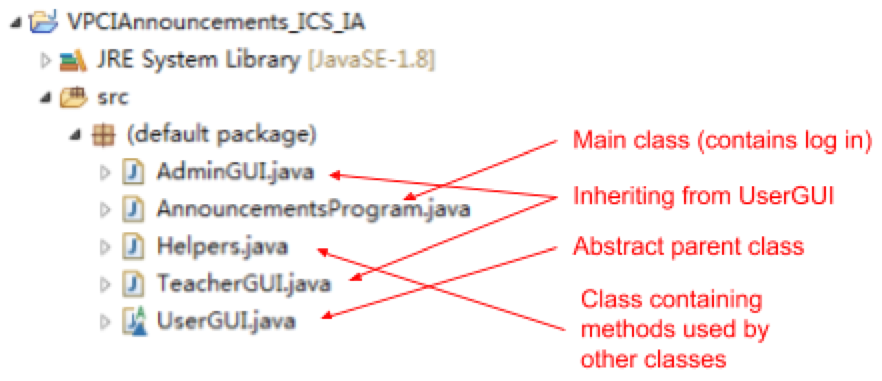
Criterion C: Development (With code citations at end)

Techniques Used

- Object Oriented Programming Techniques
  - Encapsulation
  - Inheritance
  - Polymorphism
  - Abstraction
- Graphical User Interface
- Data Structure
  - Arrays
- Error handling
- Read from and write to files
  - Encoding/Decoding
- Sending emails using SMTP
- External Libraries

Program Structure

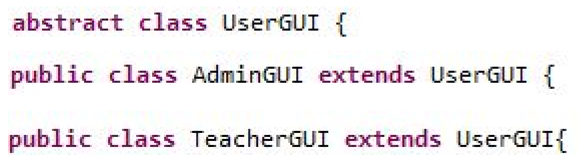


As mentioned in previous sections, I designed the project such that the functions and the graphical user interface are combined for most of the classes (while keeping in mind the disposal of objects, multi-threaded nature of Java, and the CPU processor speed of the user's computer), which explains why the final program contains only 5 classes.

OO Techniques and Graphical User Interface (GUI) Overview

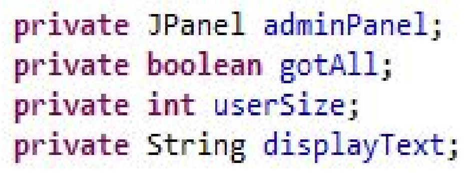
For this program, I used JFrames and JPanels, as well as other JComponents that are all provided in java swing and java awt packages, to build the graphical user interface. I chose to use flow layout manager not only because it is the default layout manager for JPanels, but also because it does not allow overlapping of components, which is desired for my program. Since I designed the JFrames of TeacherGUI and AdminGUI to be similar, I created an abstract class, UserGUI, to contain common methods and variables (including the JFrame variable userGUI).

Then, I made UserGUI the parent of the 2 specific classes such that they can interhit and use these shared attributes.

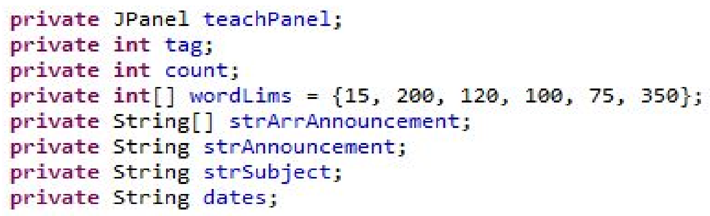


```
abstract class UserGUI {

public class AdminGUI extends UserGUI {

public class TeacherGUI extends UserGUI{
```

Despite both inheriting from UserGUI, TeacherGUI and AdminGUI both use the OO technique of encapsulation by having a private JPanel and private components, which prevents other programs from directly accessing and manipulating their GUI.

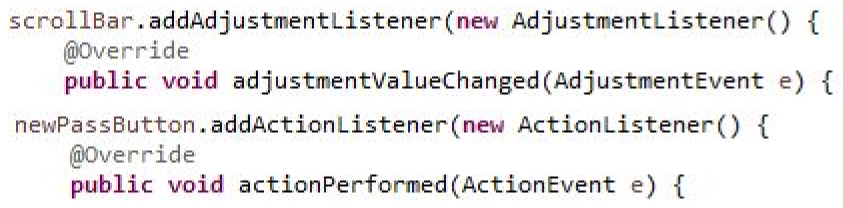


```
private JPanel adminPanel;
private boolean gotAll;
private int userSize;
private String displayText;
```



```
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 program uses the OO feature of polymorphism as well; since actionPerformed() in ActionListener and adjustmentValueChanged() in AdjustmentListener (both from java awt package) are abstract methods, they are overridden (a subclass providing a specific implementation of a method that is already provided by a superclass; shown in examples below) for the different JComponents used in the program to respond properly if the user inputs anything.



```
scrollBar.addAdjustmentListener(new AdjustmentListener() {
    @Override
    public void adjustmentValueChanged(AdjustmentEvent e) {
newPassButton.addActionListener(new ActionListener() {
    @Override
    public void actionPerformed(ActionEvent e) {
```

At the same time, constructor methods are overloaded (different functions with the same name in same scope; shown in example below) to initialize the AdminGUI regardless of the existence of parameter strId.

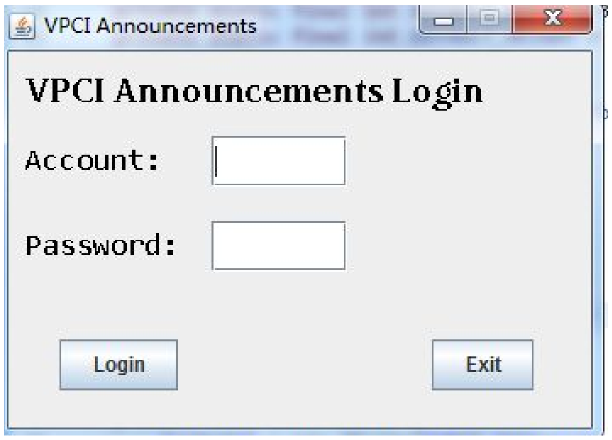
```
public AdminGUI() {
    super("Administrator");
    this.id = "Administrator";
    this.userSize = Helpers.loginInfoLength;
    this.displayText = "";
    this.gotAll = false;
    this.initPanel();
}

