

## Appendix: Source Code

### TABLE OF CONTENTS

|                          |     |
|--------------------------|-----|
| Login Window .....       | 2   |
| Reset Password .....     | 12  |
| Admin Window .....       | 21  |
| Add New Members.....     | 28  |
| Assign Teams.....        | 35  |
| Update Score.....        | 69  |
| View Rankings.....       | 77  |
| Team Ranking .....       | 83  |
| Club Ranking .....       | 93  |
| View Progress.....       | 108 |
| Team Progress .....      | 113 |
| New Season .....         | 123 |
| Team Leader Select.....  | 135 |
| Team Leader Window ..... | 145 |
| Team Member Window.....  | 161 |
| Update Progress.....     | 173 |

## Login Window



```
import java.awt.Dimension;
import java.awt.Toolkit;
import java.sql.DriverManager;
import java.sql.PreparedStatement;
import java.sql.ResultSet;
import java.sql.SQLException;
import java.util.logging.Level;
import java.util.logging.Logger;
import javax.swing.JOptionPane;
```

```
/*
 * To change this license header, choose License Headers in Project Properties.
 * To change this template file, choose Tools | Templates
 * and open the template in the editor.
 */
```

```
*/
```

```
/**
```

```
*
```

```
* @author aidenyan
```

```
*/
```

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

```
    /** preliminary static variables that would
```

```
        * be helpful for further logistics in the programs
```

```
        * */
```

```
    public static String nameUsing;
```

```

static String status;
public static String teamUsed;

/**
 * Creates new form LoginWindow
 */
public LoginWindow() {
    initComponents();
    Toolkit toolkit = getToolkit();
    Dimension size = toolkit.getScreenSize();
    setLocation(size.width / 2 - getWidth() / 2, size.height / 2 - getHeight() / 2);
//center the GUI

}

/**
 * This method is called from within the constructor to initialize the form.
 * WARNING: Do NOT modify this code. The content of this method is always
 * regenerated by the Form Editor.
 */
@SuppressWarnings("unchecked")
// <editor-fold defaultstate="collapsed" desc="Generated Code">
private void initComponents() {

    jPanel1 = new javax.swing.JPanel();
    jLabel1 = new javax.swing.JLabel();
    jLabel2 = new javax.swing.JLabel();
    jLabel3 = new javax.swing.JLabel();
    UsernameField = new javax.swing.JTextField();
    PasswordField = new javax.swing.JPasswordField();
    ResetPasswordButton = new javax.swing.JButton();
    LoginButton = new javax.swing.JButton();

    setDefaultCloseOperation(javax.swing.WindowConstants.EXIT_ON_CLOSE);

    jPanel1.setForeground(new java.awt.Color(255, 153, 0));
    jPanel1.setLocation(new java.awt.Point(0, 0));

    jLabel1.setFont(new java.awt.Font("Lucida Grande", 0, 24)); // NOI18N
    jLabel1.setText("Welcome to QHAPPY Online System!");

    jLabel2.setFont(new java.awt.Font("Lucida Grande", 0, 14)); // NOI18N
    jLabel2.setText("Username:");

```

```

jLabel3.setFont(new java.awt.Font("Lucida Grande", 0, 14)); // NOI18N
jLabel3.setText("Password:");

UsernameField.addActionListener(new java.awt.event.ActionListener() {
    public void actionPerformed(java.awt.event.ActionEvent evt) {
        UsernameFieldActionPerformed(evt);
    }
});

ResetPasswordButton.setText("Forget Password");
ResetPasswordButton.addActionListener(new
java.awt.event.ActionListener() {
    public void actionPerformed(java.awt.event.ActionEvent evt) {
        ResetPasswordButtonActionPerformed(evt);
    }
});

LoginButton.setBackground(new java.awt.Color(0, 153, 255));
LoginButton.setText("Login");
LoginButton.addActionListener(new java.awt.event.ActionListener() {
    public void actionPerformed(java.awt.event.ActionEvent evt) {
        LoginButtonActionPerformed(evt);
    }
});

javax.swing.GroupLayout jPanel1Layout = new
javax.swing.GroupLayout(jPanel1);
jPanel1.setLayout(jPanel1Layout);
jPanel1Layout.setHorizontalGroup(

jPanel1Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
    .addGroup(javax.swing.GroupLayout.Alignment.TRAILING,
jPanel1Layout.createSequentialGroup()
        .addGroup(jPanel1Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.TRAILING)
            .addGroup(jPanel1Layout.createSequentialGroup()
                .addComponent(jLabel1,
javax.swing.GroupLayout.PREFERRED_SIZE, 431,
javax.swing.GroupLayout.PREFERRED_SIZE))
            .addGroup(jPanel1Layout.createSequentialGroup()
                .addGap(87, 87, 87)

```

```

        .addGroup(jPanel1Layout.createParallelGroup(javax.swing.
GroupLayout.Alignment.LEADING)
            .addComponent(jLabel3)
            .addComponent(jLabel2))
        .addPreferredGap(javax.swing.LayoutStyle.Component
Placement.RELATED, 84, Short.MAX_VALUE)
        .addGroup(jPanel1Layout.createParallelGroup(javax.swing.
GroupLayout.Alignment.LEADING)
            .addComponent(UsernameField,
javax.swing.GroupLayout.Alignment.TRAILING,
javax.swing.GroupLayout.PREFERRED_SIZE, 293,
javax.swing.GroupLayout.PREFERRED_SIZE)
            .addComponent(PasswordField,
javax.swing.GroupLayout.Alignment.TRAILING,
javax.swing.GroupLayout.PREFERRED_SIZE, 293,
javax.swing.GroupLayout.PREFERRED_SIZE))))
        .addGap(88, 88, 88))
        .addGroup(javax.swing.GroupLayout.Alignment.TRAILING,
jPanel1Layout.createSequentialGroup()
            .addGap(0, 0, Short.MAX_VALUE)
            .addComponent(ResetPasswordButton)
            .addGap(137, 137, 137))
        .addGroup(jPanel1Layout.createParallelGroup(javax.swing.GroupLayout.
Alignment.LEADING)
            .addGroup(jPanel1Layout.createSequentialGroup()
                .addGap(133, 133, 133)
                .addComponent(LoginButton)
                .addContainerGap(413, Short.MAX_VALUE)))
        );
jPanel1Layout.setVerticalGroup(

jPanel1Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
    .addGroup(jPanel1Layout.createSequentialGroup()
        .addGap(54, 54, 54)
        .addComponent(jLabel1,
javax.swing.GroupLayout.PREFERRED_SIZE, 86,
javax.swing.GroupLayout.PREFERRED_SIZE)
        .addGap(75, 75, 75)
        .addGroup(jPanel1Layout.createParallelGroup(javax.swing.
GroupLayout.Alignment.BASELINE)
            .addComponent(jLabel2)
            .addComponent(UsernameField,
javax.swing.GroupLayout.PREFERRED_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE,

```

```

javax.swing.GroupLayout.PREFERRED_SIZE))
        .addGap(35, 35, 35)
        .addGroup(jPanel1Layout.createParallelGroup(GroupLayout.Alignment.BASELINE)
            .addComponent(jLabel3)
            .addComponent(PasswordField,
javax.swing.GroupLayout.PREFERRED_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE,
javax.swing.GroupLayout.PREFERRED_SIZE))
        .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement
.RELATED, 73, Short.MAX_VALUE)
        .addComponent(ResetPasswordButton)
        .addGap(63, 63, 63))
        .addGroup(jPanel1Layout.createParallelGroup(GroupLayout.Alignment.LEADING)
            .addGroup(javax.swing.GroupLayout.Alignment.TRAILING,
jPanel1Layout.createSequentialGroup()
                .addContainerGap(376, Short.MAX_VALUE)
                .addComponent(LoginButton)
                .addGap(62, 62, 62)))
        );

```

```

        javax.swing.GroupLayout layout = new
javax.swing.GroupLayout(getContentPane());
        getContentPane().setLayout(layout);
        layout.setHorizontalGroup(

```

```

        layout.createParallelGroup(GroupLayout.Alignment.LEADING)
            .addGroup(javax.swing.GroupLayout.Alignment.TRAILING,
layout.createSequentialGroup()
                .addContainerGap(82, Short.MAX_VALUE)
                .addComponent(jPanel1,
javax.swing.GroupLayout.PREFERRED_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE,
javax.swing.GroupLayout.PREFERRED_SIZE)
                .addGap(89, 89, 89))
        );
        layout.setVerticalGroup(

```

```

        layout.createParallelGroup(GroupLayout.Alignment.LEADING)
            .addGroup(layout.createSequentialGroup()
                .addGap(60, 60, 60)
                .addComponent(jPanel1,
javax.swing.GroupLayout.PREFERRED_SIZE,

```

```

javax.swing.GroupLayout.DEFAULT_SIZE,
javax.swing.GroupLayout.PREFERRED_SIZE)
        .addContainerGap(44, Short.MAX_VALUE))
    );

    pack();
} // </editor-fold>

private void UsernameFieldActionPerformed(java.awt.event.ActionEvent evt) {
    // TODO add your handling code here:
}

private void ResetPasswordButtonActionPerformed(java.awt.event.ActionEvent
evt) {
    new ResetPassword().setVisible(true);
    dispose();
}

private void LoginButtonActionPerformed(java.awt.event.ActionEvent evt) {
    try {
        //connect to database
        String connectionURL =
"jdbc:derby://localhost:1527/QHAPPYDatabase";
        java.sql.Connection conn =
DriverManager.getConnection(connectionURL, "username", "password");

        //initiate sql orders that would fetch member login information from
the different tables in the database
        String sql1 = "Select * from TEAM_MEMBER_LOGIN where
QDNUMBER=(?) and PASSWORD=(?)";
        String sql2 = "Select * from TEAM_LEADER_LOGIN where
QDNUMBER=(?) and PASSWORD=(?)";
        String sql3 = "Select * from ADMIN_LOGIN where
QDNUMBER=(?) and PASSWORD=(?)";

        PreparedStatement pst1 = conn.prepareStatement(sql1);
        PreparedStatement pst2 = conn.prepareStatement(sql2);
        PreparedStatement pst3 = conn.prepareStatement(sql3);

        String userName = UsernameField.getText();
        String passWord = PasswordField.getText();

        //put into the data that needs to be searched in the selected tables in my
SQL orders

```

```
pst1.setString(1, userName);
pst1.setString(2, passWord);
```

```
pst2.setString(1, userName);
pst2.setString(2, passWord);
```

```
pst3.setString(1, userName);
pst3.setString(2, passWord);
```

```
//comprise a resultset of the search results of the match between
QDNumbers and Passwords
```

```
ResultSet rs1 = pst1.executeQuery();
ResultSet rs2 = pst2.executeQuery();
ResultSet rs3 = pst3.executeQuery();
```

```
//set some preliminary boolean variables for further identification
```

```
boolean teamMemberLogin = false;
boolean teamLeaderLogin = false;
boolean adminLogin = false;
boolean found = false;
```

```
while (rs1.next()) {
```

```
    String getUsername = rs1.getString("QDNumber");
    String getPassword = rs1.getString("Password");
    String getName = rs1.getString("Name"); //re-check if usernames
```

```
and passwords match
```

```
    if (userName.equals(getUsername) &&
        passWord.equals(getPassword)) {
```

```
        /**define the static variables, set the login status
```

```
        * (either team member, team leader, or administrator)
```

```
        * same for the remaining while loops
```

```
        * */
```

```
        nameUsing = getName;
```

```
        status = "member";
```

```
        found = true;
```

```
        JOptionPane.showMessageDialog(null, "Hi " + nameUsing +
            ", Welcome to the System");
```

```
        teamMemberLogin = true;
```

```
    }
```

```
}
```

```
while (rs2.next()) {
```

```
    String getUsername = rs2.getString("QDNumber");
```

```
    String getPassword = rs2.getString("Password");
```



```

        String getName = rs2.getString("Name");
        if (passWord.equals(getPassword) &&
        userName.equals(getUsername)) {

            nameUsing = getName;
            status = "leader";
            found = true;
            teamLeaderLogin = true;
            JOptionPane.showMessageDialog(null, "Hi " + nameUsing +
            ", Welcome to the System");
        }
    }

    while (rs3.next()) {
        String getUsername = rs3.getString("QDNumber");
        String getPassword = rs3.getString("Password");
        String getName = rs3.getString("Name");
        if (userName.equals(getUsername) &&
        passWord.equals(getPassword)) {
            nameUsing = getName;
            status = "admin";
            found = true;
            adminLogin = true;
            JOptionPane.showMessageDialog(null, "Hi " + nameUsing +
            ", Welcome to the System");
        }
    }

    /**identify the types of users' status (member, leader, or administrator)
    * displace them into the corresponding navigation GUIs
    */
    if (teamMemberLogin == true) {
        new TeamMemberWindow().setVisible(true);
        dispose();
    }

    if (teamLeaderLogin == true) {
        new TeamLeaderWindow().setVisible(true);
        dispose();
    }

    if (adminLogin == true) {
        new AdminWindow().setVisible(true);
        dispose();
    }

```

```

    }

    //show error message if username and password don't match
    if (found != true) {
        JOptionPane.showMessageDialog(null, "Incorrect Username and
Password!");
    }

    } catch (SQLException ex) {
        JOptionPane.showMessageDialog(null, "Error");
    }
}

/**
 * @param args the command line arguments
 */
public static void main(String args[]) {
    /* Set the Nimbus look and feel */
    //<editor-fold defaultstate="collapsed" desc=" Look and feel setting code
(optional) ">
    /* If Nimbus (introduced in Java SE 6) is not available, stay with the default
look and feel.
    * For details see
http://download.oracle.com/javase/tutorial/uiswing/lookandfeel/plaf.html
    */
    try {
        for (javax.swing.UIManager.LookAndFeelInfo info :
javax.swing.UIManager.getInstalledLookAndFeels()) {
            if ("Nimbus".equals(info.getName())) {

javax.swing.UIManager.setLookAndFeel(info.getClassName());
                break;
            }
        }
    } catch (ClassNotFoundException ex) {

java.util.logging.Logger.getLogger(LoginWindow.class.getName()).log(java.util.loggi
ng.Level.SEVERE, null, ex);
    } catch (InstantiationException ex) {

java.util.logging.Logger.getLogger(LoginWindow.class.getName()).log(java.util.loggi
ng.Level.SEVERE, null, ex);
    } catch (IllegalAccessException ex) {

```

```
java.util.logging.Logger.getLogger(LoginWindow.class.getName()).log(java.util.logging.Level.SEVERE, null, ex);
    } catch (javax.swing.UnsupportedLookAndFeelException ex) {
```

```
java.util.logging.Logger.getLogger(LoginWindow.class.getName()).log(java.util.logging.Level.SEVERE, null, ex);
    }
//</editor-fold>
```

```
/* Create and display the form */
java.awt.EventQueue.invokeLater(new Runnable() {
    public void run() {
        new LoginWindow().setVisible(true);
    }
});
}
```

```
// Variables declaration - do not modify
private javax.swing.JButton LoginButton;
private javax.swing.JPasswordField PasswordField;
private javax.swing.JButton ResetPasswordButton;
private javax.swing.JTextField UsernameField;
private javax.swing.JLabel jLabel1;
private javax.swing.JLabel jLabel2;
private javax.swing.JLabel jLabel3;
private javax.swing.JPanel jPanel1;
// End of variables declaration
}
```

## Reset Password



**RESET PASSWORD**

QD Number:

Full Name:

New Password:

```
import java.awt.Dimension;
import java.awt.Toolkit;
import java.sql.DriverManager;
import java.sql.PreparedStatement;
import java.sql.ResultSet;
import java.sql.SQLException;
import java.util.logging.Level;
import java.util.logging.Logger;
import javax.swing.JOptionPane;
```

```
/*
 * To change this license header, choose License Headers in Project Properties.
 * To change this template file, choose Tools | Templates
 * and open the template in the editor.
 */
/**
 *
 * @author aidenyan
 */
public class ResetPassword extends javax.swing.JFrame {
```

```
/**
```

```
 * Creates new form ResetPassword
```

```

    */
    public ResetPassword() {
        initComponents();
        Toolkit toolkit = getToolkit();
        Dimension size = toolkit.getScreenSize();
        setLocation(size.width / 2 - getWidth() / 2, size.height / 2 - getHeight() / 2);
    }

    /**
     * This method is called from within the constructor to initialize the form.
     * WARNING: Do NOT modify this code. The content of this method is always
     * regenerated by the Form Editor.
     */
    @SuppressWarnings("unchecked")
    // <editor-fold defaultstate="collapsed" desc="Generated Code">
    private void initComponents() {

        jLabel1 = new javax.swing.JLabel();
        jLabel2 = new javax.swing.JLabel();
        jLabel3 = new javax.swing.JLabel();
        jLabel4 = new javax.swing.JLabel();
        QDNumberField = new javax.swing.JTextField();
        FullNameField = new javax.swing.JTextField();
        NewPasswordField = new javax.swing.JTextField();
        SubmitButton = new javax.swing.JButton();
        GetBackButton = new javax.swing.JButton();

        setDefaultCloseOperation(javax.swing.WindowConstants.EXIT_ON_CLOSE);

        jLabel1.setFont(new java.awt.Font("Lucida Grande", 1, 24)); // NOI18N
        jLabel1.setText("RESET PASSWORD");

        jLabel2.setFont(new java.awt.Font("Lucida Grande", 0, 16)); // NOI18N
        jLabel2.setText("QD Number:");

        jLabel3.setFont(new java.awt.Font("Lucida Grande", 0, 16)); // NOI18N
        jLabel3.setText("Full Name:");

        jLabel4.setFont(new java.awt.Font("Lucida Grande", 0, 16)); // NOI18N
        jLabel4.setText("New Password:");

        SubmitButton.setText("Submit");
        SubmitButton.addActionListener(new java.awt.event.ActionListener() {

```

```

        public void actionPerformed(java.awt.event.ActionEvent evt) {
            SubmitButtonActionPerformed(evt);
        }
    });

    GetBackButton.setText("Back");
    GetBackButton.addActionListener(new java.awt.event.ActionListener() {
        public void actionPerformed(java.awt.event.ActionEvent evt) {
            GetBackButtonActionPerformed(evt);
        }
    });

    javax.swing.GroupLayout layout = new
    javax.swing.GroupLayout(getContentPane());
    getContentPane().setLayout(layout);
    layout.setHorizontalGroup(

    layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
        .addGroup(layout.createSequentialGroup()
            .addGroup(layout.createParallelGroup(javax.swing.GroupLayout.
                Alignment.LEADING)
                .addGroup(layout.createSequentialGroup()
                    .addGroup(layout.createParallelGroup(javax.swing.Gro
                        upLayout.Alignment.LEADING)
                        .addComponent(jLabel1)
                        .addGroup(layout.createSequentialGroup()
                            .addGap(56, 56, 56)
                            .addGroup(layout.createParallelGroup(javax.Gro
                                upLayout.Alignment.LEADING)
                                .addComponent(jLabel2)
                                .addComponent(jLabel3)
                                .addComponent(jLabel4)
                                .addGap(76, 76, 76)
                                .addGroup(layout.createParallelGroup(javax.Gro
                                    upLayout.Alignment.LEADING, false)
                                    .addComponent(QDNumberField)
                                    .addComponent(FullNameField)
                                    .addComponent(NewPasswordField,
                                        javax.swing.GroupLayout.DEFAULT_SIZE, 273, Short.MAX_VALUE))))
                            .addGap(128, 128, 128)
                            .addComponent(SubmitButton)
                            .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement

```

```

.RELATED, 179, Short.MAX_VALUE)
    .addComponent(GetBackButton)
    .addGap(147, 147, 147))
);
layout.setVerticalGroup(

layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
    .addGroup(layout.createSequentialGroup()
        .addGap(20, 20, 20)
        .addComponent(jLabel1)
        .addGap(34, 34, 34)
        .addGroup(layout.createParallelGroup(javax.swing.GroupLayout.
Alignment.BASELINE)
            .addComponent(jLabel2)
            .addComponent(QDNumberField,
javax.swing.GroupLayout.PREFERRED_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE,
javax.swing.GroupLayout.PREFERRED_SIZE))
            .addGap(56, 56, 56)
            .addGroup(layout.createParallelGroup(javax.swing.GroupLayout.
Alignment.BASELINE)
                .addComponent(jLabel3)
                .addComponent(FullNameField,
javax.swing.GroupLayout.PREFERRED_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE,
javax.swing.GroupLayout.PREFERRED_SIZE))
                .addGap(66, 66, 66)
                .addGroup(layout.createParallelGroup(javax.swing.GroupLayout.
Alignment.BASELINE)
                    .addComponent(jLabel4)
                    .addComponent(NewPasswordField,
javax.swing.GroupLayout.PREFERRED_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE,
javax.swing.GroupLayout.PREFERRED_SIZE))
                    .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement
.RELATED, 79, Short.MAX_VALUE)
                    .addGroup(layout.createParallelGroup(javax.swing.GroupLayout.
Alignment.BASELINE)
                        .addComponent(SubmitButton)
                        .addComponent(GetBackButton))
                    .addGap(57, 57, 57))
            );

pack();

```

```

} // </editor-fold>

private void GetBackButtonActionPerformed(java.awt.event.ActionEvent evt) {
    new LoginWindow().setVisible(true);
    dispose();
}

private void SubmitButtonActionPerformed(java.awt.event.ActionEvent evt) {

    try {
        String connectionURL =
"jdbc:derby://localhost:1527/QHAPPYDatabase";
        java.sql.Connection conn =
DriverManager.getConnection(connectionURL, "username", "password");
        String sql1 = "SELECT * FROM TEAM_MEMBER_LOGIN
WHERE NAME=(?) and QDNUMBER=?";
        String sql2 = "SELECT * FROM TEAM_MEMBER_LOGIN
WHERE NAME=(?) and QDNUMBER=?";
        String sql3 = "SELECT * FROM TEAM_MEMBER_LOGIN
WHERE NAME=(?) and QDNUMBER=?";
        /** Connect to databases and all relevant login information tables
         * that would be used for verifications for existng users
         */

        PreparedStatement pst1 = conn.prepareStatement(sql1);
        PreparedStatement pst2 = conn.prepareStatement(sql2);
        PreparedStatement pst3 = conn.prepareStatement(sql3);

        //get the information that the user has input into the textfields
        String fullName = FullNameField.getText();
        String QdNumber = QDNumberField.getText();
        String newPassword = NewPasswordField.getText();

        //fill in the question marks, search if the current user does exist
        pst1.setString(1, fullName);
        pst1.setString(2, QdNumber);
        pst2.setString(1, fullName);
        pst2.setString(2, QdNumber);
        pst3.setString(1, fullName);
        pst3.setString(2, QdNumber);

        //produce a resultset of the results
        ResultSet rs1 = pst1.executeQuery();
        ResultSet rs2 = pst2.executeQuery();
    }
}

```



```

ResultSet rs3 = pst3.executeQuery();

//a boolean variables to determine whether to send an error message or
not
boolean reset = false;

/**
 * check through all resultsets which marks each database login
information
 * table, so that we know which database table should be updated
 */
while (rs1.next()) {
    String getName = rs1.getString("NAME");
    String getQdNumber = rs1.getString("QdNumber");
    if (QdNumber.equals(getQdNumber) &&
fullName.equals(getName)) {
        String sql4 = "UPDATE TEAM_MEMBER_LOGIN SET
PASSWORD=? WHERE NAME=? and QDNUMBER=?";
        PreparedStatement pst4 = conn.prepareStatement(sql4);
        pst4.setString(1, newPassword);
        pst4.setString(2, fullName);
        pst4.setString(3, QdNumber);
        //update the new password for the corresponding user in the
database

        pst4.executeUpdate();

        reset = true;
    }
}

while (rs2.next()) {
    String getName = rs2.getString("NAME");
    String getQdNumber = rs2.getString("QdNumber");
    if (QdNumber.equals(getQdNumber) &&
fullName.equals(getName)) {
        String sql5 = "UPDATE TEAM_LEADER_LOGIN SET
PASSWORD=? WHERE NAME=? and QDNUMBER=?";
        PreparedStatement pst5 = conn.prepareStatement(sql5);
        pst5.setString(1, newPassword);
        pst5.setString(2, fullName);
        pst5.setString(3, QdNumber);
        pst5.executeUpdate();
        reset = true;
    }
}

```

```

    }

    while (rs3.next()) {
        String getName = rs3.getString("NAME");
        String getQdNumber = rs3.getString("QdNumber");
        if (QdNumber.equals(getQdNumber) &&
fullName.equals(getName)) {
            String sql6 = "UPDATE ADMIN_LOGIN SET
PASSWORD=? WHERE NAME=? and QDNUMBER=?";
            PreparedStatement pst6 = conn.prepareStatement(sql6);
            pst6.setString(1, newPassword);
            pst6.setString(2, fullName);
            pst6.setString(3, QdNumber);
            pst6.executeUpdate();
            reset = true;
        }
    }
    //output success message
    if (reset == true) {
        JOptionPane.showMessageDialog(null, "Update Successfully!");
        new LoginWindow().setVisible(true);
        dispose();
    } else if (reset == false) {
        JOptionPane.showMessageDialog(null, "Wrong full name or QD
Number");
    }

    } catch (SQLException ex) {
        JOptionPane.showMessageDialog(null, "Error");
    }

}

/**
 * @param args the command line arguments
 */
public static void main(String args[]) {
    /* Set the Nimbus look and feel */
    //<editor-fold defaultstate="collapsed" desc=" Look and feel setting code
(optional) ">
    /* If Nimbus (introduced in Java SE 6) is not available, stay with the default
look and feel.
    * For details see

```

```

http://download.oracle.com/javase/tutorial/uiswing/lookandfeel/plaf.html
    */
    try {
        for (javax.swing.UIManager.LookAndFeelInfo info :
javax.swing.UIManager.getInstalledLookAndFeels()) {
            if ("Nimbus".equals(info.getName())) {

javax.swing.UIManager.setLookAndFeel(info.getClassName());
                break;

            }
        }
    } catch (ClassNotFoundException ex) {
        java.util.logging.Logger.getLogger(ResetPassword.class
            .getName()).log(java.util.logging.Level.SEVERE, null, ex);

    } catch (InstantiationException ex) {
        java.util.logging.Logger.getLogger(ResetPassword.class
            .getName()).log(java.util.logging.Level.SEVERE, null, ex);

    } catch (IllegalAccessException ex) {
        java.util.logging.Logger.getLogger(ResetPassword.class
            .getName()).log(java.util.logging.Level.SEVERE, null, ex);

    } catch (javax.swing.UnsupportedLookAndFeelException ex) {
        java.util.logging.Logger.getLogger(ResetPassword.class
            .getName()).log(java.util.logging.Level.SEVERE, null, ex);
    }
}
//</editor-fold>

/* Create and display the form */
java.awt.EventQueue.invokeLater(new Runnable() {
    public void run() {
        new ResetPassword().setVisible(true);
    }
});
}

// Variables declaration - do not modify
private javax.swing.JTextField FullNameField;
private javax.swing.JButton GetBackButton;
private javax.swing.JTextField NewPasswordField;
private javax.swing.JTextField QDNumberField;
private javax.swing.JButton SubmitButton;

```

```
private javax.swing.JLabel jLabel1;  
private javax.swing.JLabel jLabel2;  
private javax.swing.JLabel jLabel3;  
private javax.swing.JLabel jLabel4;  
// End of variables declaration  
}
```

## Admin Window



```
import java.awt.Dimension;  
import java.awt.Toolkit;
```

```
/*  
 * To change this license header, choose License Headers in Project Properties.  
 * To change this template file, choose Tools | Templates  
 * and open the template in the editor.  
 */  
/**  
 *  
 * @author aidenyan  
 */  
public class AdminWindow extends javax.swing.JFrame {  
  
    /**  
     * Creates new form adminLogin  
     */  
    public AdminWindow() {  
        initComponents();  
        Toolkit toolkit = getToolkit();  
        Dimension size = toolkit.getScreenSize();  
        setLocation(size.width / 2 - getWidth() / 2, size.height / 2 - getHeight() / 2);  
    }  
  
    /**  
     * This method is called from within the constructor to initialize the form.
```

```

* WARNING: Do NOT modify this code. The content of this method is always
* regenerated by the Form Editor.
*/
@SuppressWarnings("unchecked")
// <editor-fold defaultstate="collapsed" desc="Generated Code">
private void initComponents() {

    RetireButton = new javax.swing.JButton();
    ManageMembersButton = new javax.swing.JButton();
    ManageTeamsButton = new javax.swing.JButton();
    NewSeasonButton = new javax.swing.JButton();
    jLabel1 = new javax.swing.JLabel();
    NewMemberButton = new javax.swing.JButton();
    AssignTeamsButton = new javax.swing.JButton();
    jButton3 = new javax.swing.JButton();
    MemberProgressButton = new javax.swing.JButton();
    ViewRankButton = new javax.swing.JButton();
    GetBackButton = new javax.swing.JButton();

    RetireButton.setText("Retire");

    ManageMembersButton.setText("Manage Members");

    ManageTeamsButton.setText("Manage Teams");

    NewSeasonButton.setText("Start New Season");

    setDefaultCloseOperation(javax.swing.WindowConstants.EXIT_ON_CLOSE);

    jLabel1.setFont(new java.awt.Font("Lucida Grande", 1, 24)); // NOI18N
    jLabel1.setText("ADMINISTRATOR PAGE");

    NewMemberButton.setText("Add New Member");
    NewMemberButton.addActionListener(new
java.awt.event.ActionListener() {
        public void actionPerformed(java.awt.event.ActionEvent evt) {
            NewMemberButtonActionPerformed(evt);
        }
    });

    AssignTeamsButton.setText("Assign Teams");
    AssignTeamsButton.addActionListener(new
java.awt.event.ActionListener() {

```



```

        .addGroup(layout.createSequentialGroup()
            .addGap(42, 42, 42)
            .addComponent(MemberProgressButton)
            .addPreferredGap(javax.swing.LayoutStyle.Component
Placement.RELATED, javax.swing.GroupLayout.DEFAULT_SIZE,
Short.MAX_VALUE))
        .addGroup(javax.swing.GroupLayout.Alignment.TRAILING
, layout.createSequentialGroup()
            .addGap(57, 57, 57)
            .addGroup(layout.createParallelGroup(javax.swing.Gro
upLayout.Alignment.LEADING)
                .addComponent(NewMemberButton)
                .addGroup(layout.createSequentialGroup()
                    .addGap(13, 13, 13)
                    .addComponent(jButton3)))
                .addGap(156, 156, 156)))
        .addGroup(layout.createParallelGroup(javax.swing.GroupLayout.
Alignment.LEADING)
            .addComponent(ViewRankButton)
            .addComponent(AssignTeamsButton)
            .addGroup(layout.createSequentialGroup()
                .addGap(30, 30, 30)
                .addComponent(GetBackButton)))
            .addGap(107, 107, 107))
        .addGroup(layout.createSequentialGroup()
            .addGap(140, 140, 140)
            .addComponent(jLabel1)
            .addContainerGap(javax.swing.GroupLayout.DEFAULT_SIZE,
Short.MAX_VALUE))
    );
    layout.setVerticalGroup(

layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
        .addGroup(layout.createSequentialGroup()
            .addGap(29, 29, 29)
            .addComponent(jLabel1)
            .addGap(42, 42, 42)
            .addGroup(layout.createParallelGroup(javax.swing.GroupLayout.
Alignment.BASELINE)
                .addComponent(NewMemberButton)
                .addComponent(AssignTeamsButton))
            .addGap(59, 59, 59)
            .addGroup(layout.createParallelGroup(javax.swing.GroupLayout.
Alignment.BASELINE)

```



```

        .addComponent(jButton3)
        .addComponent(ViewRankButton))
    .addGap(69, 69, 69)
    .addGroup(layout.createParallelGroup(javax.swing.GroupLayout.
Alignment.BASELINE)
        .addComponent(MemberProgressButton)
        .addComponent(GetBackButton))
    .addContainerGap(49, Short.MAX_VALUE))
    );

    pack();
} // </editor-fold>

```

**//Check for and display to different UIs with different buttons**

```

private void GetBackButtonActionPerformed(java.awt.event.ActionEvent evt) {

    new LoginWindow().setVisible(true);
    dispose();
}

private void NewMemberButtonActionPerformed(java.awt.event.ActionEvent
evt) {
    new AddNewMembers().setVisible(true);
    dispose();
}

private void AssignTeamsButtonActionPerformed(java.awt.event.ActionEvent
evt) {
    new AssignTeams().setVisible(true);
    dispose();
}

private void jButton3ActionPerformed(java.awt.event.ActionEvent evt) {
    new UpdateScore().setVisible(true);
    dispose();
}

private void ViewRankButtonActionPerformed(java.awt.event.ActionEvent evt)
{
    new ViewRankings().setVisible(true);
    dispose();
}

private void

```

```

MemberProgressButtonActionPerformed(java.awt.event.ActionEvent evt) {
    new MembersProgress().setVisible(true);
    dispose();
}

/**
 * @param args the command line arguments
 */
public static void main(String args[]) {
    /* Set the Nimbus look and feel */
    //<editor-fold defaultstate="collapsed" desc=" Look and feel setting code
(optional) ">
    /* If Nimbus (introduced in Java SE 6) is not available, stay with the default
look and feel.
    * For details see
http://download.oracle.com/javase/tutorial/uiswing/lookandfeel/plaf.html
    */
    try {
        for (javax.swing.UIManager.LookAndFeelInfo info :
javax.swing.UIManager.getInstalledLookAndFeels()) {
            if ("Nimbus".equals(info.getName())) {

javax.swing.UIManager.setLookAndFeel(info.getClassName());
                break;
            }
        }
    } catch (ClassNotFoundException ex) {

java.util.logging.Logger.getLogger(AdminWindow.class.getName()).log(java.util.log
ging.Level.SEVERE, null, ex);
    } catch (InstantiationException ex) {

java.util.logging.Logger.getLogger(AdminWindow.class.getName()).log(java.util.log
ging.Level.SEVERE, null, ex);
    } catch (IllegalAccessException ex) {

java.util.logging.Logger.getLogger(AdminWindow.class.getName()).log(java.util.log
ging.Level.SEVERE, null, ex);
    } catch (javax.swing.UnsupportedLookAndFeelException ex) {

java.util.logging.Logger.getLogger(AdminWindow.class.getName()).log(java.util.log
ging.Level.SEVERE, null, ex);
    }
}
//</editor-fold>

```

```

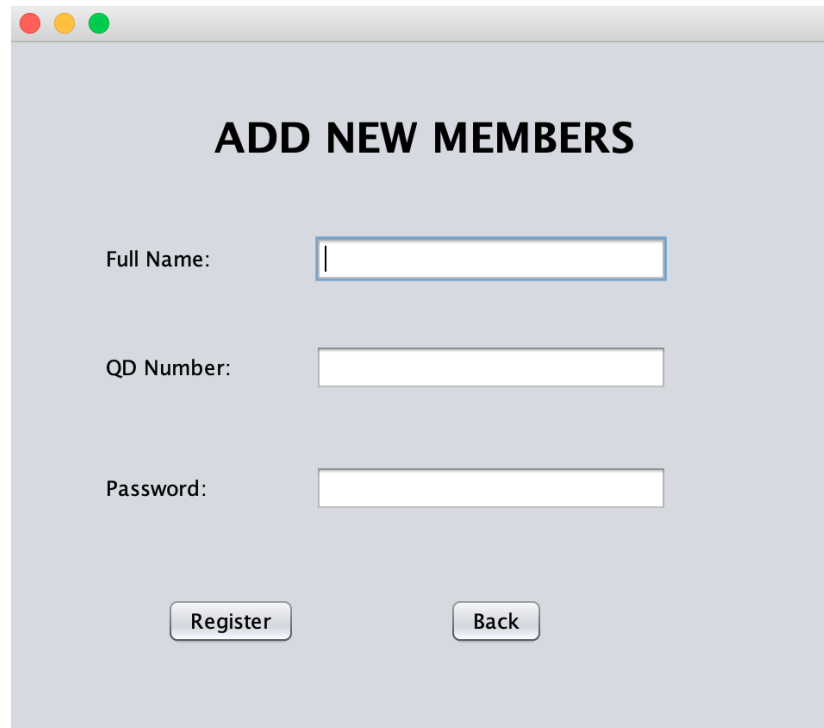
//</editor-fold>

/* Create and display the form */
java.awt.EventQueue.invokeLater(new Runnable() {
    public void run() {
        new AdminWindow().setVisible(true);
    }
});
}

// Variables declaration - do not modify
private javax.swing.JButton AssignTeamsButton;
private javax.swing.JButton GetBackButton;
private javax.swing.JButton ManageMembersButton;
private javax.swing.JButton ManageTeamsButton;
private javax.swing.JButton MemberProgressButton;
private javax.swing.JButton NewMemberButton;
private javax.swing.JButton NewSeasonButton;
private javax.swing.JButton RetireButton;
private javax.swing.JButton ViewRankButton;
private javax.swing.JButton jButton3;
private javax.swing.JLabel jLabel1;
// End of variables declaration
}

```

## Add New Members



**ADD NEW MEMBERS**

Full Name:

QD Number:

Password:

```
import java.awt.Dimension;  
import java.awt.Toolkit;  
import java.sql.DriverManager;  
import java.sql.PreparedStatement;  
import java.sql.SQLException;  
import javax.swing.JOptionPane;
```

```
/*  
 * To change this license header, choose License Headers in Project Properties.  
 * To change this template file, choose Tools | Templates  
 * and open the template in the editor.  
 */
```

```
/**
```

```
 *
```

```
 * @author aidenyan
```

```
 */
```

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

```
    /**
```

```
     * Creates new form AddNewMembers
```

```
    */
```

```
    public AddNewMembers() {  
        initComponents();  
        Toolkit toolkit = getToolkit();
```

```

        Dimension size = toolkit.getScreenSize();
        setLocation(size.width / 2 - getWidth() / 2, size.height / 2 - getHeight() / 2);
    }

    /**
     * This method is called from within the constructor to initialize the form.
     * WARNING: Do NOT modify this code. The content of this method is always
     * regenerated by the Form Editor.
     */
    @SuppressWarnings("unchecked")
    // <editor-fold defaultstate="collapsed" desc="Generated Code">
    private void initComponents() {

        jLabel1 = new javax.swing.JLabel();
        jLabel2 = new javax.swing.JLabel();
        jLabel3 = new javax.swing.JLabel();
        jLabel4 = new javax.swing.JLabel();
        QDNumberField = new javax.swing.JTextField();
        PasswordField = new javax.swing.JTextField();
        FullNameField = new javax.swing.JTextField();
        RegisterButton = new javax.swing.JButton();
        GetBackButton = new javax.swing.JButton();

        setDefaultCloseOperation(javax.swing.WindowConstants.EXIT_ON_CLOSE);

        jLabel1.setFont(new java.awt.Font("Lucida Grande", 1, 24)); // NOI18N
        jLabel1.setText("ADD NEW MEMBERS");

        jLabel2.setText("Full Name:");

        jLabel3.setText("QD Number:");

        jLabel4.setText("Password:");

        RegisterButton.setText("Register");
        RegisterButton.addActionListener(new java.awt.event.ActionListener() {
            public void actionPerformed(java.awt.event.ActionEvent evt) {
                RegisterButtonActionPerformed(evt);
            }
        });

        GetBackButton.setText("Back");
        GetBackButton.addActionListener(new java.awt.event.ActionListener() {

```



```

        .addGap(42, 42, 42)
        .addGroup(layout.createParallelGroup(javax.swing.GroupLayout.
Alignment.BASELINE)
            .addComponent(jLabel2)
            .addComponent(FullNameField,
javax.swing.GroupLayout.PREFERRED_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE,
javax.swing.GroupLayout.PREFERRED_SIZE))
        .addGap(36, 36, 36)
        .addGroup(layout.createParallelGroup(javax.swing.GroupLayout.
Alignment.BASELINE)
            .addComponent(jLabel3)
            .addComponent(QDNumberField,
javax.swing.GroupLayout.PREFERRED_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE,
javax.swing.GroupLayout.PREFERRED_SIZE))
        .addGap(43, 43, 43)
        .addGroup(layout.createParallelGroup(javax.swing.GroupLayout.
Alignment.BASELINE)
            .addComponent(jLabel4)
            .addComponent(PasswordField,
javax.swing.GroupLayout.PREFERRED_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE,
javax.swing.GroupLayout.PREFERRED_SIZE))
        .addGap(50, 50, 50)
        .addGroup(layout.createParallelGroup(javax.swing.GroupLayout.
Alignment.BASELINE)
            .addComponent(RegisterButton)
            .addComponent(GetBackButton))
        .addContainerGap(55, Short.MAX_VALUE))
    );

    pack();
} // </editor-fold>

private void GetBackButtonActionPerformed(java.awt.event.ActionEvent evt) {
    new AdminWindow().setVisible(true);
    dispose();
}

private void RegisterButtonActionPerformed(java.awt.event.ActionEvent evt) {
    try {
        //connect to database
        String connectionURL =

```

```

"jdbc:derby://localhost:1527/QHAPPYDatabase";
        java.sql.Connection conn =
DriverManager.getConnection(connectionURL, "username", "password");

        //obtain the contents written in the textfields
        String fullName = FullNameField.getText();
        String QDNumber = QDNumberField.getText();
        String Password = PasswordField.getText();

        //insert the login information to the login information table
        String sql1 = "INSERT INTO TEAM_MEMBER_LOGIN (NAME,
QDNUMBER, PASSWORD) VALUES (?, ?, ?) ";
        PreparedStatement pst1 = conn.prepareStatement(sql1);
        pst1.setString(1, fullName);
        pst1.setString(2, QDNumber);
        pst1.setString(3, Password);
        pst1.executeUpdate();

        //default the new member as a member that has not been assigned any
teams
        String sql2 = "INSERT INTO UNASSIGNED_MEMBERS (NAME,
LIT_PROGRESS, ART_PROGRESS, SCI_PROGRESS, SOCSCI_PROGRESS,
MUS_PROGRESS, MAT_PROGRESS, ECON_PROGRESS,
OVERALL_PROGRESS, POINTS, TEAM_LEADER) VALUES
(?,0,0,0,0,0,0,0,0,true)";
        PreparedStatement pst2 = conn.prepareStatement(sql2);
        pst2.setString(1, fullName);
        pst2.executeUpdate();

        //output success message
        JOptionPane.showMessageDialog(null, "New Member Added!");
        new AdminWindow().setVisible(true);
        dispose();

    } catch (SQLException ex) {
        JOptionPane.showMessageDialog(null, "Error");
    }
}

/**
 * @param args the command line arguments
 */
public static void main(String args[]) {

```



```

        /* Set the Nimbus look and feel */
        //<editor-fold defaultstate="collapsed" desc=" Look and feel setting code
(optional) ">
        /* If Nimbus (introduced in Java SE 6) is not available, stay with the default
look and feel.
        * For details see
http://download.oracle.com/javase/tutorial/uiswing/lookandfeel/plaf.html
        */
        try {
            for (javax.swing.UIManager.LookAndFeelInfo info :
javax.swing.UIManager.getInstalledLookAndFeels()) {
                if ("Nimbus".equals(info.getName())) {

javax.swing.UIManager.setLookAndFeel(info.getClassName());
                    break;

                }
            }
        } catch (ClassNotFoundException ex) {

java.util.logging.Logger.getLogger(AddNewMembers.class.getName()).log(java.util.l
ogging.Level.SEVERE, null, ex);
        } catch (InstantiationException ex) {

java.util.logging.Logger.getLogger(AddNewMembers.class.getName()).log(java.util.l
ogging.Level.SEVERE, null, ex);
        } catch (IllegalAccessException ex) {

java.util.logging.Logger.getLogger(AddNewMembers.class.getName()).log(java.util.l
ogging.Level.SEVERE, null, ex);
        } catch (javax.swing.UnsupportedLookAndFeelException ex) {

java.util.logging.Logger.getLogger(AddNewMembers.class.getName()).log(java.util.l
ogging.Level.SEVERE, null, ex);
        }
    }
    //</editor-fold>

    /* Create and display the form */
    java.awt.EventQueue.invokeLater(new Runnable() {
        public void run() {
            new AddNewMembers().setVisible(true);
        }
    });
}

```

```
// Variables declaration - do not modify
private javax.swing.JTextField FullNameField;
private javax.swing.JButton GetBackButton;
private javax.swing.JTextField PasswordField;
private javax.swing.JTextField QDNumberField;
private javax.swing.JButton RegisterButton;
private javax.swing.JLabel jLabel1;
private javax.swing.JLabel jLabel2;
private javax.swing.JLabel jLabel3;
private javax.swing.JLabel jLabel4;
// End of variables declaration
}
```

## Assign Teams

The screenshot shows a Java Swing window titled "Assign Teams". At the top, there is a "Select Team:" label, a dropdown menu with "Select a Te..." selected, and a "View" button. Below this are three distinct sections for different team types: "Honor Group:", "Scholastic Group:", and "Varsity Group:". Each section contains a "Member Name" text field and an "Add:" label followed by three "No Addition..." dropdown buttons. At the bottom of the window are two buttons: "Save Changes" and "Back".

```
import java.awt.Dimension;
import java.awt.Toolkit;
import java.sql.DriverManager;
import java.sql.PreparedStatement;
import java.sql.ResultSet;
import java.sql.SQLException;
import java.util.logging.Level;
import java.util.logging.Logger;
import javax.swing.JOptionPane;
import javax.swing.table.DefaultTableModel;
```

```
/*
 * To change this license header, choose License Headers in Project Properties.
 * To change this template file, choose Tools | Templates
 * and open the template in the editor.
 */
/**
 *
 * @author aidenyan
 */
```

