = (String)
    TablePanel.this.table.getValueAt(TablePanel.this.table.
    getSelectedRow(), 0);
198             TablePanel.selectedKey = (String)TablePanel.
    this.table.getValueAt(TablePanel.this.table.
```

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

```
 1 //
 2 // Source code recreated from a .class file by IntelliJ
   IDEA
 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
   PatternLayout("%-5p [%t]: %m%n"), file));
49          } catch (IOException var2) {
50              var2.printStackTrace();
51          }
52
53          database = new DerbyDB();
54          table = new TablePanel();
55          clipboard = Toolkit.getDefaultToolkit().
   getSystemClipboard();
56      }
57
58      public static class AltSiteListener implements
   ActionListener {
59          public AltSiteListener() {
60          }
61
62          public void actionPerformed(ActionEvent event) {
63              try {
64                  if (PassGeneratorDemo.table.getSelection
   () != null) {
65                      String newSite = JOptionPane.
   showInputDialog(PassGeneratorDemo.table, "Enter the URL of
    the new website.");
66                      if (!newSite.equals("")) {
67                          ResultSet res = PassGeneratorDemo.
   database.displayInfo(PassGeneratorDemo.table.
   getSelectionKey());
68                          res.next();
69                          PassGeneratorDemo.database.
   changeSite(PassGeneratorDemo.table.getSelectionKey(),
   newSite);
70                          PrintStream var10000 = System.out;
71                          Class var10001 = this.getClass();
72                          var10000.println(var10001 + " URL
   : " + res.getString("URL") + " Change To New Username: "
    + newSite.toString());
73                          Logger var5 = PassGeneratorDemo.
   logger;
74                          var10001 = this.getClass();
75                          var5.info(var10001 + " URL: " +
   res.getString("URL") + " Change To New Username: " +
   newSite.toString());
```

```
 76                          PassGeneratorDemo.table.
    rebuildPanel();
 77                      } else {
 78                          JOptionPane.showMessageDialog(
    PassGeneratorDemo.table, "Warning no replacement was
    entered, retry after entering a replacement.");
 79                      }
 80                  } else {
 81                      PassGeneratorDemo.table.
    displayWarning();
 82                  }
 83              } catch (Exception var4) {
 84                  System.out.println("Cancel Pressed");
 85                  PassGeneratorDemo.logger.info("Cancel
    Pressed");
 86              }
 87
 88          }
 89      }
 90
 91      public static class AltPassListener implements
    ActionListener {
 92          public AltPassListener() {
 93          }
 94
 95          public void actionPerformed(ActionEvent event) {
 96              PassGeneratorDemo.password = new Password();
 97              PassGeneratorDemo.password.generatePassword
    ();
 98              if (PassGeneratorDemo.table.getSelection
    () == null) {
 99                  PassGeneratorDemo.table.displayWarning();
100              } else {
101                  int answer = JOptionPane.
    showConfirmDialog(PassGeneratorDemo.table, "Are you sure
    you would like to regenerate this account's password?", "
    PassGenerator", 0);
102                  if (answer == 0) {
103                      int answer2 = JOptionPane.
    showConfirmDialog(PassGeneratorDemo.table, "Would you
    like to enter your own password?", "PassGenerator", 0);
104                      if (answer2 == 0) {
105                          String newPassword = JOptionPane.
    showInputDialog(PassGeneratorDemo.table, "Enter new
    password:");
106                          PassGeneratorDemo.password = new
```

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

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

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

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