public AdminGUI(String strId) {
    super(strId);
    this.id = strId;
    this.userSize = Helpers.loginInfoLength;
    this.displayText = "";
    this.gotAll = false;
    this.initPanel();
}
```

Login & User Verification

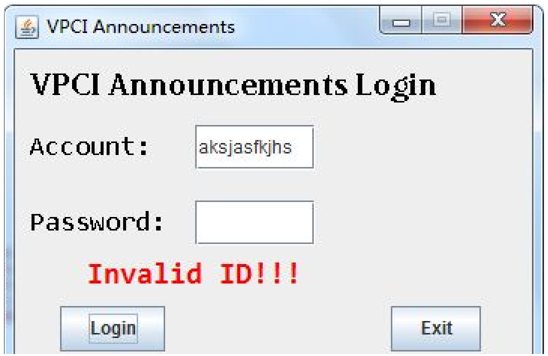
When the program is initiated, the main method in AnnouncementsProgram creates a new AnnouncementsProgram, which uses the constructor to call initializing functions of both the new Helpers instance (which initializes login resources) and this AnnouncementsProgram itself (which initializes the JFrame and JPanel as well as the fonts used). The run function of this instance is then called to display the login JFrame loginGUI.

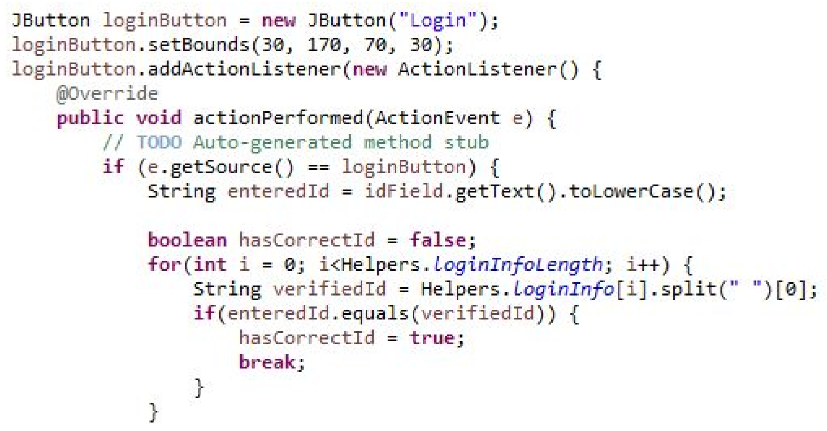
```
public static void main(String[] args) {
    AnnouncementsProgram a = new AnnouncementsProgram();
    a.run();
}
```

```
public AnnouncementsProgram() {
    Helpers helper = new Helpers();
    helper.initFonts();
    initFrame();
    initPanel();
}

public Helpers() {
    initLoginResources();
}

public void run() {
    loginGUI.setVisible(true);
}
```



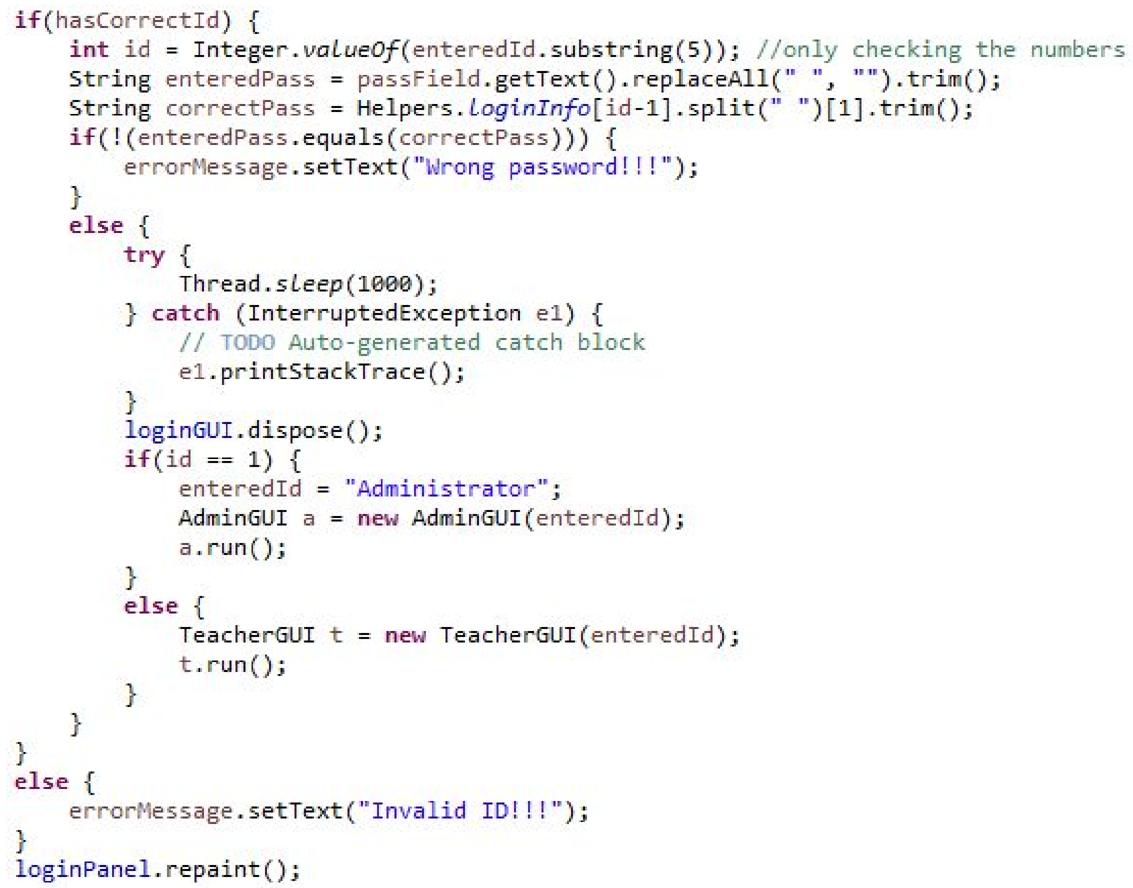
When the user enters a wrong account id enteredId and clicks the login button, an error message is shown. This is because the enteredId is not found when the loginInfo list is fully traversed (sequential search).



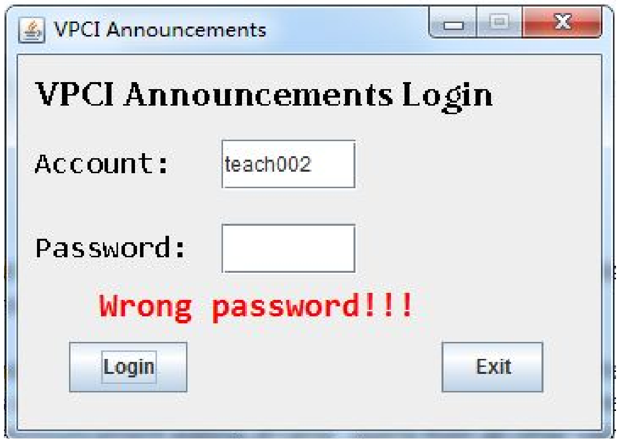


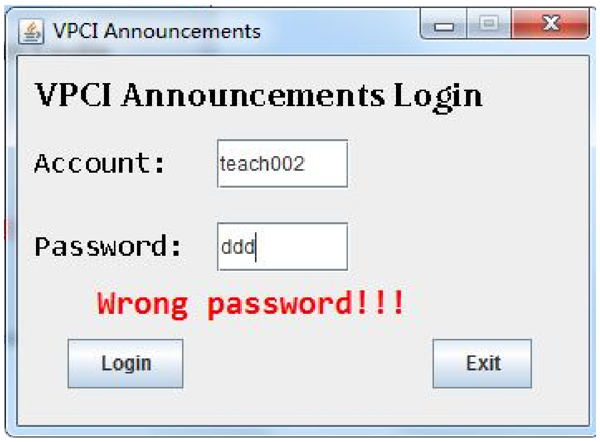
```java
JButton loginButton = new JButton("Login");
loginButton.setBounds(30, 170, 70, 30);
loginButton.addActionListener(new ActionListener() {
    @Override
    public void actionPerformed(ActionEvent e) {
        // TODO Auto-generated method stub
        if (e.getSource() == loginButton) {
            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 the user enters a valid id but has the wrong password, another error message is displayed because the password does not match the enteredId. If the user enters a valid id and password combination, the program then disposes the login JFrame loginGUI and creates an instance of AdminGUI (administrator control JFrame; has id “teach001”) or of TeachGUI (announcement editing JFrame; opens from all other valid ids).



```java
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();
```