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

```
    private int HonorCounter = 0;
    private int ScholasticCounter = 0;
```

```

private int VarsityCounter = 0;
private String teamSelected;

/**
 * Creates new form AssignTeams
 */
public AssignTeams() {
    initComponents();
    Toolkit toolkit = getToolkit();
    Dimension size = toolkit.getScreenSize();
    setLocation(size.width / 2 - getWidth() / 2, size.height / 2 - getHeight() / 2);
//center the GUI
    updateBox();
}

private void updateBox() {
    String connectionURL = "jdbc:derby://localhost:1527/QHAPPYDatabase";
    try {
        java.sql.Connection conn =
DriverManager.getConnection(connectionURL, "username", "password");
        String sql10 = "SELECT * FROM UNASSIGNED_MEMBERS";
        PreparedStatement st10 = conn.prepareStatement(sql10);
        //get all unassigned members and input them to each JComboBox
        ResultSet rs10 = st10.executeQuery();
        while (rs10.next()) {
            HonorGroupBox1.addItem(rs10.getString("NAME"));
            HonorGroupBox2.addItem(rs10.getString("NAME"));
            HonorGroupBox3.addItem(rs10.getString("NAME"));
            ScholasticGroupBox1.addItem(rs10.getString("NAME"));
            ScholasticGroupBox2.addItem(rs10.getString("NAME"));
            ScholasticGroupBox3.addItem(rs10.getString("NAME"));
            VarsityGroupBox1.addItem(rs10.getString("NAME"));
            VarsityGroupBox2.addItem(rs10.getString("NAME"));
            VarsityGroupBox3.addItem(rs10.getString("NAME"));
        }

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

}

/**

```

```

* This method is called from within the constructor to initialize the form.
* WARNING: Do NOT modify this code. The content of this method is always
* regenerated by the Form Editor.
*/

```

```

@SuppressWarnings("unchecked")
// <editor-fold defaultstate="collapsed" desc="Generated Code">
private void initComponents() {

```

```

    jScrollPane1 = new javax.swing.JScrollPane();
    jList1 = new javax.swing.JList<>();
    progressBar1 = new javax.swing.JProgressBar();
    jTextField1 = new javax.swing.JTextField();
    HonorGroupButton2 = new javax.swing.JComboBox<>();
    HonorGroupButton1 = new javax.swing.JComboBox<>();
    jLabel1 = new javax.swing.JLabel();
    jLabel2 = new javax.swing.JLabel();
    TeamBox = new javax.swing.JComboBox<>();
    ViewButton = new javax.swing.JButton();
    jScrollPane2 = new javax.swing.JScrollPane();
    HonorTable = new javax.swing.JTable();
    jLabel3 = new javax.swing.JLabel();
    jLabel4 = new javax.swing.JLabel();
    jLabel5 = new javax.swing.JLabel();
    jScrollPane3 = new javax.swing.JScrollPane();
    ScholasticTable = new javax.swing.JTable();
    jLabel6 = new javax.swing.JLabel();
    jLabel7 = new javax.swing.JLabel();
    jScrollPane4 = new javax.swing.JScrollPane();
    VarsityTable = new javax.swing.JTable();
    HonorGroupBox3 = new javax.swing.JComboBox<>();
    jLabel8 = new javax.swing.JLabel();
    SaveChangesButton = new javax.swing.JButton();
    GetBackButton = new javax.swing.JButton();
    ScholasticGroupBox1 = new javax.swing.JComboBox<>();
    HonorGroupBox2 = new javax.swing.JComboBox<>();
    HonorGroupBox1 = new javax.swing.JComboBox<>();
    ScholasticGroupBox2 = new javax.swing.JComboBox<>();
    ScholasticGroupBox3 = new javax.swing.JComboBox<>();
    VarsityGroupBox1 = new javax.swing.JComboBox<>();
    VarsityGroupBox2 = new javax.swing.JComboBox<>();
    VarsityGroupBox3 = new javax.swing.JComboBox<>();

```

```

    jList1.setModel(new javax.swing.AbstractListModel<String>() {
        String[] strings = { "Item 1", "Item 2", "Item 3", "Item 4", "Item 5" };

```

```

        public int getSize() { return strings.length; }
        public String getElementAt(int i) { return strings[i]; }
    });
    jScrollPane1.setViewportViewView(jList1);

    jTextField1.setText("jTextField1");

    HonorGroupButton2.addActionListener(new
java.awt.event.ActionListener() {
        public void actionPerformed(java.awt.event.ActionEvent evt) {
            HonorGroupButton2ActionPerformed(evt);
        }
    });

    HonorGroupButton1.addActionListener(new
java.awt.event.ActionListener() {
        public void actionPerformed(java.awt.event.ActionEvent evt) {
            HonorGroupButton1ActionPerformed(evt);
        }
    });

setDefaultCloseOperation(javax.swing.WindowConstants.EXIT_ON_CLOSE);

jLabel1.setFont(new java.awt.Font("Lucida Grande", 0, 18)); // NOI18N
jLabel1.setText("Assign Teams");

jLabel2.setText("Select Team:");

TeamBox.setModel(new javax.swing.DefaultComboBoxModel<>(new
String[] { "Select a Team...", "Team 1", "Team 2", "Team 3" }));

ViewButton.setText("View");
ViewButton.addActionListener(new java.awt.event.ActionListener() {
    public void actionPerformed(java.awt.event.ActionEvent evt) {
        ViewButtonActionPerformed(evt);
    }
});

HonorTable.setModel(new javax.swing.table.DefaultTableModel(
    new Object [][] {

    },
    new String [] {

```

```

        "Member Name"
    }
) {
    Class[] types = new Class [] {
        java.lang.String.class
    };

    public Class getColumnClass(int columnIndex) {
        return types [columnIndex];
    }
});
jScrollPane2.setViewportViewView(HonorTable);

jLabel3.setFont(new java.awt.Font("Lucida Grande", 1, 14)); // NOI18N
jLabel3.setText("Honor Group:");

jLabel4.setText("Add:");

jLabel5.setFont(new java.awt.Font("Lucida Grande", 1, 14)); // NOI18N
jLabel5.setText("Scholastic Group:");

ScholasticTable.setModel(new javax.swing.table.DefaultTableModel(
    new Object [][] {

        },
    new String [] {
        "Member Name"
    }
));
jScrollPane3.setViewportViewView(ScholasticTable);

jLabel6.setText("Add:");

jLabel7.setFont(new java.awt.Font("Lucida Grande", 1, 14)); // NOI18N
jLabel7.setText("Varsity Group:");

VarsityTable.setModel(new javax.swing.table.DefaultTableModel(
    new Object [][] {

        },
    new String [] {
        "Member Name"
    }
));

```

```

jScrollPane4.setViewportViewView(VarsityTable);

HonorGroupBox3.setModel(new
javax.swing.DefaultComboBoxModel<>(new String[] { "No Additional Member for
Honor 3" }));

jLabel8.setText("Add:");

SaveChangesButton.setText("Save Changes");
SaveChangesButton.addActionListener(new
java.awt.event.ActionListener() {
    public void actionPerformed(java.awt.event.ActionEvent evt) {
        SaveChangesButtonActionPerformed(evt);
    }
});

GetBackButton.setText("Back");
GetBackButton.addActionListener(new java.awt.event.ActionListener() {
    public void actionPerformed(java.awt.event.ActionEvent evt) {
        GetBackButtonActionPerformed(evt);
    }
});

ScholasticGroupBox1.setModel(new
javax.swing.DefaultComboBoxModel<>(new String[] { "No Additional Member for
Scholastic 1" }));

HonorGroupBox2.setModel(new
javax.swing.DefaultComboBoxModel<>(new String[] { "No Additional Member for
Honor 2" }));

HonorGroupBox1.setModel(new
javax.swing.DefaultComboBoxModel<>(new String[] { "No Additional Member for
Honor 1" }));

ScholasticGroupBox2.setModel(new
javax.swing.DefaultComboBoxModel<>(new String[] { "No Additional Member for
Scholastic 2" }));

ScholasticGroupBox3.setModel(new
javax.swing.DefaultComboBoxModel<>(new String[] { "No Additional Member for
Scholastic 3" }));

VarsityGroupBox1.setModel(new

```



```
javax.swing.DefaultComboBoxModel<>(new String[] { "No Additional Member for  
Varsity 1" }));
```

```
VarsityGroupBox2.setModel(new  
javax.swing.DefaultComboBoxModel<>(new String[] { "No Additional Member for  
Varsity 2" }));
```

```
VarsityGroupBox3.setModel(new  
javax.swing.DefaultComboBoxModel<>(new String[] { "No Additional Member for  
Varsity 3" }));
```

```
javax.swing.GroupLayout layout = new  
javax.swing.GroupLayout(getContentPane());  
getContentPane().setLayout(layout);  
layout.setHorizontalGroup(  
  
layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)  
    .addGroup(javax.swing.GroupLayout.Alignment.TRAILING,  
layout.createSequentialGroup()  
    .addGap(javax.swing.GroupLayout.DEFAULT_SIZE,  
Short.MAX_VALUE)  
    .addGroup(layout.createParallelGroup(javax.swing.GroupLayout.  
Alignment.LEADING)  
        .addComponent(jLabel7)  
        .addGroup(layout.createSequentialGroup()  
            .addComponent(ScholasticGroupBox1,  
javax.swing.GroupLayout.PREFERRED_SIZE, 117,  
javax.swing.GroupLayout.PREFERRED_SIZE)  
            .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement  
RELATED)  
            .addComponent(ScholasticGroupBox2,  
javax.swing.GroupLayout.PREFERRED_SIZE, 117,  
javax.swing.GroupLayout.PREFERRED_SIZE)  
            .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement  
RELATED)  
            .addComponent(ScholasticGroupBox3,  
javax.swing.GroupLayout.PREFERRED_SIZE, 117,  
javax.swing.GroupLayout.PREFERRED_SIZE)))  
        .addGap()  
        .addGroup(layout.createSequentialGroup()  
            .addGap(45, 45, 45)  
            .addComponent(SaveChangesButton)  
            .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement  
RELATED, javax.swing.GroupLayout.DEFAULT_SIZE, Short.MAX_VALUE)
```

```

        .addComponent(GetBackButton)
        .addGap(56, 56, 56))
    .addGroup(layout.createSequentialGroup())
    .addGroup(layout.createParallelGroup(javax.swing.GroupLayout.
Alignment.LEADING)
        .addGroup(layout.createSequentialGroup())
        .addGap(29, 29, 29)
        .addGroup(layout.createParallelGroup(javax.swing.Gro
upLayout.Alignment.LEADING)
            .addComponent(jLabel3)
            .addGroup(layout.createSequentialGroup())
            .addComponent(jLabel2)
            .addGap(18, 18, 18)
            .addGroup(layout.createParallelGroup(javax.s
wing.GroupLayout.Alignment.LEADING)
                .addGroup(layout.createSequentialGroup
())
                .addComponent(TeamBox,
javax.swing.GroupLayout.PREFERRED_SIZE, 110,
javax.swing.GroupLayout.PREFERRED_SIZE)
                .addGap(18, 18, 18)
                .addComponent(ViewButton))
                .addGroup(layout.createSequentialGroup
())
                .addGap(27, 27, 27)
                .addComponent(jLabel1))))
            .addComponent(jScrollPane2,
javax.swing.GroupLayout.PREFERRED_SIZE, 358,
javax.swing.GroupLayout.PREFERRED_SIZE)))
    .addGroup(layout.createSequentialGroup())
    .addGap(30, 30, 30)
    .addGroup(layout.createParallelGroup(javax.swing.Gro
upLayout.Alignment.LEADING)
        .addComponent(jScrollPane3,
javax.swing.GroupLayout.PREFERRED_SIZE, 363,
javax.swing.GroupLayout.PREFERRED_SIZE)
        .addComponent(jLabel5)
        .addComponent(jScrollPane4,
javax.swing.GroupLayout.PREFERRED_SIZE, 363,
javax.swing.GroupLayout.PREFERRED_SIZE)
        .addComponent(jLabel6)))
    .addGroup(layout.createSequentialGroup())
    .addGap(21, 21, 21)
    .addGroup(layout.createParallelGroup(javax.swing.Gro

```

```

upLayout.Alignment.LEADING, false)
        .addGroup(layout.createSequentialGroup()
            .addComponent(jLabel8)
            .addPreferredGap(javax.swing.LayoutStyle.C
omponentPlacement.RELATED)
            .addComponent(VarsityGroupBox1,
javax.swing.GroupLayout.PREFERRED_SIZE, 117,
javax.swing.GroupLayout.PREFERRED_SIZE)
            .addPreferredGap(javax.swing.LayoutStyle.C
omponentPlacement.RELATED)
            .addComponent(VarsityGroupBox2,
javax.swing.GroupLayout.PREFERRED_SIZE, 117,
javax.swing.GroupLayout.PREFERRED_SIZE)
            .addPreferredGap(javax.swing.LayoutStyle.C
omponentPlacement.RELATED, javax.swing.GroupLayout.DEFAULT_SIZE,
Short.MAX_VALUE)
            .addComponent(VarsityGroupBox3,
javax.swing.GroupLayout.PREFERRED_SIZE, 117,
javax.swing.GroupLayout.PREFERRED_SIZE))
        .addGroup(layout.createSequentialGroup()
            .addComponent(jLabel4)
            .addPreferredGap(javax.swing.LayoutStyle.C
omponentPlacement.RELATED)
            .addComponent(HonorGroupBox1,
javax.swing.GroupLayout.PREFERRED_SIZE, 117,
javax.swing.GroupLayout.PREFERRED_SIZE)
            .addPreferredGap(javax.swing.LayoutStyle.C
omponentPlacement.RELATED)
            .addComponent(HonorGroupBox2,
javax.swing.GroupLayout.PREFERRED_SIZE, 117,
javax.swing.GroupLayout.PREFERRED_SIZE)
            .addPreferredGap(javax.swing.LayoutStyle.C
omponentPlacement.RELATED)
            .addComponent(HonorGroupBox3,
javax.swing.GroupLayout.PREFERRED_SIZE, 117,
javax.swing.GroupLayout.PREFERRED_SIZE))))))
        .addContainerGap(12, Short.MAX_VALUE))
    );
    layout.setVerticalGroup(

    layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
        .addGroup(layout.createSequentialGroup()
            .addContainerGap()
            .addComponent(jLabel1)

```

```

        .addGap(18, 18, 18)
        .addGroup(layout.createParallelGroup(javax.swing.GroupLayout.
Alignment.BASELINE)
            .addComponent(jLabel2)
            .addComponent(TeamBox,
javax.swing.GroupLayout.PREFERRED_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE,
javax.swing.GroupLayout.PREFERRED_SIZE)
            .addComponent(ViewButton))
        .addGap(18, 18, 18)
        .addComponent(jLabel3)
        .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement
.RELATED)
            .addComponent(jScrollPane2,
javax.swing.GroupLayout.PREFERRED_SIZE, 75,
javax.swing.GroupLayout.PREFERRED_SIZE)
            .addGap(18, 18, 18)
            .addGroup(layout.createParallelGroup(javax.swing.GroupLayout.
Alignment.BASELINE)
                .addComponent(jLabel4)
                .addComponent(HonorGroupBox3,
javax.swing.GroupLayout.PREFERRED_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE,
javax.swing.GroupLayout.PREFERRED_SIZE)
                .addComponent(HonorGroupBox2,
javax.swing.GroupLayout.PREFERRED_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE,
javax.swing.GroupLayout.PREFERRED_SIZE)
                .addComponent(HonorGroupBox1,
javax.swing.GroupLayout.PREFERRED_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE,
javax.swing.GroupLayout.PREFERRED_SIZE))
            .addGap(22, 22, 22)
            .addComponent(jLabel5)
            .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement
.RELATED)
                .addComponent(jScrollPane3,
javax.swing.GroupLayout.PREFERRED_SIZE, 86,
javax.swing.GroupLayout.PREFERRED_SIZE)
                .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement
.RELATED, 16, Short.MAX_VALUE)
            .addGroup(layout.createParallelGroup(javax.swing.GroupLayout.
Alignment.BASELINE)
                .addComponent(jLabel6)

```

```

        .addComponent(ScholasticGroupBox1,
javax.swing.GroupLayout.PREFERRED_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE,
javax.swing.GroupLayout.PREFERRED_SIZE)
        .addComponent(ScholasticGroupBox2,
javax.swing.GroupLayout.PREFERRED_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE,
javax.swing.GroupLayout.PREFERRED_SIZE)
        .addComponent(ScholasticGroupBox3,
javax.swing.GroupLayout.PREFERRED_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE,
javax.swing.GroupLayout.PREFERRED_SIZE))
        .addGap(18, 18, 18)
        .addComponent(jLabel7)
        .addGap(18, 18, 18)
        .addComponent(jScrollPane4,
javax.swing.GroupLayout.PREFERRED_SIZE, 86,
javax.swing.GroupLayout.PREFERRED_SIZE)
        .addGap(18, 18, 18)
        .addGroup(layout.createParallelGroup(javax.swing.GroupLayout.
Alignment.BASELINE)
        .addComponent(jLabel8)
        .addComponent(VarsityGroupBox1,
javax.swing.GroupLayout.PREFERRED_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE,
javax.swing.GroupLayout.PREFERRED_SIZE)
        .addComponent(VarsityGroupBox2,
javax.swing.GroupLayout.PREFERRED_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE,
javax.swing.GroupLayout.PREFERRED_SIZE)
        .addComponent(VarsityGroupBox3,
javax.swing.GroupLayout.PREFERRED_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE,
javax.swing.GroupLayout.PREFERRED_SIZE))
        .addGap(38, 38, 38)
        .addGroup(layout.createParallelGroup(javax.swing.GroupLayout.
Alignment.BASELINE)
        .addComponent(SaveChangesButton)
        .addComponent(GetBackButton))
        .addGap(43, 43, 43))
    );

    pack();
} // </editor-fold>

```

```

private void HonorGroupButton1ActionPerformed(java.awt.event.ActionEvent
evt) {
    // TODO add your handling code here:
}

private void HonorGroupButton2ActionPerformed(java.awt.event.ActionEvent
evt) {
    // TODO add your handling code here:
}

private void ViewButtonActionPerformed(java.awt.event.ActionEvent evt) {

    String connectionURL = "jdbc:derby://localhost:1527/QHAPPYDatabase";
    try {
        java.sql.Connection conn =
DriverManager.getConnection(connectionURL, "username", "password");
        teamSelected = (String) TeamBox.getSelectedItem();
        /* Check which team was selected, and then put in the current
information about the team's
        academic grouping */
        if (teamSelected.equals("Team 1")) {
            String sql1 = "SELECT * FROM TEAM1_MEMBERS WHERE
ACADEMICGROUP=?";
            String sql2 = "SELECT * FROM TEAM1_MEMBERS WHERE
ACADEMICGROUP=?";
            String sql3 = "SELECT * FROM TEAM1_MEMBERS WHERE
ACADEMICGROUP=?";
            PreparedStatement st1 = conn.prepareStatement(sql1);
            PreparedStatement st2 = conn.prepareStatement(sql2);
            PreparedStatement st3 = conn.prepareStatement(sql3);
            st1.setString(1, "Honor");
            st2.setString(1, "Scholastic");
            st3.setString(1, "Varsity");
            ResultSet rs1 = st1.executeQuery();
            ResultSet rs2 = st2.executeQuery();
            ResultSet rs3 = st3.executeQuery();

            DefaultTableModel tblModel1 = (DefaultTableModel)
HonorTable.getModel();
            DefaultTableModel tblModel2 = (DefaultTableModel)
ScholasticTable.getModel();
            DefaultTableModel tblModel3 = (DefaultTableModel)
VarsityTable.getModel();

```

```

HonorCounter = 0;
ScholasticCounter = 0;
VarsityCounter = 0;
//the counters to keep track of the number of members in each
academic group

tblModel1.setRowCount(0);
tblModel2.setRowCount(0);
tblModel3.setRowCount(0);
//then put all members in the different academic groups into the
table

while (rs1.next()) {

    String theName = rs1.getString("NAME");
    String tbData[] = {theName};
    tblModel1.addRow(tbData);
    HonorCounter += 1;
    //increment the corresponding counter by one once there is
    an existing member

}
while (rs2.next()) {

    String theName = rs2.getString("NAME");
    String tbData[] = {theName};

    tblModel2.addRow(tbData);
    ScholasticCounter += 1;
}
while (rs3.next()) {

    String theName = rs3.getString("NAME");
    String tbData[] = {theName};

    tblModel3.addRow(tbData);
    VarsityCounter += 1;
}
} else if (teamSelected.equals("Team 2")) {
    String sql4 = "SELECT * FROM TEAM2_MEMBERS WHERE
ACADEMICGROUP=?";
    String sql5 = "SELECT * FROM TEAM2_MEMBERS WHERE
ACADEMICGROUP=?";
    String sql6 = "SELECT * FROM TEAM2_MEMBERS WHERE

```

```

ACADEMICGROUP=?");
        PreparedStatement st4 = conn.prepareStatement(sql4);
        PreparedStatement st5 = conn.prepareStatement(sql5);
        PreparedStatement st6 = conn.prepareStatement(sql6);
        st4.setString(1, "Honor");
        st5.setString(1, "Scholastic");
        st6.setString(1, "Varsity");
        ResultSet rs4 = st4.executeQuery();
        ResultSet rs5 = st5.executeQuery();
        ResultSet rs6 = st6.executeQuery();

        DefaultTableModel tblModel1 = (DefaultTableModel)
HonorTable.getModel();
        DefaultTableModel tblModel2 = (DefaultTableModel)
ScholasticTable.getModel();
        DefaultTableModel tblModel3 = (DefaultTableModel)
VarsityTable.getModel();

        HonorCounter = 0;
        ScholasticCounter = 0;
        VarsityCounter = 0;

        tblModel1.setRowCount(0);
        tblModel2.setRowCount(0);
        tblModel3.setRowCount(0);

        while (rs4.next()) {
            String theName = rs4.getString("NAME");
            String tbData[] = {theName};

            tblModel1.addRow(tbData);
            HonorCounter += 1;
        }
        while (rs5.next()) {
            String theName = rs5.getString("NAME");
            String tbData[] = {theName};

            tblModel2.addRow(tbData);
            ScholasticCounter += 1;
        }
        while (rs6.next()) {
            String theName = rs6.getString("NAME");
            String tbData[] = {theName};

```



```

        tblModel3.addRow(tbData);
        VarsityCounter += 1;
    }
} else if (teamSelected.equals("Team 3")) {
    String sql7 = "SELECT * FROM TEAM3_MEMBERS WHERE
ACADEMICGROUP=?";
    String sql8 = "SELECT * FROM TEAM3_MEMBERS WHERE
ACADEMICGROUP=?";
    String sql9 = "SELECT * FROM TEAM3_MEMBERS WHERE
ACADEMICGROUP=?";
    PreparedStatement st7 = conn.prepareStatement(sql7);
    PreparedStatement st8 = conn.prepareStatement(sql8);
    PreparedStatement st9 = conn.prepareStatement(sql9);
    st7.setString(1, "Honor");
    st8.setString(1, "Scholastic");
    st9.setString(1, "Varsity");
    ResultSet rs7 = st7.executeQuery();
    ResultSet rs8 = st8.executeQuery();
    ResultSet rs9 = st9.executeQuery();

    DefaultTableModel tblModel1 = (DefaultTableModel)
HonorTable.getModel();
    DefaultTableModel tblModel2 = (DefaultTableModel)
ScholasticTable.getModel();
    DefaultTableModel tblModel3 = (DefaultTableModel)
VarsityTable.getModel();

    HonorCounter = 0;
    ScholasticCounter = 0;
    VarsityCounter = 0;

    tblModel1.setRowCount(0);
    tblModel2.setRowCount(0);
    tblModel3.setRowCount(0);

    while (rs7.next()) {
        String theName = rs7.getString("NAME");
        String tbData[] = {theName};

        tblModel1.addRow(tbData);
        HonorCounter += 1;
    }
    while (rs8.next()) {
        String theName = rs8.getString("NAME");

```

```

        String tbData[] = {theName};

        tblModel2.addRow(tbData);
        ScholasticCounter += 1;
    }
    while (rs9.next()) {
        String theName = rs9.getString("NAME");
        String tbData[] = {theName};

        tblModel3.addRow(tbData);
        VarsityCounter += 1;
    }
} else {
    JOptionPane.showMessageDialog(null, "Please Select a Team!");
}
} catch (SQLException ex) {
    Logger.getLogger(AssignTeams.class.getName()).log(Level.SEVERE,
null, ex);
}
}

private void GetBackButtonActionPerformed(java.awt.event.ActionEvent evt) {
    new AdminWindow().setVisible(true);
    dispose();
}

private void SaveChangesButtonActionPerformed(java.awt.event.ActionEvent
evt) {

    String connectionURL = "jdbc:derby://localhost:1527/QHAPPYDatabase";
    try {
        boolean successful = true;
        //the boolean variable to see if the update can be made successful
        (works like a flag)
        teamSelected = (String) TeamBox.getSelectedItem();

        java.sql.Connection conn =
DriverManager.getConnection(connectionURL, "username", "password");
        //Obtain all the inputs in the jComboBoxes
        String NewHonorMember1 = (String)
HonorGroupBox1.getSelectedItem();
        String NewHonorMember2 = (String)
HonorGroupBox2.getSelectedItem();
        String NewHonorMember3 = (String)

```

```

HonorGroupBox3.getSelectedItemAt();
    String NewScholasticMember1 = (String)
ScholasticGroupBox1.getSelectedItemAt();
    String NewScholasticMember2 = (String)
ScholasticGroupBox2.getSelectedItemAt();
    String NewScholasticMember3 = (String)
ScholasticGroupBox3.getSelectedItemAt();
    String NewVarsityMember1 = (String)
VarsityGroupBox1.getSelectedItemAt();
    String NewVarsityMember2 = (String)
VarsityGroupBox2.getSelectedItemAt();
    String NewVarsityMember3 = (String)
VarsityGroupBox3.getSelectedItemAt();

    //check if the user has selected a team to operate with
    if (teamSelected.equals("Select a Team...")) {
        JOptionPane.showMessageDialog(null, "Please select a Team!");

        //make sure they haven't put in the same member into two
        different academic groups in a team
        } else if (NewHonorMember1.equals(NewHonorMember2) ||
NewHonorMember1.equals(NewHonorMember3)
            || NewHonorMember1.equals(NewScholasticMember1) ||
NewHonorMember1.equals(NewScholasticMember2)
            || NewHonorMember1.equals(NewScholasticMember3) ||
NewHonorMember1.equals(NewVarsityMember1)
            || NewHonorMember1.equals(NewVarsityMember2) ||
NewHonorMember1.equals(NewVarsityMember3)
            || NewHonorMember2.equals(NewHonorMember3)
            || NewHonorMember2.equals(NewScholasticMember1) ||
NewHonorMember2.equals(NewScholasticMember2)
            || NewHonorMember2.equals(NewScholasticMember3) ||
NewHonorMember2.equals(NewVarsityMember1)
            || NewHonorMember2.equals(NewVarsityMember2) ||
NewHonorMember2.equals(NewVarsityMember3)
            || NewHonorMember3.equals(NewScholasticMember2) ||
NewHonorMember3.equals(NewScholasticMember1)
            || NewHonorMember3.equals(NewScholasticMember3) ||
NewHonorMember3.equals(NewVarsityMember1)
            || NewHonorMember3.equals(NewVarsityMember2) ||
NewHonorMember3.equals(NewVarsityMember3)
            || NewScholasticMember1.equals(NewScholasticMember2)
            || NewScholasticMember1.equals(NewScholasticMember3) ||
NewScholasticMember1.equals(NewVarsityMember1)

```

```

        || NewScholasticMember1.equals(NewVarsityMember2) ||
NewScholasticMember1.equals(NewVarsityMember3)
        || NewScholasticMember2.equals(NewScholasticMember3) ||
NewScholasticMember2.equals(NewVarsityMember1)
        || NewScholasticMember2.equals(NewVarsityMember2) ||
NewScholasticMember2.equals(NewVarsityMember3)
        || NewScholasticMember3.equals(NewVarsityMember1)
        || NewScholasticMember3.equals(NewVarsityMember2) ||
NewScholasticMember3.equals(NewVarsityMember3)
        || NewVarsityMember1.equals(NewVarsityMember2) ||
NewVarsityMember1.equals(NewVarsityMember3)
        || NewVarsityMember2.equals(NewVarsityMember3)) {
    JOptionPane.showMessageDialog(null, "Can't put one person in
two groups!");
    successful = false;
} else {
    String sql11 = "UPDATE UNASSIGNED_MEMBERS SET
ACADEMICGROUP=(?) WHERE NAME = (?)";
    String sql12 = "UPDATE UNASSIGNED_MEMBERS SET
ACADEMICGROUP=(?) WHERE NAME = (?)";
    String sql13 = "UPDATE UNASSIGNED_MEMBERS SET
ACADEMICGROUP=(?) WHERE NAME = (?)";
    String sql14 = "UPDATE UNASSIGNED_MEMBERS SET
ACADEMICGROUP=(?) WHERE NAME = (?)";
    String sql15 = "UPDATE UNASSIGNED_MEMBERS SET
ACADEMICGROUP=(?) WHERE NAME = (?)";
    String sql16 = "UPDATE UNASSIGNED_MEMBERS SET
ACADEMICGROUP=(?) WHERE NAME = (?)";
    String sql17 = "UPDATE UNASSIGNED_MEMBERS SET
ACADEMICGROUP=(?) WHERE NAME = (?)";
    String sql18 = "UPDATE UNASSIGNED_MEMBERS SET
ACADEMICGROUP=(?) WHERE NAME = (?)";
    String sql19 = "UPDATE UNASSIGNED_MEMBERS SET
ACADEMICGROUP=(?) WHERE NAME = (?)";

    PreparedStatement st11 = conn.prepareStatement(sql11);
    PreparedStatement st12 = conn.prepareStatement(sql12);
    PreparedStatement st13 = conn.prepareStatement(sql13);
    PreparedStatement st14 = conn.prepareStatement(sql14);
    PreparedStatement st15 = conn.prepareStatement(sql15);
    PreparedStatement st16 = conn.prepareStatement(sql16);
    PreparedStatement st17 = conn.prepareStatement(sql17);
    PreparedStatement st18 = conn.prepareStatement(sql18);
    PreparedStatement st19 = conn.prepareStatement(sql19);

```

```

st11.setString(1, "Honor");
st11.setString(2, NewHonorMember1);

st12.setString(1, "Honor");
st12.setString(2, NewHonorMember2);

st13.setString(1, "Honor");
st13.setString(2, NewHonorMember3);

st14.setString(1, "Scholastic");
st14.setString(2, NewScholasticMember1);

st15.setString(1, "Scholastic");
st15.setString(2, NewScholasticMember2);

st16.setString(1, "Scholastic");
st16.setString(2, NewScholasticMember3);

st17.setString(1, "Varsity");
st17.setString(2, NewVarsityMember1);

st18.setString(1, "Varsity");
st18.setString(2, NewVarsityMember1);

st19.setString(1, "Varsity");
st19.setString(2, NewVarsityMember1);

```

members

```
//prepare to set the new academic groups for these unassigned
```

```

if (NewHonorMember1 != "No Additional Member for Honor 1")
{
    HonorCounter += 1;
    //if one member selected, then add the honorcounter by one
    and see if the number of members has exceeded 3
    if (HonorCounter <= 3) {
        if (teamSelected == "Team 1") {
            st11.executeUpdate();
            //first update this new member's academic group

```

status

```

String sql20 = "INSERT INTO
TEAM1_MEMBERS SELECT*FROM UNASSIGNED_MEMBERS WHERE
NAME=?";

```

```

        String sql21 = "DELETE FROM
UNASSIGNED_MEMBERS WHERE NAME=?";
        //delete the data from the original
unassigned_members database and move it to the corresponding team
        PreparedStatement st20 =
conn.prepareStatement(sql20);
        PreparedStatement st21 =
conn.prepareStatement(sql21);
        st20.setString(1, NewHonorMember1);
        st21.setString(1, NewHonorMember1);
        /*execute both deletion and insertion, all of the
following codes
(repeated 3 potential teams/person * 9 person = 27
times) have all the same logistics
*/

        st20.executeUpdate();
        st21.executeUpdate();

    } else if (teamSelected == "Team 2") {
        st11.executeUpdate();
        String sql22 = "INSERT INTO
TEAM2_MEMBERS SELECT*FROM UNASSIGNED_MEMBERS WHERE
NAME=?";
        String sql23 = "DELETE FROM
UNASSIGNED_MEMBERS WHERE NAME=?";
        PreparedStatement st22 =
conn.prepareStatement(sql22);
        PreparedStatement st23 =
conn.prepareStatement(sql23);
        st22.setString(1, NewHonorMember1);
        st23.setString(1, NewHonorMember1);
        st22.executeUpdate();
        st23.executeUpdate();
    } else if (teamSelected == "Team 3") {
        st11.executeUpdate();
        String sql24 = "INSERT INTO
TEAM3_MEMBERS SELECT*FROM UNASSIGNED_MEMBERS WHERE
NAME=?";
        String sql25 = "DELETE FROM
UNASSIGNED_MEMBERS WHERE NAME=?";
        PreparedStatement st24 =
conn.prepareStatement(sql24);
        PreparedStatement st25 =

```

```

conn.prepareStatement(sql25);
        st24.setString(1, NewHonorMember1);
        st25.setString(1, NewHonorMember1);
        st24.executeUpdate();
        st25.executeUpdate();
    }
    } else {
        JOptionPane.showMessageDialog(null, "Too many
members in Honor Group!!");
        successful = false;
    }
}

if (NewHonorMember2 != "No Additional Member for Honor 2")
{
    HonorCounter += 1;
    if (HonorCounter <= 3) {
        if (teamSelected == "Team 1") {
            st12.executeUpdate();
            String sql26 = "INSERT INTO
TEAM1_MEMBERS SELECT*FROM UNASSIGNED_MEMBERS WHERE
NAME=?";
            String sql27 = "DELETE FROM
UNASSIGNED_MEMBERS WHERE NAME=?";
            PreparedStatement st26 =
conn.prepareStatement(sql26);
            PreparedStatement st27 =
conn.prepareStatement(sql27);
            st26.setString(1, NewHonorMember2);
            st27.setString(1, NewHonorMember2);
            st26.executeUpdate();
            st27.executeUpdate();
        } else if (teamSelected == "Team 2") {
            st12.executeUpdate();
            String sql28 = "INSERT INTO
TEAM2_MEMBERS SELECT*FROM UNASSIGNED_MEMBERS WHERE
NAME=?";
            String sql29 = "DELETE FROM
UNASSIGNED_MEMBERS WHERE NAME=?";
            PreparedStatement st28 =
conn.prepareStatement(sql28);
            PreparedStatement st29 =
conn.prepareStatement(sql29);

```

```

        st28.setString(1, NewHonorMember2);
        st29.setString(1, NewHonorMember2);
        st28.executeUpdate();
        st29.executeUpdate();
    } else if (teamSelected == "Team 3") {
        st12.executeUpdate();
        String sql30 = "INSERT INTO
TEAM3_MEMBERS SELECT*FROM UNASSIGNED_MEMBERS WHERE
NAME=?";

        String sql31 = "DELETE FROM
UNASSIGNED_MEMBERS WHERE NAME=?";
        PreparedStatement st30 =
conn.prepareStatement(sql30);
        PreparedStatement st31 =
conn.prepareStatement(sql31);

        st30.setString(1, NewHonorMember2);
        st31.setString(1, NewHonorMember2);
        st30.executeUpdate();
        st31.executeUpdate();
    }
    } else {
        JOptionPane.showMessageDialog(null, "Too many
members in Honor Group!!");
        successful = false;
    }
}

if (NewHonorMember3 != "No Additional Member for Honor 3")
{
    HonorCounter += 1;
    if (HonorCounter <= 3) {
        if (teamSelected == "Team 1") {
            st13.executeUpdate();
            String sql32 = "INSERT INTO
TEAM1_MEMBERS SELECT*FROM UNASSIGNED_MEMBERS WHERE
NAME=?";

            String sql33 = "DELETE FROM
UNASSIGNED_MEMBERS WHERE NAME=?";
            PreparedStatement st32 =
conn.prepareStatement(sql32);
            PreparedStatement st33 =
conn.prepareStatement(sql33);

            st32.setString(1, NewHonorMember3);

```



```

        st33.setString(1, NewHonorMember3);
        st32.executeUpdate();
        st33.executeUpdate();
    } else if (teamSelected == "Team 2") {
        st13.executeUpdate();
        String sql34 = "INSERT INTO
TEAM2_MEMBERS SELECT*FROM UNASSIGNED_MEMBERS WHERE
NAME=?";

        String sql35 = "DELETE FROM
UNASSIGNED_MEMBERS WHERE NAME=?";
        PreparedStatement st34 =
conn.prepareStatement(sql34);
        PreparedStatement st35 =
conn.prepareStatement(sql35);

        st34.setString(1, NewHonorMember3);
        st35.setString(1, NewHonorMember3);
        st34.executeUpdate();
        st35.executeUpdate();
    } else if (teamSelected == "Team 3") {
        st13.executeUpdate();
        String sql36 = "INSERT INTO
TEAM3_MEMBERS SELECT*FROM UNASSIGNED_MEMBERS WHERE
NAME=?";

        String sql37 = "DELETE FROM
UNASSIGNED_MEMBERS WHERE NAME=?";
        PreparedStatement st36 =
conn.prepareStatement(sql36);
        PreparedStatement st37 =
conn.prepareStatement(sql37);

        st36.setString(1, NewHonorMember3);
        st37.setString(1, NewHonorMember3);
        st36.executeUpdate();
        st37.executeUpdate();
    }
} else {
    JOptionPane.showMessageDialog(null, "Too many
members in Honor Group!!");
    successful = false;
}

}

if (NewScholasticMember1 != "No Additional Member for
Scholastic 1") {

```

```

        ScholasticCounter += 1;
        if (ScholasticCounter <= 3) {
            if (teamSelected == "Team 1") {
                st14.executeUpdate();
                String sql38 = "INSERT INTO
TEAM1_MEMBERS SELECT*FROM UNASSIGNED_MEMBERS WHERE
NAME=?";

                String sql39 = "DELETE FROM
UNASSIGNED_MEMBERS WHERE NAME=?";
                PreparedStatement st38 =
conn.prepareStatement(sql38);
                PreparedStatement st39 =
conn.prepareStatement(sql39);
                st38.setString(1, NewScholasticMember1);
                st39.setString(1, NewScholasticMember1);
                st38.executeUpdate();
                st39.executeUpdate();
            } else if (teamSelected == "Team 2") {
                st14.executeUpdate();
                String sql40 = "INSERT INTO
TEAM2_MEMBERS SELECT*FROM UNASSIGNED_MEMBERS WHERE
NAME=?";

                String sql41 = "DELETE FROM
UNASSIGNED_MEMBERS WHERE NAME=?";
                PreparedStatement st40 =
conn.prepareStatement(sql40);
                PreparedStatement st41 =
conn.prepareStatement(sql41);
                st40.setString(1, NewScholasticMember1);
                st41.setString(1, NewScholasticMember1);
                st40.executeUpdate();
                st41.executeUpdate();
            } else if (teamSelected == "Team 3") {
                st14.executeUpdate();
                String sql42 = "INSERT INTO
TEAM3_MEMBERS SELECT*FROM UNASSIGNED_MEMBERS WHERE
NAME=?";

                String sql43 = "DELETE FROM
UNASSIGNED_MEMBERS WHERE NAME=?";
                PreparedStatement st42 =
conn.prepareStatement(sql42);
                PreparedStatement st43 =
conn.prepareStatement(sql43);
                st42.setString(1, NewScholasticMember1);

```

```

        st43.setString(1, NewScholasticMember1);
        st42.executeUpdate();
        st43.executeUpdate();
    }
} else {
    JOptionPane.showMessageDialog(null, "Too many
members in Scholastic Group!!");
    successful = false;
}

}

if (NewScholasticMember2 != "No Additional Member for
Scholastic 2") {
    ScholasticCounter += 1;
    if (ScholasticCounter <= 3) {
        if (teamSelected == "Team 1") {
            st15.executeUpdate();
            String sql44 = "INSERT INTO
TEAM1_MEMBERS SELECT*FROM UNASSIGNED_MEMBERS WHERE
NAME=?";

            String sql45 = "DELETE FROM
UNASSIGNED_MEMBERS WHERE NAME=?";
            PreparedStatement st44 =
conn.prepareStatement(sql44);
            PreparedStatement st45 =
conn.prepareStatement(sql45);

            st44.setString(1, NewScholasticMember2);
            st45.setString(1, NewScholasticMember2);
            st44.executeUpdate();
            st45.executeUpdate();
        } else if (teamSelected == "Team 2") {
            st15.executeUpdate();
            String sql46 = "INSERT INTO
TEAM2_MEMBERS SELECT*FROM UNASSIGNED_MEMBERS WHERE
NAME=?";

            String sql47 = "DELETE FROM
UNASSIGNED_MEMBERS WHERE NAME=?";
            PreparedStatement st46 =
conn.prepareStatement(sql46);
            PreparedStatement st47 =
conn.prepareStatement(sql47);

            st46.setString(1, NewScholasticMember2);
            st47.setString(1, NewScholasticMember2);

```

```

        st46.executeUpdate();
        st47.executeUpdate();
    } else if (teamSelected == "Team 3") {
        st15.executeUpdate();
        String sql48 = "INSERT INTO
TEAM3_MEMBERS SELECT*FROM UNASSIGNED_MEMBERS WHERE
NAME=?";

        String sql49 = "DELETE FROM
UNASSIGNED_MEMBERS WHERE NAME=?";
        PreparedStatement st48 =
conn.prepareStatement(sql48);
        PreparedStatement st49 =
conn.prepareStatement(sql49);
        st48.setString(1, NewScholasticMember2);
        st49.setString(1, NewScholasticMember2);
        st48.executeUpdate();
        st49.executeUpdate();
    }
} else {
    JOptionPane.showMessageDialog(null, "Too many
members in Scholastic Group!!");
    successful = false;
}

}

if (NewScholasticMember3 != "No Additional Member for
Scholastic 3") {
    ScholasticCounter += 1;
    if (ScholasticCounter <= 3) {
        if (teamSelected == "Team 1") {
            st16.executeUpdate();
            String sql50 = "INSERT INTO
TEAM1_MEMBERS SELECT*FROM UNASSIGNED_MEMBERS WHERE
NAME=?";

            String sql51 = "DELETE FROM
UNASSIGNED_MEMBERS WHERE NAME=?";
            PreparedStatement st50 =
conn.prepareStatement(sql50);
            PreparedStatement st51 =
conn.prepareStatement(sql51);
            st50.setString(1, NewScholasticMember3);
            st51.setString(1, NewScholasticMember3);
            st50.executeUpdate();

```

```

        st51.executeUpdate();
    } else if (teamSelected == "Team 2") {
        st16.executeUpdate();
        String sql52 = "INSERT INTO
TEAM2_MEMBERS SELECT*FROM UNASSIGNED_MEMBERS WHERE
NAME=?";

        String sql53 = "DELETE FROM
UNASSIGNED_MEMBERS WHERE NAME=?";
        PreparedStatement st52 =
conn.prepareStatement(sql52);
        PreparedStatement st53 =
conn.prepareStatement(sql53);

        st52.setString(1, NewScholasticMember3);
        st53.setString(1, NewScholasticMember3);
        st52.executeUpdate();
        st53.executeUpdate();
    } else if (teamSelected == "Team 3") {
        st16.executeUpdate();
        String sql54 = "INSERT INTO
TEAM3_MEMBERS SELECT*FROM UNASSIGNED_MEMBERS WHERE
NAME=?";

        String sql55 = "DELETE FROM
UNASSIGNED_MEMBERS WHERE NAME=?";
        PreparedStatement st54 =
conn.prepareStatement(sql54);
        PreparedStatement st55 =
conn.prepareStatement(sql55);

        st54.setString(1, NewScholasticMember3);
        st55.setString(1, NewScholasticMember3);
        st54.executeUpdate();
        st55.executeUpdate();
    }
} else {
    JOptionPane.showMessageDialog(null, "Too many
members in Scholastic Group!!");
    successful = false;
}

}

if (NewVarsityMember1 != "No Additional Member for Varsity
1") {
    VarsityCounter += 1;
    if (VarsityCounter <= 3) {

```

```

        if (teamSelected == "Team 1") {
            st17.executeUpdate();
            String sql56 = "INSERT INTO
TEAM1_MEMBERS SELECT*FROM UNASSIGNED_MEMBERS WHERE
NAME=?";

            String sql57 = "DELETE FROM
UNASSIGNED_MEMBERS WHERE NAME=?";
            PreparedStatement st56 =
conn.prepareStatement(sql56);
            PreparedStatement st57 =
conn.prepareStatement(sql57);

            st56.setString(1, NewVarsityMember1);
            st57.setString(1, NewVarsityMember1);
            st56.executeUpdate();
            st57.executeUpdate();
        } else if (teamSelected == "Team 2") {
            st17.executeUpdate();
            String sql58 = "INSERT INTO
TEAM2_MEMBERS SELECT*FROM UNASSIGNED_MEMBERS WHERE
NAME=?";

            String sql59 = "DELETE FROM
UNASSIGNED_MEMBERS WHERE NAME=?";
            PreparedStatement st58 =
conn.prepareStatement(sql58);
            PreparedStatement st59 =
conn.prepareStatement(sql59);

            st58.setString(1, NewVarsityMember1);
            st59.setString(1, NewVarsityMember1);
            st58.executeUpdate();
            st59.executeUpdate();
        } else if (teamSelected == "Team 3") {
            st17.executeUpdate();
            String sql60 = "INSERT INTO
TEAM3_MEMBERS SELECT*FROM UNASSIGNED_MEMBERS WHERE
NAME=?";

            String sql61 = "DELETE FROM
UNASSIGNED_MEMBERS WHERE NAME=?";
            PreparedStatement st60 =
conn.prepareStatement(sql60);
            PreparedStatement st61 =
conn.prepareStatement(sql61);

            st60.setString(1, NewVarsityMember1);
            st61.setString(1, NewVarsityMember1);
            st60.executeUpdate();