```
229 ();
230                 var10000.println(var10001 + " Username: "
     + res.getString("Username") + " Copped Successfully");
231                 Logger var6 = PassGeneratorDemo.logger;
232                 var10001 = this.getClass().getName();
233                 var6.info(var10001 + " 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
    () == null) {
267                 PassGeneratorDemo.table.displayWarning();
268             } else {
269                 try {
270                     String confirmation = JOptionPane.
    showInputDialog(PassGeneratorDemo.table, "Confirm
    deletion by typing " + PassGeneratorDemo.table.
    getSelection() + ":");
271                     if (confirmation.equalsIgnoreCase(
    PassGeneratorDemo.table.getSelection())) {
272                         ResultSet res = PassGeneratorDemo
    .database.displayInfo(PassGeneratorDemo.table.
    getSelectionKey());
273                         res.next();
274                         PrintStream var10000 = System.out
    ;
275                         Class var10001 = this.getClass();
276                         var10000.println(var10001 + "
    Account: " + res.getString("Username") + " Deleted
    Successfully");
277                         Logger var5 = PassGeneratorDemo.
    logger;
278                         var10001 = this.getClass();
279                         var5.info(var10001 + " Account: "
     + res.getString("Username") + " Deleted Successfully");
280                         PassGeneratorDemo.database.
    deleteUser(PassGeneratorDemo.table.getSelectionKey());
281                         PassGeneratorDemo.table.
    setSelection();
282                         PassGeneratorDemo.table.
    rebuildPanel();
283                     } else {
284                         JOptionPane.showMessageDialog(
    PassGeneratorDemo.table, "Incorrect account name");
285                     }
286                 } catch (Exception var4) {
287                     System.out.println("Cancel Pressed");
288                     PassGeneratorDemo.logger.info("Cancel
     Pressed");
289                 }
290             }
291
292         }
```

```
293     }
294
295     public static class AddPassListener implements
    ActionListener {
296         public AddPassListener() {
297         }
298
299         public void actionPerformed(ActionEvent event) {
300             try {
301                 String accountName;
302                 for(accountName = JOptionPane.
    showInputDialog(PassGeneratorDemo.table, "Enter Account
    Name: "); accountName.contains("'") || accountName.equals
    (""); accountName = JOptionPane.showInputDialog(
    PassGeneratorDemo.table, "Your account name empty, please
     re-enter account name: ")) {
303                 }
304
305                 String usernameName;
306                 for(usernameName = JOptionPane.
    showInputDialog(PassGeneratorDemo.table, "Enter Account
    Username: "); usernameName.contains("'") || usernameName.
    equals(""); usernameName = JOptionPane.showInputDialog(
    PassGeneratorDemo.table, "Your username name empty,
    please re-enter username name: ")) {
307                 }
308
309                 String urlName = JOptionPane.
    showInputDialog(PassGeneratorDemo.table, "Enter Website
    Name: ");
310                 PassGeneratorDemo.password = new Password
    ();
311                 PassGeneratorDemo.password.
    generatePassword();
312                 PassGeneratorDemo.key = new Password();
313                 PassGeneratorDemo.key.generatePassword();
314                 PassGeneratorDemo.database.addUser(
    accountName, usernameName, urlName, PassGeneratorDemo.
    password.toString(), PassGeneratorDemo.key.toString());
315                 PrintStream var10000 = System.out;
316                 String var10001 = this.getClass().getName
    ();
317                 var10000.println(var10001 + " Account: "
     + accountName + " Added Successfully");
318                 Logger var6 = PassGeneratorDemo.logger;
319                 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
```

```
## Root Location Option !!
#log4j.rootCategory=debug,console
#
## Package Based Logging
#log4j.logger.com.jcg.log4j.console.appender=debug,console
#log4j.additivity.com.jcg.log4j.console.appender=false
#
## Redirect Log Messages To Console !!
#log4j.appender.console=org.apache.log4j.ConsoleAppender
#log4j.appender.console.target=System.out
#log4j.appender.console.immediateFlush=true
#log4j.appender.console.encoding=UTF-8
#log4j.appender.console.layout=org.apache.log4j.
PatternLayout
#log4j.appender.console.layout.conversionPattern=%d{yyyy-
MM-dd HH:mm:ss} %-5p %c{1}:%L - %m%n
#public static Logger logger2=Logger.getLogger(Class.class
.getName());

```

```java
package com.RomTal.java;

/***
 * GUI Manager Interface
 */
public interface IView {
    /***
     * Update table of data
     */
    public void rebuildPanel();

    /***
     * get name of account
     * @return selected key of account
     */
    public String getSelection();
    /***
     * get key of account
     * @return selected name of account
     */
    public String getSelectionKey();