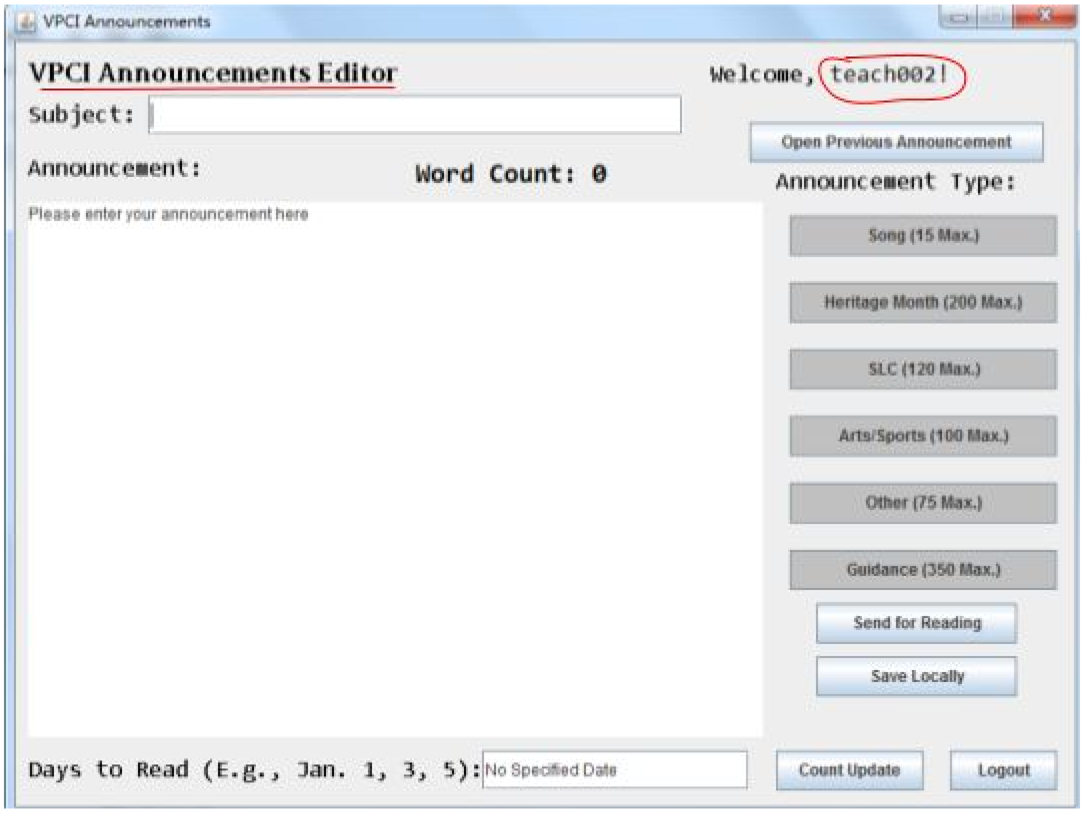




Teacher Editing JFrame

An instance of the class TeachGUI that inheritsfrom UserGUI the JFrame userGUI is created when the account entered is not the administrator("teach001") and the id is valid. The default constructor is empty because the instance requires the strId of the user. After initializing some of its private variables and JPanel (with all required JComponents added), the instance's JFrame is displayed with the appropriate id and title.

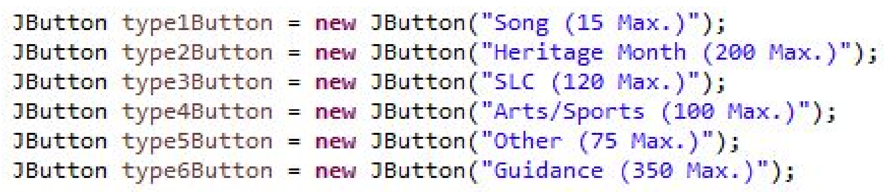
```
private JPanel teachPanel;
private int tag;
private int count;
private int[] wordLims = {15, 200, 120, 100, 75, 350};
private String[] strArrAnnouncement;
private String strAnnouncement;
private String strSubject;
private String dates;

/**
* 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();
}
```



