```java
/**
* The public AnnouncementsProgram class is a GUI and data model for login.
*/

import java.awt.Graphics;
import java.awt.event.ActionEvent;
import java.awt.event.ActionListener;
import javax.swing.JButton;
import javax.swing.JFrame;
import javax.swing.JLabel;
import javax.swing.JPanel;
import javax.swing.JTextField;

public class AnnouncementsProgram{
 protected JFrame loginGUI;
 private JPanel loginPanel;

 private static final int DEFAULT_WIDTH = 350;
 private static final int DEFAULT_HEIGHT = 250;

 /**
  * The default constructor that is used for initializing the graphical user interface for login.
  */
 public AnnouncementsProgram() {
  Helpers helper = new Helpers();
  helper.initFonts();
  initFrame();
  initPanel();
 }

 /**
  * Initializes the login JFrame.
  */
 private void initFrame() {
  loginGUI = new JFrame("VPCI Announcements");
  loginGUI.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
  loginGUI.setSize(DEFAULT_WIDTH , DEFAULT_HEIGHT);
  loginGUI.setResizable(false);
 }

 /**
  * Initializes the JPanel portion of the graphical user interface for login; all JComponents initialized and added here.
  */
 @SuppressWarnings("serial")
 private void initPanel() {
  loginPanel = new JPanel(null) {
   public void paintComponent(Graphics g) {
         super.paintComponent(g);
         g.setColor(Helpers.BLACK);
         g.setFont(Helpers.revansFont.deriveFont(20F));
         g.drawString("VPCI Announcements Login", 10, 30);
         g.setFont(Helpers.consolasFont);
         g.drawString("Account:", 10, 70);
         g.drawString("Password:", 10, 120);
   }
  };

  JLabel errorMessage = new JLabel();
  errorMessage.setFont(Helpers.consolasFontBold);
       errorMessage.setBounds(48, 130, 200, 40);
       errorMessage.setForeground(Helpers.RED);

  JTextField idField = new JTextField(10);
  idField.setBounds(120, 50, 80, 30);
       JTextField passField = new JTextField(10);
```

```
        passField.setBounds(120, 100, 80, 30);

        JButton loginButton = new JButton("Login");
        loginButton.setBounds(30, 170, 70, 30);
        loginButton.addActionListener(new ActionListener() {
         @Override
    public void actionPerformed(ActionEvent e) {
    // TODO Auto-generated method stub
    if (e.getSource() == loginButton) {
     String enteredId = idField.getText().toLowerCase();

     boolean hasCorrectId = false;
     for(int i = 0; i<Helpers.loginInfoLength; i++) {
      String verifiedId = Helpers.loginInfo[i].split(" ")[0];
      if(enteredId.equals(verifiedId)) {
       hasCorrectId = true;
       break;
      }
     }
     if(hasCorrectId) {
      int id = Integer.valueOf(enteredId.substring(5)); //only checking the numbers
      String enteredPass = passField.getText().replaceAll(" ", "").trim();
      String correctPass = Helpers.loginInfo[id-1].split(" ")[1].trim();
      if(!(enteredPass.equals(correctPass))) {
       errorMessage.setText("Wrong password!!!");
      }
      else {
       try {
        Thread.sleep(1000);
       } catch (InterruptedException e1) {
        // TODO Auto-generated catch block
        e1.printStackTrace();
       }
       loginGUI.dispose();
       if(id == 1) {
        enteredId = "Administrator";
        AdminGUI a = new AdminGUI(enteredId);
        a.run();
       }
       else {
        TeacherGUI t = new TeacherGUI(enteredId);
        t.run();
       }
      }
     }
     else {
      errorMessage.setText("Invalid ID!!!");
     }
     loginPanel.repaint();
    }
   }
});

        JButton exitButton = new JButton("Exit");
        exitButton.setBounds(250, 170, 60, 30);
        exitButton.addActionListener(new ActionListener() {
         @Override
    public void actionPerformed(ActionEvent e) {
    // TODO Auto-generated method stub
    if (e.getSource() == exitButton) {
     Helpers.exitPrompt(loginGUI);
    }
   }
});

        // Components Added using Flow Layout
```

```
        loginPanel.add(idField);
        loginPanel.add(passField);
        loginPanel.add(errorMessage);
        loginPanel.add(loginButton);
        loginPanel.add(exitButton);

        loginGUI.add(loginPanel);
    }

    /**
     * Main class that is responsible for program to run.
     * @param args is the arguments that will be ran
     */
    public static void main(String[] args) {
     AnnouncementsProgram a = new AnnouncementsProgram();
     a.run();
    }

    /**
     * The method that makes the login JFrame visible.
     */
    public void run() {
     loginGUI.setVisible(true);
    }
}
```

```java
/**
 * The abstract UserGUI class is a GUI model and data model for all users.