```

```

        st61.executeUpdate();
    }
} else {
    JOptionPane.showMessageDialog(null, "Too many
members in Varsity Group!!");
    successful = false;
}

}

if (NewVarsityMember2 != "No Additional Member for Varsity
2") {
    VarsityCounter += 1;
    if (VarsityCounter <= 3) {
        if (teamSelected == "Team 1") {
            st18.executeUpdate();
            String sql63 = "INSERT INTO
TEAM1_MEMBERS SELECT*FROM UNASSIGNED_MEMBERS WHERE
NAME=?";
            String sql64 = "DELETE FROM
UNASSIGNED_MEMBERS WHERE NAME=?";
            PreparedStatement st63 =
conn.prepareStatement(sql63);
            PreparedStatement st64 =
conn.prepareStatement(sql64);
            st63.setString(1, NewVarsityMember2);
            st64.setString(1, NewVarsityMember2);
            st63.executeUpdate();
            st64.executeUpdate();
        } else if (teamSelected == "Team 2") {
            st18.executeUpdate();
            String sql65 = "INSERT INTO
TEAM2_MEMBERS SELECT*FROM UNASSIGNED_MEMBERS WHERE
NAME=?";
            String sql66 = "DELETE FROM
UNASSIGNED_MEMBERS WHERE NAME=?";
            PreparedStatement st65 =
conn.prepareStatement(sql65);
            PreparedStatement st66 =
conn.prepareStatement(sql66);
            st65.setString(1, NewVarsityMember2);
            st66.setString(1, NewVarsityMember2);
            st65.executeUpdate();
            st66.executeUpdate();
        }
    }
}

```

```

        } else if (teamSelected == "Team 3") {
            st18.executeUpdate();
            String sql67 = "INSERT INTO
TEAM3_MEMBERS SELECT*FROM UNASSIGNED_MEMBERS WHERE
NAME=?";
            String sql68 = "DELETE FROM
UNASSIGNED_MEMBERS WHERE NAME=?";
            PreparedStatement st67 =
conn.prepareStatement(sql67);
            PreparedStatement st68 =
conn.prepareStatement(sql68);
            st67.setString(1, NewVarsityMember2);
            st68.setString(1, NewVarsityMember2);
            st67.executeUpdate();
            st68.executeUpdate();
        }
    } else {
        JOptionPane.showMessageDialog(null, "Too many
members in Varsity Group!!");
        successful = false;
    }
}

if (NewVarsityMember3 != "No Additional Member for Varsity
3") {
    VarsityCounter += 1;
    if (VarsityCounter <= 3) {
        if (teamSelected == "Team 1") {
            st19.executeUpdate();
            String sql69 = "INSERT INTO
TEAM1_MEMBERS SELECT*FROM UNASSIGNED_MEMBERS WHERE
NAME=?";
            String sql70 = "DELETE FROM
UNASSIGNED_MEMBERS WHERE NAME=?";
            PreparedStatement st69 =
conn.prepareStatement(sql69);
            PreparedStatement st70 =
conn.prepareStatement(sql70);
            st69.setString(1, NewVarsityMember3);
            st70.setString(1, NewVarsityMember3);
            st69.executeUpdate();
            st70.executeUpdate();
        } else if (teamSelected == "Team 2") {

```



```

        st19.executeUpdate();
        String sql71 = "INSERT INTO
TEAM2_MEMBERS SELECT*FROM UNASSIGNED_MEMBERS WHERE
NAME=?";

        String sql72 = "DELETE FROM
UNASSIGNED_MEMBERS WHERE NAME=?";
        PreparedStatement st71 =
conn.prepareStatement(sql71);
        PreparedStatement st72 =
conn.prepareStatement(sql72);

        st71.setString(1, NewVarsityMember3);
        st72.setString(1, NewVarsityMember3);
        st71.executeUpdate();
        st72.executeUpdate();
    } else if (teamSelected == "Team 3") {
        st19.executeUpdate();
        String sql73 = "INSERT INTO
TEAM3_MEMBERS SELECT*FROM UNASSIGNED_MEMBERS WHERE
NAME=?";

        String sql74 = "DELETE FROM
UNASSIGNED_MEMBERS WHERE NAME=?";
        PreparedStatement st73 =
conn.prepareStatement(sql73);
        PreparedStatement st74 =
conn.prepareStatement(sql74);

        st73.setString(1, NewVarsityMember3);
        st74.setString(1, NewVarsityMember3);
        st73.executeUpdate();
        st74.executeUpdate();
    }
} else {
    JOptionPane.showMessageDialog(null, "Too many
members in Varsity Group!!");
    successful = false;
}

}
if(successful == true){
    JOptionPane.showMessageDialog(null, "Updated
Successfully!");
    new AdminWindow().setVisible(true);
    dispose();
}

```

```

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

/**
 * @param args the command line arguments
 */
public static void main(String args[]) {
    /* Set the Nimbus look and feel */
    //<editor-fold defaultstate="collapsed" desc=" Look and feel setting code
(optional) ">
    /* If Nimbus (introduced in Java SE 6) is not available, stay with the default
look and feel.
    * For details see
http://download.oracle.com/javase/tutorial/uiswing/lookandfeel/plaf.html
    */
    try {
        for (javax.swing.UIManager.LookAndFeelInfo info :
javax.swing.UIManager.getInstalledLookAndFeels()) {
            if ("Nimbus".equals(info.getName())) {

javax.swing.UIManager.setLookAndFeel(info.getClassName());
                break;
            }
        }
    } catch (ClassNotFoundException ex) {

java.util.logging.Logger.getLogger(AssignTeams.class.getName()).log(java.util.loggi
ng.Level.SEVERE, null, ex);
    } catch (InstantiationException ex) {

java.util.logging.Logger.getLogger(AssignTeams.class.getName()).log(java.util.loggi
ng.Level.SEVERE, null, ex);
    } catch (IllegalAccessException ex) {

java.util.logging.Logger.getLogger(AssignTeams.class.getName()).log(java.util.loggi
ng.Level.SEVERE, null, ex);
    } catch (javax.swing.UnsupportedLookAndFeelException ex) {

java.util.logging.Logger.getLogger(AssignTeams.class.getName()).log(java.util.loggi
ng.Level.SEVERE, null, ex);

```

```

    }
    //</editor-fold>

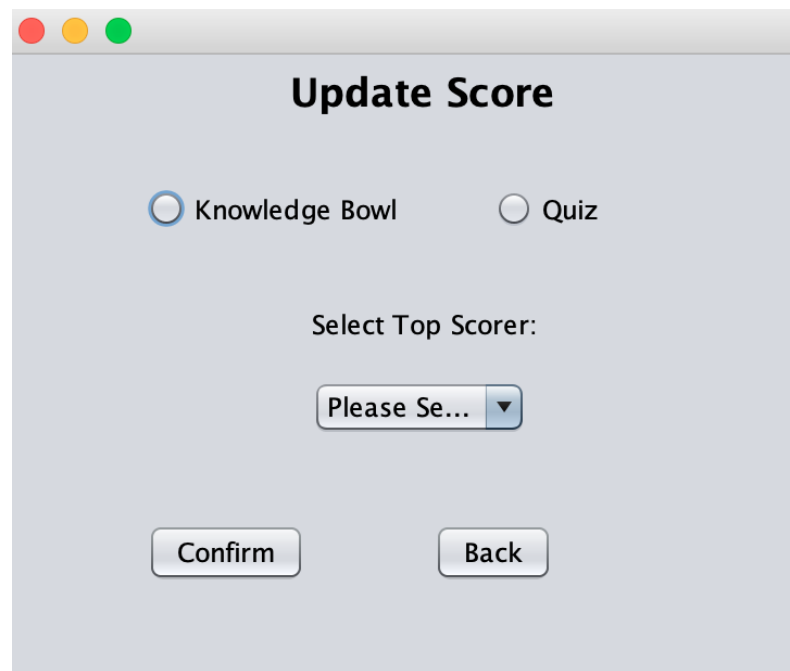
    /* Create and display the form */
    java.awt.EventQueue.invokeLater(new Runnable() {
        public void run() {
            new AssignTeams().setVisible(true);
        }
    });
}

// Variables declaration - do not modify
private javax.swing.JButton GetBackButton;
private javax.swing.JComboBox<String> HonorGroupBox1;
private javax.swing.JComboBox<String> HonorGroupBox2;
private javax.swing.JComboBox<String> HonorGroupBox3;
private javax.swing.JComboBox<String> HonorGroupButton1;
private javax.swing.JComboBox<String> HonorGroupButton2;
private javax.swing.JTable HonorTable;
private javax.swing.JButton SaveChangesButton;
private javax.swing.JComboBox<String> ScholasticGroupBox1;
private javax.swing.JComboBox<String> ScholasticGroupBox2;
private javax.swing.JComboBox<String> ScholasticGroupBox3;
private javax.swing.JTable ScholasticTable;
private javax.swing.JComboBox<String> TeamBox;
private javax.swing.JComboBox<String> VarsityGroupBox1;
private javax.swing.JComboBox<String> VarsityGroupBox2;
private javax.swing.JComboBox<String> VarsityGroupBox3;
private javax.swing.JTable VarsityTable;
private javax.swing.JButton ViewButton;
private javax.swing.JLabel jLabel1;
private javax.swing.JLabel jLabel2;
private javax.swing.JLabel jLabel3;
private javax.swing.JLabel jLabel4;
private javax.swing.JLabel jLabel5;
private javax.swing.JLabel jLabel6;
private javax.swing.JLabel jLabel7;
private javax.swing.JLabel jLabel8;
private javax.swing.JList<String> jList1;
private javax.swing.JProgressBar jProgressBar1;
private javax.swing.JScrollPane jScrollPane1;
private javax.swing.JScrollPane jScrollPane2;
private javax.swing.JScrollPane jScrollPane3;
private javax.swing.JScrollPane jScrollPane4;

```

```
private javax.swing.JTextField jTextField1;  
// End of variables declaration  
}
```

## Update Score



```
import java.awt.Dimension;
import java.awt.Toolkit;
import java.sql.DriverManager;
import java.sql.PreparedStatement;
import java.sql.ResultSet;
import java.sql.SQLException;
import java.util.logging.Level;
import java.util.logging.Logger;
import javax.swing.JOptionPane;
```

```
/*
 * To change this license header, choose License Headers in Project Properties.
 * To change this template file, choose Tools | Templates
 * and open the template in the editor.
 */
```

```
/**
 *
 * @author aidenyan
 */
```

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

```
    /**
```

```
     * Creates new form UpdateScore
```

```
     */
```

```
    public UpdateScore() {
```

```

        initComponents();
        updateCombo();
        Toolkit toolkit = getToolkit();
        Dimension size = toolkit.getScreenSize();
        setLocation(size.width / 2 - getWidth() / 2, size.height / 2 - getHeight() / 2);
//center the GUI
    }

    private void updateCombo() {
//method to update the ComboBox selection
        try {
            String connectionURL =
"jdbc:derby://localhost:1527/QHAPPYDatabase";

            java.sql.Connection conn =
DriverManager.getConnection(connectionURL, "username", "password");

            String sql1 = "SELECT * from TEAM_LEADER_LOGIN";
            String sql2 = "SELECT * from TEAM_MEMBER_LOGIN";
//get all members in the club except for the administrator (president),
put them into the ComboBox
            PreparedStatement st1 = conn.prepareStatement(sql1);
            PreparedStatement st2 = conn.prepareStatement(sql2);
            ResultSet rs1 = st1.executeQuery();
            ResultSet rs2 = st2.executeQuery();
            while (rs1.next()) {
                TopMemberBox.addItem(rs1.getString("NAME"));
            }
            while (rs2.next()) {
                TopMemberBox.addItem(rs2.getString("NAME"));
            }
        } catch (SQLException e) {
            JOptionPane.showMessageDialog(null, e);
        }
    }

/**
 * This method is called from within the constructor to initialize the form.
 * WARNING: Do NOT modify this code. The content of this method is always
 * regenerated by the Form Editor.
 */
@SuppressWarnings("unchecked")
// <editor-fold defaultstate="collapsed" desc="Generated Code">
private void initComponents() {

```

```

buttonGroup1 = new javax.swing.ButtonGroup();
buttonGroup2 = new javax.swing.ButtonGroup();
jLabel1 = new javax.swing.JLabel();
BowlButton = new javax.swing.JRadioButton();
QuizButton = new javax.swing.JRadioButton();
jLabel2 = new javax.swing.JLabel();
TopMemberBox = new javax.swing.JComboBox<>();
ConfirmButton = new javax.swing.JButton();
GetBackButton = new javax.swing.JButton();

setDefaultCloseOperation(javax.swing.WindowConstants.EXIT_ON_CLOSE);

jLabel1.setFont(new java.awt.Font("Lucida Grande", 1, 18)); // NOI18N
jLabel1.setText("Update Score");

buttonGroup1.add(BowlButton);
BowlButton.setText("Knowledge Bowl");

buttonGroup1.add(QuizButton);
QuizButton.setText("Quiz");

jLabel2.setText("Select Top Scorer:");

TopMemberBox.setModel(new
javax.swing.DefaultComboBoxModel<>(new String[] { "Please Select" }));

ConfirmButton.setText("Confirm");
ConfirmButton.addActionListener(new java.awt.event.ActionListener() {
    public void actionPerformed(java.awt.event.ActionEvent evt) {
        ConfirmButtonActionPerformed(evt);
    }
});

GetBackButton.setText("Back");
GetBackButton.addActionListener(new java.awt.event.ActionListener() {
    public void actionPerformed(java.awt.event.ActionEvent evt) {
        GetBackButtonActionPerformed(evt);
    }
});

javax.swing.GroupLayout layout = new
javax.swing.GroupLayout(getContentPane());

```





```

        .addComponent(jLabel2)
        .addGap(18, 18, 18)
        .addComponent(TopMemberBox,
javax.swing.GroupLayout.PREFERRED_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE,
javax.swing.GroupLayout.PREFERRED_SIZE)
        .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement
.RELATED, 41, Short.MAX_VALUE)
        .addGroup(layout.createParallelGroup(javax.swing.GroupLayout.
Alignment.BASELINE)
        .addComponent(ConfirmButton)
        .addComponent(GetBackButton))
        .addGap(47, 47, 47))
    );

    pack();
} // </editor-fold>

```

```

private void ConfirmButtonActionPerformed(java.awt.event.ActionEvent evt) {

    try {
        String connectionURL =
"jdbc:derby://localhost:1527/QHAPPYDatabase";
        java.sql.Connection conn =
DriverManager.getConnection(connectionURL, "username", "password");
        String theUpdatedOne = (String) TopMemberBox.getSelectedItemAt();

        //ensures that the user selects a top performer to add the points to
        if(theUpdatedOne.equals("Please Select")){
            JOptionPane.showMessageDialog(null, "Please Select a
Member!");
        } else if (BowlButton.isSelected() == true) {
            String sql3 = "UPDATE TEAM1_MEMBERS SET POINTS =
POINTS+8 WHERE NAME=?";
            String sql4 = "UPDATE TEAM2_MEMBERS SET POINTS =
POINTS+8 WHERE NAME=?";
            String sql5 = "UPDATE TEAM3_MEMBERS SET POINTS =
POINTS+8 WHERE NAME=?";
            String sql6 = "UPDATE UNASSIGNED_MEMBERS SET
POINTS = POINTS+8 WHERE NAME=?";
            //find the corresponding members in the databases, and then adds
his/her points correspondingly

```

```

        PreparedStatement st3 = conn.prepareStatement(sql3);

```

```

PreparedStatement st4 = conn.prepareStatement(sql4);
PreparedStatement st5 = conn.prepareStatement(sql5);
PreparedStatement st6 = conn.prepareStatement(sql6);

st3.setString(1, theUpdatedOne);
st4.setString(1, theUpdatedOne);
st5.setString(1, theUpdatedOne);
st6.setString(1, theUpdatedOne);

st3.executeUpdate();
st4.executeUpdate();
st5.executeUpdate();
st6.executeUpdate();
JOptionPane.showMessageDialog(null, "Update Successfully");
new AdminWindow().setVisible(true);
dispose();

} else if (QuizButton.isSelected() == true) {
    //same logics with above, but changed to 3 points
    String sql7 = "UPDATE TEAM1_MEMBERS SET POINTS =
POINTS+3 WHERE NAME=?";
    String sql8 = "UPDATE TEAM2_MEMBERS SET POINTS =
POINTS+3 WHERE NAME=?";
    String sql9 = "UPDATE TEAM3_MEMBERS SET POINTS =
POINTS+3 WHERE NAME=?";
    String sql10 = "UPDATE UNASSIGNED_MEMBERS SET
POINTS = POINTS+3 WHERE NAME=?";

    PreparedStatement st7 = conn.prepareStatement(sql7);
    PreparedStatement st8 = conn.prepareStatement(sql8);
    PreparedStatement st9 = conn.prepareStatement(sql9);
    PreparedStatement st10 = conn.prepareStatement(sql10);

    st7.setString(1, theUpdatedOne);
    st8.setString(1, theUpdatedOne);
    st9.setString(1, theUpdatedOne);
    st10.setString(1, theUpdatedOne);

    st7.executeUpdate();
    st8.executeUpdate();
    st9.executeUpdate();
    st10.executeUpdate();
    JOptionPane.showMessageDialog(null, "Update Successfully");
}

```

```

        new AdminWindow().setVisible(true);
        dispose();
    } else {
        JOptionPane.showMessageDialog(null, "Select a Kind of
Activity!");
        //the "else" here is when the user has not made a selection in the
jButtonGroup
    }
    } catch (SQLException e) {
        JOptionPane.showMessageDialog(null, e);
    }
}

private void GetBackButtonActionPerformed(java.awt.event.ActionEvent evt) {
    new AdminWindow().setVisible(true);
    dispose();
}

/**
 * @param args the command line arguments
 */
public static void main(String args[]) {
    /* Set the Nimbus look and feel */
    //<editor-fold defaultstate="collapsed" desc=" Look and feel setting code
(optional) ">
    /* If Nimbus (introduced in Java SE 6) is not available, stay with the default
look and feel.
    * For details see
http://download.oracle.com/javase/tutorial/uiswing/lookandfeel/plaf.html
    */
    try {
        for (javax.swing.UIManager.LookAndFeelInfo info :
javax.swing.UIManager.getInstalledLookAndFeels()) {
            if ("Nimbus".equals(info.getName())) {

javax.swing.UIManager.setLookAndFeel(info.getClassName());
                break;

            }
        }
    } catch (ClassNotFoundException ex) {
        java.util.logging.Logger.getLogger(UpdateScore.class

```

```

        .getName()).log(java.util.logging.Level.SEVERE, null, ex);

    } catch (InstantiationException ex) {
        java.util.logging.Logger.getLogger(UpdateScore.class
            .getName()).log(java.util.logging.Level.SEVERE, null, ex);

    } catch (IllegalAccessException ex) {
        java.util.logging.Logger.getLogger(UpdateScore.class
            .getName()).log(java.util.logging.Level.SEVERE, null, ex);

    } catch (javax.swing.UnsupportedLookAndFeelException ex) {
        java.util.logging.Logger.getLogger(UpdateScore.class
            .getName()).log(java.util.logging.Level.SEVERE, null, ex);
    }
}
//</editor-fold>

/* Create and display the form */
java.awt.EventQueue.invokeLater(new Runnable() {
    public void run() {
        new UpdateScore().setVisible(true);
    }
});
}

// Variables declaration - do not modify
private javax.swing.JRadioButton BowlButton;
private javax.swing.JButton ConfirmButton;
private javax.swing.JButton GetBackButton;
private javax.swing.JRadioButton QuizButton;
private javax.swing.JComboBox<String> TopMemberBox;
private javax.swing.ButtonGroup buttonGroup1;
private javax.swing.ButtonGroup buttonGroup2;
private javax.swing.JLabel jLabel1;
private javax.swing.JLabel jLabel2;
// End of variables declaration
}

```

## View Rankings



```
import java.awt.Dimension;  
import java.awt.Toolkit;  
import javax.swing.JOptionPane;
```

```
/*  
 * To change this license header, choose License Headers in Project Properties.  
 * To change this template file, choose Tools | Templates  
 * and open the template in the editor.  
 */
```

```
/**  
 *  
 * @author aidenyan  
 */
```

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

```
    /**
```

```
     * Creates new form ViewRankings
```

```
    */
```

```
    public ViewRankings() {  
        initComponents();  
        Toolkit toolkit = getToolkit();  
        Dimension size = toolkit.getScreenSize();  
        setLocation(size.width / 2 - getWidth() / 2, size.height / 2 - getHeight() / 2);  
    }
```

```
    /**
```

```

* This method is called from within the constructor to initialize the form.
* WARNING: Do NOT modify this code. The content of this method is always
* regenerated by the Form Editor.
*/
@SuppressWarnings("unchecked")
// <editor-fold defaultstate="collapsed" desc="Generated Code">
private void initComponents() {

    buttonGroup1 = new javax.swing.ButtonGroup();
    jLabel1 = new javax.swing.JLabel();
    TeamRankButton = new javax.swing.JRadioButton();
    ClubRankButton = new javax.swing.JRadioButton();
    jLabel2 = new javax.swing.JLabel();
    TeamBox = new javax.swing.JComboBox<>();
    AccessButton = new javax.swing.JButton();
    GetBackButton = new javax.swing.JButton();

    setDefaultCloseOperation(javax.swing.WindowConstants.EXIT_ON_CLOSE);

    jLabel1.setFont(new java.awt.Font("Lucida Grande", 1, 14)); // NOI18N
    jLabel1.setText("View Rankings");

    buttonGroup1.add(TeamRankButton);
    TeamRankButton.setText("Team Ranking");

    buttonGroup1.add(ClubRankButton);
    ClubRankButton.setText("Club Ranking");

    jLabel2.setText("Select Team:");

    TeamBox.setModel(new javax.swing.DefaultComboBoxModel<>(new
String[] { "Please Select", "Team 1", "Team 2", "Team 3" }));

    AccessButton.setText("Access");
    AccessButton.addActionListener(new java.awt.event.ActionListener() {
        public void actionPerformed(java.awt.event.ActionEvent evt) {
            AccessButtonActionPerformed(evt);
        }
    });

    GetBackButton.setText("Back");
    GetBackButton.addActionListener(new java.awt.event.ActionListener() {
        public void actionPerformed(java.awt.event.ActionEvent evt) {

```

```

        GetBackButtonActionPerformed(evt);
    }
});

    javax.swing.GroupLayout layout = new
javax.swing.GroupLayout(getContentPane());
    getContentPane().setLayout(layout);
    layout.setHorizontalGroup(

layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
        .addGroup(layout.createSequentialGroup()
            .addGroup(layout.createParallelGroup(javax.swing.GroupLayout.
Alignment.LEADING)
                .addGroup(layout.createSequentialGroup()
                    .addGap(153, 153, 153)
                    .addComponent(jLabel1))
                .addGroup(layout.createSequentialGroup()
                    .addGap(79, 79, 79)
                    .addComponent(TeamRankButton)
                    .addGap(18, 18, 18)
                    .addComponent(ClubRankButton))
                .addGroup(layout.createSequentialGroup()
                    .addGap(103, 103, 103)
                    .addComponent(jLabel2)
                    .addPreferredGap(javax.swing.LayoutStyle.Component
Placement.UNRELATED)
                    .addComponent(TeamBox,
javax.swing.GroupLayout.PREFERRED_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE,
javax.swing.GroupLayout.PREFERRED_SIZE)))
                .addContainerGap(66, Short.MAX_VALUE))
            .addGroup(layout.createSequentialGroup()
                .addContainerGap(javax.swing.GroupLayout.DEFAULT_SIZE,
Short.MAX_VALUE)
                .addComponent(AccessButton)
                .addGap(62, 62, 62)
                .addComponent(GetBackButton)
                .addGap(87, 87, 87))
        );
    layout.setVerticalGroup(

layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
        .addGroup(layout.createSequentialGroup()
            .addComponent(jLabel1)

```

```

        .addGap(28, 28, 28)
        .addGroup(layout.createParallelGroup(javax.swing.GroupLayout.
Alignment.BASELINE)
            .addComponent(TeamRankButton)
            .addComponent(ClubRankButton))
        .addGap(51, 51, 51)
        .addGroup(layout.createParallelGroup(javax.swing.GroupLayout.
Alignment.BASELINE)
            .addComponent(jLabel2)
            .addComponent(TeamBox,
javax.swing.GroupLayout.PREFERRED_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE,
javax.swing.GroupLayout.PREFERRED_SIZE))
        .addGap(33, 33, 33)
        .addGroup(layout.createParallelGroup(javax.swing.GroupLayout.
Alignment.BASELINE)
            .addComponent(AccessButton)
            .addComponent(GetBackButton))
        .addGap(0, 47, Short.MAX_VALUE))
    );

```

```

    pack();
} // </editor-fold>

```

```

private void AccessButtonActionPerformed(java.awt.event.ActionEvent evt) {
    if (TeamRankButton.isSelected() == true) {
        String currentSelect= (String) TeamBox.getSelectedItemAt();
        if(currentSelect.equals("Please Select")){
            JOptionPane.showMessageDialog(null, "Please Select a Team!");
            //let users select if they have yet to make a selection
        } else{
            //define the new teamUsed static variable
            LoginWindow.teamUsed = currentSelect;
        }
        //dispose to the team ranking UI
        new TeamRanking().setVisible(true);
        dispose();

    } else if (ClubRankButton.isSelected() == true) {
        //dispose to the club ranking UI
        new ClubRanking().setVisible(true);
        dispose();
    } else{
        //if have not select a kind of ranking, then output an error message
    }
}

```



```

        JOptionPane.showMessageDialog(null, "Please Select a Kind of
Ranking!");
    }
}

private void GetBackButtonActionPerformed(java.awt.event.ActionEvent evt) {
    new AdminWindow().setVisible(true);
    dispose();
}

/**
 * @param args the command line arguments
 */
public static void main(String args[]) {
    /* Set the Nimbus look and feel */
    //<editor-fold defaultstate="collapsed" desc=" Look and feel setting code
(optional) ">
    /* If Nimbus (introduced in Java SE 6) is not available, stay with the default
look and feel.
    * For details see
http://download.oracle.com/javase/tutorial/uiswing/lookandfeel/plaf.html
    */
    try {
        for (javax.swing.UIManager.LookAndFeelInfo info :
javax.swing.UIManager.getInstalledLookAndFeels()) {
            if ("Nimbus".equals(info.getName())) {

javax.swing.UIManager.setLookAndFeel(info.getClassName());
                break;
            }
        }
    } catch (ClassNotFoundException ex) {

java.util.logging.Logger.getLogger(ViewRankings.class.getName()).log(java.util.logg
ing.Level.SEVERE, null, ex);
    } catch (InstantiationException ex) {

java.util.logging.Logger.getLogger(ViewRankings.class.getName()).log(java.util.logg
ing.Level.SEVERE, null, ex);
    } catch (IllegalAccessException ex) {

java.util.logging.Logger.getLogger(ViewRankings.class.getName()).log(java.util.logg
ing.Level.SEVERE, null, ex);
    } catch (javax.swing.UnsupportedLookAndFeelException ex) {

```

```

java.util.logging.Logger.getLogger(ViewRankings.class.getName()).log(java.util.logging.Level.SEVERE, null, ex);
    }
    //</editor-fold>

    /* Create and display the form */
    java.awt.EventQueue.invokeLater(new Runnable() {
        public void run() {
            new ViewRankings().setVisible(true);
        }
    });
}

// Variables declaration - do not modify
private javax.swing.JButton AccessButton;
private javax.swing.JRadioButton ClubRankButton;
private javax.swing.JButton GetBackButton;
private javax.swing.JComboBox<String> TeamBox;
private javax.swing.JRadioButton TeamRankButton;
private javax.swing.ButtonGroup buttonGroup1;
private javax.swing.JLabel jLabel1;
private javax.swing.JLabel jLabel2;
// End of variables declaration
}

```

## Team Ranking

**TEAM RANKINGS**

1. Progress Ranking

| No. | Names          | Overall Progress |
|-----|----------------|------------------|
| 1   | Leo Wang       | 21               |
| 2   | Anderson Huang | 12               |
| 3   | Cherry Chen    | 2                |
| 4   | Gary           | 2                |

2. Points Ranking

| No. | Names          | Overall Progress |
|-----|----------------|------------------|
| 1   | Leo Wang       | 6                |
| 2   | Cherry Chen    | 5                |
| 3   | Anderson Huang | 1                |
| 4   | Gary           | 0                |

Back

```
import java.awt.Dimension;
import java.awt.Toolkit;
import java.sql.DriverManager;
import java.sql.PreparedStatement;
import java.sql.ResultSet;
import java.sql.SQLException;
import javax.swing.JOptionPane;
import javax.swing.table.DefaultTableModel;
```

```
/*
 * To change this license header, choose License Headers in Project Properties.
 * To change this template file, choose Tools | Templates
 * and open the template in the editor.
 */
/**
 *
 * @author aidenyan
 */
public class TeamRanking extends javax.swing.JFrame {
```

```
/**
```

```
 * Creates new form TeamRanking
```

```
*/
```

```

public TeamRanking() {
    initComponents();
    Toolkit toolkit = getToolkit();
    Dimension size = toolkit.getScreenSize();
    setLocation(size.width / 2 - getWidth() / 2, size.height / 2 - getHeight() / 2);
    compriseRankings();
}

private void compriseRankings() {
    //this is a method automatically initialized in the constructor
    try {
        String connectionURL =
"jdbc:derby://localhost:1527/QHAPPYDatabase";
        java.sql.Connection conn =
DriverManager.getConnection(connectionURL, "username", "password");

        /*treat the different teams through the chosen teamUsed
the members of the team could only access their team,
while the admin could go to different teams' rankings
*/
        if (LoginWindow.teamUsed.equals("Team 1")) {
            int pointCount = 0;
            int progressCount = 0;
            DefaultTableModel progressTable = (DefaultTableModel)
ProgressRankTable.getModel();
            DefaultTableModel pointTable = (DefaultTableModel)
PointsRankTable.getModel();

            //get the points/overall progress in descending order, so that it
could become a rank
            String sql1 = "SELECT * FROM TEAM1_MEMBERS ORDER
BY POINTS desc";
            String sql2 = "SELECT * FROM TEAM1_MEMBERS ORDER
BY OVERALL_PROGRESS desc";
            PreparedStatement st1 = conn.prepareStatement(sql1);
            PreparedStatement st2 = conn.prepareStatement(sql2);
            ResultSet rs1 = st1.executeQuery();
            ResultSet rs2 = st2.executeQuery();
            //displays the two kinds of rankings automatically
            while (rs1.next()) {
                pointCount += 1;
                String specificRank = String.valueOf(pointCount);
                String theName = rs1.getString("NAME");
                String thePoint = String.valueOf(rs1.getInt("POINTS"));

```

```

        String tbData[] = {specificRank, theName, thePoint};
        pointTable.addRow(tbData);
    }
    while (rs2.next()){
        progressCount += 1;
        String specificRank = String.valueOf(progressCount);
        String theName = rs2.getString("NAME");
        String theProgress =
String.valueOf(rs2.getInt("OVERALL_PROGRESS"));
        String tbData[] = {specificRank, theName, theProgress};
        progressTable.addRow(tbData);
    }

```

//the following codes perform the same ability with the code

above

```

    }else if (LoginWindow.teamUsed.equals("Team 2")){
        int pointCount = 0;
        int progressCount = 0;
        DefaultTableModel progressTable = (DefaultTableModel)
ProgressRankTable.getModel();
        DefaultTableModel pointTable = (DefaultTableModel)
PointsRankTable.getModel();
        String sql3 = "SELECT * FROM TEAM2_MEMBERS ORDER
BY POINTS desc";
        String sql4 = "SELECT * FROM TEAM2_MEMBERS ORDER
BY OVERALL_PROGRESS desc";
        PreparedStatement st3 = conn.prepareStatement(sql3);
        PreparedStatement st4 = conn.prepareStatement(sql4);
        ResultSet rs3 = st3.executeQuery();
        ResultSet rs4 = st4.executeQuery();
        while (rs3.next()) {
            pointCount += 1;
            String specificRank = String.valueOf(pointCount);
            String theName = rs3.getString("NAME");
            String thePoint = String.valueOf(rs3.getInt("POINTS"));
            String tbData[] = {specificRank, theName, thePoint};
            pointTable.addRow(tbData);
        }
        while (rs4.next()){
            progressCount += 1;
            String specificRank = String.valueOf(progressCount);
            String theName = rs4.getString("NAME");
            String theProgress =
String.valueOf(rs4.getInt("OVERALL_PROGRESS"));
            String tbData[] = {specificRank, theName, theProgress};

```

```

        progressTable.addRow(tbData);
    }
} else if(LoginWindow.teamUsed.equals("Team 3")){
    int pointCount = 0;
    int progressCount = 0;
    DefaultTableModel progressTable = (DefaultTableModel)
ProgressRankTable.getModel();
    DefaultTableModel pointTable = (DefaultTableModel)
PointsRankTable.getModel();
    String sql5 = "SELECT * FROM TEAM1_MEMBERS ORDER
BY POINTS desc";
    String sql6 = "SELECT * FROM TEAM1_MEMBERS ORDER
BY OVERALL_PROGRESS desc";
    PreparedStatement st5 = conn.prepareStatement(sql5);
    PreparedStatement st6 = conn.prepareStatement(sql6);
    ResultSet rs5 = st5.executeQuery();
    ResultSet rs6 = st6.executeQuery();
    while (rs5.next()) {
        pointCount += 1;
        String specificRank = String.valueOf(pointCount);
        String theName = rs5.getString("NAME");
        String thePoint = String.valueOf(rs5.getInt("POINTS"));
        String tbData[] = {specificRank, theName, thePoint};
        pointTable.addRow(tbData);
    }
    while (rs6.next()){
        progressCount += 1;
        String specificRank = String.valueOf(progressCount);
        String theName = rs6.getString("NAME");
        String theProgress =
String.valueOf(rs6.getInt("OVERALL_PROGRESS"));
        String tbData[] = {specificRank, theName, theProgress};
        progressTable.addRow(tbData);
    }
}
} catch (SQLException e) {
    JOptionPane.showMessageDialog(null, "Unsuccessful");
}

}

/**
 * This method is called from within the constructor to initialize the form.
 * WARNING: Do NOT modify this code. The content of this method is always

```

```

    * regenerated by the Form Editor.
    */
    @SuppressWarnings("unchecked")
    // <editor-fold defaultstate="collapsed" desc="Generated Code">
    private void initComponents() {

        jProgressBar1 = new javax.swing.JProgressBar();
        jLabel1 = new javax.swing.JLabel();
        jLabel2 = new javax.swing.JLabel();
        jLabel3 = new javax.swing.JLabel();
        jScrollPane1 = new javax.swing.JScrollPane();
        PointsRankTable = new javax.swing.JTable();
        jScrollPane2 = new javax.swing.JScrollPane();
        ProgressRankTable = new javax.swing.JTable();
        GetBackButton = new javax.swing.JButton();

        setDefaultCloseOperation(javax.swing.WindowConstants.EXIT_ON_CLOSE);

        jLabel1.setFont(new java.awt.Font("Lucida Grande", 1, 24)); // NOI18N
        jLabel1.setText("TEAM RANKINGS");

        jLabel2.setText("1. Progress Ranking");

        jLabel3.setText("2. Points Ranking");

        PointsRankTable.setModel(new javax.swing.table.DefaultTableModel(
            new Object [][] {

                },
            new String [] {
                "No.", "Names", "Overall Progress"
            }
        ) {
            Class[] types = new Class [] {
                java.lang.Integer.class, java.lang.Object.class,
java.lang.Integer.class
            };

            public Class getColumnClass(int columnIndex) {
                return types [columnIndex];
            }
        });
        jScrollPane1.setViewportViewView(PointsRankTable);

```

```

ProgressRankTable.setModel(new javax.swing.table.DefaultTableModel(
    new Object [][] {

        },
        new String [] {
            "No.", "Names", "Overall Progress"
        }
    ) {
        Class[] types = new Class [] {
            java.lang.Integer.class, java.lang.Object.class,
java.lang.Integer.class
        };

        public Class getColumnClass(int columnIndex) {
            return types [columnIndex];
        }
    });
jScrollPane2.setViewportViewView(ProgressRankTable);

GetBackButton.setText("Back");
GetBackButton.addActionListener(new java.awt.event.ActionListener() {
    public void actionPerformed(java.awt.event.ActionEvent evt) {
        GetBackButtonActionPerformed(evt);
    }
});

javax.swing.GroupLayout layout = new
javax.swing.GroupLayout(getContentPane());
getContentPane().setLayout(layout);
layout.setHorizontalGroup(

    layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
        .addGroup(javax.swing.GroupLayout.Alignment.TRAILING,
layout.createSequentialGroup()
        .addGap(32, 32, 32)
        .addComponent(jLabel2)
        .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement
.RELATED, 380, Short.MAX_VALUE)
        .addGroup(layout.createParallelGroup(javax.swing.GroupLayout.
Alignment.LEADING)
            .addComponent(jLabel3)
            .addComponent(jScrollPane1,
javax.swing.GroupLayout.PREFERRED_SIZE, 392,

```



```

javax.swing.GroupLayout.PREFERRED_SIZE))
    .addGap(25, 25, 25))
    .addGroup(layout.createSequentialGroup()
        .addGap(367, 367, 367)
        .addComponent(jLabel1)
        .addContainerGap(javax.swing.GroupLayout.DEFAULT_SIZE,
Short.MAX_VALUE))
    .addGroup(javax.swing.GroupLayout.Alignment.TRAILING,
layout.createSequentialGroup()
        .addContainerGap(javax.swing.GroupLayout.DEFAULT_SIZE,
Short.MAX_VALUE)
        .addComponent(GetBackButton)
        .addGap(438, 438, 438))
    .addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Ali
gment.LEADING)
        .addGroup(layout.createSequentialGroup()
            .addGap(27, 27, 27)
            .addComponent(jScrollPane2,
javax.swing.GroupLayout.PREFERRED_SIZE, 392,
javax.swing.GroupLayout.PREFERRED_SIZE)
            .addContainerGap(535, Short.MAX_VALUE)))
    );
layout.setVerticalGroup(

layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
    .addGroup(layout.createSequentialGroup()
        .addContainerGap()
        .addComponent(jLabel1)
        .addGap(71, 71, 71)
        .addGroup(layout.createParallelGroup(javax.swing.GroupLayout.
Alignment.BASELINE)
            .addComponent(jLabel2)
            .addComponent(jLabel3))
        .addGap(29, 29, 29)
        .addComponent(jScrollPane1,
javax.swing.GroupLayout.PREFERRED_SIZE, 455,
javax.swing.GroupLayout.PREFERRED_SIZE)
        .addGap(18, 18, 18)
        .addComponent(GetBackButton)
        .addContainerGap(44, Short.MAX_VALUE))
    .addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Ali
gment.LEADING)
        .addGroup(layout.createSequentialGroup()
            .addGap(151, 151, 151)

```

```

        .addComponent(jScrollPane2,
javax.swing.GroupLayout.PREFERRED_SIZE, 458,
javax.swing.GroupLayout.PREFERRED_SIZE)
        .addContainerGap(89, Short.MAX_VALUE)))
    );

    pack();
} // </editor-fold>

private void GetBackButtonActionPerformed(java.awt.event.ActionEvent evt) {
    if (null != LoginWindow.status) {
        switch (LoginWindow.status) {
            /*use different cases to displace back to the original page
since members, leaders, and administrators could all access this
page
*/
            case "leader":
                new TeamLeaderWindow().setVisible(true);
                dispose();
                break;
            case "member":
                new TeamMemberWindow().setVisible(true);
                dispose();
                break;
            case "admin":
                new AdminWindow().setVisible(true);
                dispose();
                break;
            default:
                break;
        }
    }
}

/**
 * @param args the command line arguments
 */
public static void main(String args[]) {
    /* Set the Nimbus look and feel */
    //<editor-fold defaultstate="collapsed" desc=" Look and feel setting code
(optional) ">
    /* If Nimbus (introduced in Java SE 6) is not available, stay with the default
look and feel.
    * For details see

```

```

http://download.oracle.com/javase/tutorial/uiswing/lookandfeel/plaf.html
    */
    try {
        for (javax.swing.UIManager.LookAndFeelInfo info :
javax.swing.UIManager.getInstalledLookAndFeels()) {
            if ("Nimbus".equals(info.getName())) {

javax.swing.UIManager.setLookAndFeel(info.getClassName());
                break;
            }
        }
    } catch (ClassNotFoundException ex) {

java.util.logging.Logger.getLogger(TeamRanking.class.getName()).log(java.util.loggi
ng.Level.SEVERE, null, ex);
        } catch (InstantiationException ex) {

java.util.logging.Logger.getLogger(TeamRanking.class.getName()).log(java.util.loggi
ng.Level.SEVERE, null, ex);
        } catch (IllegalAccessException ex) {

java.util.logging.Logger.getLogger(TeamRanking.class.getName()).log(java.util.loggi
ng.Level.SEVERE, null, ex);
        } catch (javax.swing.UnsupportedLookAndFeelException ex) {

java.util.logging.Logger.getLogger(TeamRanking.class.getName()).log(java.util.loggi
ng.Level.SEVERE, null, ex);
    }
    //</editor-fold>

    /* Create and display the form */
    java.awt.EventQueue.invokeLater(new Runnable() {
        public void run() {
            new TeamRanking().setVisible(true);
        }
    });
}

// Variables declaration - do not modify
private javax.swing.JButton GetBackButton;
private javax.swing.JTable PointsRankTable;
private javax.swing.JTable ProgressRankTable;
private javax.swing.JLabel jLabel1;
private javax.swing.JLabel jLabel2;

```

```
private javax.swing.JLabel jLabel3;  
private javax.swing.JProgressBar jProgressBar1;  
private javax.swing.JScrollPane jScrollPane1;  
private javax.swing.JScrollPane jScrollPane2;  
// End of variables declaration  
}
```

## Club Ranking

Club Ranking

Progress Ranking or Points Ranking? Progress Confirm

| Rank No. | Names | Specific Statistics |
|----------|-------|---------------------|
|----------|-------|---------------------|

Back

```
import java.awt.Dimension;
import java.awt.List;
import java.awt.Toolkit;
import java.sql.DriverManager;
import java.sql.PreparedStatement;
import java.sql.ResultSet;
import java.sql.SQLException;
import java.util.ArrayList;
import java.util.Collection;
import java.util.ListIterator;
import java.util.logging.Level;
import java.util.logging.Logger;
import javax.swing.JOptionPane;
import javax.swing.table.DefaultTableModel;
```

```
/*
```

```
 * To change this license header, choose License Headers in Project Properties.
```

```
 * To change this template file, choose Tools | Templates
```

```
 * and open the template in the editor.
```

```
*/
```

```

/**
 *
 * @author aidenyan
 */
public class ClubRanking extends javax.swing.JFrame {

    /**
     * Creates new form ClubRanking
     */
    public ClubRanking() {
        initComponents();
        Toolkit toolkit = getToolkit();
        Dimension size = toolkit.getScreenSize();
        setLocation(size.width / 2 - getWidth() / 2, size.height / 2 - getHeight() / 2);
    }

    private void bubbleSort(int arr[], int n) {
        /* a bubble sort method is needed since
        the whole club ranking would need to concatenate the
        databases together, rather than just using "ORDER BY" in
        sql commands for a single database
        */
        int i, j, temp;
        boolean swapped;
        for (i = 0; i < n - 1; i++) {
            swapped = false;
            for (j = 0; j < n - i - 1; j++) {
                if (arr[j] < arr[j + 1]) {
                    temp = arr[j];
                    arr[j] = arr[j + 1];
                    arr[j + 1] = temp;
                    swapped = true;
                }
            }
            if (swapped == false) {
                break;
            }
        }
    }

    /**
     * This method is called from within the constructor to initialize the form.