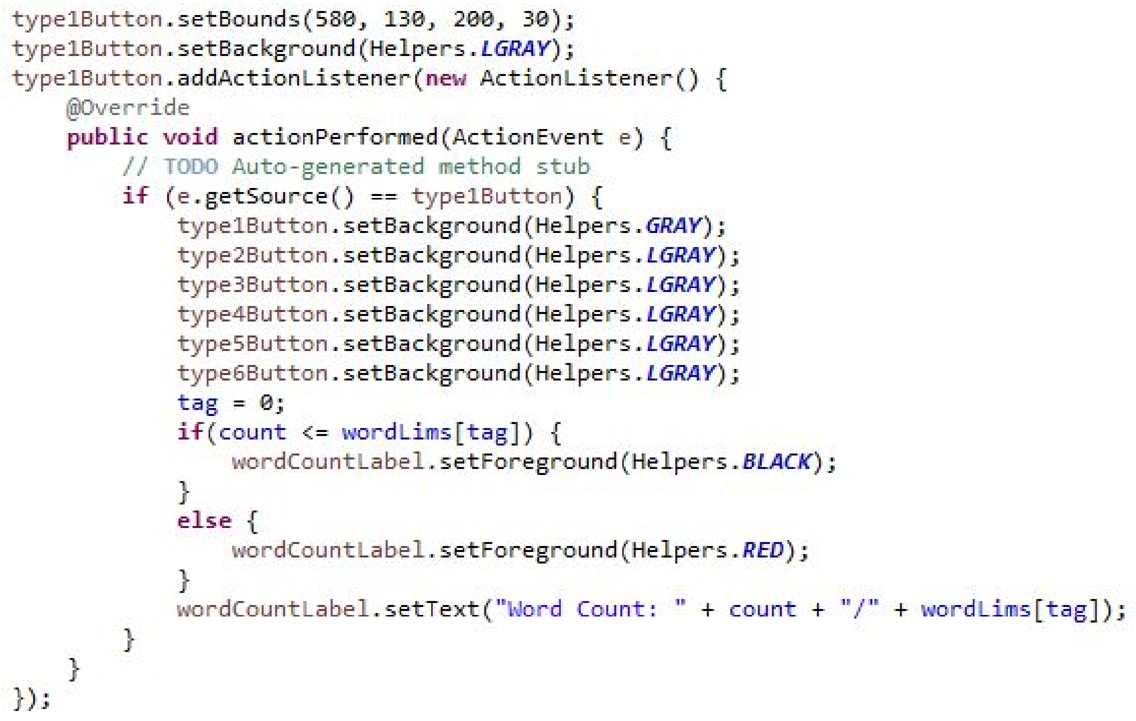
Count Update Function of TeachGUI and Word Count

When the countButton is pressed, the program updates the word count of the announcement (from JTextarea annonArea's contents; formatting not affecting count) and checks the announcement type for the word limit (defaultly set to a "song-type" announcement).

```
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]);
        }
    }
});
```