 */

import javax.swing.JFrame;

abstract class UserGUI {
 protected JFrame userGUI;
 protected String id;

 private static final int DEFAULT_WIDTH = 800;
 private static final int DEFAULT_HEIGHT = 600;

 /**
 * The default constructor if no strId is entered.
 */

 public UserGUI() {}

 /**
  * The constructor used for initializing the graphical user interface for this user.
  * @param strId is the user's id
  */
 public UserGUI(String strId) {
  this.id = strId;
  this.initFrame();
 }

 /**
  * The method that initializes the user's JFrame.
  */
 protected void initFrame() {
  userGUI = new JFrame("VPCI Announcements");
  userGUI.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
  userGUI.setSize(DEFAULT_WIDTH , DEFAULT_HEIGHT);
  userGUI.setResizable(false);
 }

 /**
  * The method that makes the user's JFrame visible.
  */
 public void run() {
  this.userGUI.setVisible(true);
 }
}
```

```java
/**
* The public AdminGUI class is a GUI and data model for the administrator.
*/

import java.awt.Graphics;
import java.awt.event.ActionEvent;
import java.awt.event.ActionListener;
import java.awt.event.AdjustmentEvent;
import java.awt.event.AdjustmentListener;
import javax.swing.JButton;
import javax.swing.JLabel;
import javax.swing.JPanel;
import javax.swing.JScrollBar;
import javax.swing.JTextArea;
import javax.swing.JTextField;

public class AdminGUI extends UserGUI {
 private JPanel adminPanel;
 private boolean gotAll;
 private int userSize;
 private String displayText;

 /**
 * The default constructor if no strId is entered; the constructor used for initializing the graphical user interface for administrator.
 */
 public AdminGUI() {
  super("Administrator");
  this.id = "Administrator";
  this.userSize = Helpers.loginInfoLength;
  this.displayText = "";
  this.gotAll = false;
  this.initPanel();
 }

 /**
 * Another constructor used for initializing the graphical user interface for administrator.
 * @param strId is the user's id
 */
 public AdminGUI(String strId) {
  super(strId);
  this.id = strId;
  this.userSize = Helpers.loginInfoLength;
  this.displayText = "";
  this.gotAll = false;
  this.initPanel();
 }

 /**
  * The method used for initializing the JPanel portion of the graphical user interface for this administrator; all JComponents initialized and added here.
  */
 @SuppressWarnings("serial")
 private void initPanel() {
  adminPanel = new JPanel(null) {
   public void paintComponent(Graphics g) {
         super.paintComponent(g);

         g.setColor(Helpers.BLACK);
         g.setFont(Helpers.revansFont.deriveFont(20F));
         g.drawString("VPCI Announcements Admin Manager", 10, 30);
         g.setFont(Helpers.consolasFont);
         g.drawString("Functions:", 10, 80);

         g.drawString("Welcome, ", 520, 30);
   }
```

```
};

JLabel idLabel = new JLabel();
idLabel.setFont(Helpers.consolasFont);
idLabel.setBounds(610, 16, 200, 20);
idLabel.setText(id + "!");
idLabel.setForeground(Helpers.BLACK);

JLabel showMessage = new JLabel();
showMessage.setFont(Helpers.consolasFontBold);
showMessage.setBounds(10, 300, 200, 40);
showMessage.setForeground(Helpers.BLACK);

JTextArea displayArea = new JTextArea("Generated passwords will appear here.");
displayArea.setBounds(250, 120, 400, 380);
displayArea.setLineWrap(true);
displayArea.setWrapStyleWord(true);
displayArea.setEditable(false);
displayArea.setOpaque(true);

JScrollBar scrollBar = new JScrollBar();
scrollBar.setBounds(660, 120, 20, 360);
scrollBar.addAdjustmentListener(new AdjustmentListener() {
 @Override
    public void adjustmentValueChanged(AdjustmentEvent e) {
     if(e.getSource() == scrollBar && gotAll) {
      displayText = "";
      try {
       int scrollValue = scrollBar.getValue();
       scrollValue = (scrollValue*(userSize-20))/90;
    for(int i = scrollValue; i < userSize; i++) {
     displayText += Helpers.loginInfo[i] + "\n";
    }
    displayArea.setText(displayText);
      } catch(ArrayIndexOutOfBoundsException e1) {
       e1.printStackTrace();
      }

     }
    }
});

JTextField idField = new JTextField("Enter ID (teachxxx) to get password");
idField.setBounds(10, 270, 200, 30);

JTextField sizeField = new JTextField("Enter new user group size here");
sizeField.setBounds(10, 500, 200, 30);

JButton newPassButton = new JButton("New Passwords (" + userSize + " users)");
newPassButton.setBounds(10, 100, 200, 50);
newPassButton.setBackground(Helpers.LGRAY);
newPassButton.addActionListener(new ActionListener() {
      @Override
 public void actionPerformed(ActionEvent e) {
  // TODO Auto-generated method stub
  if (e.getSource() == newPassButton) {
   gotAll = true;
   Helpers.generatePasswords(userGUI, userSize);
   Helpers.initLoginResources();
   displayText = "";
   for(int i = 0; i < 20; i++) {
    displayText += Helpers.loginInfo[i] + "\n";
   }
   displayArea.setText(displayText);
  }
 }
```

```
});

JButton getPassButton = new JButton("Get Password");
getPassButton.setBounds(10, 200, 200, 50);
getPassButton.setBackground(Helpers.LGRAY);
getPassButton.addActionListener(new ActionListener() {
      @Override
 public void actionPerformed(ActionEvent e) {
  // TODO Auto-generated method stub
  if (e.getSource() == getPassButton) {
   String enteredId = idField.getText().toLowerCase(); //remove to lowercase?