```

```

* WARNING: Do NOT modify this code. The content of this method is always
* regenerated by the Form Editor.
*/
@SuppressWarnings("unchecked")
// <editor-fold defaultstate="collapsed" desc="Generated Code">
private void initComponents() {

    jLabel1 = new javax.swing.JLabel();
    jLabel2 = new javax.swing.JLabel();
    SelectionBox = new javax.swing.JComboBox<>();
    AccessButton = new javax.swing.JButton();
    jScrollPane1 = new javax.swing.JScrollPane();
    RankTable = new javax.swing.JTable();
    GetBackButton = new javax.swing.JButton();

    setDefaultCloseOperation(javax.swing.WindowConstants.EXIT_ON_CLOSE);

    jLabel1.setFont(new java.awt.Font("Lucida Grande", 1, 24)); // NOI18N
    jLabel1.setText("Club Ranking");

    jLabel2.setText("Progress Ranking or Points Ranking?");

    SelectionBox.setModel(new javax.swing.DefaultComboBoxModel<>(new
String[] { "Progress", "Points" }));

    AccessButton.setText("Confirm");
    AccessButton.addActionListener(new java.awt.event.ActionListener() {
        public void actionPerformed(java.awt.event.ActionEvent evt) {
            AccessButtonActionPerformed(evt);
        }
    });

    RankTable.setModel(new javax.swing.table.DefaultTableModel(
        new Object [][] {

            },
        new String [] {
            "Rank No.", "Names", "Specific Statistics"
        }
    ) {
        Class[] types = new Class [] {
            java.lang.Integer.class, java.lang.String.class,
java.lang.String.class

```

```

    };

    public Class getColumnClass(int columnIndex) {
        return types [columnIndex];
    }
});
jScrollPane1.setViewportViewView(RankTable);

GetBackButton.setText("Back");
GetBackButton.addActionListener(new java.awt.event.ActionListener() {
    public void actionPerformed(java.awt.event.ActionEvent evt) {
        GetBackButtonActionPerformed(evt);
    }
});

javax.swing.GroupLayout layout = new
javax.swing.GroupLayout(getContentPane());
getContentPane().setLayout(layout);
layout.setHorizontalGroup(

layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
    .addGroup(layout.createSequentialGroup()
        .addGroup(layout.createParallelGroup(javax.swing.GroupLayout.
            Alignment.LEADING)
            .addGroup(layout.createSequentialGroup()
                .addGap(156, 156, 156)
                .addComponent(jLabel1))
            .addGroup(layout.createSequentialGroup()
                .addGap(21, 21, 21)
                .addGroup(layout.createParallelGroup(javax.swing.Gro
                    upLayout.Alignment.LEADING)
                        .addComponent(jScrollPane1,
                            javax.swing.GroupLayout.PREFERRED_SIZE,
                            javax.swing.GroupLayout.DEFAULT_SIZE,
                            javax.swing.GroupLayout.PREFERRED_SIZE)
                        .addGroup(layout.createSequentialGroup()
                            .addComponent(jLabel2)
                            .addPreferredGap(javax.swing.LayoutStyle.C
                                omponentPlacement.UNRELATED)
                            .addComponent(SelectionBox,
                                javax.swing.GroupLayout.PREFERRED_SIZE,
                                javax.swing.GroupLayout.DEFAULT_SIZE,
                                javax.swing.GroupLayout.PREFERRED_SIZE)
                            .addPreferredGap(javax.swing.LayoutStyle.C

```



```

omponentPlacement.RELATED)
        .addComponent(AccessButton))))
        .addGroup(layout.createSequentialGroup()
            .addGap(220, 220, 220)
            .addComponent(GetBackButton))
        .addContainerGap(78, Short.MAX_VALUE))
    );
    layout.setVerticalGroup(

    layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
        .addGroup(layout.createSequentialGroup()
            .addGap(21, 21, 21)
            .addComponent(jLabel1)
            .addGap(18, 18, 18)
            .addGroup(layout.createParallelGroup(javax.swing.GroupLayout.
Alignment.BASELINE)
                .addComponent(jLabel2)
                .addComponent(SelectionBox,
javax.swing.GroupLayout.PREFERRED_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE,
javax.swing.GroupLayout.PREFERRED_SIZE)
                .addComponent(AccessButton))
            .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement
.RELATED)
                .addComponent(jScrollPane1,
javax.swing.GroupLayout.PREFERRED_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE,
javax.swing.GroupLayout.PREFERRED_SIZE)
                .addGap(18, 18, 18)
                .addComponent(GetBackButton)
                .addContainerGap(12, Short.MAX_VALUE))
        );

    pack();
} // </editor-fold>

private void AccessButtonActionPerformed(java.awt.event.ActionEvent evt) {
    String connectionURL = "jdbc:derby://localhost:1527/QHAPPYDatabase";
    try {
        int count = 0;
        java.sql.Connection conn =
DriverManager.getConnection(connectionURL, "username", "password");
        String typeOfRank = (String) SelectionBox.getSelectedItem();

```

```

//see the type of ranking selected to be accessed
if (typeOfRank == "Progress") {
    DefaultTableModel tblModel = (DefaultTableModel)
RankTable.getModel();
    tblModel.setRowCount(0);
    //get all the members in the club, and then access their progress
statistics
    String sql1 = "SELECT * FROM TEAM1_MEMBERS";
    String sql2 = "SELECT * FROM TEAM2_MEMBERS";
    String sql3 = "SELECT * FROM TEAM3_MEMBERS";
    String sql4 = "SELECT * FROM UNASSIGNED_MEMBERS";

    PreparedStatement st1 = conn.prepareStatement(sql1);
    PreparedStatement st2 = conn.prepareStatement(sql2);
    PreparedStatement st3 = conn.prepareStatement(sql3);
    PreparedStatement st4 = conn.prepareStatement(sql4);

    ResultSet rs1 = st1.executeQuery();
    ResultSet rs2 = st2.executeQuery();
    ResultSet rs3 = st3.executeQuery();
    ResultSet rs4 = st4.executeQuery();

    /*the allProgress arraylist would then contain all the progress data
for all members
    and the UsedNames would record all the members whose names
have been prescribed
    onto the jTable. This is used because i would still be going
through databases by
    databases with the different sql commands, and if there is a tie
between two members, the
    system cannot distinguish by itself which name to put into the
table
    */
    ArrayList<Integer> allProgress = new ArrayList<>();
    ArrayList<String> UsedNames = new ArrayList<>();

    //insert all the points into the allProgress array list
    while (rs1.next()) {
        allProgress.add(rs1.getInt("OVERALL_PROGRESS"));
    }
    while (rs2.next()) {
        allProgress.add(rs2.getInt("OVERALL_PROGRESS"));
    }
}

```

```

while (rs3.next()) {
    allProgress.add(rs3.getInt("OVERALL_PROGRESS"));
}
while (rs4.next()) {
    allProgress.add(rs4.getInt("OVERALL_PROGRESS"));
}
//convert the array list into a 1-D array for bubble sort
int arrLength = allProgress.size();
int[] theArray = new int[arrLength];
for (int i = 0; i < arrLength; i++) {
    theArray[i] = allProgress.get(i);
}
bubbleSort(theArray, arrLength);

for (int i = 0; i < arrLength; i++) {
    String sql5 = "SELECT * FROM TEAM1_MEMBERS
WHERE OVERALL_PROGRESS=?";
    String sql6 = "SELECT * FROM TEAM2_MEMBERS
WHERE OVERALL_PROGRESS=?";
    String sql7 = "SELECT * FROM TEAM3_MEMBERS
WHERE OVERALL_PROGRESS=?";
    String sql8 = "SELECT * FROM
UNASSIGNED_MEMBERS WHERE OVERALL_PROGRESS=?";

    PreparedStatement st5 = conn.prepareStatement(sql5);
    PreparedStatement st6 = conn.prepareStatement(sql6);
    PreparedStatement st7 = conn.prepareStatement(sql7);
    PreparedStatement st8 = conn.prepareStatement(sql8);

    st5.setInt(1, theArray[i]);
    st6.setInt(1, theArray[i]);
    st7.setInt(1, theArray[i]);
    st8.setInt(1, theArray[i]);

    //get the members with the corresponding progress statistics,
find all of them

    ResultSet rs5 = st5.executeQuery();
    ResultSet rs6 = st6.executeQuery();
    ResultSet rs7 = st7.executeQuery();
    ResultSet rs8 = st8.executeQuery();

    String memberName = null;

```

```

String theStats = String.valueOf(theArray[i]);

while (rs5.next()) {
    int flag = 0;
    //set a flag first
    memberName = rs5.getString("NAME");
    ListIterator<String> theCheck =
UsedNames.listIterator();
    //check if the member has been recorded in the
UsedNames array list through the iterator
    while (theCheck.hasNext()) {
        String comp = theCheck.next();
        if (comp.equals(memberName)) {
            flag += 1;
            //increment the flag by 1 if it has appeared in
the UsedNames list
        }
    }
    if (flag == 0) {
        //only if the flag is 0 could the system add this
searched member into the rank
        count += 1;
        String specificRank = String.valueOf(count);
        String tbData1[] = {specificRank, memberName,
theStats};

        tblModel.addRow(tbData1);
        UsedNames.add(memberName);
    }
}
//the following codes do the same for team 2, team 3, and
unassigned members
while (rs6.next()) {
    int flag = 0;

    memberName = rs6.getString("NAME");
    ListIterator<String> theCheck =
UsedNames.listIterator();
    while (theCheck.hasNext()) {
        String comp = theCheck.next();
        if (comp.equals(memberName)) {
            flag += 1;
        }
    }
}

```

```

        if (flag == 0) {
            count += 1;
            String specificRank = String.valueOf(count);
            String tbData1[] = {specificRank, memberName,
theStats};

            tblModel.addRow(tbData1);
            UsedNames.add(memberName);
        }
    }
    while (rs7.next()) {
        int flag = 0;

        memberName = rs7.getString("NAME");
        ListIterator<String> theCheck =
UsedNames.listIterator();
        while (theCheck.hasNext()) {
            String comp = theCheck.next();
            if (comp.equals(memberName)) {
                flag += 1;
            }
        }
        if (flag == 0) {
            count += 1;
            String specificRank = String.valueOf(count);
            String tbData1[] = {specificRank, memberName,
theStats};

            tblModel.addRow(tbData1);
            UsedNames.add(memberName);
        }
    }
    while (rs8.next()) {
        int flag = 0;

        memberName = rs8.getString("NAME");
        ListIterator<String> theCheck =
UsedNames.listIterator();
        while (theCheck.hasNext()) {
            String comp = theCheck.next();
            if (comp.equals(memberName)) {
                flag += 1;
            }
        }
        if (flag == 0) {

```

```

        count += 1;
        String specificRank = String.valueOf(count);
        String tbData1[] = {specificRank, memberName,

theStats};

        tblModel.addRow(tbData1);
        UsedNames.add(memberName);
    }
}

//the following codes do the same for points rather than progress
} else {
    DefaultTableModel tblModel = (DefaultTableModel)
RankTable.getModel();
    tblModel.setRowCount(0);
    String sql1 = "SELECT * FROM TEAM1_MEMBERS";
    String sql2 = "SELECT * FROM TEAM2_MEMBERS";
    String sql3 = "SELECT * FROM TEAM3_MEMBERS";
    String sql4 = "SELECT * FROM UNASSIGNED_MEMBERS";

    PreparedStatement st1 = conn.prepareStatement(sql1);
    PreparedStatement st2 = conn.prepareStatement(sql2);
    PreparedStatement st3 = conn.prepareStatement(sql3);
    PreparedStatement st4 = conn.prepareStatement(sql4);

    ResultSet rs1 = st1.executeQuery();
    ResultSet rs2 = st2.executeQuery();
    ResultSet rs3 = st3.executeQuery();
    ResultSet rs4 = st4.executeQuery();

    ArrayList<Integer> allProgress = new ArrayList<>();
    ArrayList<String> UsedNames = new ArrayList<>();
    while (rs1.next()) {
        allProgress.add(rs1.getInt("POINTS"));
    }
    while (rs2.next()) {
        allProgress.add(rs2.getInt("POINTS"));
    }
    while (rs3.next()) {
        allProgress.add(rs3.getInt("POINTS"));
    }
    while (rs4.next()) {
        allProgress.add(rs4.getInt("POINTS"));
    }
}

```

```

        int arrLength = allProgress.size();
        int[] theArray = new int[arrLength];
        for (int i = 0; i < arrLength; i++) {
            theArray[i] = allProgress.get(i);
        }
        bubbleSort(theArray, arrLength);
        for (int i = 0; i < arrLength; i++) {
            String sql5 = "SELECT * FROM TEAM1_MEMBERS
WHERE POINTS=?";
            String sql6 = "SELECT * FROM TEAM2_MEMBERS
WHERE POINTS=?";
            String sql7 = "SELECT * FROM TEAM3_MEMBERS
WHERE POINTS=?";
            String sql8 = "SELECT * FROM
UNASSIGNED_MEMBERS WHERE POINTS=?";

            PreparedStatement st5 = conn.prepareStatement(sql5);
            PreparedStatement st6 = conn.prepareStatement(sql6);
            PreparedStatement st7 = conn.prepareStatement(sql7);
            PreparedStatement st8 = conn.prepareStatement(sql8);

            st5.setInt(1, theArray[i]);
            st6.setInt(1, theArray[i]);
            st7.setInt(1, theArray[i]);
            st8.setInt(1, theArray[i]);

            ResultSet rs5 = st5.executeQuery();
            ResultSet rs6 = st6.executeQuery();
            ResultSet rs7 = st7.executeQuery();
            ResultSet rs8 = st8.executeQuery();

            String memberName = null;

            String theStats = String.valueOf(theArray[i]);

            while (rs5.next()) {
                int flag = 0;

                memberName = rs5.getString("NAME");
                ListIterator<String> theCheck =
UsedNames.listIterator();
                while (theCheck.hasNext()) {
                    String comp = theCheck.next();
                    if (comp.equals(memberName)) {

```

```

        flag += 1;
    }
}
if (flag == 0) {
    count += 1;
    String specificRank = String.valueOf(count);
    String tbData1[] = {specificRank, memberName,
theStats};

    tblModel.addRow(tbData1);
    UsedNames.add(memberName);
}

}
while (rs6.next()) {
    int flag = 0;

    memberName = rs6.getString("NAME");
    ListIterator<String> theCheck =
UsedNames.listIterator();

    while (theCheck.hasNext()) {
        String comp = theCheck.next();
        if (comp.equals(memberName)) {
            flag += 1;
        }
    }
    if (flag == 0) {
        count += 1;
        String specificRank = String.valueOf(count);
        String tbData1[] = {specificRank, memberName,
theStats};

        tblModel.addRow(tbData1);
        UsedNames.add(memberName);
    }

}
while (rs7.next()) {
    int flag = 0;

    memberName = rs7.getString("NAME");
    ListIterator<String> theCheck =
UsedNames.listIterator();

    while (theCheck.hasNext()) {
        String comp = theCheck.next();
        if (comp.equals(memberName)) {

```



```

        flag += 1;
    }
}
if (flag == 0) {
    count += 1;
    String specificRank = String.valueOf(count);
    String tbData1[] = {specificRank, memberName,
theStats};

    tblModel.addRow(tbData1);
    UsedNames.add(memberName);
}
}
while (rs8.next()) {
    int flag = 0;

    memberName = rs8.getString("NAME");
    ListIterator<String> theCheck =
UsedNames.listIterator();
    while (theCheck.hasNext()) {
        String comp = theCheck.next();
        if (comp.equals(memberName)) {
            flag += 1;
        }
    }
    if (flag == 0) {
        count += 1;
        String specificRank = String.valueOf(count);
        String tbData1[] = {specificRank, memberName,
theStats};

        tblModel.addRow(tbData1);
        UsedNames.add(memberName);
    }
}

}

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

}

private void GetBackButtonActionPerformed(java.awt.event.ActionEvent evt) {

```

```

        new AdminWindow().setVisible(true);
        dispose();
    }

    /**
     * @param args the command line arguments
     */
    public static void main(String args[]) {
        /* Set the Nimbus look and feel */
        //<editor-fold defaultstate="collapsed" desc=" Look and feel setting code
(optional) ">
        /* If Nimbus (introduced in Java SE 6) is not available, stay with the default
look and feel.
         * For details see
http://download.oracle.com/javase/tutorial/uiswing/lookandfeel/plaf.html
         */
        try {
            for (javax.swing.UIManager.LookAndFeelInfo info :
javax.swing.UIManager.getInstalledLookAndFeels()) {
                if ("Nimbus".equals(info.getName())) {

javax.swing.UIManager.setLookAndFeel(info.getClassName());
                    break;

                }
            }
        } catch (ClassNotFoundException ex) {

java.util.logging.Logger.getLogger(ClubRanking.class.getName()).log(java.util.loggi
ng.Level.SEVERE, null, ex);
        } catch (InstantiationException ex) {

java.util.logging.Logger.getLogger(ClubRanking.class.getName()).log(java.util.loggi
ng.Level.SEVERE, null, ex);
        } catch (IllegalAccessException ex) {

java.util.logging.Logger.getLogger(ClubRanking.class.getName()).log(java.util.loggi
ng.Level.SEVERE, null, ex);
        } catch (javax.swing.UnsupportedLookAndFeelException ex) {

java.util.logging.Logger.getLogger(ClubRanking.class.getName()).log(java.util.loggi
ng.Level.SEVERE, null, ex);
        }
    }
}
//</editor-fold>

```

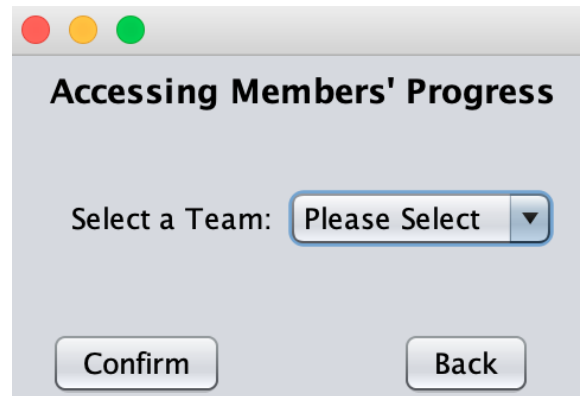
```

        /* Create and display the form */
        java.awt.EventQueue.invokeLater(new Runnable() {
            public void run() {
                new ClubRanking().setVisible(true);
            }
        });
    }

    // Variables declaration - do not modify
    private javax.swing.JButton AccessButton;
    private javax.swing.JButton GetBackButton;
    private javax.swing.JTable RankTable;
    private javax.swing.JComboBox<String> SelectionBox;
    private javax.swing.JLabel jLabel1;
    private javax.swing.JLabel jLabel2;
    private javax.swing.JScrollPane jScrollPane1;
    // End of variables declaration
}

```

## View Progress



```
import java.awt.Dimension;
import java.awt.Toolkit;
import javax.swing.JOptionPane;

/*
 * To change this license header, choose License Headers in Project Properties.
 * To change this template file, choose Tools | Templates
 * and open the template in the editor.
 */

/**
 *
 * @author aidenyan
 */
public class MembersProgress extends javax.swing.JFrame {

    /**
     * Creates new form MembersProgress
     */
    public MembersProgress() {
        initComponents();
        Toolkit toolkit = getToolkit();
        Dimension size = toolkit.getScreenSize();
        setLocation(size.width / 2 - getWidth() / 2, size.height / 2 - getHeight() / 2);
    }

    /**
     * This method is called from within the constructor to initialize the form.
     * WARNING: Do NOT modify this code. The content of this method is always
     * regenerated by the Form Editor.
     */
    @SuppressWarnings("unchecked")
```

```
// <editor-fold defaultstate="collapsed" desc="Generated Code">
private void initComponents() {

    jLabel1 = new javax.swing.JLabel();
    jLabel2 = new javax.swing.JLabel();
    TeamBox = new javax.swing.JComboBox<>();
    ConfirmButton = new javax.swing.JButton();
    GetBackButton = new javax.swing.JButton();

    setDefaultCloseOperation(javax.swing.WindowConstants.EXIT_ON_CLOSE);

    jLabel1.setFont(new java.awt.Font("Lucida Grande", 1, 14)); // NOI18N
    jLabel1.setText("Accessing Members' Progress");

    jLabel2.setText("Select a Team:");

    TeamBox.setModel(new javax.swing.DefaultComboBoxModel<>(new
String[] { "Please Select", "Team 1", "Team 2", "Team 3" }));

    ConfirmButton.setText("Confirm");
    ConfirmButton.addActionListener(new java.awt.event.ActionListener() {
        public void actionPerformed(java.awt.event.ActionEvent evt) {
            ConfirmButtonActionPerformed(evt);
        }
    });

    GetBackButton.setText("Back");
    GetBackButton.addActionListener(new java.awt.event.ActionListener() {
        public void actionPerformed(java.awt.event.ActionEvent evt) {
            GetBackButtonActionPerformed(evt);
        }
    });

    javax.swing.GroupLayout layout = new
javax.swing.GroupLayout(getContentPane());
    getContentPane().setLayout(layout);
    layout.setHorizontalGroup(
        layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
            .addGroup(layout.createSequentialGroup()
                .addContainerGap()
                .addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
                    .addComponent(jLabel1, javax.swing.GroupLayout.DEFAULT_SIZE, 300, true)
                    .addComponent(jLabel2, javax.swing.GroupLayout.DEFAULT_SIZE, 150, true)
                    .addComponent(TeamBox, javax.swing.GroupLayout.DEFAULT_SIZE, 150, true)
                    .addComponent(ConfirmButton, javax.swing.GroupLayout.DEFAULT_SIZE, 100, true)
                    .addComponent(GetBackButton, javax.swing.GroupLayout.DEFAULT_SIZE, 100, true)
                )
            )
    );
    layout.setVerticalGroup(
        layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
            .addGroup(layout.createSequentialGroup()
                .addContainerGap()
                .addComponent(jLabel1, javax.swing.GroupLayout.DEFAULT_SIZE, 40, true)
                .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED)
                .addComponent(jLabel2, javax.swing.GroupLayout.DEFAULT_SIZE, 30, true)
                .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED)
                .addComponent(TeamBox, javax.swing.GroupLayout.DEFAULT_SIZE, 30, true)
                .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED)
                .addComponent(ConfirmButton, javax.swing.GroupLayout.DEFAULT_SIZE, 30, true)
                .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED)
                .addComponent(GetBackButton, javax.swing.GroupLayout.DEFAULT_SIZE, 30, true)
            )
    );
}

private void ConfirmButtonActionPerformed(java.awt.event.ActionEvent evt) {
    // TODO add your handling code here:
}

private void GetBackButtonActionPerformed(java.awt.event.ActionEvent evt) {
    // TODO add your handling code here:
}
}

</editor-fold>
```

```

        .addGroup(layout.createSequentialGroup()
            .addGroup(layout.createParallelGroup(javax.swing.Gro
upLayout.Alignment.TRAILING)
                .addComponent(jLabel1)
                .addGroup(layout.createSequentialGroup()
                    .addComponent(jLabel2)
                    .addPreferredGap(javax.swing.LayoutStyle.C
omponentPlacement.RELATED)
                        .addComponent(TeamBox,
javax.swing.GroupLayout.PREFERRED_SIZE, 113,
javax.swing.GroupLayout.PREFERRED_SIZE)))
                    .addContainerGap(23, Short.MAX_VALUE))
                .addGroup(layout.createSequentialGroup()
                    .addComponent(ConfirmButton)
                    .addPreferredGap(javax.swing.LayoutStyle.Component
Placement.RELATED, javax.swing.GroupLayout.DEFAULT_SIZE,
Short.MAX_VALUE)
                        .addComponent(GetBackButton)
                        .addGap(33, 33, 33)))
            );
        layout.setVerticalGroup(

layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
            .addGroup(layout.createSequentialGroup()
                .addContainerGap()
                .addComponent(jLabel1)
                .addGap(33, 33, 33)
                .addGroup(layout.createParallelGroup(javax.swing.GroupLayout.
Alignment.BASELINE)
                    .addComponent(jLabel2)
                    .addComponent(TeamBox,
javax.swing.GroupLayout.PREFERRED_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE,
javax.swing.GroupLayout.PREFERRED_SIZE))
                        .addGap(35, 35, 35)
                        .addGroup(layout.createParallelGroup(javax.swing.GroupLayout.
Alignment.BASELINE)
                            .addComponent(ConfirmButton)
                            .addComponent(GetBackButton))
                            .addContainerGap(javax.swing.GroupLayout.DEFAULT_SIZE,
Short.MAX_VALUE))
                    );

        pack();

```

```

} // </editor-fold>

private void ConfirmButtonActionPerformed(java.awt.event.ActionEvent evt) {
    String theSelection = (String) TeamBox.getSelectedItemAt();

    //see if the user has made a selection on the team
    if(theSelection.equals("Please Select")){
        JOptionPane.showMessageDialog(null, "Please select a Team!");
    } else {
        /*update the static variable teamUsed so that when displaced to
        the team progress UI, the JComboBox would automatically give the
members
        in the specific team
        */
        LoginWindow.teamUsed = theSelection;
    }
    //displace to the team progress UI
    new TeamProgress().setVisible(true);
    dispose();
}

private void GetBackButtonActionPerformed(java.awt.event.ActionEvent evt) {
    new AdminWindow().setVisible(true);
    dispose();
}

/**
 * @param args the command line arguments
 */
public static void main(String args[]) {
    /* Set the Nimbus look and feel */
    //<editor-fold defaultstate="collapsed" desc=" Look and feel setting code
(optional) ">
    /* If Nimbus (introduced in Java SE 6) is not available, stay with the default
look and feel.
    * For details see
http://download.oracle.com/javase/tutorial/uiswing/lookandfeel/plaf.html
    */
    try {
        for (javax.swing.UIManager.LookAndFeelInfo info :
javax.swing.UIManager.getInstalledLookAndFeels()) {
            if ("Nimbus".equals(info.getName())) {
                javax.swing.UIManager.setLookAndFeel(info.getClassName());
            }
        }
    } catch (ClassNotFoundException ex) {
        java.util.logging.Logger.getLogger(Main.class).log(Level.SEVERE, null, ex);
    } catch (InstantiationException ex) {
        java.util.logging.Logger.getLogger(Main.class).log(Level.SEVERE, null, ex);
    } catch (IllegalAccessException ex) {
        java.util.logging.Logger.getLogger(Main.class).log(Level.SEVERE, null, ex);
    }
}

```

```

        break;
    }
}
} catch (ClassNotFoundException ex) {

java.util.logging.Logger.getLogger(MembersProgress.class.getName()).log(java.util.l
ogging.Level.SEVERE, null, ex);
    } catch (InstantiationException ex) {

java.util.logging.Logger.getLogger(MembersProgress.class.getName()).log(java.util.l
ogging.Level.SEVERE, null, ex);
    } catch (IllegalAccessException ex) {

java.util.logging.Logger.getLogger(MembersProgress.class.getName()).log(java.util.l
ogging.Level.SEVERE, null, ex);
    } catch (javax.swing.UnsupportedLookAndFeelException ex) {

java.util.logging.Logger.getLogger(MembersProgress.class.getName()).log(java.util.l
ogging.Level.SEVERE, null, ex);
    }
}
//</editor-fold>

/* Create and display the form */
java.awt.EventQueue.invokeLater(new Runnable() {
    public void run() {
        new MembersProgress().setVisible(true);
    }
});
}

// Variables declaration - do not modify
private javax.swing.JButton ConfirmButton;
private javax.swing.JButton GetBackButton;
private javax.swing.JComboBox<String> TeamBox;
private javax.swing.JLabel jLabel1;
private javax.swing.JLabel jLabel2;
// End of variables declaration
}

```



## Team Progress

Team Members' Progress

Select a Member:

| Subject | Progress(%) |
|---------|-------------|
|---------|-------------|

```
import java.awt.Dimension;
import java.awt.Toolkit;
import java.sql.DriverManager;
import java.sql.PreparedStatement;
import java.sql.SQLException;
import javax.swing.JOptionPane;
```

```
import java.sql.ResultSet;
import javax.swing.table.DefaultTableModel;
```

```
/*
 * To change this license header, choose License Headers in Project Properties.
 * To change this template file, choose Tools | Templates
 * and open the template in the editor.
 */
```

```
/**
```

```
*
```

```
* @author aidenyan
```

```
*/
```

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

```
    /**
```

```

* Creates new form TeamProgress
*/
public TeamProgress() {
    initComponents();
    Toolkit toolkit = getToolkit();
    Dimension size = toolkit.getScreenSize();
    setLocation(size.width / 2 - getWidth() / 2, size.height / 2 - getHeight() / 2);
    updateCombo();
}

private void updateCombo() {
    //a method automatically called once the program executes (put in the
    constructor) to form the combo boxes
    try {

        String connectionURL =
        "jdbc:derby://localhost:1527/QHAPPYDatabase";
        java.sql.Connection conn =
        DriverManager.getConnection(connectionURL, "username", "password");

        //add the members' names in the chosen teams into the JComboBox for
        further selection
        if (LoginWindow.teamUsed.equals("Team 1")) {

            String sql1 = "SELECT * FROM TEAM1_MEMBERS";
            PreparedStatement st1 = conn.prepareStatement(sql1);
            ResultSet rs1 = st1.executeQuery();
            while (rs1.next()) {
                MemberBox.addItem(rs1.getString("NAME"));
            }

        } else if (LoginWindow.teamUsed.equals("Team 2")) {
            String sql2 = "SELECT * FROM TEAM2_MEMBERS";
            PreparedStatement st2 = conn.prepareStatement(sql2);
            ResultSet rs2 = st2.executeQuery();
            while (rs2.next()) {
                MemberBox.addItem(rs2.getString("NAME"));
            }

        } else if (LoginWindow.teamUsed.equals("Team 3")) {
            String sql3 = "SELECT * FROM TEAM3_MEMBERS";
            PreparedStatement st3 = conn.prepareStatement(sql3);
            ResultSet rs3 = st3.executeQuery();
            while (rs3.next()) {

```

```

        MemberBox.addItem(rs3.getString("NAME"));
    }
}
} catch (SQLException e) {
    JOptionPane.showMessageDialog(null, e);
}

}

/**
 * This method is called from within the constructor to initialize the form.
 * WARNING: Do NOT modify this code. The content of this method is always
 * regenerated by the Form Editor.
 */
@SuppressWarnings("unchecked")
// <editor-fold defaultstate="collapsed" desc="Generated Code">
private void initComponents() {

    jLabel1 = new javax.swing.JLabel();
    jLabel2 = new javax.swing.JLabel();
    MemberBox = new javax.swing.JComboBox<>();
    AccessButton = new javax.swing.JButton();
    jScrollPane1 = new javax.swing.JScrollPane();
    ProgressTable = new javax.swing.JTable();
    GetBackButton = new javax.swing.JButton();

    setDefaultCloseOperation(javax.swing.WindowConstants.EXIT_ON_CLOSE);

    jLabel1.setFont(new java.awt.Font("Lucida Grande", 0, 18)); // NOI18N
    jLabel1.setText("Team Members' Progress");

    jLabel2.setFont(new java.awt.Font("Lucida Grande", 2, 13)); // NOI18N
    jLabel2.setText("Select a Member:");

    MemberBox.setModel(new javax.swing.DefaultComboBoxModel<>(new
String[] { "Please Select" }));

    AccessButton.setText("Access");
    AccessButton.addActionListener(new java.awt.event.ActionListener() {
        public void actionPerformed(java.awt.event.ActionEvent evt) {
            AccessButtonActionPerformed(evt);
        }
    });
}

```

[illegible]

```

        .addComponent(jScrollPane1,
javax.swing.GroupLayout.PREFERRED_SIZE, 375,
javax.swing.GroupLayout.PREFERRED_SIZE))
        .addGroup(layout.createSequentialGroup()
            .addGap(150, 150, 150)
            .addComponent(GetBackButton)))
        .addContainerGap(19, Short.MAX_VALUE))
    .addGroup(layout.createSequentialGroup()
        .addGap(18, 18, 18)
        .addComponent(jLabel2)
        .addGap(18, 18, 18)
        .addComponent(MemberBox, 0, 1, Short.MAX_VALUE)
        .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement
.UNRELATED)
        .addComponent(AccessButton)
        .addGap(42, 42, 42))
    );
    layout.setVerticalGroup(

layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
        .addGroup(layout.createSequentialGroup()
            .addGap(15, 15, 15)
            .addComponent(jLabel1)
            .addGap(18, 18, 18)
            .addGroup(layout.createParallelGroup(javax.swing.GroupLayout.
Alignment.BASELINE)
                .addComponent(jLabel2)
                .addComponent(MemberBox,
javax.swing.GroupLayout.PREFERRED_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE,
javax.swing.GroupLayout.PREFERRED_SIZE)
                .addComponent(AccessButton))
            .addGap(18, 18, 18)
            .addComponent(jScrollPane1,
javax.swing.GroupLayout.PREFERRED_SIZE, 241,
javax.swing.GroupLayout.PREFERRED_SIZE)
            .addGap(18, 18, 18)
            .addComponent(GetBackButton)
            .addContainerGap(17, Short.MAX_VALUE))
    );

    pack();
} // </editor-fold>

```

```

private void AccessButtonActionPerformed(java.awt.event.ActionEvent evt) {
    try {
        String connectionURL =
"jdbc:derby://localhost:1527/QHAPPYDatabase";
        java.sql.Connection conn =
DriverManager.getConnection(connectionURL, "username", "password");
        String theSelectedOne = (String) MemberBox.getSelectedItem();
        DefaultTableModel tblModel = (DefaultTableModel)
ProgressTable.getModel();

        //check if the user has made a selection of which member
        if (theSelectedOne.equals("Please Select")) {
            JOptionPane.showMessageDialog(null, "Please Select a
Member!");
        } else {
            if (LoginWindow.teamUsed.equals("Team 1")) {
                //get the name of the checked members, and find their stats
in their team member databases
                String sql4 = "SELECT * FROM TEAM1_MEMBERS
WHERE NAME=?";
                PreparedStatement st4 = conn.prepareStatement(sql4);
                st4.setString(1, theSelectedOne);
                ResultSet rs4 = st4.executeQuery();
                while (rs4.next()) {
                    tblModel.setRowCount(0);
                    //extract the information on their specific progresses
                    String Lit_Progress =
String.valueOf(rs4.getInt("LIT_PROGRESS"));
                    String Art_Progress =
String.valueOf(rs4.getInt("ART_PROGRESS"));
                    String Sci_Progress =
String.valueOf(rs4.getInt("SCI_PROGRESS"));
                    String SocSci_Progress =
String.valueOf(rs4.getInt("SOCSCI_PROGRESS"));
                    String Mus_Progress =
String.valueOf(rs4.getInt("MUS_PROGRESS"));
                    String Mat_Progress =
String.valueOf(rs4.getInt("MAT_PROGRESS"));
                    String Econ_Progress =
String.valueOf(rs4.getInt("ECON_PROGRESS"));

                    //compile a new row to show all the progresses on the
seven subjects

                    String[] allProgresses = {Lit_Progress, Art_Progress,

```

```

Sci_Progress, SocSci_Progress, Mus_Progress, Mat_Progress, Econ_Progress};
    String[] subjects = {"Literature", "Art", "Science",
    "Social Sciece", "Music", "Mathematics", "Economics"};

    for(int i = 0; i < 7; i ++){
        //add the new information into the jTable
        String tbData[] = {subjects[i],allProgresses[i]};
        tblModel.addRow(tbData);
    }
}
//following codes do the same as above for Team 2 and Team
3 members
    }else if (LoginWindow.teamUsed.equals("Team 2")){
        String sql5 = "SELECT * FROM TEAM2_MEMBERS
WHERE NAME=?";
        PreparedStatement st5 = conn.prepareStatement(sql5);
        st5.setString(1, theSelectedOne);
        ResultSet rs5 = st5.executeQuery();
        while (rs5.next()) {
            tblModel.setRowCount(0);
            String Lit_Progress =
String.valueOf(rs5.getInt("LIT_PROGRESS"));
            String Art_Progress =
String.valueOf(rs5.getInt("ART_PROGRESS"));
            String Sci_Progress =
String.valueOf(rs5.getInt("SCI_PROGRESS"));
            String SocSci_Progress =
String.valueOf(rs5.getInt("SOCSCI_PROGRESS"));
            String Mus_Progress =
String.valueOf(rs5.getInt("MUS_PROGRESS"));
            String Mat_Progress =
String.valueOf(rs5.getInt("MAT_PROGRESS"));
            String Econ_Progress =
String.valueOf(rs5.getInt("ECON_PROGRESS"));
            String[] allProgresses = {Lit_Progress, Art_Progress,
Sci_Progress, SocSci_Progress, Mus_Progress, Mat_Progress, Econ_Progress};
            String[] subjects = {"Literature", "Art", "Science",
    "Social Sciece", "Music", "Mathematics", "Economics"};
            for(int i = 0; i < 7; i ++){
                String tbData[] = {subjects[i],allProgresses[i]};
                tblModel.addRow(tbData);
            }
        }
    }
}

```

```

        }
    } else if(LoginWindow.teamUsed.equals("Team 3")){
        String sql6 = "SELECT * FROM TEAM3_MEMBERS
WHERE NAME=(?)";
        PreparedStatement st6 = conn.prepareStatement(sql6);
        st6.setString(1, theSelectedOne);
        ResultSet rs6 = st6.executeQuery();
        while (rs6.next()) {
            tblModel.setRowCount(0);
            String Lit_Progress =
String.valueOf(rs6.getInt("LIT_PROGRESS"));
            String Art_Progress =
String.valueOf(rs6.getInt("ART_PROGRESS"));
            String Sci_Progress =
String.valueOf(rs6.getInt("SCI_PROGRESS"));
            String SocSci_Progress =
String.valueOf(rs6.getInt("SOCSCI_PROGRESS"));
            String Mus_Progress =
String.valueOf(rs6.getInt("MUS_PROGRESS"));
            String Mat_Progress =
String.valueOf(rs6.getInt("MAT_PROGRESS"));
            String Econ_Progress =
String.valueOf(rs6.getInt("ECON_PROGRESS"));
            String[] allProgresses = {Lit_Progress, Art_Progress,
Sci_Progress, SocSci_Progress, Mus_Progress, Mat_Progress, Econ_Progress};
            String[] subjects = {"Literature", "Art", "Science",
"Social Sciece", "Music", "Mathematics", "Economics"};
            for(int i = 0; i < 7; i++){
                String tbData[] = {subjects[i],allProgresses[i]};
                tblModel.addRow(tbData);
            }
        }
    }
}
} catch (SQLException e) {
    JOptionPane.showMessageDialog(null, "Shown Unsuccessfully");
}
}

private void GetBackButtonActionPerformed(java.awt.event.ActionEvent evt) {
    new AdminWindow().setVisible(true);
    dispose();
}

```



```

/**
 * @param args the command line arguments
 */
public static void main(String args[]) {
    /* Set the Nimbus look and feel */
    //<editor-fold defaultstate="collapsed" desc=" Look and feel setting code
(optional) ">
    /* If Nimbus (introduced in Java SE 6) is not available, stay with the default
look and feel.
    * For details see
http://download.oracle.com/javase/tutorial/uiswing/lookandfeel/plaf.html
    */
    try {
        for (javax.swing.UIManager.LookAndFeelInfo info :
javax.swing.UIManager.getInstalledLookAndFeels()) {
            if ("Nimbus".equals(info.getName())) {

javax.swing.UIManager.setLookAndFeel(info.getClassName());
                break;
            }
        }
    } catch (ClassNotFoundException ex) {

java.util.logging.Logger.getLogger(TeamProgress.class.getName()).log(java.util.loggi
ng.Level.SEVERE, null, ex);
    } catch (InstantiationException ex) {

java.util.logging.Logger.getLogger(TeamProgress.class.getName()).log(java.util.loggi
ng.Level.SEVERE, null, ex);
    } catch (IllegalAccessException ex) {

java.util.logging.Logger.getLogger(TeamProgress.class.getName()).log(java.util.loggi
ng.Level.SEVERE, null, ex);
    } catch (javax.swing.UnsupportedLookAndFeelException ex) {

java.util.logging.Logger.getLogger(TeamProgress.class.getName()).log(java.util.loggi
ng.Level.SEVERE, null, ex);
    }
    //</editor-fold>

    /* Create and display the form */
    java.awt.EventQueue.invokeLater(new Runnable() {
        public void run() {

```

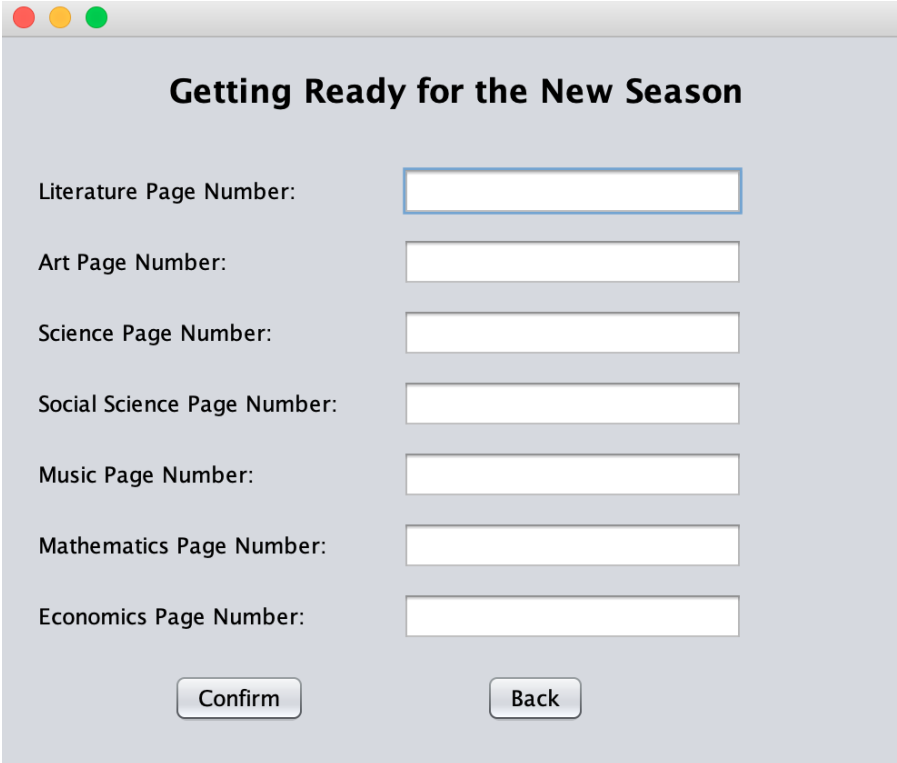
```

        new TeamProgress().setVisible(true);
    }
});
}

// Variables declaration - do not modify
private javax.swing.JButton AccessButton;
private javax.swing.JButton GetBackButton;
private javax.swing.JComboBox<String> MemberBox;
private javax.swing.JTable ProgressTable;
private javax.swing.JLabel jLabel1;
private javax.swing.JLabel jLabel2;
private javax.swing.JScrollPane jScrollPane1;
// End of variables declaration
}

```

## New Season



**Getting Ready for the New Season**

Literature Page Number:

Art Page Number:

Science Page Number:

Social Science Page Number:

Music Page Number:

Mathematics Page Number:

Economics Page Number:

```
import java.awt.Dimension;
import java.awt.Toolkit;
import java.sql.DriverManager;
import java.sql.PreparedStatement;
import java.sql.SQLException;
import javax.swing.JOptionPane;
```

```
/*
 * To change this license header, choose License Headers in Project Properties.
 * To change this template file, choose Tools | Templates
 * and open the template in the editor.
 */
```

```
/**
 *
 * @author aidenyan
 */
```

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

```
    /**
     * Creates new form NewSeason
     */
```

```
    public NewSeason() {
```

```

        initComponents();
        Toolkit toolkit = getToolkit();
        Dimension size = toolkit.getScreenSize();
        setLocation(size.width / 2 - getWidth() / 2, size.height / 2 - getHeight() / 2);
//center the GUI
    }

    /**
     * This method is called from within the constructor to initialize the form.
     * WARNING: Do NOT modify this code. The content of this method is always
     * regenerated by the Form Editor.
     */
    @SuppressWarnings("unchecked")
    // <editor-fold defaultstate="collapsed" desc="Generated Code">
    private void initComponents() {

        jLabel9 = new javax.swing.JLabel();
        jLabel1 = new javax.swing.JLabel();
        jLabel2 = new javax.swing.JLabel();
        jLabel3 = new javax.swing.JLabel();
        jLabel4 = new javax.swing.JLabel();
        jLabel5 = new javax.swing.JLabel();
        jLabel6 = new javax.swing.JLabel();
        jLabel7 = new javax.swing.JLabel();
        jLabel8 = new javax.swing.JLabel();
        LitPageNum = new javax.swing.JTextField();
        ArtPageNum = new javax.swing.JTextField();
        SciPageNum = new javax.swing.JTextField();
        SocSciPageNum = new javax.swing.JTextField();
        MusPageNum = new javax.swing.JTextField();
        MatPageNum = new javax.swing.JTextField();
        EconPageNum = new javax.swing.JTextField();
        ConfirmButton = new javax.swing.JButton();
        GetBackButton = new javax.swing.JButton();

        jLabel9.setText("Economics Page Number:");

        setDefaultCloseOperation(javax.swing.WindowConstants.EXIT_ON_CLOSE);

        jLabel1.setFont(new java.awt.Font("Lucida Grande", 1, 18)); // NOI18N
        jLabel1.setText("Getting Ready for the New Season");

        jLabel2.setText("Literature Page Number:");

```



```

        .addComponent(jLabel3)
        .addComponent(jLabel4)
        .addComponent(jLabel5)
        .addComponent(jLabel6)
        .addComponent(jLabel7)
        .addGroup(layout.createParallelGroup(javax.swing
.GroupLayout.Alignment.TRAILING)
        .addComponent(ConfirmButton)
        .addComponent(jLabel8)))
        .addGroup(layout.createParallelGroup(javax.swing.Gro
upLayout.Alignment.LEADING)
        .addGroup(layout.createSequentialGroup()
            .addGap(35, 35, 35)
            .addGroup(layout.createParallelGroup(javax.s
wing.GroupLayout.Alignment.LEADING, false)
                .addComponent(LitPageNum)
                .addComponent(ArtPageNum)
                .addComponent(SciPageNum)
                .addComponent(SocSciPageNum)
                .addComponent(MusPageNum)
                .addComponent(MatPageNum)
                .addComponent(EconPageNum,
javax.swing.GroupLayout.DEFAULT_SIZE, Short.MAX_VALUE)))
            .addGroup(layout.createSequentialGroup()
                .addGap(81, 81, 81)
                .addComponent(GetBackButton))))
        .addContainerGap(93, Short.MAX_VALUE))
    );
    layout.setVerticalGroup(

layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
        .addGroup(layout.createSequentialGroup()
            .addGap(18, 18, 18)
            .addComponent(jLabel1)
            .addGap(31, 31, 31)
            .addGroup(layout.createParallelGroup(javax.swing.GroupLayout.
Alignment.BASELINE)
                .addComponent(jLabel2)
                .addComponent(LitPageNum,
javax.swing.GroupLayout.PREFERRED_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE,
javax.swing.GroupLayout.PREFERRED_SIZE))
            .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement
.UNRELATED)

```

```

        .addGroup(layout.createParallelGroup(javax.swing.GroupLayout.
Alignment.BASELINE)
            .addComponent(jLabel3)
            .addComponent(ArtPageNum,
javax.swing.GroupLayout.PREFERRED_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE,
javax.swing.GroupLayout.PREFERRED_SIZE))
        .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement
.UNRELATED)
        .addGroup(layout.createParallelGroup(javax.swing.GroupLayout.
Alignment.BASELINE)
            .addComponent(jLabel4)
            .addComponent(SciPageNum,
javax.swing.GroupLayout.PREFERRED_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE,
javax.swing.GroupLayout.PREFERRED_SIZE))
        .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement
.UNRELATED)
        .addGroup(layout.createParallelGroup(javax.swing.GroupLayout.
Alignment.BASELINE)
            .addComponent(jLabel5)
            .addComponent(SocSciPageNum,
javax.swing.GroupLayout.PREFERRED_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE,
javax.swing.GroupLayout.PREFERRED_SIZE))
        .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement
.UNRELATED)
        .addGroup(layout.createParallelGroup(javax.swing.GroupLayout.
Alignment.BASELINE)
            .addComponent(jLabel6)
            .addComponent(MusPageNum,
javax.swing.GroupLayout.PREFERRED_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE,
javax.swing.GroupLayout.PREFERRED_SIZE))
        .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement
.UNRELATED)
        .addGroup(layout.createParallelGroup(javax.swing.GroupLayout.
Alignment.BASELINE)
            .addComponent(jLabel7)
            .addComponent(MatPageNum,
javax.swing.GroupLayout.PREFERRED_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE,
javax.swing.GroupLayout.PREFERRED_SIZE))
        .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement

```

```

.UNRELATED)
        .addGroup(layout.createParallelGroup(javax.swing.GroupLayout.
Alignment.BASELINE)
            .addComponent(jLabel8)
            .addComponent(EconPageNum,
javax.swing.GroupLayout.PREFERRED_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE,
javax.swing.GroupLayout.PREFERRED_SIZE))
        .addGap(18, 18, 18)
        .addGroup(layout.createParallelGroup(javax.swing.GroupLayout.
Alignment.BASELINE)
            .addComponent(ConfirmButton)
            .addComponent(GetBackButton))
        .addGap(0, 28, Short.MAX_VALUE))
    );

    pack();
} // </editor-fold>

private void ConfirmButtonActionPerformed(java.awt.event.ActionEvent evt) {
    try {
        String connectionURL =
"jdbc:derby://localhost:1527/QHAPPYDatabase";
        java.sql.Connection conn =
DriverManager.getConnection(connectionURL, "username", "password");

        int LitPgNum = Integer.valueOf(LitPageNum.getText());
        int ArtPgNum = Integer.valueOf(ArtPageNum.getText());
        int SciPgNum = Integer.valueOf(SciPageNum.getText());
        int SocSciPgNum = Integer.valueOf(SocSciPageNum.getText());
        int MusPgNum = Integer.valueOf(MusPageNum.getText());
        int MatPgNum = Integer.valueOf(MatPageNum.getText());
        int EconPgNum = Integer.valueOf(EconPageNum.getText());
        //update all the new page numbers into the database table

        String sql1 = "UPDATE PAGE_NUM SET LIT_PAGE = (?) WHERE
LINE_NUM = (?)";
        String sql2 = "UPDATE PAGE_NUM SET ART_PAGE = (?) WHERE
LINE_NUM = (?)";
        String sql3 = "UPDATE PAGE_NUM SET SCI_PAGE = (?) WHERE
LINE_NUM = (?)";
        String sql4 = "UPDATE PAGE_NUM SET SOCSCI_PAGE = (?)
WHERE LINE_NUM = (?)";
        String sql5 = "UPDATE PAGE_NUM SET MUS_PAGE = (?)