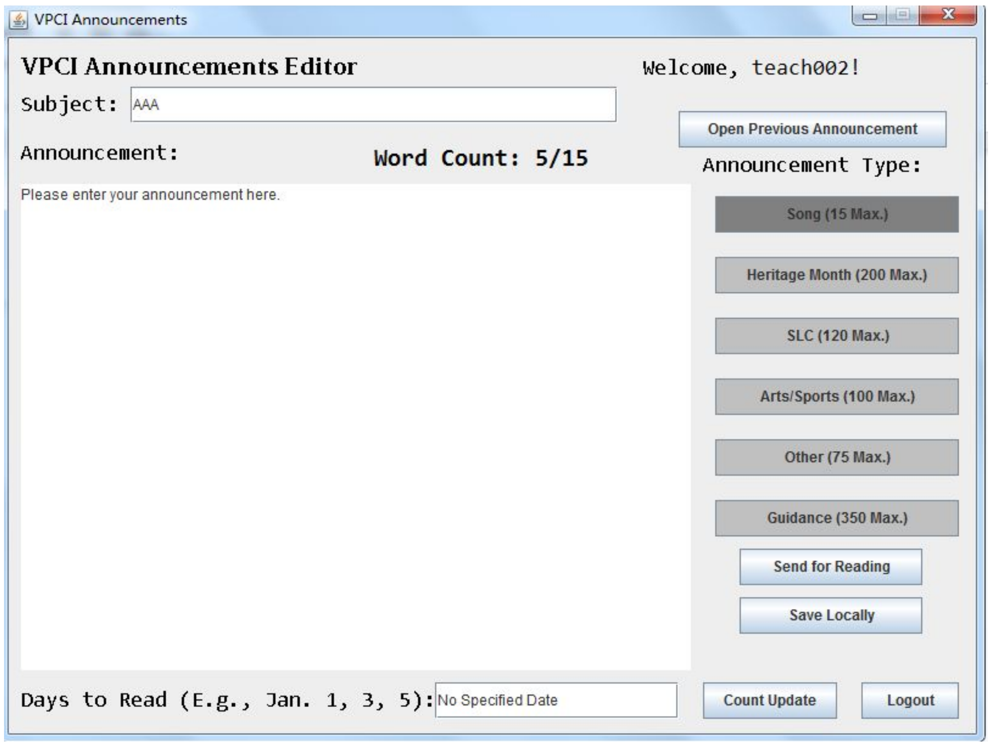


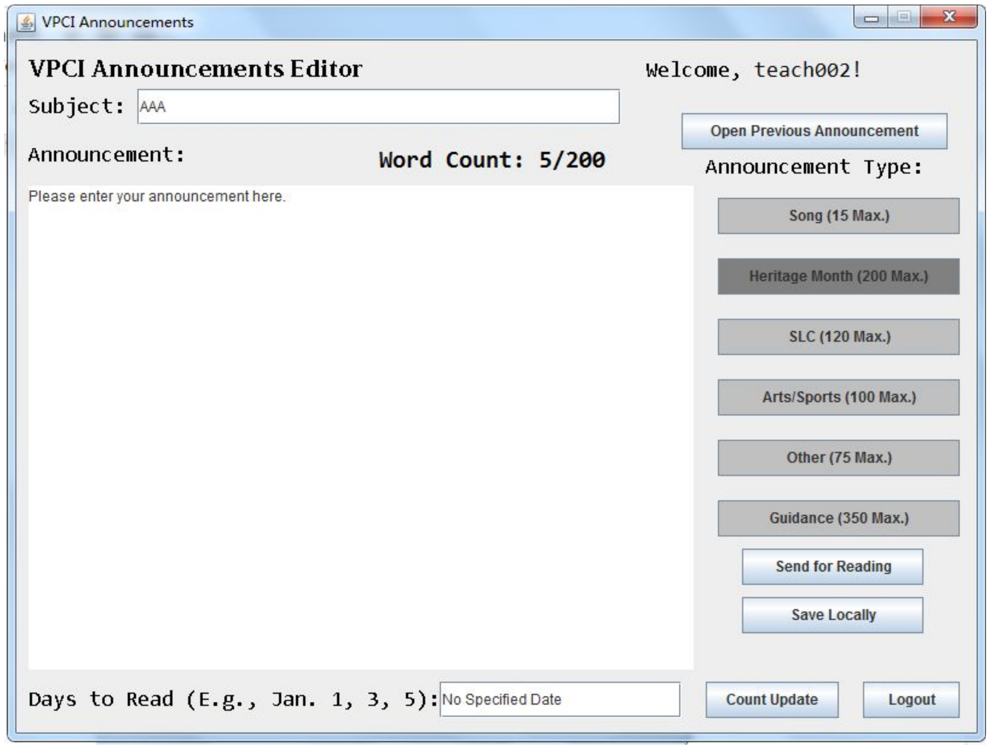
```java
JButton type1Button = new JButton("Song (15 Max.)");
JButton type2Button = new JButton("Heritage Month (200 Max.)");
JButton type3Button = new JButton("SLC (120 Max.)");
JButton type4Button = new JButton("Arts/Sports (100 Max.)");
JButton type5Button = new JButton("Other (75 Max.)");
JButton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



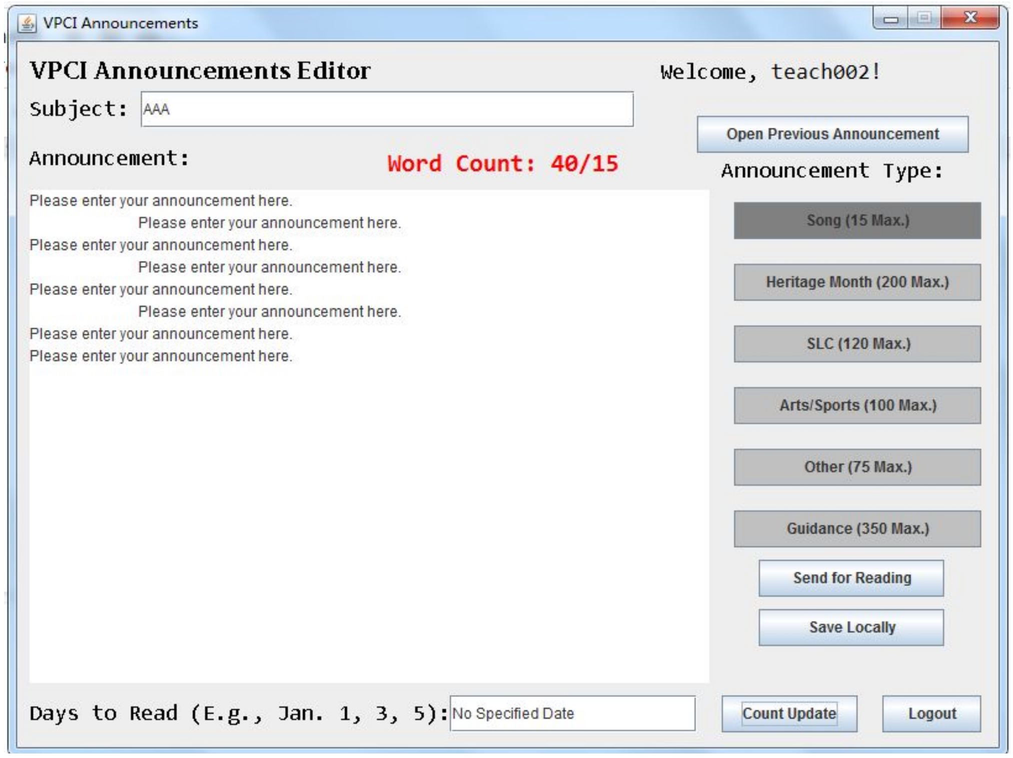
```java
type1Button.setBounds(580, 130, 200, 30);
type1Button.setBackground(Helpers.LGRAY);
type1Button.addActionListener(new ActionListener() {
    @Override
    public void actionPerformed(ActionEvent e) {
        // TODO Auto-generated method stub
        if (e.getSource() == type1Button) {
            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]);
        }
    }
});
```

The type can be changed by clicking on the buttons, and the button for current type has a dark background (type1Button code below).





If the word count is over word limit, the message becomes red.



Subject: AAA

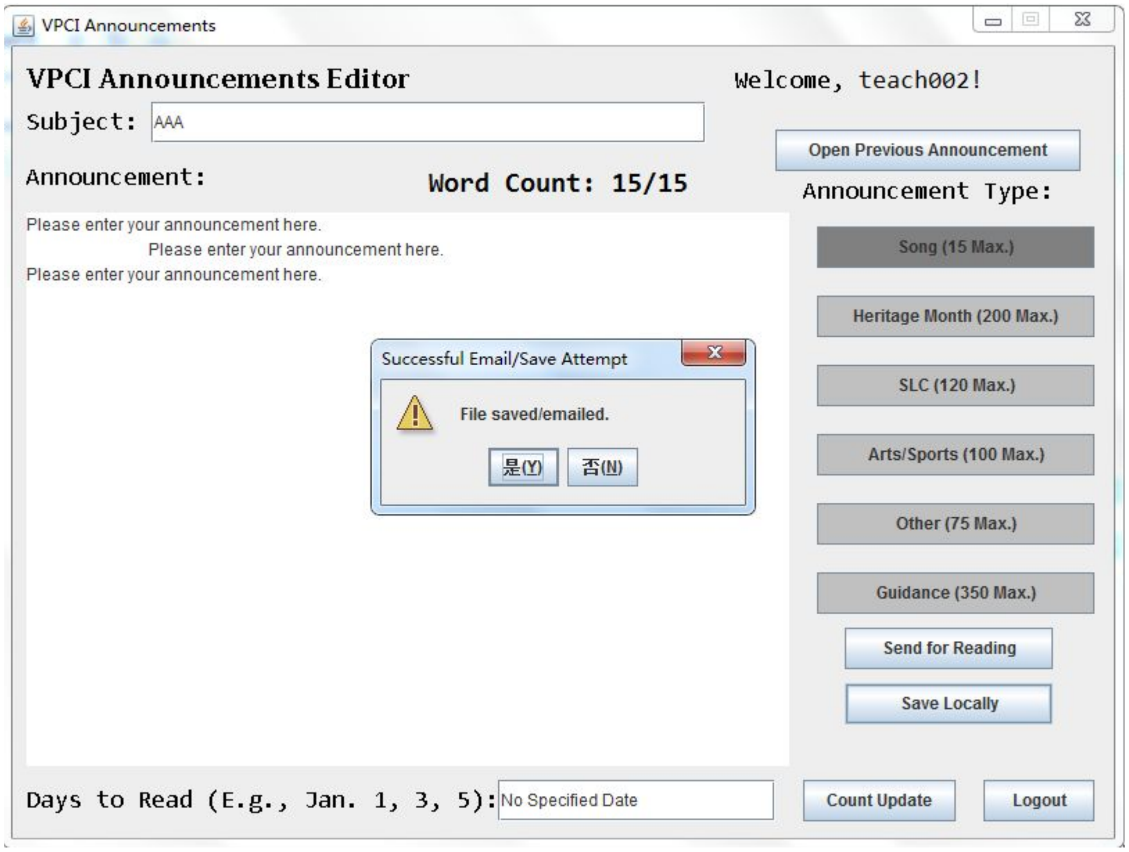
```
Please enter your announcement here.
                    Please enter your announcement here.
Please enter your announcement here.
                    Please enter your announcement here.
Please enter your announcement here.
                    Please enter your announcement here.
Please enter your announcement here.
Please enter your announcement here.
```

Days to Read (E.g., Jan. 1, 3, 5): No Specified Date

<u>Saving Function of TeachGUI</u>

To save an announcement when the saveButton is pressed, TeachGUI uses the static writeAnnouncement() method from Helpers. After updating strSubject (subject of announcement) with desired dates and the word count, the program checks whether the announcement is within its type's word limit. The program then uses JOptionPanes to inform user about the successful/failed attempt to save announcement.