   int intId = 0;

   boolean hasCorrectId = false;
   for(int i = 0; i < userSize; i++) {
    String verifiedId = Helpers.loginInfo[i].split(" ")[0];
    if(enteredId.equals(verifiedId)) {
     hasCorrectId = true;
     intId = i+1;
     break;
    }
   }
   if(hasCorrectId) {
    showMessage.setForeground(Helpers.BLACK);
    showMessage.setText("Password: " + Helpers.loginInfo[intId-1].split(" ")[1]);
   }
   else {
    showMessage.setForeground(Helpers.RED);
    showMessage.setText("Invalid ID!!!");
   }
   adminPanel.repaint();
  }
 }
});

JButton getPassesButton = new JButton("Get All Passwords");
getPassesButton.setBounds(10, 350, 200, 50);
getPassesButton.setBackground(Helpers.LGRAY);
getPassesButton.addActionListener(new ActionListener() {
      @Override
 public void actionPerformed(ActionEvent e) {
  // TODO Auto-generated method stub
  if (e.getSource() == getPassesButton) {
   gotAll = true;
   Helpers.initLoginResources();
   displayText = "";
   for(int i = 0; i < 20; i++) {
    displayText += Helpers.loginInfo[i] + "\n";
   }
   displayArea.setText(displayText);
  }
 }
});

JButton newSizeButton = new JButton("New User Group Size");
newSizeButton.setBounds(10, 450, 200, 30);
newSizeButton.setBackground(Helpers.LGRAY);
newSizeButton.addActionListener(new ActionListener() {
      @Override
 public void actionPerformed(ActionEvent e) {
  // TODO Auto-generated method stub
  if (e.getSource() == newSizeButton) {
   String enteredSize = sizeField.getText().trim();
   try {
    userSize = Integer.parseInt(enteredSize);
    newPassButton.doClick();
```

```
    Helpers.initLoginResources();
    displayText = "";
    for(int i = 0; i < 20; i++) {
     displayText += Helpers.loginInfo[i] + "\n";
    }
    displayArea.setText(displayText);
    newPassButton.setText("New Passwords (" + userSize + " users)");
   } catch(NumberFormatException e1) {
    showMessage.setText("Integer user size!");
   }

  }
 }
});

JButton logoutButton = new JButton("Logout");
logoutButton.setBounds(650, 500, 100, 50);
logoutButton.addActionListener(new ActionListener() {
      @Override
 public void actionPerformed(ActionEvent e) {
  // TODO Auto-generated method stub
  if (e.getSource() == logoutButton) {
   Helpers.logoutPrompt(userGUI);
   userGUI.dispose();
   AnnouncementsProgram a = new AnnouncementsProgram();
   a.run();
  }
 }
});

// Components Added using Flow Layout
adminPanel.add(idLabel);
adminPanel.add(showMessage);

adminPanel.add(displayArea);
adminPanel.add(scrollBar);
adminPanel.add(idField);
adminPanel.add(sizeField);

adminPanel.add(newPassButton);
adminPanel.add(getPassButton);
adminPanel.add(getPassesButton);
adminPanel.add(newSizeButton);
adminPanel.add(logoutButton);

userGUI.add(adminPanel);

 }
}
```

```java
/**
* The public TeacherGUI class is a GUI and data model for all teachers that are wishing to send an announcement.
*/

import java.awt.Graphics;
import java.awt.event.ActionEvent;
import java.awt.event.ActionListener;
import javax.swing.JButton;
import javax.swing.JLabel;
import javax.swing.JPanel;
import javax.swing.JTextArea;
import javax.swing.JTextField;

public class TeacherGUI extends UserGUI{
 private JPanel teachPanel;
 private int tag;
 private int count;
 private int[] wordLims = {15, 200, 120, 100, 75, 350};
 private String[] strArrAnnouncement;
 private String strAnnouncement;
 private String strSubject;
 private String dates;

 /**
 * The default constructor if no strId is entered.
 */
 public TeacherGUI() {}

 /**
  * The constructor used for initializing the graphical user interface for this teacher; calls initPanel() and creates an instance of
Helpers.