    /***
     * set selection to null after deleting account
     */
    public void setSelection();

    /***
     * view/copy parameter of account
     * @param password password of an account
     */
    public void displayCopy(String password);

    /***
     * popup warning if account not been selected
     */
    public void displayWarning();
}

```

```java
package com.RomTal.java;

/***
 * Password Generator Manger Interface
 */
public interface IModel {
    /***
     * Generate a new Password
     */
    public void generatePassword();

    /***
     * Generate a new password from a specific characters dictionary
     * @param startingChar Character dictionary
     */
    public void generatePassword(char startingChar);

    /***
     * Generate a new password in length that specific length(minimum 16 characters)
     * @param length Password length (minimum 16 characters)
     */
    public void generatePassword(int length);

    /***
     * checking if password equal to string
     * @param compareString password to compare
     * @return
     */
    public boolean equals(String compareString);

    /***
     * checking if password include symbols
     * @return
     */
    public boolean hasSymbols();

    /***
     * convert to string
     * @return
     */
    public String toString();
}

```

```
 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
   database for table");
40                          DatabaseMetaData databaseMetadata
    = conn.getMetaData();
41                          ResultSet resultSet =
   databaseMetadata.getTables(null, null, "DATA", null);//
   checking if table data exist
42                          if (resultSet.next()) {
43                              System.out.println("TABLE
   ALREADY EXISTS");
44                          } else {
45                              state.execute("create table
   data(account varchar(60) ,username varchar(60) ,url
   varchar(120) ,password varchar(60),KeyVal varchar(60))");
46                          }
47                          flag++;
48                      }
49                  ResultSet res = state.executeQuery("SELECT
    account FROM data");
50                  if(!res.next()) {
51                      System.out.println("Building data
   table.");
52                      logger.info(this.getClass().getName
   () + " Building data table");
53                  }
54              }
55              catch (SQLException e)
56              {
57                  JOptionPane.showMessageDialog(null, e.
   getMessage());
58              }
59          }
60      }
61
62      /***
63       * Close connection to database
64       * @throws SQLException
65       */
66      public void closeConnection() throws SQLException {
67          if (conn == null) {
68              getConnection();
69          }
70          conn.close();
71          conn=null;
72          System.out.println("Layout panel successfully
   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
    account, username, url, password, KeyVal FROM data");
 86         return res;
 87     }
 88
 89     /***
 90      * get a reference to account match key in database
 91      * @param KeyVal key of account
 92      * @return
 93      * @throws SQLException
 94      */
 95     public ResultSet displayInfo(String KeyVal) throws
    SQLException {
 96         if (conn == null) {
 97             getConnection();
 98         }
 99         Statement state = conn.createStatement();
100         ResultSet res = state.executeQuery("SELECT
    account, username, url, password FROM data WHERE KeyVal
     = '" + KeyVal + "'");
101         return res;
102     }
103
104     /***
105      *
106      * @return get amount of account saved in database
107      * @throws SQLException
108      */
109     public int displayAmount() throws SQLException {
110         if (conn == null) {
111             getConnection();
112         }
113         int count = 0;
114         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,
    String url, String password, String KeyVal) {
131         if (conn == null) {
132             getConnection();
133         }
134         try
135         {
136             PreparedStatement prep = conn.
    prepareStatement("INSERT INTO data VALUES(?,?,?,?,?)");
137             prep.setString(1, account);
138             prep.setString(2, username);
139             prep.setString(3, url);
140             prep.setString(4, password);
141             prep.setString(5, KeyVal);
142             prep.execute();
143         }
144         catch (SQLException e)
145         {
146             JOptionPane.showMessageDialog(null, e.
    getMessage());
147         }
148     }
149 
150     /***
151      * Change username of account match key in database
152      * @param KeyVal key of account
153      * @param newUsername New username to change for
154      */
155     public void changeUser(String KeyVal, String
    newUsername) {
```

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

```java
package com.RomTal.java;

import java.security.SecureRandom;