```java
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);
            }
        }
    }
});
```


VPCI Announcements

VPCI Announcements Editor

Welcome, teach002!

Subject: AAA

Open Previous Announcement

Announcement:

Word Count: 15/15

Announcement Type:

Please enter your announcement here.
Please enter your announcement here.
Please enter your announcement here.

Successful Email/Save Attempt

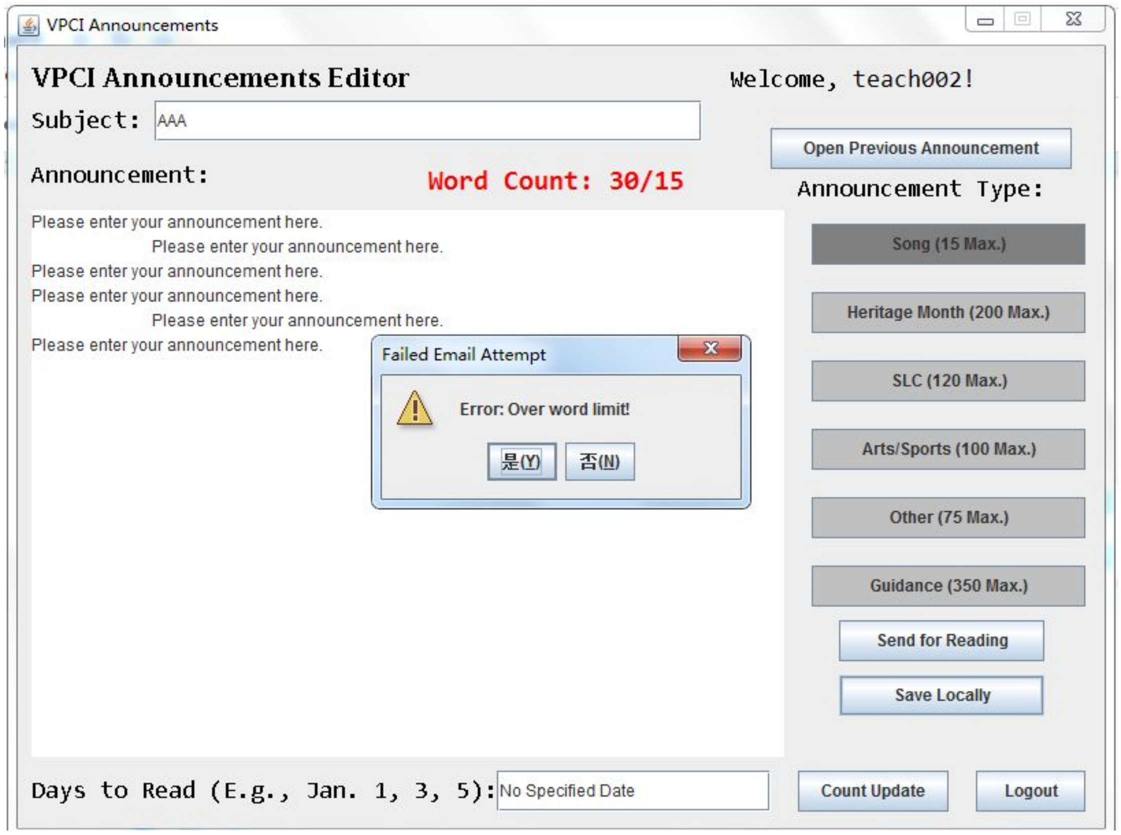
File saved/emailed.

是(Y) 否(N)

Song (15 Max.)
Heritage Month (200 Max.)
SLC (120 Max.)
Arts/Sports (100 Max.)
Other (75 Max.)
Guidance (350 Max.)
Send for Reading
Save Locally

Days to Read (E.g., Jan. 1, 3, 5): No Specified Date

Count Update
Logout


VPCI Announcements

VPCI Announcements Editor

Welcome, teach002!

Subject: AAA

Open Previous Announcement

Announcement:

Word Count: 30/15

Announcement Type:

Please enter your announcement here.
Please enter your announcement here.
Please enter your announcement here.
Please enter your announcement here.
Please enter your announcement here.
Please enter your announcement here.

Failed Email Attempt

Error: Over word limit!

是(Y) 否(N)

Song (15 Max.)
Heritage Month (200 Max.)
SLC (120 Max.)
Arts/Sports (100 Max.)
Other (75 Max.)
Guidance (350 Max.)
Send for Reading
Save Locally

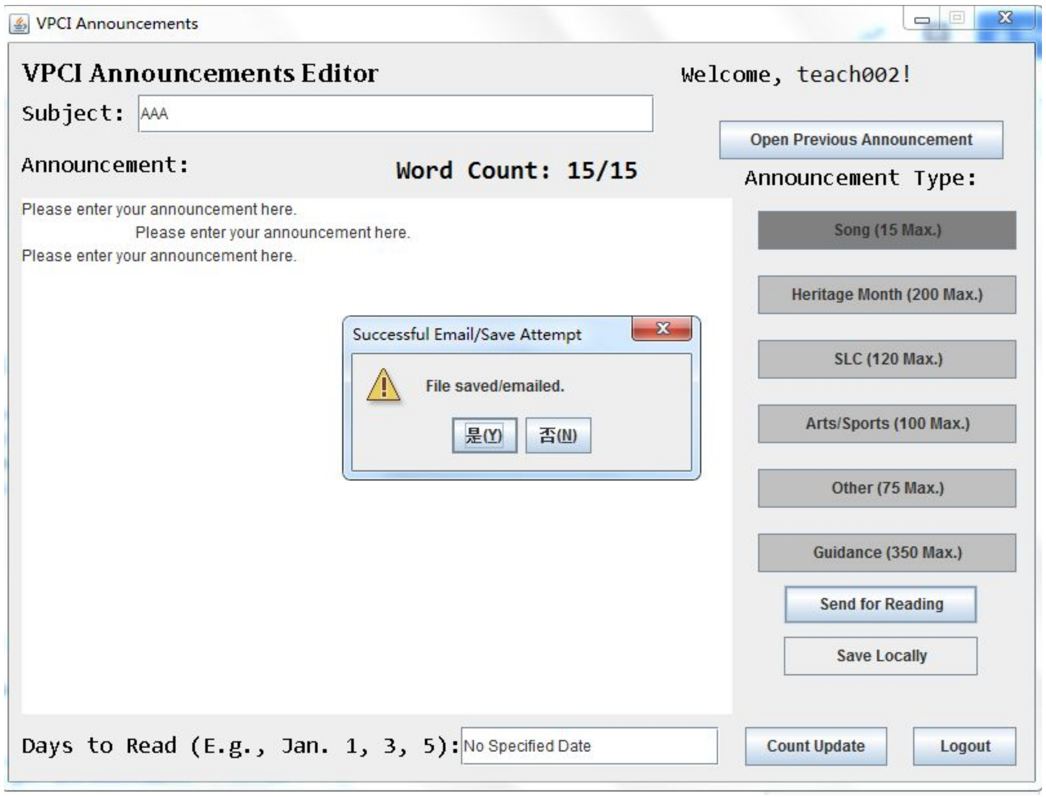
Days to Read (E.g., Jan. 1, 3, 5): No Specified Date

Count Update
Logout

Sending Function of TeachGUI

To send an announcement when the sendButton is pressed, TeachGUI uses the static sendEmail() method from Helpers. The saveButton is clicked (to save the announcement before sending), and the program sends an email if the word count is ok. The program uses the same JOptionPanes to inform user about their attempt.