  * @param strId is the user's id
  */
 public TeacherGUI(String strId) {
  super(strId);
  this.id = strId;
  this.count = 0;
  this.initPanel();
 }


 /**
  * The method used for initializing the JPanel portion of the graphical user interface for this teacher; all JComponents initialized
and added here.
  */
 @SuppressWarnings("serial")
 private void initPanel() {
  teachPanel = new JPanel(null) {
   public void paintComponent(Graphics g) {
         super.paintComponent(g);

         g.setColor(Helpers.BLACK);
         g.setFont(Helpers.revansFont.deriveFont(20F));
         g.drawString("VPCI Announcements Editor", 10, 30);
         g.setFont(Helpers.consolasFont);
         g.drawString("Subject:", 10, 60);
         g.drawString("Announcement:", 10, 100);
         g.drawString("Days to Read (E.g., Jan. 1, 3, 5):", 10, 550);
         g.drawString("Announcement Type:", 570, 110);
         g.drawString("Welcome, ", 520, 30);
   }
  };

  JLabel idLabel = new JLabel();
  idLabel.setFont(Helpers.consolasFont);
```

```
idLabel.setBounds(610, 16, 200, 20);
idLabel.setText(id + "!");
idLabel.setForeground(Helpers.BLACK);

JLabel wordCountLabel = new JLabel();
wordCountLabel.setFont(Helpers.consolasFontBold);
wordCountLabel.setBounds(300, 90, 200, 20);
wordCountLabel.setText("Word Count: " + count);
wordCountLabel.setForeground(Helpers.BLACK);

JTextField subjectField = new JTextField(40);
subjectField.setBounds(100, 40, 400, 30);

JTextArea announArea = new JTextArea("Please enter your announcement here");
announArea.setBounds(10, 120, 550, 400);
announArea.setLineWrap(true);
announArea.setWrapStyleWord(true);
announArea.setEditable(true);
announArea.setOpaque(true);

JTextField dateField = new JTextField("No Specified Date");
dateField.setEditable(true);
dateField.setBounds(350, 530, 200, 30);

JButton type1Button = new JButton("Song (15 Max.)");
JButton type2Button = new JButton("Heritage Month (200 Max.)");
JButton type3Button = new JButton("SLC (120 Max.)");
JButton type4Button = new JButton("Arts/Sports (100 Max.)");
JButton type5Button = new JButton("Other (75 Max.)");
JButton type6Button = new JButton("Guidance (350 Max.)");


type1Button.setBounds(580, 130, 200, 30);
type1Button.setBackground(Helpers.LGRAY);
type1Button.addActionListener(new ActionListener() {
      @Override
 public void actionPerformed(ActionEvent e) {
 // TODO Auto-generated method stub
 if (e.getSource() == type1Button) {
  type1Button.setBackground(Helpers.GRAY);
  type2Button.setBackground(Helpers.LGRAY);
  type3Button.setBackground(Helpers.LGRAY);
  type4Button.setBackground(Helpers.LGRAY);
  type5Button.setBackground(Helpers.LGRAY);
  type6Button.setBackground(Helpers.LGRAY);
  tag = 0;
  if(count <= wordLims[tag]) {
   wordCountLabel.setForeground(Helpers.BLACK);
  }
  else {
   wordCountLabel.setForeground(Helpers.RED);
  }
  wordCountLabel.setText("Word Count: " + count + "/" + wordLims[tag]);
 }
 }
});

type2Button.setBounds(580, 180, 200, 30);
type2Button.setBackground(Helpers.LGRAY);
type2Button.addActionListener(new ActionListener() {
      @Override
 public void actionPerformed(ActionEvent e) {
 // TODO Auto-generated method stub
 if (e.getSource() == type2Button) {
  type1Button.setBackground(Helpers.LGRAY);
  type2Button.setBackground(Helpers.GRAY);
```

```
   type3Button.setBackground(Helpers.LGRAY);
   type4Button.setBackground(Helpers.LGRAY);
   type5Button.setBackground(Helpers.LGRAY);
   type6Button.setBackground(Helpers.LGRAY);
   tag = 1;
   if(count <= wordLims[tag]) {
    wordCountLabel.setForeground(Helpers.BLACK);
   }
   else {
    wordCountLabel.setForeground(Helpers.RED);
   }
   wordCountLabel.setText("Word Count: " + count + "/" + wordLims[tag]);
  }
 }
});

type3Button.setBounds(580, 230, 200, 30);
type3Button.setBackground(Helpers.LGRAY);
type3Button.addActionListener(new ActionListener() {
      @Override
 public void actionPerformed(ActionEvent e) {
  // TODO Auto-generated method stub
  if (e.getSource() == type3Button) {
   type1Button.setBackground(Helpers.LGRAY);
   type2Button.setBackground(Helpers.LGRAY);
   type3Button.setBackground(Helpers.GRAY);
   type4Button.setBackground(Helpers.LGRAY);
   type5Button.setBackground(Helpers.LGRAY);
   type6Button.setBackground(Helpers.LGRAY);
   tag = 2;