```



```

WHERE LINE_NUM = (?";
        String sql6 = "UPDATE PAGE_NUM SET MAT_PAGE = (?)
WHERE LINE_NUM = (?";
        String sql7 = "UPDATE PAGE_NUM SET ECON_PAGE = (?)
WHERE LINE_NUM = (?";

```

```

        PreparedStatement ps1 = conn.prepareStatement(sql1);
        PreparedStatement ps2 = conn.prepareStatement(sql2);
        PreparedStatement ps3 = conn.prepareStatement(sql3);
        PreparedStatement ps4 = conn.prepareStatement(sql4);
        PreparedStatement ps5 = conn.prepareStatement(sql5);
        PreparedStatement ps6 = conn.prepareStatement(sql6);
        PreparedStatement ps7 = conn.prepareStatement(sql7);

```

```

        ps1.setInt(1, LitPgNum);
        ps1.setInt(2, 1);
        ps2.setInt(1, ArtPgNum);
        ps2.setInt(2, 1);
        ps3.setInt(1, SciPgNum);
        ps3.setInt(2, 1);
        ps4.setInt(1, SocSciPgNum);
        ps4.setInt(2, 1);
        ps5.setInt(1, MusPgNum);
        ps5.setInt(2, 1);
        ps6.setInt(1, MatPgNum);
        ps6.setInt(2, 1);
        ps7.setInt(1, EconPgNum);
        ps7.setInt(2, 1);

```

```

        ps1.executeUpdate();
        ps2.executeUpdate();
        ps3.executeUpdate();
        ps4.executeUpdate();
        ps5.executeUpdate();
        ps6.executeUpdate();
        ps7.executeUpdate();

```

```

//put all members into UNASSIGNED_MEMBERS for the new season
and delete last season's records

```

```

        String sql8 = "INSERT INTO UNASSIGNED_MEMBERS
SELECT*FROM TEAM1_MEMBERS";

```

```
String sql9 = "INSERT INTO UNASSIGNED_MEMBERS  
SELECT*FROM TEAM2_MEMBERS";
```

```
String sql10 = "INSERT INTO UNASSIGNED_MEMBERS  
SELECT*FROM TEAM3_MEMBERS";
```

```
PreparedStatement ps8 = conn.prepareStatement(sql8);  
PreparedStatement ps9 = conn.prepareStatement(sql9);  
PreparedStatement ps10 = conn.prepareStatement(sql10);
```

```
ps8.executeUpdate();  
ps9.executeUpdate();  
ps10.executeUpdate();
```

```
String sql11 = "DELETE FROM TEAM1_MEMBERS";  
PreparedStatement ps11 = conn.prepareStatement(sql11);  
ps11.executeUpdate();
```

```
String sql12 = "DELETE FROM TEAM2_MEMBERS";  
PreparedStatement ps12 = conn.prepareStatement(sql12);  
ps12.executeUpdate();
```

```
String sql13 = "DELETE FROM TEAM3_MEMBERS";  
PreparedStatement ps13 = conn.prepareStatement(sql13);  
ps13.executeUpdate();
```

```
String sql14 = "UPDATE UNASSIGNED_MEMBERS SET POINTS =  
(?)";
```

```
PreparedStatement ps14 = conn.prepareStatement(sql14);  
ps14.setInt(1, 0);  
ps14.executeUpdate();
```

```
String sql15 = "UPDATE UNASSIGNED_MEMBERS SET  
LIT_PROGRESS = (?)";
```

```
String sql16 = "UPDATE UNASSIGNED_MEMBERS SET  
ART_PROGRESS = (?)";
```

```
String sql17 = "UPDATE UNASSIGNED_MEMBERS SET  
SCI_PROGRESS = (?)";
```

```
String sql18 = "UPDATE UNASSIGNED_MEMBERS SET  
SOCSCI_PROGRESS = (?)";
```

```
String sql19 = "UPDATE UNASSIGNED_MEMBERS SET  
MUS_PROGRESS = (?)";
```

```
String sql20 = "UPDATE UNASSIGNED_MEMBERS SET
```

```

MAT_PROGRESS = (?");
        String sql21 = "UPDATE UNASSIGNED_MEMBERS SET
ECON_PROGRESS = (?";
        String sql22 = "UPDATE UNASSIGNED_MEMBERS SET
OVERALL_PROGRESS = (?";
        String sql23 = "UPDATE UNASSIGNED_MEMBERS SET
TEAM_LEADER = (?";
        String sql24 = "UPDATE UNASSIGNED_MEMBERS SET
ACADEMICGROUP = (?";

```

```

PreparedStatement ps15 = conn.prepareStatement(sql15);
PreparedStatement ps16 = conn.prepareStatement(sql16);
PreparedStatement ps17 = conn.prepareStatement(sql17);
PreparedStatement ps18 = conn.prepareStatement(sql18);
PreparedStatement ps19 = conn.prepareStatement(sql19);
PreparedStatement ps20 = conn.prepareStatement(sql20);
PreparedStatement ps21 = conn.prepareStatement(sql21);
PreparedStatement ps22 = conn.prepareStatement(sql22);
PreparedStatement ps23 = conn.prepareStatement(sql23);
PreparedStatement ps24 = conn.prepareStatement(sql24);

```

```

ps15.setInt(1,0);
ps16.setInt(1,0);
ps17.setInt(1,0);
ps18.setInt(1,0);
ps19.setInt(1,0);
ps20.setInt(1,0);
ps21.setInt(1,0);
ps22.setInt(1,0);
ps23.setBoolean(1,false);
ps24.setString(1,null);

```

```

ps15.executeUpdate();
ps16.executeUpdate();
ps17.executeUpdate();
ps18.executeUpdate();
ps19.executeUpdate();
ps20.executeUpdate();
ps21.executeUpdate();
ps22.executeUpdate();
ps23.executeUpdate();
ps24.executeUpdate();

```

```

String sql25 = "INSERT INTO TEAM_MEMBER_LOGIN

```

```

SELECT*FROM TEAM_LEADER_LOGIN";
    String sql26 = "DELETE FROM TEAM_LEADER_LOGIN";
    PreparedStatement ps25 = conn.prepareStatement(sql25);
    PreparedStatement ps26 = conn.prepareStatement(sql26);
    ps25.executeUpdate();
    ps26.executeUpdate();

    JOptionPane.showMessageDialog(null, "New Season READY!");

    new AdminWindow().setVisible(true);
    dispose();

    } catch (SQLException ex) {
        JOptionPane.showMessageDialog(null, "Error");
    }
}

private void GetBackButtonActionPerformed(java.awt.event.ActionEvent evt) {
    new AdminWindow().setVisible(true);
    dispose();
}

/**
 * @param args the command line arguments
 */
public static void main(String args[]) {
    /* Set the Nimbus look and feel */
    //<editor-fold defaultstate="collapsed" desc=" Look and feel setting code
(optional) ">
    /* If Nimbus (introduced in Java SE 6) is not available, stay with the default
look and feel.
    * For details see
http://download.oracle.com/javase/tutorial/uiswing/lookandfeel/plaf.html
    */
    try {
        for (javax.swing.UIManager.LookAndFeelInfo info :
javax.swing.UIManager.getInstalledLookAndFeels()) {
            if ("Nimbus".equals(info.getName())) {

javax.swing.UIManager.setLookAndFeel(info.getClassName());
                break;
            }
        }
    }
}

```

```

        }
    } catch (ClassNotFoundException ex) {

java.util.logging.Logger.getLogger(NewSeason.class.getName()).log(java.util.logging
.Level.SEVERE, null, ex);
    } catch (InstantiationException ex) {

java.util.logging.Logger.getLogger(NewSeason.class.getName()).log(java.util.logging
.Level.SEVERE, null, ex);
    } catch (IllegalAccessException ex) {

java.util.logging.Logger.getLogger(NewSeason.class.getName()).log(java.util.logging
.Level.SEVERE, null, ex);
    } catch (javax.swing.UnsupportedLookAndFeelException ex) {

java.util.logging.Logger.getLogger(NewSeason.class.getName()).log(java.util.logging
.Level.SEVERE, null, ex);
    }
}
//</editor-fold>

/* Create and display the form */
java.awt.EventQueue.invokeLater(new Runnable() {
    public void run() {
        new NewSeason().setVisible(true);
    }
});
}

// Variables declaration - do not modify
private javax.swing.JTextField ArtPageNum;
private javax.swing.JButton ConfirmButton;
private javax.swing.JTextField EconPageNum;
private javax.swing.JButton GetBackButton;
private javax.swing.JTextField LitPageNum;
private javax.swing.JTextField MatPageNum;
private javax.swing.JTextField MusPageNum;
private javax.swing.JTextField SciPageNum;
private javax.swing.JTextField SocSciPageNum;
private javax.swing.JLabel jLabel1;
private javax.swing.JLabel jLabel2;
private javax.swing.JLabel jLabel3;
private javax.swing.JLabel jLabel4;
private javax.swing.JLabel jLabel5;
private javax.swing.JLabel jLabel6;

```

```
private javax.swing.JLabel jLabel7;  
private javax.swing.JLabel jLabel8;  
private javax.swing.JLabel jLabel9;  
// End of variables declaration  
}
```

## Team Leader Select



```
import java.awt.Dimension;
import java.awt.Toolkit;
import java.sql.DriverManager;
import java.sql.PreparedStatement;
import java.sql.ResultSet;
import java.sql.SQLException;
import java.util.logging.Level;
import java.util.logging.Logger;
import javax.swing.JOptionPane;
```

```
/*
 * To change this license header, choose License Headers in Project Properties.
 * To change this template file, choose Tools | Templates
 * and open the template in the editor.
 */
/**
 *
 * @author aidenyan
 */
```

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

    private String theTeam;
```

```

/**
 * Creates new form TeamLeader
 */
public TeamLeader() {
    initComponents();
    Toolkit toolkit = getToolkit();
    Dimension size = toolkit.getScreenSize();
    setLocation(size.width / 2 - getWidth() / 2, size.height / 2 - getHeight() / 2);

    //centre GUI
}

private void updateLeaderBox() {
    LeaderBox.removeAllItems();
    String connectionURL = "jdbc:derby://localhost:1527/QHAPPYDatabase";
    try {
        java.sql.Connection conn =
DriverManager.getConnection(connectionURL, "username", "password");
        if (theTeam.equals("Team 1")) {
            String sql1 = "SELECT * FROM TEAM1_MEMBERS";

            PreparedStatement st1 = conn.prepareStatement(sql1);
            //get all team 1 members and input them to each JComboBox
            ResultSet rs1 = st1.executeQuery();

            while (rs1.next()) {
                LeaderBox.addItem(rs1.getString("NAME"));
            }

        } else if (theTeam.equals("Team 2")) {
            String sql2 = "SELECT * FROM TEAM2_MEMBERS";

            PreparedStatement st2 = conn.prepareStatement(sql2);
            //get all team 2 members and input them to each JComboBox
            ResultSet rs2 = st2.executeQuery();

            while (rs2.next()) {
                LeaderBox.addItem(rs2.getString("NAME"));
            }

        } else if (theTeam.equals("Team 3")) {
            String sql3 = "SELECT * FROM TEAM3_MEMBERS";

            PreparedStatement st3 = conn.prepareStatement(sql3);

```



```

//get all team 3 members and input them to each JComboBox
ResultSet rs3 = st3.executeQuery();

while (rs3.next()) {
    LeaderBox.addItem(rs3.getString("NAME"));
}

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

/**
 * This method is called from within the constructor to initialize the form.
 * WARNING: Do NOT modify this code. The content of this method is always
 * regenerated by the Form Editor.
 */
@SuppressWarnings("unchecked")
// <editor-fold defaultstate="collapsed" desc="Generated Code">
private void initComponents() {

    JLabel1 = new javax.swing.JLabel();
    JLabel2 = new javax.swing.JLabel();
    TeamBox = new javax.swing.JComboBox<>();
    ConfirmButton = new javax.swing.JButton();
    JLabel3 = new javax.swing.JLabel();
    LeaderBox = new javax.swing.JComboBox<>();
    JButton1 = new javax.swing.JButton();
    ConfirmLeaderButton = new javax.swing.JButton();

    setDefaultCloseOperation(javax.swing.WindowConstants.EXIT_ON_CLOSE);

    JLabel1.setFont(new java.awt.Font("Lucida Grande", 1, 18)); // NOI18N
    JLabel1.setText("Select Team Leader");

    JLabel2.setText("Which Team to Operate?");

    TeamBox.setModel(new javax.swing.DefaultComboBoxModel<>(new
String[] { "Please Select", "Team 1", "Team 2", "Team 3" }));

    ConfirmButton.setText("Confirm");

```

```

        ConfirmButton.addActionListener(new java.awt.event.ActionListener() {
            public void actionPerformed(java.awt.event.ActionEvent evt) {
                ConfirmButtonActionPerformed(evt);
            }
        });

        jLabel3.setText("Who would you like to appoint as the captain?");

        jButton1.setText("Back");
        jButton1.addActionListener(new java.awt.event.ActionListener() {
            public void actionPerformed(java.awt.event.ActionEvent evt) {
                jButton1ActionPerformed(evt);
            }
        });

        ConfirmLeaderButton.setText("Confirm");
        ConfirmLeaderButton.addActionListener(new
java.awt.event.ActionListener() {
            public void actionPerformed(java.awt.event.ActionEvent evt) {
                ConfirmLeaderButtonActionPerformed(evt);
            }
        });

        javax.swing.GroupLayout layout = new
javax.swing.GroupLayout(getContentPane());
        getContentPane().setLayout(layout);
        layout.setHorizontalGroup(

layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
            .addGroup(javax.swing.GroupLayout.Alignment.TRAILING,
layout.createSequentialGroup()
                .addContainerGap()
                .addComponent(jLabel1)
                .addGap(100, 100, 100)
                .addGroup(layout.createParallelGroup(javax.swing.GroupLayout.
Alignment.LEADING)
                    .addGroup(layout.createSequentialGroup()
                        .addComponent(TextBox,
javax.swing.GroupLayout.PREFERRED_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE,
javax.swing.GroupLayout.PREFERRED_SIZE)

```

```

        .addGap(37, 37, 37)
        .addComponent(ConfirmButton))
    .addGroup(layout.createSequentialGroup())
    .addGap(22, 22, 22)
    .addGroup(layout.createParallelGroup(javax.swing.Gro
upLayout.Alignment.LEADING)
        .addComponent(jLabel3)
        .addComponent(jLabel2)))
    .addGroup(layout.createSequentialGroup())
    .addGap(136, 136, 136)
    .addComponent(LeaderBox,
javax.swing.GroupLayout.PREFERRED_SIZE, 82,
javax.swing.GroupLayout.PREFERRED_SIZE)))
    .addContainerGap(67, Short.MAX_VALUE))
    .addGroup(layout.createSequentialGroup())
    .addGap(48, 48, 48)
    .addComponent(ConfirmLeaderButton)
    .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement
.RELATED, javax.swing.GroupLayout.DEFAULT_SIZE, Short.MAX_VALUE)
    .addComponent(jButton1)
    .addGap(64, 64, 64))
    );
    layout.setVerticalGroup(

layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
    .addGroup(layout.createSequentialGroup())
    .addGap(18, 18, 18)
    .addComponent(jLabel1)
    .addGap(18, 18, 18)
    .addComponent(jLabel2)
    .addGap(18, 18, 18)
    .addGroup(layout.createParallelGroup(javax.swing.GroupLayout.
Alignment.BASELINE)
        .addComponent(TeamBox,
javax.swing.GroupLayout.PREFERRED_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE,
javax.swing.GroupLayout.PREFERRED_SIZE)
        .addComponent(ConfirmButton))
    .addGap(18, 18, 18)
    .addComponent(jLabel3)
    .addGap(18, 18, 18)
    .addComponent(LeaderBox,
javax.swing.GroupLayout.PREFERRED_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE,

```

```

javax.swing.GroupLayout.PREFERRED_SIZE)
        .addGap(18, 18, 18)
        .addGroup(layout.createParallelGroup(javax.swing.GroupLayout.
Alignment.BASELINE)
        .addComponent(ConfirmLeaderButton)
        .addComponent(jButton1))
        .addContainerGap(22, Short.MAX_VALUE))
    );

    pack();
} // </editor-fold>

```

```

private void ConfirmButtonActionPerformed(java.awt.event.ActionEvent evt) {
    String currentSelect = (String) TeamBox.getSelectedItemAt();
    if (currentSelect.equals("Please Select")) {
        JOptionPane.showMessageDialog(null, "Please Select a Team!");
        //let users select if they have yet to make a selection
    } else {
        //define the new teamUsed static variable
        theTeam = currentSelect;
    }
    updateLeaderBox();
}

```

```

private void ConfirmLeaderButtonActionPerformed(java.awt.event.ActionEvent
evt) {
    String connectionURL = "jdbc:derby://localhost:1527/QHAPPYDatabase";
    try {
        java.sql.Connection conn =
DriverManager.getConnection(connectionURL, "username", "password");

```

```

        //once the team is selected, input the team members and let the admin
to select the new leader

```

```

        String currentLeaderSelect = (String) LeaderBox.getSelectedItemAt();
        boolean hadLeader = false;
        if (theTeam.equals("Team 1")) {
            String sql4 = "SELECT * from TEAM1_MEMBERS";
            PreparedStatement st4 = conn.prepareStatement(sql4);
            ResultSet rs4 = st4.executeQuery();
            while (rs4.next()) {
                if (rs4.getBoolean("TEAM_LEADER") == true) {
                    hadLeader = true;

```

```

        }
    }
    if (hadLeader) {
        JOptionPane.showMessageDialog(null, "You already had a
leader for Team 1!");
    } else {
        String sql5 = "UPDATE TEAM1_MEMBERS SET
TEAM_LEADER = (?) WHERE NAME = (?)";
        PreparedStatement st5 = conn.prepareStatement(sql5);
        st5.setBoolean(1, true);
        st5.setString(2, currentLeaderSelect);
        st5.executeUpdate();
        String sql10 = "INSERT INTO TEAM_LEADER_LOGIN
SELECT*FROM TEAM_MEMBER_LOGIN WHERE NAME =(?)";
        PreparedStatement st10 = conn.prepareStatement(sql10);
        st10.setString(1, currentLeaderSelect);
        st10.executeUpdate();

        String sql11 = "DELETE FROM
TEAM_MEMBER_LOGIN WHERE NAME = (?)";
        PreparedStatement st11 = conn.prepareStatement(sql11);
        st11.setString(1, currentLeaderSelect);
        st11.executeUpdate();

        JOptionPane.showMessageDialog(null,"New Leader Chosen
for Team 1!");
    }
} else if (theTeam.equals("Team 2")) {
    String sql6 = "SELECT * from TEAM2_MEMBERS";
    PreparedStatement st6 = conn.prepareStatement(sql6);
    ResultSet rs6 = st6.executeQuery();
    while (rs6.next()) {
        if (rs6.getBoolean("TEAM_LEADER") == true) {
            hadLeader = true;
        }
    }
    if (hadLeader) {
        JOptionPane.showMessageDialog(null, "You already had a
leader for Team 2!");
    } else {
        String sql7 = "UPDATE TEAM2_MEMBERS SET
TEAM_LEADER = (?) WHERE NAME = (?)";
        PreparedStatement st7 = conn.prepareStatement(sql7);
        st7.setBoolean(1, true);

```

```

        st7.setString(2, currentLeaderSelect);
        st7.executeUpdate();
        String sql10 = "INSERT INTO TEAM_LEADER_LOGIN
SELECT*FROM TEAM_MEMBER_LOGIN WHERE NAME =(?)";
        PreparedStatement st10 = conn.prepareStatement(sql10);
        st10.setString(1, currentLeaderSelect);
        st10.executeUpdate();

        String sql11 = "DELETE FROM
TEAM_MEMBER_LOGIN WHERE NAME = (?)";
        PreparedStatement st11 = conn.prepareStatement(sql11);
        st11.setString(1, currentLeaderSelect);
        st11.executeUpdate();

        JOptionPane.showMessageDialog(null,"New Leader Chosen
for Team 2!");
    }
    }else if (theTeam.equals("Team 3")) {
        String sql8 = "SELECT * from TEAM3_MEMBERS";
        PreparedStatement st8 = conn.prepareStatement(sql8);
        ResultSet rs8 = st8.executeQuery();
        while (rs8.next()) {
            if (rs8.getBoolean("TEAM_LEADER") == true) {
                hadLeader = true;
            }
        }
        if (hadLeader) {
            JOptionPane.showMessageDialog(null, "You already had a
leader for Team 3!");
        } else {
            String sql9 = "UPDATE TEAM3_MEMBERS SET
TEAM_LEADER = (?) WHERE NAME = (?)";
            PreparedStatement st9 = conn.prepareStatement(sql9);
            st9.setBoolean(1, true);
            st9.setString(2, currentLeaderSelect);
            st9.executeUpdate();
            String sql10 = "INSERT INTO TEAM_LEADER_LOGIN
SELECT*FROM TEAM_MEMBER_LOGIN WHERE NAME =(?)";
            PreparedStatement st10 = conn.prepareStatement(sql10);
            st10.setString(1, currentLeaderSelect);
            st10.executeUpdate();

            String sql11 = "DELETE FROM
TEAM_MEMBER_LOGIN WHERE NAME = (?)";

```

```

        PreparedStatement st11 = conn.prepareStatement(sql11);
        st11.setString(1, currentLeaderSelect);
        st11.executeUpdate();

        JOptionPane.showMessageDialog(null,"New Leader Chosen
for Team 3!");
    }
}
new AdminWindow().setVisible(true);
dispose();

} catch (SQLException ex) {
    JOptionPane.showMessageDialog(null, "error");
}
}

private void jButton1ActionPerformed(java.awt.event.ActionEvent evt) {
    new AdminWindow().setVisible(true);
    dispose();
}

/**
 * @param args the command line arguments
 */
public static void main(String args[]) {
    /* Set the Nimbus look and feel */
    //<editor-fold defaultstate="collapsed" desc=" Look and feel setting code
(optional) ">
    /* If Nimbus (introduced in Java SE 6) is not available, stay with the default
look and feel.
    * For details see
http://download.oracle.com/javase/tutorial/uiswing/lookandfeel/plaf.html
    */
    try {
        for (javax.swing.UIManager.LookAndFeelInfo info :
javax.swing.UIManager.getInstalledLookAndFeels()) {
            if ("Nimbus".equals(info.getName())) {

javax.swing.UIManager.setLookAndFeel(info.getClassName());
                break;

            }
        }
    }
    catch (ClassNotFoundException ex) {

```

```

        java.util.logging.Logger.getLogger(TeamLeader.class
            .getName()).log(java.util.logging.Level.SEVERE, null, ex);

    } catch (InstantiationException ex) {
        java.util.logging.Logger.getLogger(TeamLeader.class
            .getName()).log(java.util.logging.Level.SEVERE, null, ex);

    } catch (IllegalAccessException ex) {
        java.util.logging.Logger.getLogger(TeamLeader.class
            .getName()).log(java.util.logging.Level.SEVERE, null, ex);

    } catch (javax.swing.UnsupportedLookAndFeelException ex) {
        java.util.logging.Logger.getLogger(TeamLeader.class
            .getName()).log(java.util.logging.Level.SEVERE, null, ex);
    }
}
//</editor-fold>

/* Create and display the form */
java.awt.EventQueue.invokeLater(new Runnable() {
    public void run() {
        new TeamLeader().setVisible(true);
    }
});
}

// Variables declaration - do not modify
private javax.swing.JButton ConfirmButton;
private javax.swing.JButton ConfirmLeaderButton;
private javax.swing.JComboBox<String> LeaderBox;
private javax.swing.JComboBox<String> TeamBox;
private javax.swing.JButton jButton1;
private javax.swing.JLabel jLabel1;
private javax.swing.JLabel jLabel2;
private javax.swing.JLabel jLabel3;
// End of variables declaration
}

```



## Team Leader Window

**TEAM LEADER PAGE**

Your Current Progress: 21

Ranking in Team: 2

Ranking in Club: 2

In-Team Ranking Club Ranking

Update Progress Back

Top Performer in Team: Cherry Chen

Worst Performer in Team: Gary

Team Overall Progress

```
import java.awt.Dimension;
import java.awt.Toolkit;
import java.sql.Array;
import java.sql.DriverManager;
import java.sql.PreparedStatement;
import java.sql.ResultSet;
import java.sql.SQLException;
import java.util.ArrayList;
import java.util.ListIterator;
import javax.swing.JOptionPane;
```

```
/*
 * To change this license header, choose License Headers in Project Properties.
 * To change this template file, choose Tools | Templates
 * and open the template in the editor.
 */
/**
 *
 * @author aidenyan
```

```

*/
public class TeamLeaderWindow extends javax.swing.JFrame {

    /**
     * Creates new form teamLeaderLogin
     */
    public TeamLeaderWindow() {
        initComponents();
        DirectDisplay();
        Toolkit toolkit = getToolkit();
        Dimension size = toolkit.getScreenSize();
        setLocation(size.width / 2 - getWidth() / 2, size.height / 2 - getHeight() / 2);

        //center the GUI
    }

    private void DirectDisplay() {
        try {
            String connectionURL =
"jdbc:derby://localhost:1527/QHAPPYDatabase";
            java.sql.Connection conn =
DriverManager.getConnection(connectionURL, "username", "password");

            String currentTeam = LoginWindow.teamUsed;
            String currentName = LoginWindow.nameUsing;

            //Display the corresponding messages (e.g., best/worst performer,
current rank&progress)
            if (currentTeam.equals("Team 1")) {
                String sql1 = "SELECT * FROM TEAM1_MEMBERS WHERE
NAME = (?)";
                PreparedStatement st1 = conn.prepareStatement(sql1);
                st1.setString(1, currentName);
                ResultSet rs1 = st1.executeQuery();
                while (rs1.next()) {
                    String theProgress =
String.valueOf(rs1.getInt("OVERALL_PROGRESS"));
                    CurrentProgressField.setText(theProgress);
                }

                //first order the point by descending order, so the first one is the
best performer
                //descending order also helps to generate the club ranking and
output the user's club rank

```

```

        String sql2 = "SELECT * FROM TEAM1_MEMBERS ORDER
BY POINTS desc";
        PreparedStatement st2 = conn.prepareStatement(sql2);
        ResultSet rs2 = st2.executeQuery();
        int bestRankKeeper = 0;
        int worstRankKeeper = 0;
        while (rs2.next()) {
            bestRankKeeper += 1;
            if (rs2.getString("NAME").equals(currentName)) {
                TeamRankField.setText(String.valueOf(bestRankKeeper));
            }
            if (bestRankKeeper == 1){
                TopPerformerField.setText(rs2.getString("NAME"));
            }
        }
    }

```

//ascending order to get the worst performer

```

        String sql3 = "SELECT * FROM TEAM1_MEMBERS ORDER
BY POINTS asc";
        PreparedStatement st3 = conn.prepareStatement(sql3);
        ResultSet rs3 = st3.executeQuery();
        while(rs3.next()){
            worstRankKeeper += 1;
            if(worstRankKeeper == 1){
                WorstPerformerField.setText(rs3.getString("NAME"));
            }
        }
    }

```

//repeat for team 2 and team 3, unassigned members do not need  
because there are no leaders for unassigned members

```

    } else if (currentTeam.equals("Team 2")) {
        String sql1 = "SELECT * FROM TEAM2_MEMBERS WHERE
NAME = (?)";
        PreparedStatement st1 = conn.prepareStatement(sql1);
        st1.setString(1, currentName);
        ResultSet rs1 = st1.executeQuery();
        while (rs1.next()) {
            String theProgress =
String.valueOf(rs1.getInt("OVERALL_PROGRESS"));

```

```

        CurrentProgressField.setText(theProgress);
    }
    String sql2 = "SELECT * FROM TEAM2_MEMBERS ORDER
BY POINTS desc";
    PreparedStatement st2 = conn.prepareStatement(sql2);
    ResultSet rs2 = st2.executeQuery();
    int rankKeeper = 0;
    while (rs2.next()) {
        rankKeeper += 1;
        if (rs2.getString("NAME").equals(currentName)) {
            TeamRankField.setText(String.valueOf(rankKeeper));
        }
    }
    rs2.beforeFirst();
    TopPerformerField.setText(rs2.getString("NAME"));

    String sql3 = "SELECT * FROM TEAM2_MEMBERS ORDER
BY POINTS asc";
    PreparedStatement st3 = conn.prepareStatement(sql3);
    ResultSet rs3 = st3.executeQuery();
    WorstPerformerField.setText(rs3.getString("NAME"));

    } else if (currentTeam.equals("Team 3")) {
        String sql1 = "SELECT * FROM TEAM3_MEMBERS WHERE
NAME = (?)";
        PreparedStatement st1 = conn.prepareStatement(sql1);
        st1.setString(1, currentName);
        ResultSet rs1 = st1.executeQuery();
        while (rs1.next()) {
            String theProgress =
String.valueOf(rs1.getInt("OVERALL_PROGRESS"));
            CurrentProgressField.setText(theProgress);
        }
        String sql2 = "SELECT * FROM TEAM3_MEMBERS ORDER
BY POINTS desc";
        PreparedStatement st2 = conn.prepareStatement(sql2);
        ResultSet rs2 = st2.executeQuery();
        int rankKeeper = 0;
        while (rs2.next()) {
            rankKeeper += 1;
            if (rs2.getString("NAME").equals(currentName)) {
                TeamRankField.setText(String.valueOf(rankKeeper));
            }
        }
    }

```

```

    }
    rs2.beforeFirst();
    TopPerformerField.setText(rs2.getString("NAME"));

    String sql3 = "SELECT * FROM TEAM3_MEMBERS ORDER
BY POINTS asc";
    PreparedStatement st3 = conn.prepareStatement(sql3);
    ResultSet rs3 = st3.executeQuery();
    WorstPerformerField.setText(rs3.getString("NAME"));

}

```

//generate the club ranking, in the same way with the ClubRanking UI  
(the swing table changed to an arraylist of array that contains the rank, the name, and  
the point)

```

String sql1 = "SELECT * FROM TEAM1_MEMBERS";
String sql2 = "SELECT * FROM TEAM2_MEMBERS";
String sql3 = "SELECT * FROM TEAM3_MEMBERS";
String sql4 = "SELECT * FROM UNASSIGNED_MEMBERS";

```

```

PreparedStatement st1 = conn.prepareStatement(sql1);
PreparedStatement st2 = conn.prepareStatement(sql2);
PreparedStatement st3 = conn.prepareStatement(sql3);
PreparedStatement st4 = conn.prepareStatement(sql4);

```

```

ResultSet rs1 = st1.executeQuery();
ResultSet rs2 = st2.executeQuery();
ResultSet rs3 = st3.executeQuery();
ResultSet rs4 = st4.executeQuery();

```

```

ArrayList<Integer> allPoint = new ArrayList<>();
ArrayList<String> UsedNames = new ArrayList<>();
ArrayList<String[]> theRank = new ArrayList<String[]>();

```

```

while (rs1.next()) {
    allPoint.add(rs1.getInt("POINTS"));
}
while (rs2.next()) {
    allPoint.add(rs2.getInt("POINTS"));
}
while (rs3.next()) {
    allPoint.add(rs3.getInt("POINTS"));
}

```

```

    }
    while (rs4.next()) {
        allPoint.add(rs4.getInt("POINTS"));
    }
    int arrLength = allPoint.size();
    int[] theArray = new int[arrLength];
    for (int i = 0; i < arrLength; i++) {
        theArray[i] = allPoint.get(i);
    }
    bubbleSort(theArray, arrLength);

    int clubRankKeeper = 0;

    for (int i = 0; i < arrLength; i++) {
        String sql5 = "SELECT * FROM TEAM1_MEMBERS WHERE
POINTS=?";
        String sql6 = "SELECT * FROM TEAM2_MEMBERS WHERE
POINTS=?";
        String sql7 = "SELECT * FROM TEAM3_MEMBERS WHERE
POINTS=?";
        String sql8 = "SELECT * FROM UNASSIGNED_MEMBERS
WHERE POINTS=?";

        PreparedStatement st5 = conn.prepareStatement(sql5);
        PreparedStatement st6 = conn.prepareStatement(sql6);
        PreparedStatement st7 = conn.prepareStatement(sql7);
        PreparedStatement st8 = conn.prepareStatement(sql8);

        st5.setInt(1, theArray[i]);
        st6.setInt(1, theArray[i]);
        st7.setInt(1, theArray[i]);
        st8.setInt(1, theArray[i]);

        ResultSet rs5 = st5.executeQuery();
        ResultSet rs6 = st6.executeQuery();
        ResultSet rs7 = st7.executeQuery();
        ResultSet rs8 = st8.executeQuery();

        String memberName = null;
        String theStats = String.valueOf(theArray[i]);

        while (rs5.next()) {
            boolean flag = false;
            memberName = rs5.getString("NAME");

```

```

        ListIterator<String> theCheck = UsedNames.listIterator();
        while (theCheck.hasNext()) {
            String comp = theCheck.next();
            if (comp.equals(memberName)) {
                flag = true;
            }
        }

        if (!flag) {
            clubRankKeeper += 1;
            String converted = String.valueOf(clubRankKeeper);
            String[] inputData = {converted, memberName,

theStats};

            theRank.add(inputData);
            UsedNames.add(memberName);
        }
    }
    while (rs6.next()) {
        boolean flag = false;
        memberName = rs6.getString("NAME");
        ListIterator<String> theCheck = UsedNames.listIterator();
        while (theCheck.hasNext()) {
            String comp = theCheck.next();
            if (comp.equals(memberName)) {
                flag = true;
            }
        }

        if (!flag) {
            clubRankKeeper += 1;
            String converted = String.valueOf(clubRankKeeper);
            String[] inputData = {converted, memberName,

theStats};

            theRank.add(inputData);
            UsedNames.add(memberName);
        }
    }
    while (rs7.next()) {
        boolean flag = false;
        memberName = rs7.getString("NAME");
        ListIterator<String> theCheck = UsedNames.listIterator();
        while (theCheck.hasNext()) {
            String comp = theCheck.next();
            if (comp.equals(memberName)) {

```

```

        flag = true;
    }
}

if (!flag) {
    clubRankKeeper += 1;
    String converted = String.valueOf(clubRankKeeper);
    String[] inputData = {converted, memberName,

theStats};

    theRank.add(inputData);
    UsedNames.add(memberName);
}
}
while (rs8.next()) {
    boolean flag = false;
    memberName = rs8.getString("NAME");
    ListIterator<String> theCheck = UsedNames.listIterator();
    while (theCheck.hasNext()) {
        String comp = theCheck.next();
        if (comp.equals(memberName)) {
            flag = true;
        }
    }

    if (!flag) {
        clubRankKeeper += 1;
        String converted = String.valueOf(clubRankKeeper);
        String[] inputData = {converted, memberName,

theStats};

        theRank.add(inputData);
        UsedNames.add(memberName);
    }
}
ListIterator<String[]> theCheck = theRank.listIterator();
while (theCheck.hasNext()) {
    String[] currentCheck = theCheck.next();
    if (currentCheck[1].equals(currentName)) {
        ClubRankField.setText(currentCheck[0]);
    }
}
}
} catch (SQLException ex) {
    JOptionPane.showMessageDialog(null, "error");
}

```



```

    }
}

```

```

private void bubbleSort(int arr[], int n) {
    /* a bubble sort method is needed since
    the whole club ranking would need to concatenate the
    databases together, rather than just using "ORDER BY" in
    sql commands for a single database
    */
    int i, j, temp;
    boolean swapped;
    for (i = 0; i < n - 1; i++) {
        swapped = false;
        for (j = 0; j < n - i - 1; j++) {
            if (arr[j] < arr[j + 1]) {
                temp = arr[j];
                arr[j] = arr[j + 1];
                arr[j + 1] = temp;
                swapped = true;
            }
        }

        if (swapped == false) {
            break;
        }
    }
}

```

```

/**
 * This method is called from within the constructor to initialize the form.
 * WARNING: Do NOT modify this code. The content of this method is always
 * regenerated by the Form Editor.
 */

```

```

@SuppressWarnings("unchecked")
// <editor-fold defaultstate="collapsed" desc="Generated Code">
private void initComponents() {

```

```

    jLabel1 = new javax.swing.JLabel();
    jLabel2 = new javax.swing.JLabel();
    jLabel3 = new javax.swing.JLabel();
    jLabel4 = new javax.swing.JLabel();
    TeamRankButton = new javax.swing.JButton();
    UpdateProgressButton = new javax.swing.JButton();

```

```

GetBackButton = new javax.swing.JButton();
ClubRankButton = new javax.swing.JButton();
jSeparator1 = new javax.swing.JSeparator();
jLabel5 = new javax.swing.JLabel();
jLabel6 = new javax.swing.JLabel();
TopPerformerField = new javax.swing.JTextField();
WorstPerformerField = new javax.swing.JTextField();
CurrentProgressField = new javax.swing.JTextField();
TeamRankField = new javax.swing.JTextField();
ClubRankField = new javax.swing.JTextField();
OverallProgressButton = new javax.swing.JButton();

setDefaultCloseOperation(javax.swing.WindowConstants.EXIT_ON_CLOSE);

jLabel1.setFont(new java.awt.Font("Lucida Grande", 1, 24)); // NOI18N
jLabel1.setText("TEAM LEADER PAGE");

jLabel2.setText("Your Current Progress:");

jLabel3.setText("Ranking in Team:");

jLabel4.setText("Ranking in Club:");

TeamRankButton.setText("In-Team Ranking");
TeamRankButton.addActionListener(new java.awt.event.ActionListener() {
    public void actionPerformed(java.awt.event.ActionEvent evt) {
        TeamRankButtonActionPerformed(evt);
    }
});

UpdateProgressButton.setText("Update Progress");
UpdateProgressButton.addActionListener(new
java.awt.event.ActionListener() {
    public void actionPerformed(java.awt.event.ActionEvent evt) {
        UpdateProgressButtonActionPerformed(evt);
    }
});

GetBackButton.setText("Back");
GetBackButton.addActionListener(new java.awt.event.ActionListener() {
    public void actionPerformed(java.awt.event.ActionEvent evt) {
        GetBackButtonActionPerformed(evt);
    }
}

```

```
});

ClubRankButton.setText("Club Ranking");
ClubRankButton.addActionListener(new java.awt.event.ActionListener() {
    public void actionPerformed(java.awt.event.ActionEvent evt) {
        ClubRankButtonActionPerformed(evt);
    }
});

jLabel5.setText("Top Performer in Team:");

jLabel6.setText("Worst Performer in Team:");

OverallProgressButton.setText("Team Overall Progress");
OverallProgressButton.addActionListener(new
java.awt.event.ActionListener() {
    public void actionPerformed(java.awt.event.ActionEvent evt) {
        OverallProgressButtonActionPerformed(evt);
    }
});

javax.swing.GroupLayout layout = new
javax.swing.GroupLayout(getContentPane());
getContentPane().setLayout(layout);
layout.setHorizontalGroup(

layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
    .addComponent(jSeparator1)
    .addGroup(layout.createSequentialGroup()
        .addGroup(layout.createParallelGroup(javax.swing.GroupLayout.
Alignment.LEADING)
            .addGroup(layout.createSequentialGroup()
                .addGap(123, 123, 123)
                .addComponent(jLabel1))
            .addGroup(layout.createSequentialGroup()
                .addGap(53, 53, 53)
                .addGroup(layout.createParallelGroup(javax.s
GroupLayout.Alignment.LEADING)
                    .addGroup(layout.createSequentialGroup()
                        .addComponent(TeamRankButton)
                        .addComponent(UpdateProgressButton,
javax.swing.GroupLayout.Alignment.LEADING)))
```

```

        .addGroup(layout.createParallelGroup(javax.s
wing.GroupLayout.Alignment.LEADING)
    ()
        .addGap(110, 110, 110)
        .addComponent(GetBackButton))
    ()
        .addGap(84, 84, 84)
        .addComponent(ClubRankButton)))
)
        .addGroup(layout.createSequentialGroup())
        .addGroup(layout.createParallelGroup(javax.s
wing.GroupLayout.Alignment.LEADING)
            .addComponent(jLabel2)
            .addComponent(jLabel3)
            .addComponent(jLabel4))
            .addGap(47, 47, 47)
            .addGroup(layout.createParallelGroup(javax.s
wing.GroupLayout.Alignment.LEADING, false)
                .addComponent(CurrentProgressField)
                .addComponent(TeamRankField)
                .addComponent(ClubRankField,
javax.swing.GroupLayout.DEFAULT_SIZE, 200, Short.MAX_VALUE))))
        .addGroup(layout.createSequentialGroup())
            .addGap(53, 53, 53)
            .addGroup(layout.createParallelGroup(javax.swing.Gro
upLayout.Alignment.TRAILING)
                .addGroup(layout.createSequentialGroup())
                    .addComponent(jLabel6)
                    .addGap(18, 18, 18)
                    .addComponent(WorstPerformerField,
javax.swing.GroupLayout.PREFERRED_SIZE, 190,
javax.swing.GroupLayout.PREFERRED_SIZE))
                .addGroup(layout.createSequentialGroup())
                    .addComponent(jLabel5)
                    .addGap(31, 31, 31)
                    .addComponent(TopPerformerField,
javax.swing.GroupLayout.PREFERRED_SIZE, 189,
javax.swing.GroupLayout.PREFERRED_SIZE))))
        .addGroup(layout.createSequentialGroup())
            .addGap(158, 158, 158)
            .addComponent(OverallProgressButton)))
        .addContainerGap(70, Short.MAX_VALUE))

```