/***
 * Password Generator Manger
 */
public class Password implements IModel {
//  passwords Generator at least 10 characters long.
    private final byte PASSWORD_LENGTH = 16;
    //This string holds all the possible values that a generated password might contain.
    private final String CHARACTER_DICTIONARY = "abcdefghijklmnopqrstuvwxyz"
                                              + "ABCDEFGHIJKLMNOPQRSTUVWXYZ"
                                              + "1234567890"
                                              + "!?@#$%";
    private String password;
    private SecureRandom secureGenerator = new SecureRandom();
    //Constructor that generates a 16 character string, that will be used to create a password.zB7yI6@quL5SS6pd
    public Password() {
        password = "";
    }
    public Password(String existingPass) {
        password = existingPass;
    }
    public Password(Password object2) {
        password = object2.password;
    }
    @Override
    public void generatePassword() {
        String[] arrString = new String[PASSWORD_LENGTH]; //array that will be used for the generation of the password.
        while (!(this.hasSymbols())) { //Loop that checks for the password containing a symbol.
            for (int i = 0; i < PASSWORD_LENGTH; i++) {
                arrString[i] = Character.toString(CHARACTER_DICTIONARY.charAt(secureGenerator.nextInt(CHARACTER_DICTIONARY.length())));
            }
            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              for (int i = 1; i < PASSWORD_LENGTH; i++) {
44                  arrString[i] = Character.toString(
    CHARACTER_DICTIONARY.charAt(secureGenerator.nextInt(
    CHARACTER_DICTIONARY.length())));
45              }
46              password = String.join("", arrString);
47          }
48      }
49      @Override
50      public void generatePassword(int length) {
51          String[] arrString = new String[length]; //array
    that will be used for the generation of the password.
52          while (!(this.hasSymbols())) { //Loop that checks
    for the password containing a symbol.
53              for (int i = 0; i < PASSWORD_LENGTH; i++) {
54                  arrString[i] = Character.toString(
    CHARACTER_DICTIONARY.charAt(secureGenerator.nextInt(
    CHARACTER_DICTIONARY.length())));
55              }
56              password = String.join("", arrString);
57          }
58      }
59      @Override
60      public boolean equals(String compareString) {
61          return password == compareString;
62      }
63      @Override
64      public boolean hasSymbols() {
65          return (password.contains("?") || password.
    contains("!") || password.contains("@") || password.
    contains("#") || password.contains("$") || password.
    contains("%"));
66      }
67      @Override
68      public String toString() {
```

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

```java
package com.RomTal.java;

/***
 * Main Application Interface
 */
public interface IViewModel {
    public static void main(String[] args) throws
Exception{ new PassGeneratorDemo();}
}

```

```
 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.
   SINGLE_SELECTION);
42         ListSelectionModel listSelectionModel = table.
   getSelectionModel();
43         listSelectionModel.addListSelectionListener(new
```

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

```
 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
    Credentials");
 94         JButton altUser = new JButton("Change Username");
 95         JButton altPass = new JButton("Change Password");
 96         JButton altSite = new JButton("Change Website");
 97         addPass.addActionListener(new PassGeneratorDemo.
    AddPassListener());
 98         delPass.addActionListener(new PassGeneratorDemo.
    DelPassListener());
 99         seePass.addActionListener(new PassGeneratorDemo.
    SeePassListener());
100         altUser.addActionListener(new PassGeneratorDemo.
    AltUserListener());
101         altPass.addActionListener(new PassGeneratorDemo.
    AltPassListener());
102         altSite.addActionListener(new PassGeneratorDemo.
    AltSiteListener());
103         //set border around buttons
104         buttonPanel.setBorder(BorderFactory.
    createTitledBorder("Operation "));
105         //adds buttons to the panel
106         buttonPanel.add(addPass);
107         buttonPanel.add(delPass);
108         buttonPanel.add(seePass);
109         buttonPanel.add(altUser);
110         buttonPanel.add(altPass);
111         buttonPanel.add(altSite);
112     }
113     @Override
114     public String getSelection() {
115         return selectedAccount;
116     }
117     @Override
118     public String getSelectionKey() {
119         return selectedKey;
120     }
121     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
    PassGeneratorDemo.CopyPassListener());
133         copyUsername.addActionListener(new
    PassGeneratorDemo.CopyUserListener());
134         copyUrl.addActionListener(new PassGeneratorDemo.
    CopyUrlListener());
135         passText = new JLabel("Password:");
136         display.add(passText);
137         passText = new JLabel(password);
138         display.add(passText);
139         display.add(copyUsername);
140         display.add(copyPassword);
141         display.add(copyUrl);
142         JOptionPane.showMessageDialog(this, display);
143     };
144     @Override
145     public void displayWarning(){
146         JOptionPane.showMessageDialog(this, "Warning no
    account was selected, retry after selecting an account");
147     };
148     private class TableModel extends AbstractTableModel {
149         public int getColumnCount() {
150             return columnNames.length;
151         }
152         public int getRowCount() {
153             return data.length;
154         }
155         public String getColumnName(int col) {
156             return columnNames[col];
157         }
158         public String getValueAt(int rowNum, int colNum
    ) {
159             return data[rowNum][colNum];
160         }
```

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