   if(count <= wordLims[tag]) {
    wordCountLabel.setForeground(Helpers.BLACK);
   }
   else {
    wordCountLabel.setForeground(Helpers.RED);
   }
   wordCountLabel.setText("Word Count: " + count + "/" + wordLims[tag]);
  }
 }
});

type4Button.setBounds(580, 280, 200, 30);
type4Button.setBackground(Helpers.LGRAY);
type4Button.addActionListener(new ActionListener() {
      @Override
 public void actionPerformed(ActionEvent e) {
  // TODO Auto-generated method stub
  if (e.getSource() == type4Button) {
   type1Button.setBackground(Helpers.LGRAY);
   type2Button.setBackground(Helpers.LGRAY);
   type3Button.setBackground(Helpers.LGRAY);
   type4Button.setBackground(Helpers.GRAY);
   type5Button.setBackground(Helpers.LGRAY);
   type6Button.setBackground(Helpers.LGRAY);
   tag = 3;
   if(count <= wordLims[tag]) {
    wordCountLabel.setForeground(Helpers.BLACK);
   }
   else {
    wordCountLabel.setForeground(Helpers.RED);
   }
   wordCountLabel.setText("Word Count: " + count + "/" + wordLims[tag]);
  }
 }
});
```

```
type5Button.setBounds(580, 330, 200, 30);
type5Button.setBackground(Helpers.LGRAY);
type5Button.addActionListener(new ActionListener() {
      @Override
 public void actionPerformed(ActionEvent e) {
  // TODO Auto-generated method stub
  if (e.getSource() == type5Button) {
   type1Button.setBackground(Helpers.LGRAY);
   type2Button.setBackground(Helpers.LGRAY);
   type3Button.setBackground(Helpers.LGRAY);
   type4Button.setBackground(Helpers.LGRAY);
   type5Button.setBackground(Helpers.GRAY);
   type6Button.setBackground(Helpers.LGRAY);
   tag = 4;
   if(count <= wordLims[tag]) {
    wordCountLabel.setForeground(Helpers.BLACK);
   }
   else {
    wordCountLabel.setForeground(Helpers.RED);
   }
   wordCountLabel.setText("Word Count: " + count + "/" + wordLims[tag]);
  }
 }
});

type6Button.setBounds(580, 380, 200, 30);
type6Button.setBackground(Helpers.LGRAY);
type6Button.addActionListener(new ActionListener() {
      @Override
 public void actionPerformed(ActionEvent e) {
  // TODO Auto-generated method stub
  if (e.getSource() == type6Button) {
   type1Button.setBackground(Helpers.LGRAY);
   type2Button.setBackground(Helpers.LGRAY);
   type3Button.setBackground(Helpers.LGRAY);
   type4Button.setBackground(Helpers.LGRAY);
   type5Button.setBackground(Helpers.LGRAY);
   type6Button.setBackground(Helpers.GRAY);
   tag = 5;
   if(count <= wordLims[tag]) {
    wordCountLabel.setForeground(Helpers.BLACK);
   }
   else {
    wordCountLabel.setForeground(Helpers.RED);
   }
   wordCountLabel.setText("Word Count: " + count + "/" + wordLims[tag]);
  }
 }
});

JButton countButton = new JButton("Count Update");
countButton.setBounds(570, 530, 110, 30);
countButton.addActionListener(new ActionListener() {
      @Override
 public void actionPerformed(ActionEvent e) {
  // TODO Auto-generated method stub
  if (e.getSource() == countButton) {
   strAnnouncement = announArea.getText().trim().replaceAll("\\s+", " ").replaceAll("\t", "");
   strArrAnnouncement = strAnnouncement.split(" ");
   count = strArrAnnouncement.length;
   if(count <= wordLims[tag]) {
    wordCountLabel.setForeground(Helpers.BLACK);
   }
   else {
    wordCountLabel.setForeground(Helpers.RED);
```

```
   }
   wordCountLabel.setText("Word Count: " + count + "/" + wordLims[tag]);
  }
 }
});

JButton saveButton = new JButton("Save Locally");
saveButton.setBounds(600, 460, 150, 30);
saveButton.addActionListener(new ActionListener() {
      @Override
 public void actionPerformed(ActionEvent e) {
  // TODO Auto-generated method stub
  if (e.getSource() == saveButton) {
   countButton.doClick();

   strSubject = subjectField.getText().trim().replaceAll("\\s+", " ").replaceAll("\n", "").replaceAll("\t", "");

   dates = dateField.getText().trim().replaceAll("\\s+", " ").replaceAll("\n", "").replaceAll("\t", "");

   strSubject = dates + "-" + strSubject;
   if(count > wordLims[tag]) {
    Helpers.emailOverPrompt(userGUI);
   }
   else {
    Helpers.emailOKPrompt(userGUI);
    Helpers.writeAnnoncement(id, tag, strSubject, strAnnouncement);
   }
  }
 }
});

JButton sendButton = new JButton("Send for Reading");
sendButton.setBounds(600, 420, 150, 30);
sendButton.addActionListener(new ActionListener() {
      @Override
 public void actionPerformed(ActionEvent e) {