```
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);
            }
        }
    }
});
```


VPCI Announcements

VPCI Announcements Editor

Welcome, teach002!

Subject: AAA

Announcement:

Word Count: 15/15

Please enter your announcement here.
Please enter your announcement here.
Please enter your announcement here.

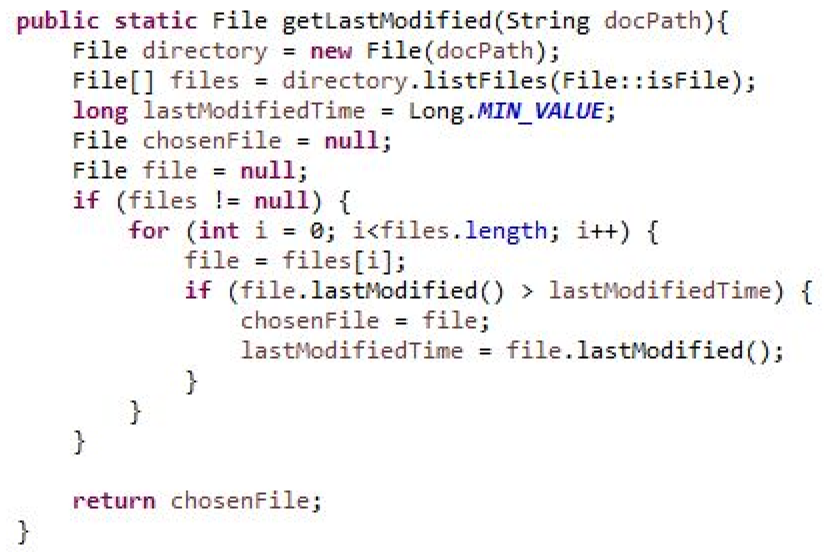
Successful Email/Save Attempt

File saved/emailed.

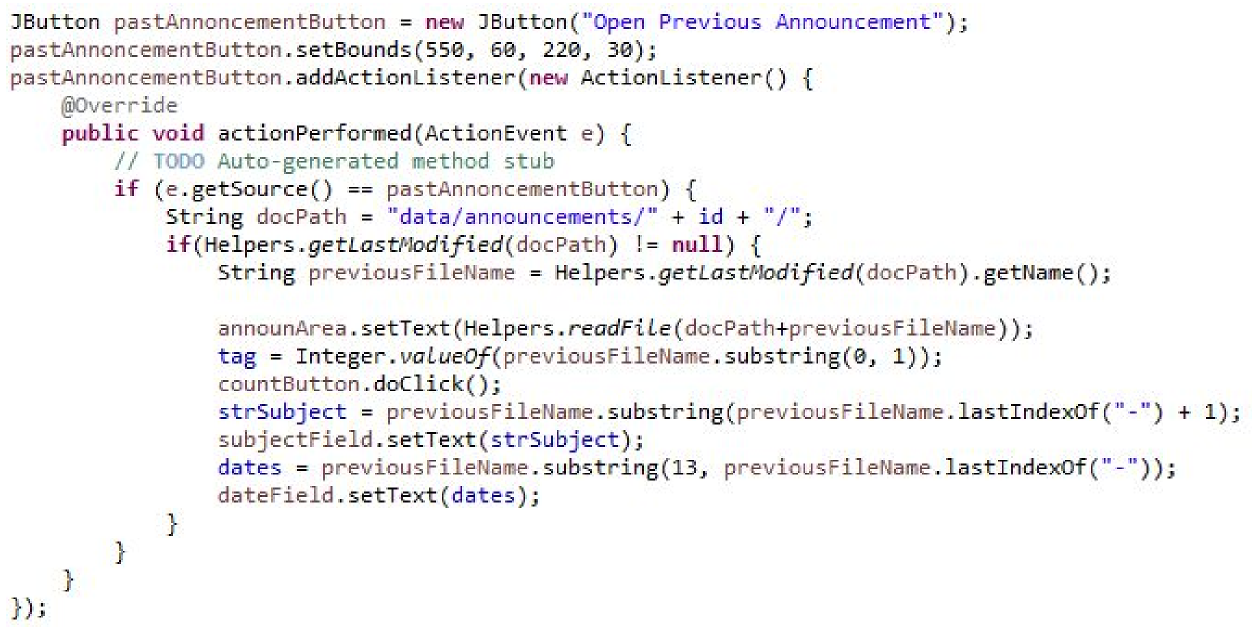
Days to Read (E.g., Jan. 1, 3, 5): No Specified Date

Previous Announcement Function (using Java io and nio packages)

To open the previous announcement, TeachGUI uses the getLastModified() method in Helpers to get the previously saved announcement and outputs its contents in the JTextfields and JTextareas.