```java
package com.RomTal.java;

import org.apache.log4j.FileAppender;
import org.apache.log4j.Logger;
import org.apache.log4j.PatternLayout;
import javax.swing.*;
import java.awt.*;
import java.awt.datatransfer.Clipboard;
import java.awt.datatransfer.StringSelection;
import java.awt.event.ActionEvent;
import java.awt.event.ActionListener;
import java.io.IOException;
import java.io.PrintWriter;
import java.sql.ResultSet;
import java.sql.SQLException;

/***
 * Main Application
 */
public class PassGeneratorDemo implements IViewModel{
    public static Logger logger;
    static {
        String file = "logs.txt";
        logger = Logger.getLogger(Class.class.getName());
        try {
            PrintWriter writer = new PrintWriter(file);
            writer.print("");
            writer.close();
            logger.addAppender(new FileAppender(new
PatternLayout("%-5p [%t]: %m%n"), file));
        }
        catch (IOException e) {
            e.printStackTrace();
        }
    }
    private static DerbyDB database = new DerbyDB();
    private static TablePanel table= new TablePanel();
    private static Clipboard clipboard = Toolkit.
getDefaultToolkit().getSystemClipboard();
    private static Password password;
    private static Password key;
    /***
     * Add new account (implements listener in Add account
 button)
     */
    public static class AddPassListener implements
```

```
43 ActionListener {
44         public void actionPerformed(ActionEvent event) {
45             try {
46                 String accountName = JOptionPane.
   showInputDialog(table, "Enter Account Name: ");
47                 while (accountName.contains("'") ||
   accountName.equals("")) {
48                     accountName = JOptionPane.
   showInputDialog(table, "Your account name empty, please re
   -enter account name: ");
49                 }
50                 String usernameName = JOptionPane.
   showInputDialog(table, "Enter Account Username: ");
51                 while (usernameName.contains("'") ||
   usernameName.equals("")) {
52                     usernameName = JOptionPane.
   showInputDialog(table, "Your username name empty, please
   re-enter username name: ");
53                 }
54                 String urlName = JOptionPane.
   showInputDialog(table, "Enter Website Name: ");
55                 password = new Password();
56                 password.generatePassword();
57                 key = new Password();
58                 key.generatePassword();
59                 database.addUser(accountName, usernameName
   , urlName, password.toString(), key.toString());
60                 System.out.println(this.getClass().getName
   () + " Account: " + accountName + " Added Successfully");
61                 logger.info(this.getClass().getName() + "
   Account: " + accountName + " Added Successfully");
62                 table.rebuildPanel();
63             } catch (Exception e) {
64                 System.out.println("Cancel Pressed");
65                 logger.info("Cancel Pressed");
66             }
67             password = null;
68 
69         }
70     }
71     /***
72      * deleting account (implements listener in delete
   account button)
73      */
74     public static class DelPassListener implements
   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.
    showInputDialog(table, "Confirm deletion by typing " +
    table.getSelection() + ":");
 82                     if (confirmation.equalsIgnoreCase(
    table.getSelection())) {
 83                         res = database.displayInfo(table.
    getSelectionKey());
 84                         res.next();
 85                         System.out.println(this.getClass
    () + " Account: " + res.getString("Username") + " Deleted
     Successfully");
 86                         logger.info(this.getClass() + "
    Account: " + res.getString("Username") + " Deleted
    Successfully");
 87                         database.deleteUser(table.
    getSelectionKey());
 88                         table.setSelection();
 89                         table.rebuildPanel();
 90                     } else {
 91                         JOptionPane.showMessageDialog(
    table, "Incorrect account name");
 92                     }
 93                 } catch (Exception e) {
 94                     System.out.println("Cancel Pressed");
 95                     logger.info("Cancel Pressed");
 96                 }
 97             }
 98         }
 99     }
100     /***
101      * view/copy account credentials (implements listener
     in copy/view credentials button)
102      */
103     public static class SeePassListener implements
    ActionListener {
104         public void actionPerformed(ActionEvent event) {
105             ResultSet res;
106             try {
107                 if (table.getSelection() == null) {
108                     table.displayWarning();
```

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

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

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

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

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

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