  // TODO Auto-generated method stub
  if (e.getSource() == sendButton) {
   saveButton.doClick();
   if(count <= wordLims[tag]) {
    Helpers.sendEmail(strSubject, tag, id, strAnnouncement);
   }
  }
 }
});

JButton logoutButton = new JButton("Logout");
logoutButton.setBounds(700, 530, 80, 30);
logoutButton.addActionListener(new ActionListener() {
      @Override
 public void actionPerformed(ActionEvent e) {
  // TODO Auto-generated method stub
  if (e.getSource() == logoutButton) {
   Helpers.logoutPrompt(userGUI);
   userGUI.dispose();
   AnnouncementsProgram a1 = new AnnouncementsProgram();
   a1.run();
  }
 }
});

JButton pastAnnoncementButton = new JButton("Open Previous Announcement");
pastAnnoncementButton.setBounds(550, 60, 220, 30);
pastAnnoncementButton.addActionListener(new ActionListener() {
      @Override
 public void actionPerformed(ActionEvent e) {
```

```
    // TODO Auto-generated method stub
    if (e.getSource() == pastAnnoncementButton) {
     String docPath = "data/announcements/" + id + "/";
     if(Helpers.getLastModified(docPath) != null) {
      String previousFileName = Helpers.getLastModified(docPath).getName();

      announArea.setText(Helpers.readFile(docPath+previousFileName));
      tag = Integer.valueOf(previousFileName.substring(0, 1));
      countButton.doClick();
      strSubject = previousFileName.substring(previousFileName.lastIndexOf("-") + 1);
      subjectField.setText(strSubject);
      dates = previousFileName.substring(13, previousFileName.lastIndexOf("-"));
      dateField.setText(dates);
     }
    }
   }
  });

  // Components Added using Flow Layout
  teachPanel.add(idLabel);
  teachPanel.add(wordCountLabel);

  teachPanel.add(subjectField);
  teachPanel.add(announArea);
  teachPanel.add(dateField);

  teachPanel.add(type1Button);
  teachPanel.add(type2Button);
  teachPanel.add(type3Button);
  teachPanel.add(type4Button);
  teachPanel.add(type5Button);
  teachPanel.add(type6Button);

  teachPanel.add(logoutButton);
  teachPanel.add(saveButton);
  teachPanel.add(sendButton);
  teachPanel.add(countButton);
  teachPanel.add(pastAnnoncementButton);

  userGUI.add(teachPanel);
 }
}
```

```java
/**
* The public Helpers class contains many methods that are used in the product.
*/

import java.awt.Color;
import java.awt.Font;
import java.awt.FontFormatException;
import java.io.BufferedReader;
import java.io.BufferedWriter;
import java.io.File;
import java.io.FileInputStream;
import java.io.FileNotFoundException;
import java.io.FileReader;
import java.io.FileWriter;
import java.io.IOException;
import java.io.InputStream;
import java.nio.file.Files;
import java.nio.file.Path;
import java.nio.file.Paths;
import java.time.LocalDate;
import java.time.format.DateTimeFormatter;

import javax.mail.Message;
import javax.mail.MessagingException;
import javax.mail.PasswordAuthentication;
import javax.mail.Session;
import javax.mail.Transport;
import javax.mail.internet.InternetAddress;
import javax.mail.internet.MimeMessage;

import java.util.Base64;
import java.util.Properties;

import javax.swing.JFrame;
import javax.swing.JOptionPane;

public class Helpers {

 public static Font revansFont;
 public static Font consolasFont;
 public static Font consolasFontBold;

 public static final Color BLACK = Color.BLACK;
 public static final Color RED = Color.RED;
 public static final Color GRAY = Color.GRAY;
 public static final Color LGRAY = Color.LIGHT_GRAY;

 public static String[] loginInfo;
 public static int loginInfoLength;

 public static final String EMAIL = "vpciannouncements@gmail.com";

 /**
  * The default constructor that initializes the fonts and login resources.
  */
 public Helpers() {
  initFonts();
  initLoginResources();
 }

 /**
  * Opens an option pane that shows the successful attempt of emailing the announcement/saving the file.
  * @param f for the JFrame that is currently opened
  */
 public static void emailOKPrompt(JFrame f) {
```

```
 JOptionPane.showConfirmDialog(f, "File saved/emailed.", "Successful Email/Save Attempt", JOptionPane.YES_OPTION,
JOptionPane.WARNING_MESSAGE, null);
 }

 /**
  * Opens an option pane that shows the unsuccessful attempt of emailing the announcement/saving the file due to the length
being over word limit.