```

    );
    layout.setVerticalGroup(

layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
        .addGroup(layout.createSequentialGroup()
            .addGap(27, 27, 27)
            .addComponent(jLabel1)
            .addGap(34, 34, 34)
            .addGroup(layout.createParallelGroup(javax.swing.GroupLayout.
Alignment.BASELINE)
                .addComponent(jLabel2)
                .addComponent(CurrentProgressField,
javax.swing.GroupLayout.PREFERRED_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE,
javax.swing.GroupLayout.PREFERRED_SIZE))
            .addGap(18, 18, 18)
            .addGroup(layout.createParallelGroup(javax.swing.GroupLayout.
Alignment.BASELINE)
                .addComponent(jLabel3)
                .addComponent(TeamRankField,
javax.swing.GroupLayout.PREFERRED_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE,
javax.swing.GroupLayout.PREFERRED_SIZE))
            .addGap(18, 18, 18)
            .addGroup(layout.createParallelGroup(javax.swing.GroupLayout.
Alignment.BASELINE)
                .addComponent(jLabel4)
                .addComponent(ClubRankField,
javax.swing.GroupLayout.PREFERRED_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE,
javax.swing.GroupLayout.PREFERRED_SIZE))
            .addGap(37, 37, 37)
            .addGroup(layout.createParallelGroup(javax.swing.GroupLayout.
Alignment.BASELINE)
                .addComponent(TeamRankButton)
                .addComponent(ClubRankButton))
            .addGap(18, 18, 18)
            .addGroup(layout.createParallelGroup(javax.swing.GroupLayout.
Alignment.BASELINE)
                .addComponent(UpdateProgressButton)
                .addComponent(GetBackButton))
            .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement
.UNRELATED)
            .addComponent(jSeparator1,

```

```

javax.swing.GroupLayout.PREFERRED_SIZE, 10,
javax.swing.GroupLayout.PREFERRED_SIZE)
        .addGap(18, 18, 18)
        .addGroup(layout.createParallelGroup(javax.swing.GroupLayout.
Alignment.BASELINE)
        .addComponent(jLabel5)
        .addComponent(TopPerformerField,
javax.swing.GroupLayout.PREFERRED_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE,
javax.swing.GroupLayout.PREFERRED_SIZE))
        .addGap(18, 18, 18)
        .addGroup(layout.createParallelGroup(javax.swing.GroupLayout.
Alignment.BASELINE)
        .addComponent(jLabel6)
        .addComponent(WorstPerformerField,
javax.swing.GroupLayout.PREFERRED_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE,
javax.swing.GroupLayout.PREFERRED_SIZE))
        .addGap(18, 18, 18)
        .addComponent(OverallProgressButton)
        .addContainerGap(17, Short.MAX_VALUE))
    );

```

```

        pack();
    } // </editor-fold>

```

```

private void GetBackButtonActionPerformed(java.awt.event.ActionEvent evt) {
    new LoginWindow().setVisible(true);
    dispose();
}

```

```

private void TeamRankButtonActionPerformed(java.awt.event.ActionEvent evt)
{
    new TeamRanking().setVisible(true);
    dispose();
}

```

```

private void ClubRankButtonActionPerformed(java.awt.event.ActionEvent evt)
{
    new ClubRanking().setVisible(true);
    dispose();
}

```

```

private void UpdateProgressButtonActionPerformed(java.awt.event.ActionEvent

```

```

    evt) {
        new UpdateProgress().setVisible(true);
        dispose();
    }

    private void OverallProgressButtonActionPerformed(java.awt.event.ActionEvent
    evt) {
        new TeamProgress().setVisible(true);
        dispose();
    }

    /**
     * @param args the command line arguments
     */
    public static void main(String args[]) {
        /* Set the Nimbus look and feel */
        //<editor-fold defaultstate="collapsed" desc=" Look and feel setting code
        (optional) ">
        /* If Nimbus (introduced in Java SE 6) is not available, stay with the default
        look and feel.
         * For details see
        http://download.oracle.com/javase/tutorial/uiswing/lookandfeel/plaf.html
         */
        try {
            for (javax.swing.UIManager.LookAndFeelInfo info :
            javax.swing.UIManager.getInstalledLookAndFeels()) {
                if ("Nimbus".equals(info.getName())) {

                    javax.swing.UIManager.setLookAndFeel(info.getClassName());
                    break;

                }
            }
        } catch (ClassNotFoundException ex) {
            java.util.logging.Logger.getLogger(TeamLeaderWindow.class
            .getName()).log(java.util.logging.Level.SEVERE, null, ex);

        } catch (InstantiationException ex) {
            java.util.logging.Logger.getLogger(TeamLeaderWindow.class
            .getName()).log(java.util.logging.Level.SEVERE, null, ex);

        } catch (IllegalAccessException ex) {
            java.util.logging.Logger.getLogger(TeamLeaderWindow.class
            .getName()).log(java.util.logging.Level.SEVERE, null, ex);

```

```

    } catch (javax.swing.UnsupportedLookAndFeelException ex) {
        java.util.logging.Logger.getLogger(TeamLeaderWindow.class
            .getName()).log(java.util.logging.Level.SEVERE, null, ex);
    }
//</editor-fold>
//</editor-fold>
//</editor-fold>
//</editor-fold>

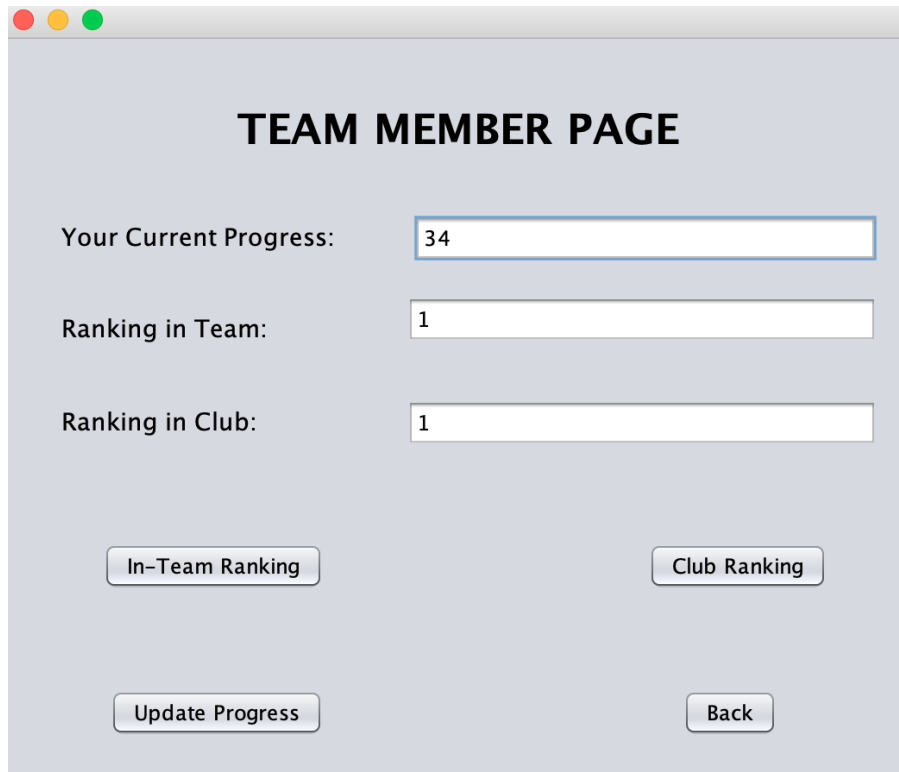
/* Create and display the form */
java.awt.EventQueue.invokeLater(new Runnable() {
    public void run() {
        new TeamLeaderWindow().setVisible(true);
    }
});
}

// Variables declaration - do not modify
private javax.swing.JButton ClubRankButton;
private javax.swing.JTextField ClubRankField;
private javax.swing.JTextField CurrentProgressField;
private javax.swing.JButton GetBackButton;
private javax.swing.JButton OverallProgressButton;
private javax.swing.JButton TeamRankButton;
private javax.swing.JTextField TeamRankField;
private javax.swing.JTextField TopPerformerField;
private javax.swing.JButton UpdateProgressButton;
private javax.swing.JTextField WorstPerformerField;
private javax.swing.JLabel jLabel1;
private javax.swing.JLabel jLabel2;
private javax.swing.JLabel jLabel3;
private javax.swing.JLabel jLabel4;
private javax.swing.JLabel jLabel5;
private javax.swing.JLabel jLabel6;
private javax.swing.JSeparator jSeparator1;
// End of variables declaration
}

```



## Team Member Window



**TEAM MEMBER PAGE**

Your Current Progress:

Ranking in Team:

Ranking in Club:

\*Because the code of Team Member Window is basically the same as the Team Leader Window (with team leader window having some more functions), so I won't have annotations on the source code below:

```
import java.awt.Dimension;
import java.awt.Toolkit;
import java.sql.DriverManager;
import java.sql.PreparedStatement;
import java.sql.ResultSet;
import java.sql.SQLException;
import java.util.logging.Level;
import java.util.logging.Logger;
import javax.swing.JOptionPane;
```

```
/*
 * To change this license header, choose License Headers in Project Properties.
 * To change this template file, choose Tools | Templates
 * and open the template in the editor.
 */
/**
 *
 * @author aidenyan
 */
```

```

public class LoginWindow extends javax.swing.JFrame {

    /**
     * preliminary static variables that would be helpful for further logistics
     * in the programs
     *
     */
    public static String nameUsing;
    static String status;
    public static String teamUsed = null;

    /**
     * Creates new form LoginWindow
     */
    public LoginWindow() {
        initComponents();
        Toolkit toolkit = getToolkit();
        Dimension size = toolkit.getScreenSize();
        setLocation(size.width / 2 - getWidth() / 2, size.height / 2 - getHeight() / 2);
//center the GUI

    }

    /**
     * This method is called from within the constructor to initialize the form.
     * WARNING: Do NOT modify this code. The content of this method is always
     * regenerated by the Form Editor.
     */
    @SuppressWarnings("unchecked")
    // <editor-fold defaultstate="collapsed" desc="Generated Code">
    private void initComponents() {

        jPanel1 = new javax.swing.JPanel();
        jLabel1 = new javax.swing.JLabel();
        jLabel2 = new javax.swing.JLabel();
        jLabel3 = new javax.swing.JLabel();
        UsernameField = new javax.swing.JTextField();
        PasswordField = new javax.swing.JPasswordField();
        ResetPasswordButton = new javax.swing.JButton();
        LoginButton = new javax.swing.JButton();

        setDefaultCloseOperation(javax.swing.WindowConstants.EXIT_ON_CLOSE);

```

```

jPanel1.setForeground(new java.awt.Color(255, 153, 0));
jPanel1.setLocation(new java.awt.Point(0, 0));

jLabel1.setFont(new java.awt.Font("Lucida Grande", 0, 24)); // NOI18N
jLabel1.setText("Welcome to QHAPPY Online System!");

jLabel2.setFont(new java.awt.Font("Lucida Grande", 0, 14)); // NOI18N
jLabel2.setText("Username:");

jLabel3.setFont(new java.awt.Font("Lucida Grande", 0, 14)); // NOI18N
jLabel3.setText("Password:");

UsernameField.addActionListener(new java.awt.event.ActionListener() {
    public void actionPerformed(java.awt.event.ActionEvent evt) {
        UsernameFieldActionPerformed(evt);
    }
});

ResetPasswordButton.setText("Forget Password");
ResetPasswordButton.addActionListener(new
java.awt.event.ActionListener() {
    public void actionPerformed(java.awt.event.ActionEvent evt) {
        ResetPasswordButtonActionPerformed(evt);
    }
});

LoginButton.setBackground(new java.awt.Color(0, 153, 255));
LoginButton.setText("Login");
LoginButton.addActionListener(new java.awt.event.ActionListener() {
    public void actionPerformed(java.awt.event.ActionEvent evt) {
        LoginButtonActionPerformed(evt);
    }
});

javax.swing.GroupLayout jPanel1Layout = new
javax.swing.GroupLayout(jPanel1);
jPanel1.setLayout(jPanel1Layout);
jPanel1Layout.setHorizontalGroup(

jPanel1Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
    .addGroup(javax.swing.GroupLayout.Alignment.TRAILING,
jPanel1Layout.createSequentialGroup()
        .addContainerGap()
        .addGroup(jPanel1Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.TRAILING)
            .addComponent(jLabel3, javax.swing.GroupLayout.DEFAULT_SIZE, 150, true)
            .addComponent(jLabel2, javax.swing.GroupLayout.DEFAULT_SIZE, 150, true)
            .addComponent(jLabel1, javax.swing.GroupLayout.DEFAULT_SIZE, 150, true)
            .addComponent(LoginButton, javax.swing.GroupLayout.DEFAULT_SIZE, 150, true)
            .addComponent(ResetPasswordButton, javax.swing.GroupLayout.DEFAULT_SIZE, 150, true)
        )
        .addContainerGap(150, true)
    )
);

```

```

        .addGroup(jPanel1Layout.createSequentialGroup()
            .addContainerGap(javax.swing.GroupLayout.DEFAULT
T_SIZE, Short.MAX_VALUE)
            .addComponent(jLabel1,
javax.swing.GroupLayout.PREFERRED_SIZE, 431,
javax.swing.GroupLayout.PREFERRED_SIZE))
        .addGroup(jPanel1Layout.createSequentialGroup()
            .addGap(87, 87, 87)
            .addGroup(jPanel1Layout.createParallelGroup(javax.sw
ing.GroupLayout.Alignment.LEADING)
                .addComponent(jLabel3)
                .addComponent(jLabel2))
            .addPreferredGap(javax.swing.LayoutStyle.Component
Placement.RELATED, 84, Short.MAX_VALUE)
            .addGroup(jPanel1Layout.createParallelGroup(javax.sw
ing.GroupLayout.Alignment.LEADING)
                .addComponent(UsernameField,
javax.swing.GroupLayout.Alignment.TRAILING,
javax.swing.GroupLayout.PREFERRED_SIZE, 293,
javax.swing.GroupLayout.PREFERRED_SIZE)
                .addComponent(PasswordField,
javax.swing.GroupLayout.Alignment.TRAILING,
javax.swing.GroupLayout.PREFERRED_SIZE, 293,
javax.swing.GroupLayout.PREFERRED_SIZE))))
            .addGap(88, 88, 88))
        .addGroup(javax.swing.GroupLayout.Alignment.TRAILING,
jPanel1Layout.createSequentialGroup()
            .addGap(0, 0, Short.MAX_VALUE)
            .addComponent(ResetPasswordButton)
            .addGap(137, 137, 137))
        .addGroup(jPanel1Layout.createParallelGroup(javax.swing.GroupLay
out.Alignment.LEADING)
            .addGroup(jPanel1Layout.createSequentialGroup()
                .addGap(133, 133, 133)
                .addComponent(LoginButton)
                .addContainerGap(413, Short.MAX_VALUE)))
    );
    jPanel1Layout.setVerticalGroup(

jPanel1Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
    .addGroup(jPanel1Layout.createSequentialGroup()
        .addGap(54, 54, 54)
        .addComponent(jLabel1,
javax.swing.GroupLayout.PREFERRED_SIZE, 86,

```

```

javax.swing.GroupLayout.PREFERRED_SIZE)
    .addGap(75, 75, 75)
    .addGroup(jPanel1Layout.createParallelGroup(javax.swing.Grou
pLayout.Alignment.BASELINE)
        .addComponent(jLabel2)
        .addComponent(UsernameField,
javax.swing.GroupLayout.PREFERRED_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE,
javax.swing.GroupLayout.PREFERRED_SIZE))
    .addGap(35, 35, 35)
    .addGroup(jPanel1Layout.createParallelGroup(javax.swing.Grou
pLayout.Alignment.BASELINE)
        .addComponent(jLabel3)
        .addComponent>PasswordField,
javax.swing.GroupLayout.PREFERRED_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE,
javax.swing.GroupLayout.PREFERRED_SIZE))
    .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement
.RELATED, 73, Short.MAX_VALUE)
    .addComponent(ResetPasswordButton)
    .addGap(63, 63, 63))
    .addGroup(jPanel1Layout.createParallelGroup(javax.swing.Grou
pLayout.Alignment.LEADING)
        .addGroup(javax.swing.GroupLayout.Alignment.TRAILING,
jPanel1Layout.createSequentialGroup()
            .addContainerGap(376, Short.MAX_VALUE)
            .addComponent(LoginButton)
            .addGap(62, 62, 62)))
    );

    javax.swing.GroupLayout layout = new
javax.swing.GroupLayout(getContentPane());
    getContentPane().setLayout(layout);
    layout.setHorizontalGroup(

layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
        .addGroup(javax.swing.GroupLayout.Alignment.TRAILING,
layout.createSequentialGroup()
            .addContainerGap(82, Short.MAX_VALUE)
            .addComponent(jPanel1,
javax.swing.GroupLayout.PREFERRED_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE,
javax.swing.GroupLayout.PREFERRED_SIZE)
            .addGap(89, 89, 89))

```

```

    );
    layout.setVerticalGroup(

layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
        .addGroup(layout.createSequentialGroup()
            .addGap(60, 60, 60)
            .addComponent(jPanel1,
javax.swing.GroupLayout.PREFERRED_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE,
javax.swing.GroupLayout.PREFERRED_SIZE)
            .addContainerGap(44, Short.MAX_VALUE))
        );

    pack();
} // </editor-fold>

private void UsernameFieldActionPerformed(java.awt.event.ActionEvent evt) {
    // TODO add your handling code here:
}

private void ResetPasswordButtonActionPerformed(java.awt.event.ActionEvent
evt) {
    new ResetPassword().setVisible(true);
    dispose();
}

private void LoginButtonActionPerformed(java.awt.event.ActionEvent evt) {
    try {
        //connect to database
        String connectionURL =
"jdbc:derby://localhost:1527/QHAPPYDatabase";
        java.sql.Connection conn =
DriverManager.getConnection(connectionURL, "username", "password");

        //initiate sql orders that would fetch member login information from
the different tables in the database
        String sql1 = "Select * from TEAM_MEMBER_LOGIN where
QDNUMBER=(?) and PASSWORD=(?)";
        String sql2 = "Select * from TEAM_LEADER_LOGIN where
QDNUMBER=(?) and PASSWORD=(?)";
        String sql3 = "Select * from ADMIN_LOGIN where
QDNUMBER=(?) and PASSWORD=(?)";

        PreparedStatement pst1 = conn.prepareStatement(sql1);

```

```

PreparedStatement pst2 = conn.prepareStatement(sql2);
PreparedStatement pst3 = conn.prepareStatement(sql3);

String userName = UsernameField.getText();
String passWord = PasswordField.getText();

//put into the data that needs to be searched in the selected tables in my
SQL orders
pst1.setString(1, userName);
pst1.setString(2, passWord);

pst2.setString(1, userName);
pst2.setString(2, passWord);

pst3.setString(1, userName);
pst3.setString(2, passWord);

//comprise a resultset of the search results of the match between
QDNumbers and Passwords
ResultSet rs1 = pst1.executeQuery();
ResultSet rs2 = pst2.executeQuery();
ResultSet rs3 = pst3.executeQuery();

//set some preliminary boolean variables for further identification
boolean teamMemberLogin = false;
boolean teamLeaderLogin = false;
boolean adminLogin = false;
boolean found = false;

while (rs1.next()) {
    String getUsername = rs1.getString("QDNumber");
    String getPassword = rs1.getString("Password");
    String getName = rs1.getString("Name");
    //re-check if usernames and passwords match
    if (userName.equals(getUsername) &&
passWord.equals(getPassword)) {

        String sql4 = "SELECT * FROM TEAM1_MEMBERS
WHERE NAME = (?)";
        String sql5 = "SELECT * FROM TEAM2_MEMBERS
WHERE NAME = (?)";
        String sql6 = "SELECT * FROM TEAM3_MEMBERS
WHERE NAME = (?)";
        String sql7 = "SELECT * FROM

```

```

UNASSIGNED_MEMBERS WHERE NAME = (?)";

        PreparedStatement st4 = conn.prepareStatement(sql4);
        PreparedStatement st5 = conn.prepareStatement(sql5);
        PreparedStatement st6 = conn.prepareStatement(sql6);
        PreparedStatement st7 = conn.prepareStatement(sql7);
        st4.setString(1, getName);
        st5.setString(1, getName);
        st6.setString(1, getName);
        st7.setString(1, getName);

        ResultSet rs4 = st4.executeQuery();
        ResultSet rs5 = st5.executeQuery();
        ResultSet rs6 = st6.executeQuery();
        ResultSet rs7 = st7.executeQuery();
        while (rs4.next()) {
            teamUsed = "Team 1";
        }
        while (rs5.next()) {
            teamUsed = "Team 2";
        }
        while (rs6.next()) {
            teamUsed = "Team 3";
        }
        while (rs7.next()) {
            teamUsed = "Unassigned";
        }

        nameUsing = getName;
        status = "member";
        found = true;
        JOptionPane.showMessageDialog(null, "Hi " + nameUsing +
", Welcome to the System");
        teamMemberLogin = true;
    }
}

while (rs2.next()) {
    String getUsername = rs2.getString("QDNumber");
    String getPassword = rs2.getString("Password");
    String getName = rs2.getString("Name");
    if (passWord.equals(getPassword) &&
userName.equals(getUsername)) {
        String sql4 = "SELECT * FROM TEAM1_MEMBERS
WHERE NAME = (?)";

```



```

        String sql5 = "SELECT * FROM TEAM2_MEMBERS
WHERE NAME = (?)";
        String sql6 = "SELECT * FROM TEAM3_MEMBERS
WHERE NAME = (?)";
        PreparedStatement st4 = conn.prepareStatement(sql4);
        PreparedStatement st5 = conn.prepareStatement(sql5);
        PreparedStatement st6 = conn.prepareStatement(sql6);
        st4.setString(1, getName);
        st5.setString(1, getName);
        st6.setString(1, getName);

        ResultSet rs4 = st4.executeQuery();
        ResultSet rs5 = st5.executeQuery();
        ResultSet rs6 = st6.executeQuery();
        while (rs4.next()) {
            teamUsed = "Team 1";
        }
        while (rs5.next()) {
            teamUsed = "Team 2";
        }
        while (rs6.next()) {
            teamUsed = "Team 3";
        }

        nameUsing = getName;
        status = "leader";
        found = true;
        teamLeaderLogin = true;
        JOptionPane.showMessageDialog(null, "Hi " + nameUsing +
", Welcome to the System");
    }
}

while (rs3.next()) {
    String getUsername = rs3.getString("QDNumber");
    String getPassword = rs3.getString("Password");
    String getName = rs3.getString("Name");
    if (userName.equals(getUsername) &&
passWord.equals(getPassword)) {
        nameUsing = getName;
        status = "admin";
        found = true;
        adminLogin = true;
        JOptionPane.showMessageDialog(null, "Hi " + nameUsing +

```

```

", Welcome to the System");
    }
}

/**
 * identify the types of users' status (member, leader, or
 * administrator) displace them into the corresponding navigation
 * GUIs
 */
if (teamMemberLogin == true) {
    new TeamMemberWindow().setVisible(true);
    dispose();
}

if (teamLeaderLogin == true) {
    new TeamLeaderWindow().setVisible(true);
    dispose();
}

if (adminLogin == true) {
    new AdminWindow().setVisible(true);
    dispose();
}

//show error message if username and password don't match
if (found != true) {
    JOptionPane.showMessageDialog(null, "Incorrect Username and
Password!");
}

} catch (SQLException ex) {
    JOptionPane.showMessageDialog(null, "Error");
}
}

/**
 * @param args the command line arguments
 */
public static void main(String args[]) {
    /* Set the Nimbus look and feel */
    //<editor-fold defaultstate="collapsed" desc=" Look and feel setting code
(optional) ">
    /* If Nimbus (introduced in Java SE 6) is not available, stay with the default

```

look and feel.

\* For details see

<http://download.oracle.com/javase/tutorial/uiswing/lookandfeel/plaf.html>

```
*/
try {
    for (javax.swing.UIManager.LookAndFeelInfo info :
        javax.swing.UIManager.getInstalledLookAndFeels()) {
        if ("Nimbus".equals(info.getName())) {
            javax.swing.UIManager.setLookAndFeel(info.getClassName());
            break;
        }
    }
} catch (ClassNotFoundException ex) {

java.util.logging.Logger.getLogger(LoginWindow.class.getName()).log(java.util.loggi
ng.Level.SEVERE, null, ex);
    } catch (InstantiationException ex) {

java.util.logging.Logger.getLogger(LoginWindow.class.getName()).log(java.util.loggi
ng.Level.SEVERE, null, ex);
    } catch (IllegalAccessException ex) {

java.util.logging.Logger.getLogger(LoginWindow.class.getName()).log(java.util.loggi
ng.Level.SEVERE, null, ex);
    } catch (javax.swing.UnsupportedLookAndFeelException ex) {

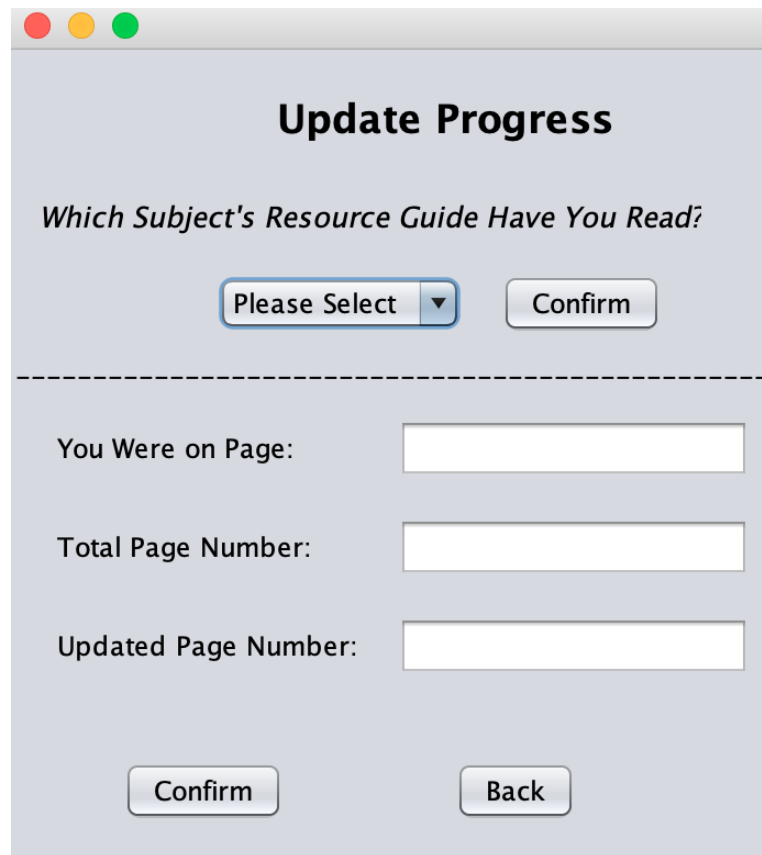
java.util.logging.Logger.getLogger(LoginWindow.class.getName()).log(java.util.loggi
ng.Level.SEVERE, null, ex);
    }
}
//</editor-fold>

/* Create and display the form */
java.awt.EventQueue.invokeLater(new Runnable() {
    public void run() {
        new LoginWindow().setVisible(true);
    }
});
}

// Variables declaration - do not modify
private javax.swing.JButton LoginButton;
private javax.swing.JPasswordField PasswordField;
private javax.swing.JButton ResetPasswordButton;
```

```
private javax.swing.JTextField UsernameField;  
private javax.swing.JLabel jLabel1;  
private javax.swing.JLabel jLabel2;  
private javax.swing.JLabel jLabel3;  
private javax.swing.JPanel jPanel1;  
// End of variables declaration  
}
```

## Update Progress



**Update Progress**

*Which Subject's Resource Guide Have You Read?*

Please Select ▼ Confirm

---

You Were on Page:

Total Page Number:

Updated Page Number:

Confirm Back

```
import java.awt.Dimension;
import java.awt.Toolkit;
import java.sql.DriverManager;
import java.sql.PreparedStatement;
import java.sql.ResultSet;
import java.sql.SQLException;
import javax.swing.JOptionPane;
```

```
/*
 * To change this license header, choose License Headers in Project Properties.
 * To change this template file, choose Tools | Templates
 * and open the template in the editor.
 */
/**
 *
 * @author aidenyan
 */
public class UpdateProgress extends javax.swing.JFrame {
```

```
/**
 * Creates new form UpdateProgress
```

```

    */
private String subject;

public UpdateProgress() {
    initComponents();
    Toolkit toolkit = getToolkit();
    Dimension size = toolkit.getScreenSize();
    setLocation(size.width / 2 - getWidth() / 2, size.height / 2 - getHeight() / 2);
//center the GUI

}

/**
 * This method is called from within the constructor to initialize the form.
 * WARNING: Do NOT modify this code. The content of this method is always
 * regenerated by the Form Editor.
 */
@SuppressWarnings("unchecked")
// <editor-fold defaultstate="collapsed" desc="Generated Code">
private void initComponents() {

    jLabel1 = new javax.swing.JLabel();
    jLabel2 = new javax.swing.JLabel();
    SubjectBox = new javax.swing.JComboBox<>();
    ConfirmUpdateButton = new javax.swing.JButton();
    jLabel3 = new javax.swing.JLabel();
    jLabel4 = new javax.swing.JLabel();
    jLabel5 = new javax.swing.JLabel();
    jLabel6 = new javax.swing.JLabel();
    NewPageField = new javax.swing.JTextField();
    GetBackButton = new javax.swing.JButton();
    ConfirmBookButton = new javax.swing.JButton();
    WerePageField = new javax.swing.JTextField();
    TotPageField = new javax.swing.JTextField();

    setDefaultCloseOperation(javax.swing.WindowConstants.EXIT_ON_CLOSE);

    jLabel1.setFont(new java.awt.Font("Lucida Grande", 1, 18)); // NOI18N
    jLabel1.setText("Update Progress");

    jLabel2.setFont(new java.awt.Font("Lucida Grande", 2, 13)); // NOI18N
    jLabel2.setText("Which Subject's Resource Guide Have You Read?");

```

```

        SubjectBox.setModel(new javax.swing.DefaultComboBoxModel<>(new
String[] { "Please Select", "Literature", "Art", "Science", "Social Science", "Music",
"Mathematics", "Economics" }));

        ConfirmUpdateButton.setText("Confirm");
        ConfirmUpdateButton.addActionListener(new
java.awt.event.ActionListener() {
            public void actionPerformed(java.awt.event.ActionEvent evt) {
                ConfirmUpdateButtonActionPerformed(evt);
            }
        });

        jLabel3.setText("-----");

        jLabel4.setText("You Were on Page:");

        jLabel5.setText("Total Page Number:");

        jLabel6.setText("Updated Page Number:");

        GetBackButton.setText("Back");
        GetBackButton.addActionListener(new java.awt.event.ActionListener() {
            public void actionPerformed(java.awt.event.ActionEvent evt) {
                GetBackButtonActionPerformed(evt);
            }
        });

        ConfirmBookButton.setText("Confirm");
        ConfirmBookButton.addActionListener(new
java.awt.event.ActionListener() {
            public void actionPerformed(java.awt.event.ActionEvent evt) {
                ConfirmBookButtonActionPerformed(evt);
            }
        });

        javax.swing.GroupLayout layout = new
javax.swing.GroupLayout(getContentPane());
        getContentPane().setLayout(layout);
        layout.setHorizontalGroup(

layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
            .addGroup(layout.createSequentialGroup()
                .addContainerGap()
                .addComponent(jLabel1, javax.swing.GroupLayout.DEFAULT_SIZE,
180, true)
                .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED)
                .addComponent(jLabel2, javax.swing.GroupLayout.DEFAULT_SIZE,
180, true)
                .addGap(10, 10, 10)
                .addComponent(jLabel3, javax.swing.GroupLayout.DEFAULT_SIZE,
180, true)
                .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.UNRELATED)
                .addComponent(jLabel4, javax.swing.GroupLayout.DEFAULT_SIZE,
180, true)
                .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.UNRELATED)
                .addComponent(jLabel5, javax.swing.GroupLayout.DEFAULT_SIZE,
180, true)
                .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.UNRELATED)
                .addComponent(jLabel6, javax.swing.GroupLayout.DEFAULT_SIZE,
180, true)
                .addGap(10, 10, 10)
                .addComponent(GetBackButton, javax.swing.GroupLayout.DEFAULT
SIZE, 100, true)
                .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.UNRELATED)
                .addComponent(ConfirmUpdateButton, javax.swing.GroupLayout.D
EFAULT_SIZE, 100, true)
                .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.UNRELATED)
                .addComponent(ConfirmBookButton, javax.swing.GroupLayout.DEF
AULT_SIZE, 100, true)
            )
        );
    }

    private void GetBackButtonActionPerformed(java.awt.event.ActionEvent evt)
{
        // TODO add your handling code here:
    }

    private void ConfirmUpdateButtonActionPerformed(java.awt.event.ActionEve
nt evt) {
        // TODO add your handling code here:
    }

    private void ConfirmBookButtonActionPerformed(java.awt.event.ActionEve
nt evt) {
        // TODO add your handling code here:
    }
}

```

```

        .addGroup(javax.swing.GroupLayout.Alignment.TRAILING
, layout.createSequentialGroup()
        .addGap(0, 4, Short.MAX_VALUE)
        .addComponent(jLabel3))
        .addGroup(layout.createSequentialGroup()
        .addGroup(layout.createParallelGroup(javax.swing.Gro
upLayout.Alignment.LEADING)
        .addGroup(layout.createSequentialGroup()
        .addGap(123, 123, 123)
        .addComponent(jLabel1))
        .addGroup(layout.createSequentialGroup()
        .addGap(15, 15, 15)
        .addComponent(jLabel2))
        .addGroup(layout.createSequentialGroup()
        .addGap(53, 53, 53)
        .addComponent(ConfirmUpdateButton)
        .addGap(78, 78, 78)
        .addComponent(GetBackButton)))
        .addGap(0, 0, Short.MAX_VALUE)))
        .addContainerGap())
        .addGroup(layout.createSequentialGroup()
        .addGap(23, 23, 23)
        .addGroup(layout.createParallelGroup(javax.swing.GroupLayout.
Alignment.LEADING)
        .addComponent(jLabel6)
        .addComponent(jLabel4)
        .addComponent(jLabel5))
        .addGap(18, 18, 18)
        .addGroup(layout.createParallelGroup(javax.swing.GroupLayout.
Alignment.LEADING, false)
        .addComponent(NewPageField,
javax.swing.GroupLayout.DEFAULT_SIZE, 160, Short.MAX_VALUE)
        .addComponent(WerePageField)
        .addComponent(TotPageField))
        .addContainerGap(javax.swing.GroupLayout.DEFAULT_SIZE,
Short.MAX_VALUE))
        .addGroup(layout.createSequentialGroup()
        .addContainerGap(javax.swing.GroupLayout.DEFAULT_SIZE,
Short.MAX_VALUE)
        .addComponent(SubjectBox,
javax.swing.GroupLayout.PREFERRED_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE,
javax.swing.GroupLayout.PREFERRED_SIZE)
        .addGap(18, 18, 18)

```



```

        .addComponent(ConfirmBookButton)
        .addGap(55, 55, 55))
    );
    layout.setVerticalGroup(

layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
        .addGroup(layout.createSequentialGroup()
            .addGap(20, 20, 20)
            .addComponent(jLabel1)
            .addGap(26, 26, 26)
            .addComponent(jLabel2)
            .addGap(18, 18, 18)
            .addGroup(layout.createParallelGroup(javax.swing.GroupLayout.
Alignment.BASELINE)
                .addComponent(SubjectBox,
javax.swing.GroupLayout.PREFERRED_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE,
javax.swing.GroupLayout.PREFERRED_SIZE)
                .addComponent(ConfirmBookButton))
            .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement
.UNRELATED)
            .addComponent(jLabel3)
            .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement
.UNRELATED)
            .addGroup(layout.createParallelGroup(javax.swing.GroupLayout.
Alignment.BASELINE)
                .addComponent(jLabel4)
                .addComponent(WerePageField,
javax.swing.GroupLayout.PREFERRED_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE,
javax.swing.GroupLayout.PREFERRED_SIZE))
            .addGap(18, 18, 18)
            .addGroup(layout.createParallelGroup(javax.swing.GroupLayout.
Alignment.BASELINE)
                .addComponent(jLabel5)
                .addComponent(TotPageField,
javax.swing.GroupLayout.PREFERRED_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE,
javax.swing.GroupLayout.PREFERRED_SIZE))
            .addGap(18, 18, 18)
            .addGroup(layout.createParallelGroup(javax.swing.GroupLayout.
Alignment.BASELINE)
                .addComponent(jLabel6)
                .addComponent(NewPageField,

```

```

javax.swing.GroupLayout.PREFERRED_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE,
javax.swing.GroupLayout.PREFERRED_SIZE))
        .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement
.RELATED, 39, Short.MAX_VALUE)
        .addGroup(layout.createParallelGroup(javax.swing.GroupLayout.
Alignment.BASELINE)
            .addComponent(ConfirmUpdateButton)
            .addComponent(GetBackButton))
        .addGap(23, 23, 23))
    );

    pack();
} // </editor-fold>

```

```

private void ConfirmUpdateButtonActionPerformed(java.awt.event.ActionEvent
evt) {
    String currentUser = LoginWindow.nameUsing;
    String currentTeam = LoginWindow.teamUsed;
    try {

```

```

        String connectionURL =
        "jdbc:derby://localhost:1527/QHAPPYDatabase";
        java.sql.Connection conn =
        DriverManager.getConnection(connectionURL, "username", "password");
        if (subject.equals("Literature")) {

```

```

//initiate the local variables

```

```

        int allCurrentProgress = 0;
        int allPageNum = 0;
        ResultSet rs2 = null;

```

```

        String sql1 = "SELECT * FROM PAGE_NUM WHERE
LINE_NUM = (?)";

```

```

        PreparedStatement st1 = conn.prepareStatement(sql1);
        st1.setInt(1, 1);
        ResultSet rs1 = st1.executeQuery();

```

```

//get the total number of page for literature

```

```

        if (LoginWindow.teamUsed.equals("Team 1")) {

```

```

            int nowPage = Integer.valueOf(NewPageField.getText());

```

```

int oriPage = Integer.valueOf(WerePageField.getText());
int totPage = Integer.valueOf(TotPageField.getText());

//validating the input
if (nowPage < oriPage) {
    JOptionPane.showMessageDialog(null, "YOU READ
BACKWARDS?");
} else if (nowPage > totPage) {
    JOptionPane.showMessageDialog(null, "YOU READ
MORE THAN THE EXISTING PAGE NUMBER?");
} else if (nowPage < 0) {
    JOptionPane.showMessageDialog(null, "YOU READ
NEGATIVE PAGE NUMBERS???");
} else {
    //if all validations passed, then update the new progress
    String sql3 = "UPDATE TEAM1_MEMBERS SET
LIT_PROGRESS = (?) WHERE NAME = (?)";
    PreparedStatement st3 = conn.prepareStatement(sql3);
    st3.setInt(1, nowPage);
    st3.setString(2, currentUser);
    st3.executeUpdate();

    String sql2 = "SELECT * FROM TEAM1_MEMBERS
WHERE NAME = (?)";
    PreparedStatement st2 = conn.prepareStatement(sql2);
    st2.setString(1, currentUser);
    rs2 = st2.executeQuery();

    //if finished the guide, then add 3 points

    if (nowPage == totPage) {
        String sql4 = "UPDATE TEAM1_MEMBERS
SET POINTS = POINTS+3 WHERE NAME = (?)";
        PreparedStatement st4 =
conn.prepareStatement(sql4);
        st4.setString(1, currentUser);
        st4.executeUpdate();
        JOptionPane.showMessageDialog(null,
"Congratulations on finishing LIT! You have gained 3 points!");
    }
    JOptionPane.showMessageDialog(null, "Update
Successfully");

    if (LoginWindow.status.equals("member")) {
        new TeamMemberWindow().setVisible(true);
    }
}

```

```

        dispose();
    } else if (LoginWindow.status.equals("leader")) {
        new TeamLeaderWindow().setVisible(true);
        dispose();
    }
}
}
//repeat for team 2, team 3, and unassigned members and for other
subjects of resource guides

```

```

else if (LoginWindow.teamUsed.equals("Team 2")) {
    int nowPage = Integer.valueOf(NewPageField.getText());
    int oriPage = Integer.valueOf(WerePageField.getText());
    int totPage = Integer.valueOf(TotPageField.getText());
    if (nowPage < oriPage) {
        JOptionPane.showMessageDialog(null, "YOU READ
BACKWARDS?");
    } else if (nowPage > totPage) {
        JOptionPane.showMessageDialog(null, "YOU READ
MORE THAN THE EXISTING PAGE NUMBER?");
    } else if (nowPage < 0) {
        JOptionPane.showMessageDialog(null, "YOU READ
NEGATIVE PAGE NUMBERS???");
    } else {
        String sql3 = "UPDATE TEAM2_MEMBERS SET
LIT_PROGRESS = (?) WHERE NAME = (?)";
        PreparedStatement st3 = conn.prepareStatement(sql3);
        st3.setInt(1, nowPage);
        st3.setString(2, currentUser);
        st3.executeUpdate();

        String sql2 = "SELECT * FROM TEAM2_MEMBERS
WHERE NAME = (?)";

        PreparedStatement st2 = conn.prepareStatement(sql2);
        st2.setString(1, currentUser);
        rs2 = st2.executeQuery();

        if (nowPage == totPage) {
            String sql4 = "UPDATE TEAM2_MEMBERS
SET POINTS = POINTS+3 WHERE NAME = (?)";
            PreparedStatement st4 =
conn.prepareStatement(sql4);
            st4.setString(1, currentUser);
            st4.executeUpdate();

```

```

        JOptionPane.showMessageDialog(null,
"Congratulations on finishing LIT! You have gained 3 points!");
    }
    JOptionPane.showMessageDialog(null, "Update
Successfully");

    if (LoginWindow.status.equals("member")) {
        new TeamMemberWindow().setVisible(true);
        dispose();
    } else if (LoginWindow.status.equals("leader")) {
        new TeamLeaderWindow().setVisible(true);
        dispose();
    }
}
} else if (LoginWindow.teamUsed.equals("Team 3")) {
    int nowPage = Integer.valueOf(NewPageField.getText());
    int oriPage = Integer.valueOf(WerePageField.getText());
    int totPage = Integer.valueOf(TotPageField.getText());
    if (nowPage < oriPage) {
        JOptionPane.showMessageDialog(null, "YOU READ
BACKWARDS?");
    } else if (nowPage > totPage) {
        JOptionPane.showMessageDialog(null, "YOU READ
MORE THAN THE EXISTING PAGE NUMBER?");
    } else if (nowPage < 0) {
        JOptionPane.showMessageDialog(null, "YOU READ
NEGATIVE PAGE NUMBERS???");
    } else {
        String sql3 = "UPDATE TEAM3_MEMBERS SET
LIT_PROGRESS = (?) WHERE NAME = (?)";
        PreparedStatement st3 = conn.prepareStatement(sql3);
        st3.setInt(1, nowPage);
        st3.setString(2, currentUser);
        st3.executeUpdate();

        String sql2 = "SELECT * FROM TEAM3_MEMBERS
WHERE NAME = (?)";

        PreparedStatement st2 = conn.prepareStatement(sql2);
        st2.setString(1, currentUser);
        rs2 = st2.executeQuery();

        if (nowPage == totPage) {
            String sql4 = "UPDATE TEAM3_MEMBERS
SET POINTS = POINTS+3 WHERE NAME = (?)";
            PreparedStatement st4 =

```

```

conn.prepareStatement(sql4);
                                st4.setString(1, currentUser);
                                st4.executeUpdate();
                                JOptionPane.showMessageDialog(null,
"Congratulations on finishing LIT! You have gained 3 points!");
                                }
                                JOptionPane.showMessageDialog(null, "Update
Successfully");
                                if (LoginWindow.status.equals("member")) {
                                    new TeamMemberWindow().setVisible(true);
                                    dispose();
                                } else if (LoginWindow.status.equals("leader")) {
                                    new TeamLeaderWindow().setVisible(true);
                                    dispose();
                                }
                                }
                                } else if (LoginWindow.teamUsed.equals("Unassigned")) {
                                    int nowPage = Integer.valueOf(NewPageField.getText());
                                    int oriPage = Integer.valueOf(WerePageField.getText());
                                    int totPage = Integer.valueOf(TotPageField.getText());
                                    if (nowPage < oriPage) {
                                        JOptionPane.showMessageDialog(null, "YOU READ
BACKWARDS?");
                                    } else if (nowPage > totPage) {
                                        JOptionPane.showMessageDialog(null, "YOU READ
MORE THAN THE EXISTING PAGE NUMBER?");
                                    } else if (nowPage < 0) {
                                        JOptionPane.showMessageDialog(null, "YOU READ
NEGATIVE PAGE NUMBERS???");
                                    } else {
                                        String sql3 = "UPDATE UNASSIGNED_MEMBERS
SET LIT_PROGRESS = (?) WHERE NAME = (?)";
                                        PreparedStatement st3 = conn.prepareStatement(sql3);
                                        st3.setInt(1, nowPage);
                                        st3.setString(2, currentUser);
                                        st3.executeUpdate();

                                        String sql2 = "SELECT * FROM
UNASSIGNED_MEMBERS WHERE NAME = (?)";
                                        PreparedStatement st2 = conn.prepareStatement(sql2);
                                        st2.setString(1, currentUser);
                                        rs2 = st2.executeQuery();