```java
package com.RomTal.java;

import org.junit.jupiter.api.AfterEach;
import org.junit.jupiter.api.BeforeEach;
import org.junit.jupiter.api.Test;

import java.sql.ResultSet;
import java.sql.SQLException;

import static org.junit.jupiter.api.Assertions.*;

class DerbyDBTest {

    public DerbyDB database;
    public String Key="KeyChecking";
    public ResultSet res;


    @BeforeEach
    void setUp() {
        database = new DerbyDB();
    }

    @AfterEach
    void tearDown() {
        database = null;
    }

    @Test
    void closeConnection() throws SQLException {
        database.closeConnection();
    }

    @Test
    void displayUsers() throws SQLException {
        database.displayUsers();
    }

    @Test
    void displayInfo() throws SQLException {
        database.displayInfo(Key);
    }

    @Test
    void displayAmount() throws SQLException {
        database.displayAmount();
```

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

```java
package com.RomTal.java;

import org.junit.jupiter.api.AfterEach;
import org.junit.jupiter.api.BeforeEach;
import org.junit.jupiter.api.Test;

import static org.junit.jupiter.api.Assertions.*;

class PasswordTest {

    public static Password password;
    int test = 20;
    char test1='R';
    String Pass="PassChecking";
    @BeforeEach
    void setUp() {
        password = new Password();
    }

    @AfterEach
    void tearDown() {
        password=null;
    }

    @Test
    void generatePassword() {
        password.generatePassword();
    }

    @Test
    void testGeneratePassword() {
        password.generatePassword(test);
    }

    @Test
    void testGeneratePassword1() {
        password.generatePassword(test1);
    }

    @Test
    void testEquals() {
        password.equals(Pass);
    }

    @Test
    void hasSymbols() {
```

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

```java
package com.RomTal.java;

import org.junit.jupiter.api.AfterEach;
import org.junit.jupiter.api.BeforeEach;
import org.junit.jupiter.api.Test;

class TablePanelTest {

    public TablePanel table;
    String Pass="PassChecking";
    @BeforeEach
    void setUp() {
        table = new TablePanel();
    }

    @AfterEach
    void tearDown() {
         table= null;
    }

    @Test
    void testRebuildPanel() {
        table.rebuildPanel();
    }

    @Test
    void testGetSelection() {
        table.getSelection();
    }

    @Test
    void testGetSelectionKey() {
        table.getSelectionKey();
    }

    @Test
    void testSetSelection() {
        table.setSelection();
    }

    @Test
    void testDisplayCopy() {
        table.displayCopy(Pass);
    }
}
```

```java
package com.RomTal.java;

import org.junit.jupiter.api.AfterEach;
import org.junit.jupiter.api.BeforeEach;
import org.junit.jupiter.api.Test;

import static org.junit.jupiter.api.Assertions.*;

class PassGeneratorDemoTest {

    public PassGeneratorDemo Application;
    @BeforeEach
    void setUp() {
        Application = new PassGeneratorDemo();
    }

    @AfterEach
    void tearDown() {
        Application = null;
    }

    @Test
    void main() {
    }
}
```