  * @param f for the JFrame that is currently opened
  */
 public static void emailOverPrompt(JFrame f) {
  JOptionPane.showConfirmDialog(f, "Error: Over word limit!", "Failed Email Attempt", JOptionPane.YES_OPTION,
JOptionPane.WARNING_MESSAGE, null);
 }

 /**
  * Encodes the message using java.util.Base64's encoder.
  * @param dataWhole is the data that is waiting to be encoded
  * @return the encoded data
  */
 public static String encode(String dataWhole) {
  String[] data = dataWhole.split("\n");
  int numLines = data.length;
  String encodedData = "";
  String encodedLine;
  for(int i = 0; i < numLines; i++) {
   encodedLine = Base64.getEncoder().encodeToString(data[i].getBytes());
   encodedData += encodedLine + "\n";
  }
  return encodedData;
 }

 /**
  * Opens an option pane that asks the user to confirm his/her exit.
  * @param f for the JFrame that is currently opened
  */
 public static void exitPrompt(JFrame f) {
  int confirm = JOptionPane.showOptionDialog(f, "Are you sure?", "Confirm Exit", JOptionPane.YES_NO_OPTION,
JOptionPane.WARNING_MESSAGE, null, null, null);
  if (confirm == JOptionPane.YES_OPTION) { //another way to say 0 for yes
   System.exit(0);
  }
 }

 /**
  * Encodes the message using java.util.Base64's encoder.
  * @param dataWhole is the data that is waiting to be decoded
  * @return the decoded data
  */
 public static String decode(String dataWhole) {
  String[] data = dataWhole.split("\n");
  int numLines = data.length;
  String decodedData = "";
  String decodedLine = "";
  for(int i = 0; i < numLines; i++) {
   decodedLine = new String(Base64.getDecoder().decode(data[i]));
   decodedData += decodedLine + "\n";
  }
  return decodedData;
 }

 /**
  * Generates passwords randomly for the users and stores it in a file.
  * @param f for the JFrame that is currently opened
  * @param size is the desired new user group size
  */
 public static void generatePasswords(JFrame f, int size) {
```

```
 String allData = "";
 for (int i = 1; i <= size; i++) {
  String pass = "";
  double randomNum;
  for (int j = 1; j<=8; j++) {
   randomNum = Math.random();
   if (randomNum > 0.7) {
    pass += (char) (65 + (int)(Math.random()*26)); //add random upper case letter to password
   }
   else if (randomNum < 0.3){
    pass += (char) (97 + (int)(Math.random()*26)); //add random lower case letter to password
   }
   else {
    pass += (int)(Math.random()*10); //add random digit to password; implicit casting from int to string
   }
  }
  //teach_ _ _ will be the user names
  //pass will be the passwords of the users
  allData += String.format("teach%03d %s%n", i, pass);
 }
 String docName = "data/login/loginInfo" + getYear();
 writeFile(docName, allData); //creates a new file for the newly generated passwords
 JOptionPane.showConfirmDialog(f, "Passwords Generated!", "Successful Password Generation", JOptionPane.YES_OPTION,
JOptionPane.WARNING_MESSAGE, null);
 }


 /**
  * Returns the file that is last modified.
  * @param docPath is the path that is being searched
  * @return the file that is last modified
  */
 public static File getLastModified(String docPath){
    File directory = new File(docPath);
    File[] files = directory.listFiles(File::isFile);
    long lastModifiedTime = Long.MIN_VALUE;
    File chosenFile = null;
    File file = null;
    if (files != null) {
       for (int i = 0; i<files.length; i++) {
        file = files[i];
           if (file.lastModified() > lastModifiedTime) {
              chosenFile = file;
              lastModifiedTime = file.lastModified();
           }
       }
    }

    return chosenFile;
 }

 /**
  * Gets the current date.
  * @return the current date
  */
 public static String getDate() {
  LocalDate date = LocalDate.now();
  String strDate = "";
  DateTimeFormatter dtf = DateTimeFormatter.ISO_LOCAL_DATE;
  strDate = dtf.format(date);
  return strDate;
 }

 /**
  * Gets the current year.
  * @return the current year
```

```
 */
protected static String getYear() {
 String strYear = "";
 DateTimeFormatter dtf = DateTimeFormatter.ofPattern("yyyy");
 LocalDate date = LocalDate.now();
 strYear = dtf.format(date);
 return strYear;
}

/**
 * Initializes the fonts used in the program.
 */
public void initFonts() {
 consolasFont = new Font("Consolas", Font.PLAIN, 18);
 consolasFontBold = new Font("Consolas", Font.BOLD, 20);

 File titleFontFile = new File("data/font/Revans-Medium.ttf");
 //add other font?

 try {
  InputStream fontStream = new FileInputStream(titleFontFile);
  revansFont = Font.createFont(Font.TRUETYPE_FONT, fontStream);
 } catch (FontFormatException | IOException e) {
  // TODO Auto-generated catch block
  e.printStackTrace();
 }

}

/**
 * Initializes the login resources used in the program.