                                        if (nowPage == totPage) {

```

```

        String sql4 = "UPDATE
UNASSIGNED_MEMBERS SET POINTS = POINTS+3 WHERE NAME = (?)";
        PreparedStatement st4 =
conn.prepareStatement(sql4);
        st4.setString(1, currentUser);
        st4.executeUpdate();
        JOptionPane.showMessageDialog(null,
"Congratulations on finishing LIT! You have gained 3 points!");
    }
    JOptionPane.showMessageDialog(null, "Update
Successfully");

    if (LoginWindow.status.equals("member")) {
        new TeamMemberWindow().setVisible(true);
        dispose();
    } else if (LoginWindow.status.equals("leader")) {
        new TeamLeaderWindow().setVisible(true);
        dispose();
    }
}
}

```

**//also update the overall progress**

```

while (rs2.next()) {
    int lit = rs2.getInt("LIT_PROGRESS");
    int art = rs2.getInt("ART_PROGRESS");
    int sci = rs2.getInt("SCI_PROGRESS");
    int socsci = rs2.getInt("SOCSCI_PROGRESS");
    int mus = rs2.getInt("MUS_PROGRESS");
    int mat = rs2.getInt("MAT_PROGRESS");
    int econ = rs2.getInt("ECON_PROGRESS");
    allCurrentProgress = lit + art + sci + socsci + mus + mat +
econ;
}

while (rs1.next()) {
    int tot_lit = rs1.getInt("LIT_PAGE");
    int tot_art = rs1.getInt("ART_PAGE");
    int tot_sci = rs1.getInt("SCI_PAGE");
    int tot_socsci = rs1.getInt("SOCSCI_PAGE");
    int tot_mus = rs1.getInt("MUS_PAGE");
    int tot_mat = rs1.getInt("MAT_PAGE");
    int tot_econ = rs1.getInt("ECON_PAGE");
    allPageNum = tot_lit + tot_art + tot_sci + tot_socsci +

```

```

tot_mus + tot_mat + tot_econ;
    }

    int updateOverall = (allCurrentProgress * 100) / allPageNum;
    String sql4 = "UPDATE TEAM1_MEMBERS SET
OVERALL_PROGRESS = (?) WHERE NAME = (?)";
    String sql5 = "UPDATE TEAM2_MEMBERS SET
OVERALL_PROGRESS = (?) WHERE NAME = (?)";
    String sql6 = "UPDATE TEAM3_MEMBERS SET
OVERALL_PROGRESS = (?) WHERE NAME = (?)";
    String sql7 = "UPDATE UNASSIGNED_MEMBERS SET
OVERALL_PROGRESS = (?) WHERE NAME = (?)";
    PreparedStatement st4 = conn.prepareStatement(sql4);
    PreparedStatement st5 = conn.prepareStatement(sql5);
    PreparedStatement st6 = conn.prepareStatement(sql6);
    PreparedStatement st7 = conn.prepareStatement(sql7);
    st4.setInt(1, updateOverall);
    st4.setString(2, currentUser);
    st5.setInt(1, updateOverall);
    st5.setString(2, currentUser);
    st6.setInt(1, updateOverall);
    st6.setString(2, currentUser);
    st7.setInt(1, updateOverall);
    st7.setString(2, currentUser);
    st4.executeUpdate();
    st5.executeUpdate();
    st6.executeUpdate();
    st7.executeUpdate();

    }
    if (subject.equals("Art")) {

        int allCurrentProgress = 0;
        int allPageNum = 0;
        ResultSet rs2 = null;

        String sql1 = "SELECT * FROM PAGE_NUM WHERE
LINE_NUM = (?)";
        PreparedStatement st1 = conn.prepareStatement(sql1);
        st1.setInt(1, 1);
        ResultSet rs1 = st1.executeQuery();

        if (LoginWindow.teamUsed.equals("Team 1")) {
            int nowPage = Integer.valueOf(NewPageField.getText());

```



```

        int oriPage = Integer.valueOf(WerePageField.getText());
        int totPage = Integer.valueOf(TotPageField.getText());
        if (nowPage < oriPage) {
            JOptionPane.showMessageDialog(null, "YOU READ
BACKWARDS?");
        } else if (nowPage > totPage) {
            JOptionPane.showMessageDialog(null, "YOU READ
MORE THAN THE EXISTING PAGE NUMBER?");
        } else if (nowPage < 0) {
            JOptionPane.showMessageDialog(null, "YOU READ
NEGATIVE PAGE NUMBERS???");
        } else {
            String sql3 = "UPDATE TEAM1_MEMBERS SET
ART_PROGRESS = (?) WHERE NAME = (?)";
            PreparedStatement st3 = conn.prepareStatement(sql3);
            st3.setInt(1, nowPage);
            st3.setString(2, currentUser);
            st3.executeUpdate();

            String sql2 = "SELECT * FROM TEAM1_MEMBERS
WHERE NAME = (?)";
            PreparedStatement st2 = conn.prepareStatement(sql2);
            st2.setString(1, currentUser);
            rs2 = st2.executeQuery();

            if (nowPage == totPage) {
                String sql4 = "UPDATE TEAM1_MEMBERS
SET POINTS = POINTS+3 WHERE NAME = (?)";
                PreparedStatement st4 =
conn.prepareStatement(sql4);
                st4.setString(1, currentUser);
                st4.executeUpdate();
                JOptionPane.showMessageDialog(null,
"Congratulations on finishing ART! You have gained 3 points!");
            }
            JOptionPane.showMessageDialog(null, "Update
Successfully");

            if (LoginWindow.status.equals("member")) {
                new TeamMemberWindow().setVisible(true);
                dispose();
            } else if (LoginWindow.status.equals("leader")) {
                new TeamLeaderWindow().setVisible(true);
                dispose();
            }
        }
    }

```

```

    }
    } else if (LoginWindow.teamUsed.equals("Team 2")) {
        int nowPage = Integer.valueOf(NewPageField.getText());
        int oriPage = Integer.valueOf(WerePageField.getText());
        int totPage = Integer.valueOf(TotPageField.getText());
        if (nowPage < oriPage) {
            JOptionPane.showMessageDialog(null, "YOU READ
BACKWARDS?");
        } else if (nowPage > totPage) {
            JOptionPane.showMessageDialog(null, "YOU READ
MORE THAN THE EXISTING PAGE NUMBER?");
        } else if (nowPage < 0) {
            JOptionPane.showMessageDialog(null, "YOU READ
NEGATIVE PAGE NUMBERS???");
        } else {
            String sql3 = "UPDATE TEAM2_MEMBERS SET
ART_PROGRESS = (?) WHERE NAME = (?)";
            PreparedStatement st3 = conn.prepareStatement(sql3);
            st3.setInt(1, nowPage);
            st3.setString(2, currentUser);
            st3.executeUpdate();

            String sql2 = "SELECT * FROM TEAM2_MEMBERS
WHERE NAME = (?)";
            PreparedStatement st2 = conn.prepareStatement(sql2);
            st2.setString(1, currentUser);
            rs2 = st2.executeQuery();

            if (nowPage == totPage) {
                String sql4 = "UPDATE TEAM2_MEMBERS
SET POINTS = POINTS+3 WHERE NAME = (?)";
                PreparedStatement st4 =
conn.prepareStatement(sql4);
                st4.setString(1, currentUser);
                st4.executeUpdate();
                JOptionPane.showMessageDialog(null,
"Congratulations on finishing ART! You have gained 3 points!");
            }
            JOptionPane.showMessageDialog(null, "Update
Successfully");

            if (LoginWindow.status.equals("member")) {
                new TeamMemberWindow().setVisible(true);
                dispose();
            } else if (LoginWindow.status.equals("leader")) {

```

```

        new TeamLeaderWindow().setVisible(true);
        dispose();
    }
}
} else if (LoginWindow.teamUsed.equals("Team 3")) {
    int nowPage = Integer.valueOf(NewPageField.getText());
    int oriPage = Integer.valueOf(WerePageField.getText());
    int totPage = Integer.valueOf(TotPageField.getText());
    if (nowPage < oriPage) {
        JOptionPane.showMessageDialog(null, "YOU READ
BACKWARDS?");
    } else if (nowPage > totPage) {
        JOptionPane.showMessageDialog(null, "YOU READ
MORE THAN THE EXISTING PAGE NUMBER?");
    } else if (nowPage < 0) {
        JOptionPane.showMessageDialog(null, "YOU READ
NEGATIVE PAGE NUMBERS???");
    } else {
        String sql3 = "UPDATE TEAM3_MEMBERS SET
ART_PROGRESS = (?) WHERE NAME = (?)";
        PreparedStatement st3 = conn.prepareStatement(sql3);
        st3.setInt(1, nowPage);
        st3.setString(2, currentUser);
        st3.executeUpdate();

        String sql2 = "SELECT * FROM TEAM3_MEMBERS
WHERE NAME = (?)";

        PreparedStatement st2 = conn.prepareStatement(sql2);
        st2.setString(1, currentUser);
        rs2 = st2.executeQuery();

        if (nowPage == totPage) {
            String sql4 = "UPDATE TEAM3_MEMBERS
SET POINTS = POINTS+3 WHERE NAME = (?)";
            PreparedStatement st4 =
conn.prepareStatement(sql4);
            st4.setString(1, currentUser);
            st4.executeUpdate();
            JOptionPane.showMessageDialog(null,
"Congratulations on finishing ART! You have gained 3 points!");
        }
        JOptionPane.showMessageDialog(null, "Update
Successfully");

        if (LoginWindow.status.equals("member")) {

```

```

        new TeamMemberWindow().setVisible(true);
        dispose();
    } else if (LoginWindow.status.equals("leader")) {
        new TeamLeaderWindow().setVisible(true);
        dispose();
    }
}
} else if (LoginWindow.teamUsed.equals("Unassigned")) {
    int nowPage = Integer.valueOf(NewPageField.getText());
    int oriPage = Integer.valueOf(WerePageField.getText());
    int totPage = Integer.valueOf(TotPageField.getText());
    if (nowPage < oriPage) {
        JOptionPane.showMessageDialog(null, "YOU READ
BACKWARDS?");
    } else if (nowPage > totPage) {
        JOptionPane.showMessageDialog(null, "YOU READ
MORE THAN THE EXISTING PAGE NUMBER?");
    } else if (nowPage < 0) {
        JOptionPane.showMessageDialog(null, "YOU READ
NEGATIVE PAGE NUMBERS???");
    } else {
        String sql3 = "UPDATE UNASSIGNED_MEMBERS
SET ART_PROGRESS = (?) WHERE NAME = (?)";
        PreparedStatement st3 = conn.prepareStatement(sql3);
        st3.setInt(1, nowPage);
        st3.setString(2, currentUser);
        st3.executeUpdate();

        String sql2 = "SELECT * FROM
UNASSIGNED_MEMBERS WHERE NAME = (?)";
        PreparedStatement st2 = conn.prepareStatement(sql2);
        st2.setString(1, currentUser);
        rs2 = st2.executeQuery();

        if (nowPage == totPage) {
            String sql4 = "UPDATE
UNASSIGNED_MEMBERS SET POINTS = POINTS+3 WHERE NAME = (?)";
            PreparedStatement st4 =
conn.prepareStatement(sql4);
            st4.setString(1, currentUser);
            st4.executeUpdate();
            JOptionPane.showMessageDialog(null,
"Congratulations on finishing ART! You have gained 3 points!");
        }
    }
}

```

```

        JOptionPane.showMessageDialog(null, "Update
Successfully");

        if (LoginWindow.status.equals("member")) {
            new TeamMemberWindow().setVisible(true);
            dispose();
        } else if (LoginWindow.status.equals("leader")) {
            new TeamLeaderWindow().setVisible(true);
            dispose();
        }
    }
}

while (rs2.next()) {
    int lit = rs2.getInt("LIT_PROGRESS");
    int art = rs2.getInt("ART_PROGRESS");
    int sci = rs2.getInt("SCI_PROGRESS");
    int socsci = rs2.getInt("SOCSCI_PROGRESS");
    int mus = rs2.getInt("MUS_PROGRESS");
    int mat = rs2.getInt("MAT_PROGRESS");
    int econ = rs2.getInt("ECON_PROGRESS");
    allCurrentProgress = lit + art + sci + socsci + mus + mat +
econ;
}

while (rs1.next()) {
    int tot_lit = rs1.getInt("LIT_PAGE");
    int tot_art = rs1.getInt("ART_PAGE");
    int tot_sci = rs1.getInt("SCI_PAGE");
    int tot_socsci = rs1.getInt("SOCSCI_PAGE");
    int tot_mus = rs1.getInt("MUS_PAGE");
    int tot_mat = rs1.getInt("MAT_PAGE");
    int tot_econ = rs1.getInt("ECON_PAGE");
    allPageNum = tot_lit + tot_art + tot_sci + tot_socsci +
tot_mus + tot_mat + tot_econ;
}

int updateOverall = (allCurrentProgress * 100) / allPageNum;
String sql4 = "UPDATE TEAM1_MEMBERS SET
OVERALL_PROGRESS = (?) WHERE NAME = (?)";
String sql5 = "UPDATE TEAM2_MEMBERS SET
OVERALL_PROGRESS = (?) WHERE NAME = (?)";
String sql6 = "UPDATE TEAM3_MEMBERS SET
OVERALL_PROGRESS = (?) WHERE NAME = (?)";
String sql7 = "UPDATE UNASSIGNED_MEMBERS SET

```

```

OVERALL_PROGRESS = (?) WHERE NAME = (?)";
        PreparedStatement st4 = conn.prepareStatement(sql4);
        PreparedStatement st5 = conn.prepareStatement(sql5);
        PreparedStatement st6 = conn.prepareStatement(sql6);
        PreparedStatement st7 = conn.prepareStatement(sql7);
        st4.setInt(1, updateOverall);
        st4.setString(2, currentUser);
        st5.setInt(1, updateOverall);
        st5.setString(2, currentUser);
        st6.setInt(1, updateOverall);
        st6.setString(2, currentUser);
        st7.setInt(1, updateOverall);
        st7.setString(2, currentUser);
        st4.executeUpdate();
        st5.executeUpdate();
        st6.executeUpdate();
        st7.executeUpdate();

    }

    if (subject.equals("Science")) {

        int allCurrentProgress = 0;
        int allPageNum = 0;
        ResultSet rs2 = null;

        String sql1 = "SELECT * FROM PAGE_NUM WHERE
LINE_NUM = (?)";
        PreparedStatement st1 = conn.prepareStatement(sql1);
        st1.setInt(1, 1);
        ResultSet rs1 = st1.executeQuery();

        if (LoginWindow.teamUsed.equals("Team 1")) {
            int nowPage = Integer.valueOf(NewPageField.getText());
            int oriPage = Integer.valueOf(WerePageField.getText());
            int totPage = Integer.valueOf(TotPageField.getText());
            if (nowPage < oriPage) {
                JOptionPane.showMessageDialog(null, "YOU READ
BACKWARDS?");
            } else if (nowPage > totPage) {
                JOptionPane.showMessageDialog(null, "YOU READ
MORE THAN THE EXISTING PAGE NUMBER?");
            } else if (nowPage < 0) {
                JOptionPane.showMessageDialog(null, "YOU READ

```

```

NEGATIVE PAGE NUMBERS???" );
        } else {
            String sql3 = "UPDATE TEAM1_MEMBERS SET
SCI_PROGRESS = (?) WHERE NAME = (?)";
            PreparedStatement st3 = conn.prepareStatement(sql3);
            st3.setInt(1, nowPage);
            st3.setString(2, currentUser);
            st3.executeUpdate();

            String sql2 = "SELECT * FROM TEAM1_MEMBERS
WHERE NAME = (?)";
            PreparedStatement st2 = conn.prepareStatement(sql2);
            st2.setString(1, currentUser);
            rs2 = st2.executeQuery();

            if (nowPage == totPage) {
                String sql4 = "UPDATE TEAM1_MEMBERS
SET POINTS = POINTS+3 WHERE NAME = (?)";
                PreparedStatement st4 =
conn.prepareStatement(sql4);
                st4.setString(1, currentUser);
                st4.executeUpdate();
                JOptionPane.showMessageDialog(null,
"Congratulations on finishing SCI! You have gained 3 points!");
            }
            JOptionPane.showMessageDialog(null, "Update
Successfully");

            if (LoginWindow.status.equals("member")) {
                new TeamMemberWindow().setVisible(true);
                dispose();
            } else if (LoginWindow.status.equals("leader")) {
                new TeamLeaderWindow().setVisible(true);
                dispose();
            }
        }
    } else if (LoginWindow.teamUsed.equals("Team 2")) {
        int nowPage = Integer.valueOf(NewPageField.getText());
        int oriPage = Integer.valueOf(WerePageField.getText());
        int totPage = Integer.valueOf(TotPageField.getText());
        if (nowPage < oriPage) {
            JOptionPane.showMessageDialog(null, "YOU READ
BACKWARDS?");
        } else if (nowPage > totPage) {
            JOptionPane.showMessageDialog(null, "YOU READ

```

```

MORE THAN THE EXISTING PAGE NUMBER?");
        } else if (nowPage < 0) {
            JOptionPane.showMessageDialog(null, "YOU READ
NEGATIVE PAGE NUMBERS???");
        } else {
            String sql3 = "UPDATE TEAM2_MEMBERS SET
SCI_PROGRESS = (?) WHERE NAME = (?)";
            PreparedStatement st3 = conn.prepareStatement(sql3);
            st3.setInt(1, nowPage);
            st3.setString(2, currentUser);
            st3.executeUpdate();

            String sql2 = "SELECT * FROM TEAM2_MEMBERS
WHERE NAME = (?)";
            PreparedStatement st2 = conn.prepareStatement(sql2);
            st2.setString(1, currentUser);
            rs2 = st2.executeQuery();

            if (nowPage == totPage) {
                String sql4 = "UPDATE TEAM2_MEMBERS
SET POINTS = POINTS+3 WHERE NAME = (?)";
                PreparedStatement st4 =
conn.prepareStatement(sql4);
                st4.setString(1, currentUser);
                st4.executeUpdate();
                JOptionPane.showMessageDialog(null,
"Congratulations on finishing SCI! You have gained 3 points!");
            }
            JOptionPane.showMessageDialog(null, "Update
Successfully");

            if (LoginWindow.status.equals("member")) {
                new TeamMemberWindow().setVisible(true);
                dispose();
            } else if (LoginWindow.status.equals("leader")) {
                new TeamLeaderWindow().setVisible(true);
                dispose();
            }
        }
    } else if (LoginWindow.teamUsed.equals("Team 3")) {
        int nowPage = Integer.valueOf(NewPageField.getText());
        int oriPage = Integer.valueOf(WerePageField.getText());
        int totPage = Integer.valueOf(TotPageField.getText());
        if (nowPage < oriPage) {
            JOptionPane.showMessageDialog(null, "YOU READ

```



```

BACKWARDS?");
        } else if (nowPage > totPage) {
            JOptionPane.showMessageDialog(null, "YOU READ
MORE THAN THE EXISTING PAGE NUMBER?");
        } else if (nowPage < 0) {
            JOptionPane.showMessageDialog(null, "YOU READ
NEGATIVE PAGE NUMBERS???");
        } else {
            String sql3 = "UPDATE TEAM3_MEMBERS SET
SCI_PROGRESS = (?) WHERE NAME = (?)";
            PreparedStatement st3 = conn.prepareStatement(sql3);
            st3.setInt(1, nowPage);
            st3.setString(2, currentUser);
            st3.executeUpdate();

            String sql2 = "SELECT * FROM TEAM3_MEMBERS
WHERE NAME = (?)";
            PreparedStatement st2 = conn.prepareStatement(sql2);
            st2.setString(1, currentUser);
            rs2 = st2.executeQuery();

            if (nowPage == totPage) {
                String sql4 = "UPDATE TEAM3_MEMBERS
SET POINTS = POINTS+3 WHERE NAME = (?)";
                PreparedStatement st4 =
conn.prepareStatement(sql4);
                st4.setString(1, currentUser);
                st4.executeUpdate();
                JOptionPane.showMessageDialog(null,
"Congratulations on finishing SCI! You have gained 3 points!");
            }
            JOptionPane.showMessageDialog(null, "Update
Successfully");

            if (LoginWindow.status.equals("member")) {
                new TeamMemberWindow().setVisible(true);
                dispose();
            } else if (LoginWindow.status.equals("leader")) {
                new TeamLeaderWindow().setVisible(true);
                dispose();
            }
        }
    } else if (LoginWindow.teamUsed.equals("Unassigned")) {
        int nowPage = Integer.valueOf(NewPageField.getText());
        int oriPage = Integer.valueOf(WerePageField.getText());

```

```

        int totPage = Integer.valueOf(TotPageField.getText());
        if (nowPage < oriPage) {
            JOptionPane.showMessageDialog(null, "YOU READ
BACKWARDS?");
        } else if (nowPage > totPage) {
            JOptionPane.showMessageDialog(null, "YOU READ
MORE THAN THE EXISTING PAGE NUMBER?");
        } else if (nowPage < 0) {
            JOptionPane.showMessageDialog(null, "YOU READ
NEGATIVE PAGE NUMBERS???");
        } else {
            String sql3 = "UPDATE UNASSIGNED_MEMBERS
SET SCI_PROGRESS = (?) WHERE NAME = (?)";
            PreparedStatement st3 = conn.prepareStatement(sql3);
            st3.setInt(1, nowPage);
            st3.setString(2, currentUser);
            st3.executeUpdate();

            String sql2 = "SELECT * FROM
UNASSIGNED_MEMBERS WHERE NAME = (?)";
            PreparedStatement st2 = conn.prepareStatement(sql2);
            st2.setString(1, currentUser);
            rs2 = st2.executeQuery();

            if (nowPage == totPage) {
                String sql4 = "UPDATE
UNASSIGNED_MEMBERS SET POINTS = POINTS+3 WHERE NAME = (?)";
                PreparedStatement st4 =
conn.prepareStatement(sql4);
                st4.setString(1, currentUser);
                st4.executeUpdate();
                JOptionPane.showMessageDialog(null,
"Congratulations on finishing SCI! You have gained 3 points!");
            }
            JOptionPane.showMessageDialog(null, "Update
Successfully");

            if (LoginWindow.status.equals("member")) {
                new TeamMemberWindow().setVisible(true);
                dispose();
            } else if (LoginWindow.status.equals("leader")) {
                new TeamLeaderWindow().setVisible(true);
                dispose();
            }
        }
    }
}

```

```

    }

    while (rs2.next()) {
        int lit = rs2.getInt("LIT_PROGRESS");
        int art = rs2.getInt("ART_PROGRESS");
        int sci = rs2.getInt("SCI_PROGRESS");
        int socsci = rs2.getInt("SOCSCI_PROGRESS");
        int mus = rs2.getInt("MUS_PROGRESS");
        int mat = rs2.getInt("MAT_PROGRESS");
        int econ = rs2.getInt("ECON_PROGRESS");
        allCurrentProgress = lit + art + sci + socsci + mus + mat +
econ;
    }

    while (rs1.next()) {
        int tot_lit = rs1.getInt("LIT_PAGE");
        int tot_art = rs1.getInt("ART_PAGE");
        int tot_sci = rs1.getInt("SCI_PAGE");
        int tot_socsci = rs1.getInt("SOCSCI_PAGE");
        int tot_mus = rs1.getInt("MUS_PAGE");
        int tot_mat = rs1.getInt("MAT_PAGE");
        int tot_econ = rs1.getInt("ECON_PAGE");
        allPageNum = tot_lit + tot_art + tot_sci + tot_socsci +
tot_mus + tot_mat + tot_econ;
    }

    int updateOverall = (allCurrentProgress * 100) / allPageNum;
    String sql4 = "UPDATE TEAM1_MEMBERS SET
OVERALL_PROGRESS = (?) WHERE NAME = (?)";
    String sql5 = "UPDATE TEAM2_MEMBERS SET
OVERALL_PROGRESS = (?) WHERE NAME = (?)";
    String sql6 = "UPDATE TEAM3_MEMBERS SET
OVERALL_PROGRESS = (?) WHERE NAME = (?)";
    String sql7 = "UPDATE UNASSIGNED_MEMBERS SET
OVERALL_PROGRESS = (?) WHERE NAME = (?)";
    PreparedStatement st4 = conn.prepareStatement(sql4);
    PreparedStatement st5 = conn.prepareStatement(sql5);
    PreparedStatement st6 = conn.prepareStatement(sql6);
    PreparedStatement st7 = conn.prepareStatement(sql7);
    st4.setInt(1, updateOverall);
    st4.setString(2, currentUser);
    st5.setInt(1, updateOverall);
    st5.setString(2, currentUser);
    st6.setInt(1, updateOverall);

```

```

        st6.setString(2, currentUser);
        st7.setInt(1, updateOverall);
        st7.setString(2, currentUser);
        st4.executeUpdate();
        st5.executeUpdate();
        st6.executeUpdate();
        st7.executeUpdate();

    }

    if (subject.equals("Social Science")) {

        int allCurrentProgress = 0;
        int allPageNum = 0;
        ResultSet rs2 = null;

        String sql1 = "SELECT * FROM PAGE_NUM WHERE
LINE_NUM = (?)";
        PreparedStatement st1 = conn.prepareStatement(sql1);
        st1.setInt(1, 1);
        ResultSet rs1 = st1.executeQuery();

        if (LoginWindow.teamUsed.equals("Team 1")) {
            int nowPage = Integer.valueOf(NewPageField.getText());
            int oriPage = Integer.valueOf(WerePageField.getText());
            int totPage = Integer.valueOf(TotPageField.getText());
            if (nowPage < oriPage) {
                JOptionPane.showMessageDialog(null, "YOU READ
BACKWARDS?");
            } else if (nowPage > totPage) {
                JOptionPane.showMessageDialog(null, "YOU READ
MORE THAN THE EXISTING PAGE NUMBER?");
            } else if (nowPage < 0) {
                JOptionPane.showMessageDialog(null, "YOU READ
NEGATIVE PAGE NUMBERS???");
            } else {
                String sql3 = "UPDATE TEAM1_MEMBERS SET
SOCSCI_PROGRESS = (?) WHERE NAME = (?)";
                PreparedStatement st3 = conn.prepareStatement(sql3);
                st3.setInt(1, nowPage);
                st3.setString(2, currentUser);
                st3.executeUpdate();

                String sql2 = "SELECT * FROM TEAM1_MEMBERS

```

```

WHERE NAME = (?)";

        PreparedStatement st2 = conn.prepareStatement(sql2);
        st2.setString(1, currentUser);
        rs2 = st2.executeQuery();

        if (nowPage == totPage) {
            String sql4 = "UPDATE TEAM1_MEMBERS
SET POINTS = POINTS+3 WHERE NAME = (?)";
            PreparedStatement st4 =
conn.prepareStatement(sql4);

            st4.setString(1, currentUser);
            st4.executeUpdate();
            JOptionPane.showMessageDialog(null,
"Congratulations on finishing SOCSCI! You have gained 3 points!");
        }
        JOptionPane.showMessageDialog(null, "Update
Successfully");

        if (LoginWindow.status.equals("member")) {
            new TeamMemberWindow().setVisible(true);
            dispose();
        } else if (LoginWindow.status.equals("leader")) {
            new TeamLeaderWindow().setVisible(true);
            dispose();
        }
    }
} else if (LoginWindow.teamUsed.equals("Team 2")) {
    int nowPage = Integer.valueOf(NewPageField.getText());
    int oriPage = Integer.valueOf(WerePageField.getText());
    int totPage = Integer.valueOf(TotPageField.getText());
    if (nowPage < oriPage) {
        JOptionPane.showMessageDialog(null, "YOU READ
BACKWARDS?");
    } else if (nowPage > totPage) {
        JOptionPane.showMessageDialog(null, "YOU READ
MORE THAN THE EXISTING PAGE NUMBER?");
    } else if (nowPage < 0) {
        JOptionPane.showMessageDialog(null, "YOU READ
NEGATIVE PAGE NUMBERS???");
    } else {
        String sql3 = "UPDATE TEAM2_MEMBERS SET
SOCSCI_PROGRESS = (?) WHERE NAME = (?)";
        PreparedStatement st3 = conn.prepareStatement(sql3);
        st3.setInt(1, nowPage);
        st3.setString(2, currentUser);
    }
}

```

```

        st3.executeUpdate();

        String sql2 = "SELECT * FROM TEAM2_MEMBERS
WHERE NAME = (?)";

        PreparedStatement st2 = conn.prepareStatement(sql2);
        st2.setString(1, currentUser);
        rs2 = st2.executeQuery();

        if (nowPage == totPage) {
            String sql4 = "UPDATE TEAM2_MEMBERS
SET POINTS = POINTS+3 WHERE NAME = (?)";
            PreparedStatement st4 =
conn.prepareStatement(sql4);
            st4.setString(1, currentUser);
            st4.executeUpdate();
            JOptionPane.showMessageDialog(null,
"Congratulations on finishing SOCSCI! You have gained 3 points!");
        }
        JOptionPane.showMessageDialog(null, "Update
Successfully");

        if (LoginWindow.status.equals("member")) {
            new TeamMemberWindow().setVisible(true);
            dispose();
        } else if (LoginWindow.status.equals("leader")) {
            new TeamLeaderWindow().setVisible(true);
            dispose();
        }
    }
} else if (LoginWindow.teamUsed.equals("Team 3")) {
    int nowPage = Integer.valueOf(NewPageField.getText());
    int oriPage = Integer.valueOf(WerePageField.getText());
    int totPage = Integer.valueOf(TotPageField.getText());
    if (nowPage < oriPage) {
        JOptionPane.showMessageDialog(null, "YOU READ
BACKWARDS?");
    } else if (nowPage > totPage) {
        JOptionPane.showMessageDialog(null, "YOU READ
MORE THAN THE EXISTING PAGE NUMBER?");
    } else if (nowPage < 0) {
        JOptionPane.showMessageDialog(null, "YOU READ
NEGATIVE PAGE NUMBERS???");
    } else {
        String sql3 = "UPDATE TEAM3_MEMBERS SET
SOCSCI_PROGRESS = (?) WHERE NAME = (?)";

```

```

        PreparedStatement st3 = conn.prepareStatement(sql3);
        st3.setInt(1, nowPage);
        st3.setString(2, currentUser);
        st3.executeUpdate();

        String sql2 = "SELECT * FROM TEAM3_MEMBERS
WHERE NAME = (?)";

        PreparedStatement st2 = conn.prepareStatement(sql2);
        st2.setString(1, currentUser);
        rs2 = st2.executeQuery();

        if (nowPage == totPage) {
            String sql4 = "UPDATE TEAM3_MEMBERS
SET POINTS = POINTS+3 WHERE NAME = (?)";
            PreparedStatement st4 =
conn.prepareStatement(sql4);
            st4.setString(1, currentUser);
            st4.executeUpdate();
            JOptionPane.showMessageDialog(null,
"Congratulations on finishing SOCSOI! You have gained 3 points!");
        }
        JOptionPane.showMessageDialog(null, "Update
Successfully");

        if (LoginWindow.status.equals("member")) {
            new TeamMemberWindow().setVisible(true);
            dispose();
        } else if (LoginWindow.status.equals("leader")) {
            new TeamLeaderWindow().setVisible(true);
            dispose();
        }
    }
} else if (LoginWindow.teamUsed.equals("Unassigned")) {
    int nowPage = Integer.valueOf(NewPageField.getText());
    int oriPage = Integer.valueOf(WerePageField.getText());
    int totPage = Integer.valueOf(TotPageField.getText());
    if (nowPage < oriPage) {
        JOptionPane.showMessageDialog(null, "YOU READ
BACKWARDS?");
    } else if (nowPage > totPage) {
        JOptionPane.showMessageDialog(null, "YOU READ
MORE THAN THE EXISTING PAGE NUMBER?");
    } else if (nowPage < 0) {
        JOptionPane.showMessageDialog(null, "YOU READ
NEGATIVE PAGE NUMBERS???");
    }
}

```

```

        } else {
            String sql3 = "UPDATE UNASSIGNED_MEMBERS
SET SOCSCI_PROGRESS = (?) WHERE NAME = (?)";
            PreparedStatement st3 = conn.prepareStatement(sql3);
            st3.setInt(1, nowPage);
            st3.setString(2, currentUser);
            st3.executeUpdate();

            String sql2 = "SELECT * FROM
UNASSIGNED_MEMBERS WHERE NAME = (?)";
            PreparedStatement st2 = conn.prepareStatement(sql2);
            st2.setString(1, currentUser);
            rs2 = st2.executeQuery();

            if (nowPage == totPage) {
                String sql4 = "UPDATE
UNASSIGNED_MEMBERS SET POINTS = POINTS+3 WHERE NAME = (?)";
                PreparedStatement st4 =
conn.prepareStatement(sql4);
                st4.setString(1, currentUser);
                st4.executeUpdate();
                JOptionPane.showMessageDialog(null,
"Congratulations on finishing SOCSCI! You have gained 3 points!");
            }
            JOptionPane.showMessageDialog(null, "Update
Successfully");

            if (LoginWindow.status.equals("member")) {
                new TeamMemberWindow().setVisible(true);
                dispose();
            } else if (LoginWindow.status.equals("leader")) {
                new TeamLeaderWindow().setVisible(true);
                dispose();
            }
        }
    }

    while (rs2.next()) {
        int lit = rs2.getInt("LIT_PROGRESS");
        int art = rs2.getInt("ART_PROGRESS");
        int sci = rs2.getInt("SCI_PROGRESS");
        int socsci = rs2.getInt("SOCSCI_PROGRESS");
        int mus = rs2.getInt("MUS_PROGRESS");
        int mat = rs2.getInt("MAT_PROGRESS");
    }
}

```



```

        int econ = rs2.getInt("ECON_PROGRESS");
        allCurrentProgress = lit + art + sci + socsci + mus + mat +
econ;
    }

    while (rs1.next()) {
        int tot_lit = rs1.getInt("LIT_PAGE");
        int tot_art = rs1.getInt("ART_PAGE");
        int tot_sci = rs1.getInt("SCI_PAGE");
        int tot_socsci = rs1.getInt("SOCSCI_PAGE");
        int tot_mus = rs1.getInt("MUS_PAGE");
        int tot_mat = rs1.getInt("MAT_PAGE");
        int tot_econ = rs1.getInt("ECON_PAGE");
        allPageNum = tot_lit + tot_art + tot_sci + tot_socsci +
tot_mus + tot_mat + tot_econ;
    }

    int updateOverall = (allCurrentProgress * 100) / allPageNum;
    String sql4 = "UPDATE TEAM1_MEMBERS SET
OVERALL_PROGRESS = (?) WHERE NAME = (?)";
    String sql5 = "UPDATE TEAM2_MEMBERS SET
OVERALL_PROGRESS = (?) WHERE NAME = (?)";
    String sql6 = "UPDATE TEAM3_MEMBERS SET
OVERALL_PROGRESS = (?) WHERE NAME = (?)";
    String sql7 = "UPDATE UNASSIGNED_MEMBERS SET
OVERALL_PROGRESS = (?) WHERE NAME = (?)";
    PreparedStatement st4 = conn.prepareStatement(sql4);
    PreparedStatement st5 = conn.prepareStatement(sql5);
    PreparedStatement st6 = conn.prepareStatement(sql6);
    PreparedStatement st7 = conn.prepareStatement(sql7);
    st4.setInt(1, updateOverall);
    st4.setString(2, currentUser);
    st5.setInt(1, updateOverall);
    st5.setString(2, currentUser);
    st6.setInt(1, updateOverall);
    st6.setString(2, currentUser);
    st7.setInt(1, updateOverall);
    st7.setString(2, currentUser);
    st4.executeUpdate();
    st5.executeUpdate();
    st6.executeUpdate();
    st7.executeUpdate();
}

```

```

        if (subject.equals("Music")) {

            int allCurrentProgress = 0;
            int allPageNum = 0;
            ResultSet rs2 = null;

            String sql1 = "SELECT * FROM PAGE_NUM WHERE
LINE_NUM = (?)";
            PreparedStatement st1 = conn.prepareStatement(sql1);
            st1.setInt(1, 1);
            ResultSet rs1 = st1.executeQuery();

            if (LoginWindow.teamUsed.equals("Team 1")) {
                int nowPage = Integer.valueOf(NewPageField.getText());
                int oriPage = Integer.valueOf(WerePageField.getText());
                int totPage = Integer.valueOf(TotPageField.getText());
                if (nowPage < oriPage) {
                    JOptionPane.showMessageDialog(null, "YOU READ
BACKWARDS?");
                } else if (nowPage > totPage) {
                    JOptionPane.showMessageDialog(null, "YOU READ
MORE THAN THE EXISTING PAGE NUMBER?");
                } else if (nowPage < 0) {
                    JOptionPane.showMessageDialog(null, "YOU READ
NEGATIVE PAGE NUMBERS???");
                } else {
                    String sql3 = "UPDATE TEAM1_MEMBERS SET
MUS_PROGRESS = (?) WHERE NAME = (?)";
                    PreparedStatement st3 = conn.prepareStatement(sql3);
                    st3.setInt(1, nowPage);
                    st3.setString(2, currentUser);
                    st3.executeUpdate();

                    String sql2 = "SELECT * FROM TEAM1_MEMBERS
WHERE NAME = (?)";
                    PreparedStatement st2 = conn.prepareStatement(sql2);
                    st2.setString(1, currentUser);
                    rs2 = st2.executeQuery();

                    if (nowPage == totPage) {
                        String sql4 = "UPDATE TEAM1_MEMBERS
SET POINTS = POINTS+3 WHERE NAME = (?)";
                        PreparedStatement st4 =

```

```

conn.prepareStatement(sql4);
                                st4.setString(1, currentUser);
                                st4.executeUpdate();
                                JOptionPane.showMessageDialog(null,
"Congratulations on finishing MUS! You have gained 3 points!");
                                }
                                JOptionPane.showMessageDialog(null, "Update
Successfully");
                                if (LoginWindow.status.equals("member")) {
                                    new TeamMemberWindow().setVisible(true);
                                    dispose();
                                } else if (LoginWindow.status.equals("leader")) {
                                    new TeamLeaderWindow().setVisible(true);
                                    dispose();
                                }
                                }
                                } else if (LoginWindow.teamUsed.equals("Team 2")) {
                                    int nowPage = Integer.valueOf(NewPageField.getText());
                                    int oriPage = Integer.valueOf(WerePageField.getText());
                                    int totPage = Integer.valueOf(TotPageField.getText());
                                    if (nowPage < oriPage) {
                                        JOptionPane.showMessageDialog(null, "YOU READ
BACKWARDS?");
                                    } else if (nowPage > totPage) {
                                        JOptionPane.showMessageDialog(null, "YOU READ
MORE THAN THE EXISTING PAGE NUMBER?");
                                    } else if (nowPage < 0) {
                                        JOptionPane.showMessageDialog(null, "YOU READ
NEGATIVE PAGE NUMBERS???");
                                    } else {
                                        String sql3 = "UPDATE TEAM2_MEMBERS SET
MUS_PROGRESS = (?) WHERE NAME = (?)";
                                        PreparedStatement st3 = conn.prepareStatement(sql3);
                                        st3.setInt(1, nowPage);
                                        st3.setString(2, currentUser);
                                        st3.executeUpdate();

                                        String sql2 = "SELECT * FROM TEAM2_MEMBERS
WHERE NAME = (?)";
                                        PreparedStatement st2 = conn.prepareStatement(sql2);
                                        st2.setString(1, currentUser);
                                        rs2 = st2.executeQuery();

                                        if (nowPage == totPage) {

```

```

        String sql4 = "UPDATE TEAM2_MEMBERS
SET POINTS = POINTS+3 WHERE NAME = (?)";
        PreparedStatement st4 =
conn.prepareStatement(sql4);
        st4.setString(1, currentUser);
        st4.executeUpdate();
        JOptionPane.showMessageDialog(null,
"Congratulations on finishing MUS! You have gained 3 points!");
    }
    JOptionPane.showMessageDialog(null, "Update
Successfully");

    if (LoginWindow.status.equals("member")) {
        new TeamMemberWindow().setVisible(true);
        dispose();
    } else if (LoginWindow.status.equals("leader")) {
        new TeamLeaderWindow().setVisible(true);
        dispose();
    }
}
} else if (LoginWindow.teamUsed.equals("Team 3")) {
    int nowPage = Integer.valueOf(NewPageField.getText());
    int oriPage = Integer.valueOf(WerePageField.getText());
    int totPage = Integer.valueOf(TotPageField.getText());
    if (nowPage < oriPage) {
        JOptionPane.showMessageDialog(null, "YOU READ
BACKWARDS?");
    } else if (nowPage > totPage) {
        JOptionPane.showMessageDialog(null, "YOU READ
MORE THAN THE EXISTING PAGE NUMBER?");
    } else if (nowPage < 0) {
        JOptionPane.showMessageDialog(null, "YOU READ
NEGATIVE PAGE NUMBERS???");
    } else {
        String sql3 = "UPDATE TEAM3_MEMBERS SET
MUS_PROGRESS = (?) WHERE NAME = (?)";
        PreparedStatement st3 = conn.prepareStatement(sql3);
        st3.setInt(1, nowPage);
        st3.setString(2, currentUser);
        st3.executeUpdate();

        String sql2 = "SELECT * FROM TEAM3_MEMBERS
WHERE NAME = (?)";
        PreparedStatement st2 = conn.prepareStatement(sql2);
        st2.setString(1, currentUser);

```

```

rs2 = st2.executeQuery();

if (nowPage == totPage) {
    String sql4 = "UPDATE TEAM3_MEMBERS
SET POINTS = POINTS+3 WHERE NAME = (?)";
    PreparedStatement st4 =
conn.prepareStatement(sql4);
    st4.setString(1, currentUser);
    st4.executeUpdate();
    JOptionPane.showMessageDialog(null,
"Congratulations on finishing MUS! You have gained 3 points!");
}
JOptionPane.showMessageDialog(null, "Update
Successfully");

if (LoginWindow.status.equals("member")) {
    new TeamMemberWindow().setVisible(true);
    dispose();
} else if (LoginWindow.status.equals("leader")) {
    new TeamLeaderWindow().setVisible(true);
    dispose();
}
}
} else if (LoginWindow.teamUsed.equals("Unassigned")) {
    int nowPage = Integer.valueOf(NewPageField.getText());
    int oriPage = Integer.valueOf(WerePageField.getText());
    int totPage = Integer.valueOf(TotPageField.getText());
    if (nowPage < oriPage) {
        JOptionPane.showMessageDialog(null, "YOU READ
BACKWARDS?");
    } else if (nowPage > totPage) {
        JOptionPane.showMessageDialog(null, "YOU READ
MORE THAN THE EXISTING PAGE NUMBER?");
    } else if (nowPage < 0) {
        JOptionPane.showMessageDialog(null, "YOU READ
NEGATIVE PAGE NUMBERS???");
    } else {
        String sql3 = "UPDATE UNASSIGNED_MEMBERS
SET MUS_PROGRESS = (?) WHERE NAME = (?)";
        PreparedStatement st3 = conn.prepareStatement(sql3);
        st3.setInt(1, nowPage);
        st3.setString(2, currentUser);
        st3.executeUpdate();

        String sql2 = "SELECT * FROM

```

```

UNASSIGNED_MEMBERS WHERE NAME = (?)";
        PreparedStatement st2 = conn.prepareStatement(sql2);
        st2.setString(1, currentUser);
        rs2 = st2.executeQuery();

        if (nowPage == totPage) {
            String sql4 = "UPDATE
UNASSIGNED_MEMBERS SET POINTS = POINTS+3 WHERE NAME = (?)";
            PreparedStatement st4 =
conn.prepareStatement(sql4);

            st4.setString(1, currentUser);
            st4.executeUpdate();
            JOptionPane.showMessageDialog(null,
"Congratulations on finishing MUS! You have gained 3 points!");
        }
        JOptionPane.showMessageDialog(null, "Update
Successfully");

        if (LoginWindow.status.equals("member")) {
            new TeamMemberWindow().setVisible(true);
            dispose();
        } else if (LoginWindow.status.equals("leader")) {
            new TeamLeaderWindow().setVisible(true);
            dispose();
        }
    }
}

while (rs2.next()) {
    int lit = rs2.getInt("LIT_PROGRESS");
    int art = rs2.getInt("ART_PROGRESS");
    int sci = rs2.getInt("SCI_PROGRESS");
    int socsci = rs2.getInt("SOCSCI_PROGRESS");
    int mus = rs2.getInt("MUS_PROGRESS");
    int mat = rs2.getInt("MAT_PROGRESS");
    int econ = rs2.getInt("ECON_PROGRESS");
    allCurrentProgress = lit + art + sci + socsci + mus + mat +
econ;

}

while (rs1.next()) {
    int tot_lit = rs1.getInt("LIT_PAGE");
    int tot_art = rs1.getInt("ART_PAGE");
    int tot_sci = rs1.getInt("SCI_PAGE");
    int tot_socsci = rs1.getInt("SOCSCI_PAGE");

```

```

        int tot_mus = rs1.getInt("MUS_PAGE");
        int tot_mat = rs1.getInt("MAT_PAGE");
        int tot_econ = rs1.getInt("ECON_PAGE");
        allPageNum = tot_lit + tot_art + tot_sci + tot_socsci +
tot_mus + tot_mat + tot_econ;
    }

    int updateOverall = (allCurrentProgress * 100) / allPageNum;
    String sql4 = "UPDATE TEAM1_MEMBERS SET
OVERALL_PROGRESS = (?) WHERE NAME = (?)";
    String sql5 = "UPDATE TEAM2_MEMBERS SET
OVERALL_PROGRESS = (?) WHERE NAME = (?)";
    String sql6 = "UPDATE TEAM3_MEMBERS SET
OVERALL_PROGRESS = (?) WHERE NAME = (?)";
    String sql7 = "UPDATE UNASSIGNED_MEMBERS SET
OVERALL_PROGRESS = (?) WHERE NAME = (?)";
    PreparedStatement st4 = conn.prepareStatement(sql4);
    PreparedStatement st5 = conn.prepareStatement(sql5);
    PreparedStatement st6 = conn.prepareStatement(sql6);
    PreparedStatement st7 = conn.prepareStatement(sql7);
    st4.setInt(1, updateOverall);
    st4.setString(2, currentUser);
    st5.setInt(1, updateOverall);
    st5.setString(2, currentUser);
    st6.setInt(1, updateOverall);
    st6.setString(2, currentUser);
    st7.setInt(1, updateOverall);
    st7.setString(2, currentUser);
    st4.executeUpdate();
    st5.executeUpdate();
    st6.executeUpdate();
    st7.executeUpdate();

}

if (subject.equals("Mathematics")) {

    int allCurrentProgress = 0;
    int allPageNum = 0;
    ResultSet rs2 = null;

    String sql1 = "SELECT * FROM PAGE_NUM WHERE
LINE_NUM = (?)";
    PreparedStatement st1 = conn.prepareStatement(sql1);

```

```

        st1.setInt(1, 1);
        ResultSet rs1 = st1.executeQuery();

        if (LoginWindow.teamUsed.equals("Team 1")) {
            int nowPage = Integer.valueOf(NewPageField.getText());
            int oriPage = Integer.valueOf(WerePageField.getText());
            int totPage = Integer.valueOf(TotPageField.getText());
            if (nowPage < oriPage) {
                JOptionPane.showMessageDialog(null, "YOU READ
BACKWARDS?");
            } else if (nowPage > totPage) {
                JOptionPane.showMessageDialog(null, "YOU READ
MORE THAN THE EXISTING PAGE NUMBER?");
            } else if (nowPage < 0) {
                JOptionPane.showMessageDialog(null, "YOU READ
NEGATIVE PAGE NUMBERS???");
            } else {
                String sql3 = "UPDATE TEAM1_MEMBERS SET
MAT_PROGRESS = (?) WHERE NAME = (?)";
                PreparedStatement st3 = conn.prepareStatement(sql3);
                st3.setInt(1, nowPage);
                st3.setString(2, currentUser);
                st3.executeUpdate();

                String sql2 = "SELECT * FROM TEAM1_MEMBERS
WHERE NAME = (?)";
                PreparedStatement st2 = conn.prepareStatement(sql2);
                st2.setString(1, currentUser);
                rs2 = st2.executeQuery();

                if (nowPage == totPage) {
                    String sql4 = "UPDATE TEAM1_MEMBERS
SET POINTS = POINTS+3 WHERE NAME = (?)";
                    PreparedStatement st4 =
conn.prepareStatement(sql4);
                    st4.setString(1, currentUser);
                    st4.executeUpdate();
                    JOptionPane.showMessageDialog(null,
"Congratulations on finishing MAT! You have gained 3 points!");
                }
                JOptionPane.showMessageDialog(null, "Update
Successfully");

                if (LoginWindow.status.equals("member")) {
                    new TeamMemberWindow().setVisible(true);

```



```

        dispose();
    } else if (LoginWindow.status.equals("leader")) {
        new TeamLeaderWindow().setVisible(true);
        dispose();
    }
}
} else if (LoginWindow.teamUsed.equals("Team 2")) {
    int nowPage = Integer.valueOf(NewPageField.getText());
    int oriPage = Integer.valueOf(WerePageField.getText());
    int totPage = Integer.valueOf(TotPageField.getText());
    if (nowPage < oriPage) {
        JOptionPane.showMessageDialog(null, "YOU READ
BACKWARDS?");
    } else if (nowPage > totPage) {
        JOptionPane.showMessageDialog(null, "YOU READ
MORE THAN THE EXISTING PAGE NUMBER?");
    } else if (nowPage < 0) {
        JOptionPane.showMessageDialog(null, "YOU READ
NEGATIVE PAGE NUMBERS???");
    } else {
        String sql3 = "UPDATE TEAM2_MEMBERS SET
MAT_PROGRESS = (?) WHERE NAME = (?)";
        PreparedStatement st3 = conn.prepareStatement(sql3);
        st3.setInt(1, nowPage);
        st3.setString(2, currentUser);
        st3.executeUpdate();

        String sql2 = "SELECT * FROM TEAM2_MEMBERS
WHERE NAME = (?)";
        PreparedStatement st2 = conn.prepareStatement(sql2);
        st2.setString(1, currentUser);
        rs2 = st2.executeQuery();

        if (nowPage == totPage) {
            String sql4 = "UPDATE TEAM2_MEMBERS
SET POINTS = POINTS+3 WHERE NAME = (?)";
            PreparedStatement st4 =
conn.prepareStatement(sql4);
            st4.setString(1, currentUser);
            st4.executeUpdate();
            JOptionPane.showMessageDialog(null,
"Congratulations on finishing MAT! You have gained 3 points!");
        }
        JOptionPane.showMessageDialog(null, "Update

```

```

Successfully");

        if (LoginWindow.status.equals("member")) {
            new TeamMemberWindow().setVisible(true);
            dispose();
        } else if (LoginWindow.status.equals("leader")) {
            new TeamLeaderWindow().setVisible(true);
            dispose();
        }
    }
} else if (LoginWindow.teamUsed.equals("Team 3")) {
    int nowPage = Integer.valueOf(NewPageField.getText());
    int oriPage = Integer.valueOf(WerePageField.getText());
    int totPage = Integer.valueOf(TotPageField.getText());
    if (nowPage < oriPage) {
        JOptionPane.showMessageDialog(null, "YOU READ
BACKWARDS?");
    } else if (nowPage > totPage) {
        JOptionPane.showMessageDialog(null, "YOU READ
MORE THAN THE EXISTING PAGE NUMBER?");
    } else if (nowPage < 0) {
        JOptionPane.showMessageDialog(null, "YOU READ
NEGATIVE PAGE NUMBERS???");
    } else {
        String sql3 = "UPDATE TEAM3_MEMBERS SET
MAT_PROGRESS = (?) WHERE NAME = (?)";
        PreparedStatement st3 = conn.prepareStatement(sql3);
        st3.setInt(1, nowPage);
        st3.setString(2, currentUser);
        st3.executeUpdate();

        String sql2 = "SELECT * FROM TEAM3_MEMBERS
WHERE NAME = (?)";
        PreparedStatement st2 = conn.prepareStatement(sql2);
        st2.setString(1, currentUser);
        rs2 = st2.executeQuery();

        if (nowPage == totPage) {
            String sql4 = "UPDATE TEAM3_MEMBERS
SET POINTS = POINTS+3 WHERE NAME = (?)";
            PreparedStatement st4 =
conn.prepareStatement(sql4);
            st4.setString(1, currentUser);
            st4.executeUpdate();
            JOptionPane.showMessageDialog(null,

```

```

"Congratulations on finishing MAT! You have gained 3 points!");
    }
    JOptionPane.showMessageDialog(null, "Update
Successfully");

    if (LoginWindow.status.equals("member")) {
        new TeamMemberWindow().setVisible(true);
        dispose();
    } else if (LoginWindow.status.equals("leader")) {
        new TeamLeaderWindow().setVisible(true);
        dispose();
    }
}
} else if (LoginWindow.teamUsed.equals("Unassigned")) {
    int nowPage = Integer.valueOf(NewPageField.getText());
    int oriPage = Integer.valueOf(WerePageField.getText());
    int totPage = Integer.valueOf(TotPageField.getText());
    if (nowPage < oriPage) {
        JOptionPane.showMessageDialog(null, "YOU READ
BACKWARDS?");
    } else if (nowPage > totPage) {
        JOptionPane.showMessageDialog(null, "YOU READ
MORE THAN THE EXISTING PAGE NUMBER?");
    } else if (nowPage < 0) {
        JOptionPane.showMessageDialog(null, "YOU READ
NEGATIVE PAGE NUMBERS???");
    } else {
        String sql3 = "UPDATE UNASSIGNED_MEMBERS
SET MAT_PROGRESS = (?) WHERE NAME = (?)";
        PreparedStatement st3 = conn.prepareStatement(sql3);
        st3.setInt(1, nowPage);
        st3.setString(2, currentUser);
        st3.executeUpdate();

        String sql2 = "SELECT * FROM
UNASSIGNED_MEMBERS WHERE NAME = (?)";
        PreparedStatement st2 = conn.prepareStatement(sql2);
        st2.setString(1, currentUser);
        rs2 = st2.executeQuery();

        if (nowPage == totPage) {
            String sql4 = "UPDATE
UNASSIGNED_MEMBERS SET POINTS = POINTS+3 WHERE NAME = (?)";
            PreparedStatement st4 =
conn.prepareStatement(sql4);

```

```

        st4.setString(1, currentUser);
        st4.executeUpdate();
        JOptionPane.showMessageDialog(null,
"Congratulations on finishing MAT! You have gained 3 points!");
    }
    JOptionPane.showMessageDialog(null, "Update
Successfully");

    if (LoginWindow.status.equals("member")) {
        new TeamMemberWindow().setVisible(true);
        dispose();
    } else if (LoginWindow.status.equals("leader")) {
        new TeamLeaderWindow().setVisible(true);
        dispose();
    }
}

while (rs2.next()) {
    int lit = rs2.getInt("LIT_PROGRESS");
    int art = rs2.getInt("ART_PROGRESS");
    int sci = rs2.getInt("SCI_PROGRESS");
    int socsci = rs2.getInt("SOCSCI_PROGRESS");
    int mus = rs2.getInt("MUS_PROGRESS");
    int mat = rs2.getInt("MAT_PROGRESS");
    int econ = rs2.getInt("ECON_PROGRESS");
    allCurrentProgress = lit + art + sci + socsci + mus + mat +
econ;

}

while (rs1.next()) {
    int tot_lit = rs1.getInt("LIT_PAGE");
    int tot_art = rs1.getInt("ART_PAGE");
    int tot_sci = rs1.getInt("SCI_PAGE");
    int tot_socsci = rs1.getInt("SOCSCI_PAGE");
    int tot_mus = rs1.getInt("MUS_PAGE");
    int tot_mat = rs1.getInt("MAT_PAGE");
    int tot_econ = rs1.getInt("ECON_PAGE");
    allPageNum = tot_lit + tot_art + tot_sci + tot_socsci +
tot_mus + tot_mat + tot_econ;
}

int updateOverall = (allCurrentProgress * 100) / allPageNum;
String sql4 = "UPDATE TEAM1_MEMBERS SET
OVERALL_PROGRESS = (?) WHERE NAME = (?)";

```

```

        String sql5 = "UPDATE TEAM2_MEMBERS SET
OVERALL_PROGRESS = (?) WHERE NAME = (?)";
        String sql6 = "UPDATE TEAM3_MEMBERS SET
OVERALL_PROGRESS = (?) WHERE NAME = (?)";
        String sql7 = "UPDATE UNASSIGNED_MEMBERS SET
OVERALL_PROGRESS = (?) WHERE NAME = (?)";
        PreparedStatement st4 = conn.prepareStatement(sql4);
        PreparedStatement st5 = conn.prepareStatement(sql5);
        PreparedStatement st6 = conn.prepareStatement(sql6);
        PreparedStatement st7 = conn.prepareStatement(sql7);
        st4.setInt(1, updateOverall);
        st4.setString(2, currentUser);
        st5.setInt(1, updateOverall);
        st5.setString(2, currentUser);
        st6.setInt(1, updateOverall);
        st6.setString(2, currentUser);
        st7.setInt(1, updateOverall);
        st7.setString(2, currentUser);
        st4.executeUpdate();
        st5.executeUpdate();
        st6.executeUpdate();
        st7.executeUpdate();

    }

    if (subject.equals("Economics")) {

        int allCurrentProgress = 0;
        int allPageNum = 0;
        ResultSet rs2 = null;

        String sql1 = "SELECT * FROM PAGE_NUM WHERE
LINE_NUM = (?)";
        PreparedStatement st1 = conn.prepareStatement(sql1);
        st1.setInt(1, 1);
        ResultSet rs1 = st1.executeQuery();

        if (LoginWindow.teamUsed.equals("Team 1")) {
            int nowPage = Integer.valueOf(NewPageField.getText());
            int oriPage = Integer.valueOf(WerePageField.getText());
            int totPage = Integer.valueOf(TotPageField.getText());
            if (nowPage < oriPage) {
                JOptionPane.showMessageDialog(null, "YOU READ
BACKWARDS?");
            }
        }
    }
}

```

```

        } else if (nowPage > totPage) {
            JOptionPane.showMessageDialog(null, "YOU READ
MORE THAN THE EXISTING PAGE NUMBER?");
        } else if (nowPage < 0) {
            JOptionPane.showMessageDialog(null, "YOU READ
NEGATIVE PAGE NUMBERS???");
        } else {
            String sql3 = "UPDATE TEAM1_MEMBERS SET
ECON_PROGRESS = (?) WHERE NAME = (?)";
            PreparedStatement st3 = conn.prepareStatement(sql3);
            st3.setInt(1, nowPage);
            st3.setString(2, currentUser);
            st3.executeUpdate();

            String sql2 = "SELECT * FROM TEAM1_MEMBERS
WHERE NAME = (?)";
            PreparedStatement st2 = conn.prepareStatement(sql2);
            st2.setString(1, currentUser);
            rs2 = st2.executeQuery();

            if (nowPage == totPage) {
                String sql4 = "UPDATE TEAM1_MEMBERS
SET POINTS = POINTS+3 WHERE NAME = (?)";
                PreparedStatement st4 =
conn.prepareStatement(sql4);
                st4.setString(1, currentUser);
                st4.executeUpdate();
                JOptionPane.showMessageDialog(null,
"Congratulations on finishing ECON! You have gained 3 points!");
            }
            JOptionPane.showMessageDialog(null, "Update
Successfully");

            if (LoginWindow.status.equals("member")) {
                new TeamMemberWindow().setVisible(true);
                dispose();
            } else if (LoginWindow.status.equals("leader")) {
                new TeamLeaderWindow().setVisible(true);
                dispose();
            }
        }
    } else if (LoginWindow.teamUsed.equals("Team 2")) {
        int nowPage = Integer.valueOf(NewPageField.getText());
        int oriPage = Integer.valueOf(WerePageField.getText());
        int totPage = Integer.valueOf(TotPageField.getText());

```

```

        if (nowPage < oriPage) {
            JOptionPane.showMessageDialog(null, "YOU READ
BACKWARDS?");
        } else if (nowPage > totPage) {
            JOptionPane.showMessageDialog(null, "YOU READ
MORE THAN THE EXISTING PAGE NUMBER?");
        } else if (nowPage < 0) {
            JOptionPane.showMessageDialog(null, "YOU READ
NEGATIVE PAGE NUMBERS???");
        } else {
            String sql3 = "UPDATE TEAM2_MEMBERS SET
ECON_PROGRESS = (?) WHERE NAME = (?)";
            PreparedStatement st3 = conn.prepareStatement(sql3);
            st3.setInt(1, nowPage);
            st3.setString(2, currentUser);
            st3.executeUpdate();

            String sql2 = "SELECT * FROM TEAM2_MEMBERS
WHERE NAME = (?)";
            PreparedStatement st2 = conn.prepareStatement(sql2);
            st2.setString(1, currentUser);
            rs2 = st2.executeQuery();

            if (nowPage == totPage) {
                String sql4 = "UPDATE TEAM2_MEMBERS
SET POINTS = POINTS+3 WHERE NAME = (?)";
                PreparedStatement st4 =
conn.prepareStatement(sql4);
                st4.setString(1, currentUser);
                st4.executeUpdate();
                JOptionPane.showMessageDialog(null,
"Congratulations on finishing ECON! You have gained 3 points!");
            }
            JOptionPane.showMessageDialog(null, "Update
Successfully");

            if (LoginWindow.status.equals("member")) {
                new TeamMemberWindow().setVisible(true);
                dispose();
            } else if (LoginWindow.status.equals("leader")) {
                new TeamLeaderWindow().setVisible(true);
                dispose();
            }
        }
    } else if (LoginWindow.teamUsed.equals("Team 3")) {

```

```

        int nowPage = Integer.valueOf(NewPageField.getText());
        int oriPage = Integer.valueOf(WerePageField.getText());
        int totPage = Integer.valueOf(TotPageField.getText());
        if (nowPage < oriPage) {
            JOptionPane.showMessageDialog(null, "YOU READ
BACKWARDS?");
        } else if (nowPage > totPage) {
            JOptionPane.showMessageDialog(null, "YOU READ
MORE THAN THE EXISTING PAGE NUMBER?");
        } else if (nowPage < 0) {
            JOptionPane.showMessageDialog(null, "YOU READ
NEGATIVE PAGE NUMBERS???");
        } else {
            String sql3 = "UPDATE TEAM3_MEMBERS SET
ECON_PROGRESS = (?) WHERE NAME = (?)";
            PreparedStatement st3 = conn.prepareStatement(sql3);
            st3.setInt(1, nowPage);
            st3.setString(2, currentUser);
            st3.executeUpdate();

            String sql2 = "SELECT * FROM TEAM3_MEMBERS
WHERE NAME = (?)";
            PreparedStatement st2 = conn.prepareStatement(sql2);
            st2.setString(1, currentUser);
            rs2 = st2.executeQuery();

            if (nowPage == totPage) {
                String sql4 = "UPDATE TEAM3_MEMBERS
SET POINTS = POINTS+3 WHERE NAME = (?)";
                PreparedStatement st4 =
conn.prepareStatement(sql4);
                st4.setString(1, currentUser);
                st4.executeUpdate();
                JOptionPane.showMessageDialog(null,
"Congratulations on finishing ECON! You have gained 3 points!");
            }
            JOptionPane.showMessageDialog(null, "Update
Successfully");

            if (LoginWindow.status.equals("member")) {
                new TeamMemberWindow().setVisible(true);
                dispose();
            } else if (LoginWindow.status.equals("leader")) {
                new TeamLeaderWindow().setVisible(true);
                dispose();
            }
        }
    }
}

```



```

    }
}
} else if (LoginWindow.teamUsed.equals("Unassigned")) {
    int nowPage = Integer.valueOf(NewPageField.getText());
    int oriPage = Integer.valueOf(WerePageField.getText());
    int totPage = Integer.valueOf(TotPageField.getText());
    if (nowPage < oriPage) {
        JOptionPane.showMessageDialog(null, "YOU READ
BACKWARDS?");
    } else if (nowPage > totPage) {
        JOptionPane.showMessageDialog(null, "YOU READ
MORE THAN THE EXISTING PAGE NUMBER?");
    } else if (nowPage < 0) {
        JOptionPane.showMessageDialog(null, "YOU READ
NEGATIVE PAGE NUMBERS???");
    } else {
        String sql3 = "UPDATE UNASSIGNED_MEMBERS
SET ECON_PROGRESS = (?) WHERE NAME = (?)";
        PreparedStatement st3 = conn.prepareStatement(sql3);
        st3.setInt(1, nowPage);
        st3.setString(2, currentUser);
        st3.executeUpdate();

        String sql2 = "SELECT * FROM
UNASSIGNED_MEMBERS WHERE NAME = (?)";
        PreparedStatement st2 = conn.prepareStatement(sql2);
        st2.setString(1, currentUser);
        rs2 = st2.executeQuery();

        if (nowPage == totPage) {
            String sql4 = "UPDATE
UNASSIGNED_MEMBERS SET POINTS = POINTS+3 WHERE NAME = (?)";
            PreparedStatement st4 =
conn.prepareStatement(sql4);
            st4.setString(1, currentUser);
            st4.executeUpdate();
            JOptionPane.showMessageDialog(null,
"Congratulations on finishing ECON! You have gained 3 points!");
        }
        JOptionPane.showMessageDialog(null, "Update
Successfully");

        if (LoginWindow.status.equals("member")) {
            new TeamMemberWindow().setVisible(true);
            dispose();
        }
    }
}
}

```

```

        } else if (LoginWindow.status.equals("leader")) {
            new TeamLeaderWindow().setVisible(true);
            dispose();
        }
    }
}

while (rs2.next()) {
    int lit = rs2.getInt("LIT_PROGRESS");
    int art = rs2.getInt("ART_PROGRESS");
    int sci = rs2.getInt("SCI_PROGRESS");
    int socsci = rs2.getInt("SOCSCI_PROGRESS");
    int mus = rs2.getInt("MUS_PROGRESS");
    int mat = rs2.getInt("MAT_PROGRESS");
    int econ = rs2.getInt("ECON_PROGRESS");
    allCurrentProgress = lit + art + sci + socsci + mus + mat +
econ;

}

while (rs1.next()) {
    int tot_lit = rs1.getInt("LIT_PAGE");
    int tot_art = rs1.getInt("ART_PAGE");
    int tot_sci = rs1.getInt("SCI_PAGE");
    int tot_socsci = rs1.getInt("SOCSCI_PAGE");
    int tot_mus = rs1.getInt("MUS_PAGE");
    int tot_mat = rs1.getInt("MAT_PAGE");
    int tot_econ = rs1.getInt("ECON_PAGE");
    allPageNum = tot_lit + tot_art + tot_sci + tot_socsci +
tot_mus + tot_mat + tot_econ;
}

int updateOverall = (allCurrentProgress * 100) / allPageNum;
String sql4 = "UPDATE TEAM1_MEMBERS SET
OVERALL_PROGRESS = (?) WHERE NAME = (?)";
String sql5 = "UPDATE TEAM2_MEMBERS SET
OVERALL_PROGRESS = (?) WHERE NAME = (?)";
String sql6 = "UPDATE TEAM3_MEMBERS SET
OVERALL_PROGRESS = (?) WHERE NAME = (?)";
String sql7 = "UPDATE UNASSIGNED_MEMBERS SET
OVERALL_PROGRESS = (?) WHERE NAME = (?)";
PreparedStatement st4 = conn.prepareStatement(sql4);
PreparedStatement st5 = conn.prepareStatement(sql5);
PreparedStatement st6 = conn.prepareStatement(sql6);
PreparedStatement st7 = conn.prepareStatement(sql7);

```

```

        st4.setInt(1, updateOverall);
        st4.setString(2, currentUser);
        st5.setInt(1, updateOverall);
        st5.setString(2, currentUser);
        st6.setInt(1, updateOverall);
        st6.setString(2, currentUser);
        st7.setInt(1, updateOverall);
        st7.setString(2, currentUser);
        st4.executeUpdate();
        st5.executeUpdate();
        st6.executeUpdate();
        st7.executeUpdate();

    }

    } catch (SQLException ex) {
        JOptionPane.showMessageDialog(null, "error");
    }
}

private void ConfirmBookButtonActionPerformed(java.awt.event.ActionEvent
evt) {
    String currentSelect = (String) SubjectBox.getSelectedItem();
    if (currentSelect.equals("Please Select")) {
        JOptionPane.showMessageDialog(null, "Please Select a Subject!");
        //see if the user have yet to make a selection
    } else {
        subject = currentSelect;
        if (subject.equals("Literature")) {
            try {
                String connectionURL =
                "jdbc:derby://localhost:1527/QHAPPYDatabase";
                java.sql.Connection conn =
                DriverManager.getConnection(connectionURL, "username", "password");
                String sql1 = "SELECT * FROM PAGE_NUM WHERE
                LINE_NUM = (?)";
                PreparedStatement st1 = conn.prepareStatement(sql1);
                st1.setInt(1, 1);
                ResultSet rs1 = st1.executeQuery();
                while (rs1.next()) {
                    String theTotPgNum =
                String.valueOf(rs1.getInt("LIT_PAGE"));
                    //get and display the total page for literature
                    TotPageField.setText(theTotPgNum);
                }
            }
        }
    }
}

```

```

    }
    if (LoginWindow.teamUsed.equals("Team 1")) {
        String sql2 = "SELECT * FROM TEAM1_MEMBERS
WHERE NAME = (?)";

        PreparedStatement st2 = conn.prepareStatement(sql2);
        st2.setString(1, LoginWindow.nameUsing);
        ResultSet rs2 = st2.executeQuery();
        while (rs2.next()) {
            String theCurrentPgNum =
String.valueOf(rs2.getInt("LIT_PROGRESS"));
            //get and display the user's current progress in
reading literature

            WerePageField.setText(theCurrentPgNum);
        }

        //repeat for team2, team3, and unassigned members and
for all subjects
    } else if (LoginWindow.teamUsed.equals("Team 2")) {
        String sql2 = "SELECT * FROM TEAM2_MEMBERS
WHERE NAME = (?)";

        PreparedStatement st2 = conn.prepareStatement(sql2);
        st2.setString(1, LoginWindow.nameUsing);
        ResultSet rs2 = st2.executeQuery();
        while (rs2.next()) {
            String theCurrentPgNum =
String.valueOf(rs2.getInt("LIT_PROGRESS"));
            WerePageField.setText(theCurrentPgNum);
        }
    } else if (LoginWindow.teamUsed.equals("Team 3")) {
        String sql2 = "SELECT * FROM TEAM3_MEMBERS
WHERE NAME = (?)";

        PreparedStatement st2 = conn.prepareStatement(sql2);
        st2.setString(1, LoginWindow.nameUsing);
        ResultSet rs2 = st2.executeQuery();
        while (rs2.next()) {
            String theCurrentPgNum =
String.valueOf(rs2.getInt("LIT_PROGRESS"));
            WerePageField.setText(theCurrentPgNum);
        }
    } else if (LoginWindow.teamUsed.equals("Unassigned")) {
        String sql2 = "SELECT * FROM

```

```

UNASSIGNED_MEMBERS WHERE NAME = (?)";
        PreparedStatement st2 = conn.prepareStatement(sql2);
        st2.setString(1, LoginWindow.nameUsing);
        ResultSet rs2 = st2.executeQuery();
        while (rs2.next()) {
            String theCurrentPgNum =
String.valueOf(rs2.getInt("LIT_PROGRESS"));
            WerePageField.setText(theCurrentPgNum);
        }
    }
} catch (SQLException ex) {
    JOptionPane.showMessageDialog(null, "error");
}

} else if (subject.equals("Art")) {
    try {
        String connectionURL =
"jdbc:derby://localhost:1527/QHAPPYDatabase";
        java.sql.Connection conn =
DriverManager.getConnection(connectionURL, "username", "password");
        String sql1 = "SELECT * FROM PAGE_NUM WHERE
LINE_NUM = (?)";
        PreparedStatement st1 = conn.prepareStatement(sql1);
        st1.setInt(1, 1);
        ResultSet rs1 = st1.executeQuery();
        while (rs1.next()) {
            String theTotPgNum =
String.valueOf(rs1.getInt("ART_PAGE"));
            TotPageField.setText(theTotPgNum);
        }
        if (LoginWindow.teamUsed.equals("Team 1")) {
            String sql2 = "SELECT * FROM TEAM1_MEMBERS
WHERE NAME = (?)";
            PreparedStatement st2 = conn.prepareStatement(sql2);
            st2.setString(1, LoginWindow.nameUsing);
            ResultSet rs2 = st2.executeQuery();
            while (rs2.next()) {
                String theCurrentPgNum =
String.valueOf(rs2.getInt("ART_PROGRESS"));
                WerePageField.setText(theCurrentPgNum);
            }
        } else if (LoginWindow.teamUsed.equals("Team 2")) {
            String sql2 = "SELECT * FROM TEAM2_MEMBERS

```

```

WHERE NAME = (?";

        PreparedStatement st2 = conn.prepareStatement(sql2);
        st2.setString(1, LoginWindow.nameUsing);
        ResultSet rs2 = st2.executeQuery();
        while (rs2.next()) {
            String theCurrentPgNum =
String.valueOf(rs2.getInt("ART_PROGRESS"));
            WerePageField.setText(theCurrentPgNum);

        }
    } else if (LoginWindow.teamUsed.equals("Team 3")) {
        String sql2 = "SELECT * FROM TEAM3_MEMBERS
WHERE NAME = (?";

        PreparedStatement st2 = conn.prepareStatement(sql2);
        st2.setString(1, LoginWindow.nameUsing);
        ResultSet rs2 = st2.executeQuery();
        while (rs2.next()) {
            String theCurrentPgNum =
String.valueOf(rs2.getInt("ART_PROGRESS"));
            WerePageField.setText(theCurrentPgNum);

        }
    } else if (LoginWindow.teamUsed.equals("Unassigned")) {
        String sql2 = "SELECT * FROM
UNASSIGNED_MEMBERS WHERE NAME = (?";

        PreparedStatement st2 = conn.prepareStatement(sql2);
        st2.setString(1, LoginWindow.nameUsing);
        ResultSet rs2 = st2.executeQuery();
        while (rs2.next()) {
            String theCurrentPgNum =
String.valueOf(rs2.getInt("ART_PROGRESS"));
            WerePageField.setText(theCurrentPgNum);

        }
    }
} catch (SQLException ex) {
    JOptionPane.showMessageDialog(null, "error");
}

    } else if (subject.equals("Science")) {
        try {
            String connectionURL =
"jdbc:derby://localhost:1527/QHAPPYDatabase";
            java.sql.Connection conn =
DriverManager.getConnection(connectionURL, "username", "password");

```

```

String sql1 = "SELECT * FROM PAGE_NUM WHERE
LINE_NUM = (?)";
PreparedStatement st1 = conn.prepareStatement(sql1);
st1.setInt(1, 1);
ResultSet rs1 = st1.executeQuery();
while (rs1.next()) {
    String theTotPgNum =
String.valueOf(rs1.getInt("SCI_PAGE"));
    TotPageField.setText(theTotPgNum);
}
if (LoginWindow.teamUsed.equals("Team 1")) {
    String sql2 = "SELECT * FROM TEAM1_MEMBERS
WHERE NAME = (?)";
    PreparedStatement st2 = conn.prepareStatement(sql2);
    st2.setString(1, LoginWindow.nameUsing);
    ResultSet rs2 = st2.executeQuery();
    while (rs2.next()) {
        String theCurrentPgNum =
String.valueOf(rs2.getInt("SCI_PROGRESS"));
        WerePageField.setText(theCurrentPgNum);
    }
} else if (LoginWindow.teamUsed.equals("Team 2")) {
    String sql2 = "SELECT * FROM TEAM2_MEMBERS
WHERE NAME = (?)";
    PreparedStatement st2 = conn.prepareStatement(sql2);
    st2.setString(1, LoginWindow.nameUsing);
    ResultSet rs2 = st2.executeQuery();
    while (rs2.next()) {
        String theCurrentPgNum =
String.valueOf(rs2.getInt("SCI_PROGRESS"));
        WerePageField.setText(theCurrentPgNum);
    }
} else if (LoginWindow.teamUsed.equals("Team 3")) {
    String sql2 = "SELECT * FROM TEAM3_MEMBERS
WHERE NAME = (?)";
    PreparedStatement st2 = conn.prepareStatement(sql2);
    st2.setString(1, LoginWindow.nameUsing);
    ResultSet rs2 = st2.executeQuery();
    while (rs2.next()) {
        String theCurrentPgNum =
String.valueOf(rs2.getInt("SCI_PROGRESS"));
        WerePageField.setText(theCurrentPgNum);
    }
}

```

```

        }
    } else if (LoginWindow.teamUsed.equals("Unassigned")) {
        String sql2 = "SELECT * FROM
UNASSIGNED_MEMBERS WHERE NAME = (?)";
        PreparedStatement st2 = conn.prepareStatement(sql2);
        st2.setString(1, LoginWindow.nameUsing);
        ResultSet rs2 = st2.executeQuery();
        while (rs2.next()) {
            String theCurrentPgNum =
String.valueOf(rs2.getInt("SCI_PROGRESS"));
            WerePageField.setText(theCurrentPgNum);
        }
    }
} catch (SQLException ex) {
    JOptionPane.showMessageDialog(null, "error");
}

} else if (subject.equals("Social Science")) {
    try {
        String connectionURL =
"jdbc:derby://localhost:1527/QHAPPYDatabase";
        java.sql.Connection conn =
DriverManager.getConnection(connectionURL, "username", "password");
        String sql1 = "SELECT * FROM PAGE_NUM WHERE
LINE_NUM = (?)";
        PreparedStatement st1 = conn.prepareStatement(sql1);
        st1.setInt(1, 1);
        ResultSet rs1 = st1.executeQuery();
        while (rs1.next()) {
            String theTotPgNum =
String.valueOf(rs1.getInt("SOCSCI_PAGE"));
            TotPageField.setText(theTotPgNum);
        }
        if (LoginWindow.teamUsed.equals("Team 1")) {
            String sql2 = "SELECT * FROM TEAM1_MEMBERS
WHERE NAME = (?)";
            PreparedStatement st2 = conn.prepareStatement(sql2);
            st2.setString(1, LoginWindow.nameUsing);
            ResultSet rs2 = st2.executeQuery();
            while (rs2.next()) {
                String theCurrentPgNum =
String.valueOf(rs2.getInt("SOCSCI_PROGRESS"));
                WerePageField.setText(theCurrentPgNum);
            }
        }
    }
}

```



```

        }
    } else if (LoginWindow.teamUsed.equals("Team 2")) {
        String sql2 = "SELECT * FROM TEAM2_MEMBERS
WHERE NAME = (?)";
        PreparedStatement st2 = conn.prepareStatement(sql2);
        st2.setString(1, LoginWindow.nameUsing);
        ResultSet rs2 = st2.executeQuery();
        while (rs2.next()) {
            String theCurrentPgNum =
String.valueOf(rs2.getInt("SOCSCI_PROGRESS"));
            WerePageField.setText(theCurrentPgNum);
        }
    } else if (LoginWindow.teamUsed.equals("Team 3")) {
        String sql2 = "SELECT * FROM TEAM3_MEMBERS
WHERE NAME = (?)";
        PreparedStatement st2 = conn.prepareStatement(sql2);
        st2.setString(1, LoginWindow.nameUsing);
        ResultSet rs2 = st2.executeQuery();
        while (rs2.next()) {
            String theCurrentPgNum =
String.valueOf(rs2.getInt("SOCSCI_PROGRESS"));
            WerePageField.setText(theCurrentPgNum);
        }
    } else if (LoginWindow.teamUsed.equals("Unassigned")) {
        String sql2 = "SELECT * FROM
UNASSIGNED_MEMBERS WHERE NAME = (?)";
        PreparedStatement st2 = conn.prepareStatement(sql2);
        st2.setString(1, LoginWindow.nameUsing);
        ResultSet rs2 = st2.executeQuery();
        while (rs2.next()) {
            String theCurrentPgNum =
String.valueOf(rs2.getInt("SOCSCI_PROGRESS"));
            WerePageField.setText(theCurrentPgNum);
        }
    }
} catch (SQLException ex) {
    JOptionPane.showMessageDialog(null, "error");
}

} else if (subject.equals("Music")) {
    try {

```

```

        String connectionURL =
"jdbc:derby://localhost:1527/QHAPPYDatabase";
        java.sql.Connection conn =
DriverManager.getConnection(connectionURL, "username", "password");
        String sql1 = "SELECT * FROM PAGE_NUM WHERE
LINE_NUM = (?)";
        PreparedStatement st1 = conn.prepareStatement(sql1);
        st1.setInt(1, 1);
        ResultSet rs1 = st1.executeQuery();
        while (rs1.next()) {
            String theTotPgNum =
String.valueOf(rs1.getInt("MUS_PAGE"));
            TotPageField.setText(theTotPgNum);
        }
        if (LoginWindow.teamUsed.equals("Team 1")) {
            String sql2 = "SELECT * FROM TEAM1_MEMBERS
WHERE NAME = (?)";
            PreparedStatement st2 = conn.prepareStatement(sql2);
            st2.setString(1, LoginWindow.nameUsing);
            ResultSet rs2 = st2.executeQuery();
            while (rs2.next()) {
                String theCurrentPgNum =
String.valueOf(rs2.getInt("MUS_PROGRESS"));
                WerePageField.setText(theCurrentPgNum);
            }
        } else if (LoginWindow.teamUsed.equals("Team 2")) {
            String sql2 = "SELECT * FROM TEAM2_MEMBERS
WHERE NAME = (?)";
            PreparedStatement st2 = conn.prepareStatement(sql2);
            st2.setString(1, LoginWindow.nameUsing);
            ResultSet rs2 = st2.executeQuery();
            while (rs2.next()) {
                String theCurrentPgNum =
String.valueOf(rs2.getInt("MUS_PROGRESS"));
                WerePageField.setText(theCurrentPgNum);
            }
        } else if (LoginWindow.teamUsed.equals("Team 3")) {
            String sql2 = "SELECT * FROM TEAM3_MEMBERS
WHERE NAME = (?)";
            PreparedStatement st2 = conn.prepareStatement(sql2);
            st2.setString(1, LoginWindow.nameUsing);
            ResultSet rs2 = st2.executeQuery();

```

```

        while (rs2.next()) {
            String theCurrentPgNum =
String.valueOf(rs2.getInt("MUS_PROGRESS"));
            WerePageField.setText(theCurrentPgNum);

        }
    } else if (LoginWindow.teamUsed.equals("Unassigned")) {
        String sql2 = "SELECT * FROM
UNASSIGNED_MEMBERS WHERE NAME = (?)";
        PreparedStatement st2 = conn.prepareStatement(sql2);
        st2.setString(1, LoginWindow.nameUsing);
        ResultSet rs2 = st2.executeQuery();
        while (rs2.next()) {
            String theCurrentPgNum =
String.valueOf(rs2.getInt("MUS_PROGRESS"));
            WerePageField.setText(theCurrentPgNum);
        }
    }
} catch (SQLException ex) {
    JOptionPane.showMessageDialog(null, "error");
}

} else if (subject.equals("Mathematics")) {
    try {
        String connectionURL =
"jdbc:derby://localhost:1527/QHAPPYDatabase";
        java.sql.Connection conn =
DriverManager.getConnection(connectionURL, "username", "password");
        String sql1 = "SELECT * FROM PAGE_NUM WHERE
LINE_NUM = (?)";
        PreparedStatement st1 = conn.prepareStatement(sql1);
        st1.setInt(1, 1);
        ResultSet rs1 = st1.executeQuery();
        while (rs1.next()) {
            String theTotPgNum =
String.valueOf(rs1.getInt("MAT_PAGE"));
            TotPageField.setText(theTotPgNum);
        }
        if (LoginWindow.teamUsed.equals("Team 1")) {
            String sql2 = "SELECT * FROM TEAM1_MEMBERS
WHERE NAME = (?)";
            PreparedStatement st2 = conn.prepareStatement(sql2);
            st2.setString(1, LoginWindow.nameUsing);
            ResultSet rs2 = st2.executeQuery();

```

```

        while (rs2.next()) {
            String theCurrentPgNum =
String.valueOf(rs2.getInt("MAT_PROGRESS"));
            WerePageField.setText(theCurrentPgNum);

        }
    } else if (LoginWindow.teamUsed.equals("Team 2")) {
        String sql2 = "SELECT * FROM TEAM2_MEMBERS
WHERE NAME = (?)";
        PreparedStatement st2 = conn.prepareStatement(sql2);
        st2.setString(1, LoginWindow.nameUsing);
        ResultSet rs2 = st2.executeQuery();
        while (rs2.next()) {
            String theCurrentPgNum =
String.valueOf(rs2.getInt("MAT_PROGRESS"));
            WerePageField.setText(theCurrentPgNum);

        }
    } else if (LoginWindow.teamUsed.equals("Team 3")) {
        String sql2 = "SELECT * FROM TEAM3_MEMBERS
WHERE NAME = (?)";
        PreparedStatement st2 = conn.prepareStatement(sql2);
        st2.setString(1, LoginWindow.nameUsing);
        ResultSet rs2 = st2.executeQuery();
        while (rs2.next()) {
            String theCurrentPgNum =
String.valueOf(rs2.getInt("MAT_PROGRESS"));
            WerePageField.setText(theCurrentPgNum);

        }
    } else if (LoginWindow.teamUsed.equals("Unassigned")) {
        String sql2 = "SELECT * FROM
UNASSIGNED_MEMBERS WHERE NAME = (?)";
        PreparedStatement st2 = conn.prepareStatement(sql2);
        st2.setString(1, LoginWindow.nameUsing);
        ResultSet rs2 = st2.executeQuery();
        while (rs2.next()) {
            String theCurrentPgNum =
String.valueOf(rs2.getInt("MAT_PROGRESS"));
            WerePageField.setText(theCurrentPgNum);

        }
    }
} catch (SQLException ex) {
    JOptionPane.showMessageDialog(null, "error");
}

```

```

    }

    } else if (subject.equals("Economics")) {
        try {
            String connectionURL =
"jdbc:derby://localhost:1527/QHAPPYDatabase";
            java.sql.Connection conn =
DriverManager.getConnection(connectionURL, "username", "password");
            String sql1 = "SELECT * FROM PAGE_NUM WHERE
LINE_NUM = (?)";

            PreparedStatement st1 = conn.prepareStatement(sql1);
            st1.setInt(1, 1);
            ResultSet rs1 = st1.executeQuery();
            while (rs1.next()) {
                String theTotPgNum =
String.valueOf(rs1.getInt("ECON_PAGE"));
                TotPageField.setText(theTotPgNum);
            }
            if (LoginWindow.teamUsed.equals("Team 1")) {
                String sql2 = "SELECT * FROM TEAM1_MEMBERS
WHERE NAME = (?)";

                PreparedStatement st2 = conn.prepareStatement(sql2);
                st2.setString(1, LoginWindow.nameUsing);
                ResultSet rs2 = st2.executeQuery();
                while (rs2.next()) {
                    String theCurrentPgNum =
String.valueOf(rs2.getInt("ECON_PROGRESS"));
                    WerePageField.setText(theCurrentPgNum);
                }
            }
            } else if (LoginWindow.teamUsed.equals("Team 2")) {
                String sql2 = "SELECT * FROM TEAM2_MEMBERS
WHERE NAME = (?)";

                PreparedStatement st2 = conn.prepareStatement(sql2);
                st2.setString(1, LoginWindow.nameUsing);
                ResultSet rs2 = st2.executeQuery();
                while (rs2.next()) {
                    String theCurrentPgNum =
String.valueOf(rs2.getInt("ECON_PROGRESS"));
                    WerePageField.setText(theCurrentPgNum);
                }
            }
            } else if (LoginWindow.teamUsed.equals("Team 3")) {
                String sql2 = "SELECT * FROM TEAM3_MEMBERS

```

```

WHERE NAME = (?)";

        PreparedStatement st2 = conn.prepareStatement(sql2);
        st2.setString(1, LoginWindow.nameUsing);
        ResultSet rs2 = st2.executeQuery();
        while (rs2.next()) {
            String theCurrentPgNum =
String.valueOf(rs2.getInt("ECON_PROGRESS"));
            WerePageField.setText(theCurrentPgNum);

        }
    } else if (LoginWindow.teamUsed.equals("Unassigned")) {
        String sql2 = "SELECT * FROM
UNASSIGNED_MEMBERS WHERE NAME = (?)";
        PreparedStatement st2 = conn.prepareStatement(sql2);
        st2.setString(1, LoginWindow.nameUsing);
        ResultSet rs2 = st2.executeQuery();
        while (rs2.next()) {
            String theCurrentPgNum =
String.valueOf(rs2.getInt("ECON_PROGRESS"));
            WerePageField.setText(theCurrentPgNum);

        }
    }
} catch (SQLException ex) {
    JOptionPane.showMessageDialog(null, "error");
}

}

}

private void GetBackButtonActionPerformed(java.awt.event.ActionEvent evt) {
    if (LoginWindow.status.equals("member")) {
        new TeamMemberWindow().setVisible(true);
        dispose();
    } else if (LoginWindow.status.equals("leader")) {
        new TeamLeaderWindow().setVisible(true);
        dispose();
    }
}

/**
 * @param args the command line arguments
 */

```

```

    public static void main(String args[]) {
        /* Set the Nimbus look and feel */
        //<editor-fold defaultstate="collapsed" desc=" Look and feel setting code
(optional) ">
        /* If Nimbus (introduced in Java SE 6) is not available, stay with the default
look and feel.
        * For details see
http://download.oracle.com/javase/tutorial/uiswing/lookandfeel/plaf.html
        */
        try {
            for (javax.swing.UIManager.LookAndFeelInfo info :
javax.swing.UIManager.getInstalledLookAndFeels()) {
                if ("Nimbus".equals(info.getName())) {

javax.swing.UIManager.setLookAndFeel(info.getClassName());
                    break;

                }
            }
        } catch (ClassNotFoundException ex) {
            java.util.logging.Logger.getLogger(UpdateProgress.class
                .getName()).log(java.util.logging.Level.SEVERE, null, ex);

        } catch (InstantiationException ex) {
            java.util.logging.Logger.getLogger(UpdateProgress.class
                .getName()).log(java.util.logging.Level.SEVERE, null, ex);

        } catch (IllegalAccessException ex) {
            java.util.logging.Logger.getLogger(UpdateProgress.class
                .getName()).log(java.util.logging.Level.SEVERE, null, ex);

        } catch (javax.swing.UnsupportedLookAndFeelException ex) {
            java.util.logging.Logger.getLogger(UpdateProgress.class
                .getName()).log(java.util.logging.Level.SEVERE, null, ex);
        }
        //</editor-fold>

        /* Create and display the form */
        java.awt.EventQueue.invokeLater(new Runnable() {
            public void run() {
                new UpdateProgress().setVisible(true);
            }
        });
    }
}

```

```
// Variables declaration - do not modify
private javax.swing.JButton ConfirmBookButton;
private javax.swing.JButton ConfirmUpdateButton;
private javax.swing.JButton GetBackButton;
private javax.swing.JTextField NewPageField;
private javax.swing.JComboBox<String> SubjectBox;
private javax.swing.JTextField TotPageField;
private javax.swing.JTextField WerePageField;
private javax.swing.JLabel jLabel1;
private javax.swing.JLabel jLabel2;
private javax.swing.JLabel jLabel3;
private javax.swing.JLabel jLabel4;
private javax.swing.JLabel jLabel5;
private javax.swing.JLabel jLabel6;
// End of variables declaration
}
```