 */
public static void initLoginResources() {
 String docName = "data/login/loginInfo" + Helpers.getYear();
 loginInfo = Helpers.readFile(docName).split("\n");
 loginInfoLength = loginInfo.length;
}

/**
 * Opens an option pane that asks the user to confirm his/her logout.
 * @param f for the JFrame that is currently opened
 */
public static void logoutPrompt(JFrame f) {
 int confirm = JOptionPane.showOptionDialog(f, "Are you sure?", "Confirm Logout", JOptionPane.YES_NO_OPTION,
JOptionPane.WARNING_MESSAGE, null, null, null);
 if (confirm == JOptionPane.YES_OPTION) {
  f.dispose();
 }
}


/**
 * Reads the contents of the file with the specific docName.
 * @param docName is the name of the file that is to be read from
 * @return the contents of the file in a string
 */
public static String readFile(String docName) {
 File r = new File(docName);
 FileReader fr = null;
 String allData = "";
 String decodedData = null;
 try {
  fr = new FileReader(r);
  BufferedReader br = new BufferedReader(fr);
  String nextLine;
  while ((nextLine = br.readLine()) != null) {
```

```
   allData += nextLine;
   allData += "\n";
  }
  decodedData = decode(allData);
  br.close();
 }
 catch (FileNotFoundException e) {
  //System.out.println("File not found");
  e.printStackTrace();
 }
 catch (IOException e) {
  //System.out.println("IO exception; reading error");
  e.printStackTrace();
 }
 return decodedData;
}

/**
 * Sends the announcement as an email to the announcements team's email account.
 * @param subject is the subject of the announcement
 * @param tag is the tag of the announcement
 * @param id is the account id of the user
 * @param content is the content of the announcement
 */
public static void sendEmail(String subject, int tag, String id, String content) {
 subject = tag + "-" + subject + "-" + id;
      // Sender's email ID needs to be mentioned
      String username = Helpers.EMAIL;
      final String password = "AnnouncementsVP";

      // Assuming you are sending email through relay.jangosmtp.net
      String host = "smtp.gmail.com";
      String port = "587";

      Properties props = new Properties();
      //props.put("mail.debug", "true");
      props.put("mail.smtp.auth", "true");
      props.put("mail.smtp.starttls.enable", "true");
      props.put("mail.smtp.host", host);
      props.put("mail.smtp.port", port);

      // Get the Session object.
      Session session = Session.getInstance(props,
       new javax.mail.Authenticator() {
          public PasswordAuthentication getPasswordAuthentication() {
           return new PasswordAuthentication(username, password);
          }
       }
      );

      try {
      // Create a default MimeMessage object.
      Message message = new MimeMessage(session);

      // Set From: header field of the header.
      message.setFrom(new InternetAddress(Helpers.EMAIL));

      // Set To: header field of the header.
      message.setRecipients(Message.RecipientType.TO, InternetAddress.parse(Helpers.EMAIL));

      // Set Subject: header field
       message.setSubject(subject);

      // Now set the actual message
      message.setText(content);
```

```
        // Send message
        Transport.send(message);

        } catch (MessagingException e) {
         throw new RuntimeException(e);
        }
}


/**
 * Saves the current announcement into a file.
 * @param username is the account id of the user
 * @param intTag is the tag of the announcement
 * @param subject is the subject of the announcement
 * @param allData is the content of the announcement
 */
public static void writeAnnoncement(String username, int intTag, String subject, String allData) {
 Path path = Paths.get("data/announcements/" + username + "/");
 try {
  Files.createDirectories(path);
 } catch (IOException e1) {
  // TODO Auto-generated catch block
  e1.printStackTrace();
 }
 String docName = "" + path + "/" + intTag + "-" + getDate() + "-" + subject;
 File w = new File(docName);
 FileWriter fw = null;
 try {
  fw = new FileWriter(w);
  BufferedWriter bw = new BufferedWriter(fw);
  String encodedData = encode(allData);
  bw.write(encodedData);
  bw.close();
 }
 catch (IOException e) {
  //System.out.println("IO exception; writing error");
  e.printStackTrace();
 }
}


/**
 * Writes a file with specified docName and specified data in allData.
 * @param docName is the name of the document
 * @param allData is the contents to be written into the file
 */
public static void writeFile(String docName, String allData) {
 int slashIndex = docName.lastIndexOf('/');
 Path path = Paths.get(docName.substring(0, slashIndex+1));
 try {
  Files.createDirectories(path);
 } catch (IOException e1) {
  // TODO Auto-generated catch block
  e1.printStackTrace();
 }
 File w = new File(docName);
 FileWriter fw = null;
 try {
  fw = new FileWriter(w,false);
  BufferedWriter bw = new BufferedWriter(fw);
  String encodedData = encode(allData);
  bw.write(encodedData);
  bw.close();
 }
 catch (IOException e) {
  //System.out.println("IO exception; writing error1");
```

```
    e.printStackTrace();
   }
  }

}
```