```java
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;
}
```



```java
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);
            }
        }
    }
});
```

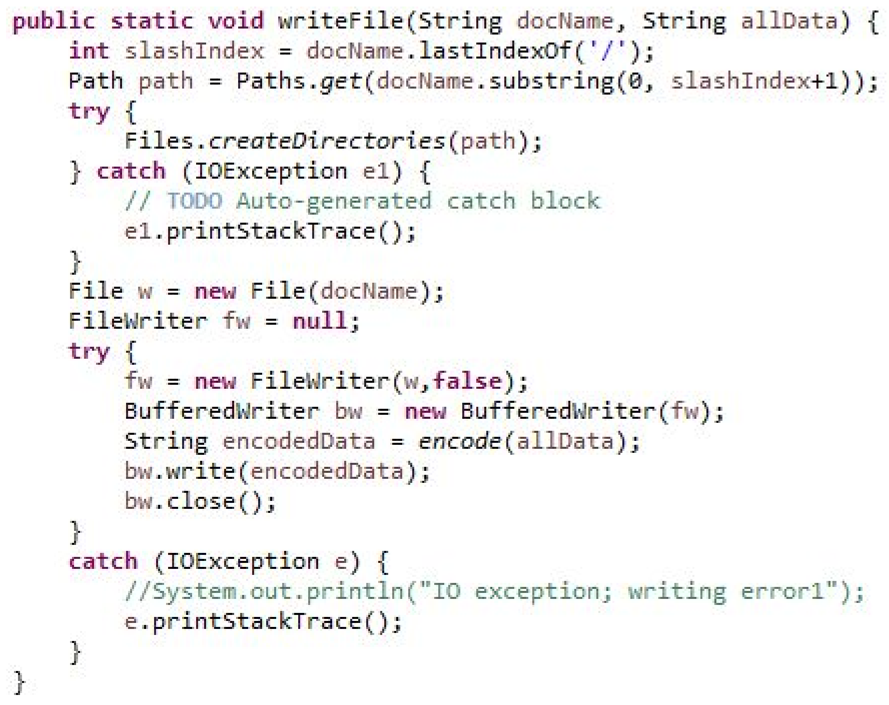
Saving and Reading Files Locally (using Java io and nio packages)

Helpers class contains static methods writeFile(), writeAnnouncement() and readFile().

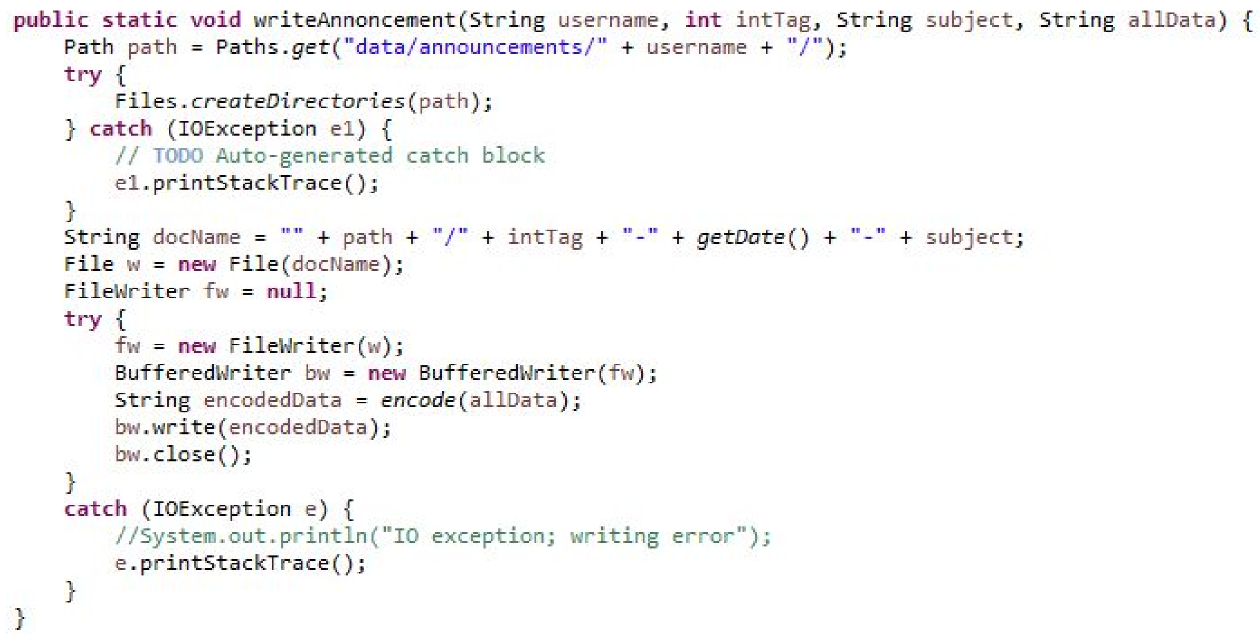
The two writing methods are similar:

1. Program checks if the path of the document (to be written) exists; if not, a folder is created.

2. A file is created with the docName (generated from different variables in writeAnnouncement).
3. An instance of BufferedWriter is created for the file.
4. The contents are encoded/encrypted using encode() method in the same class.
5. The file is written in the file and BufferedWriter is closed.

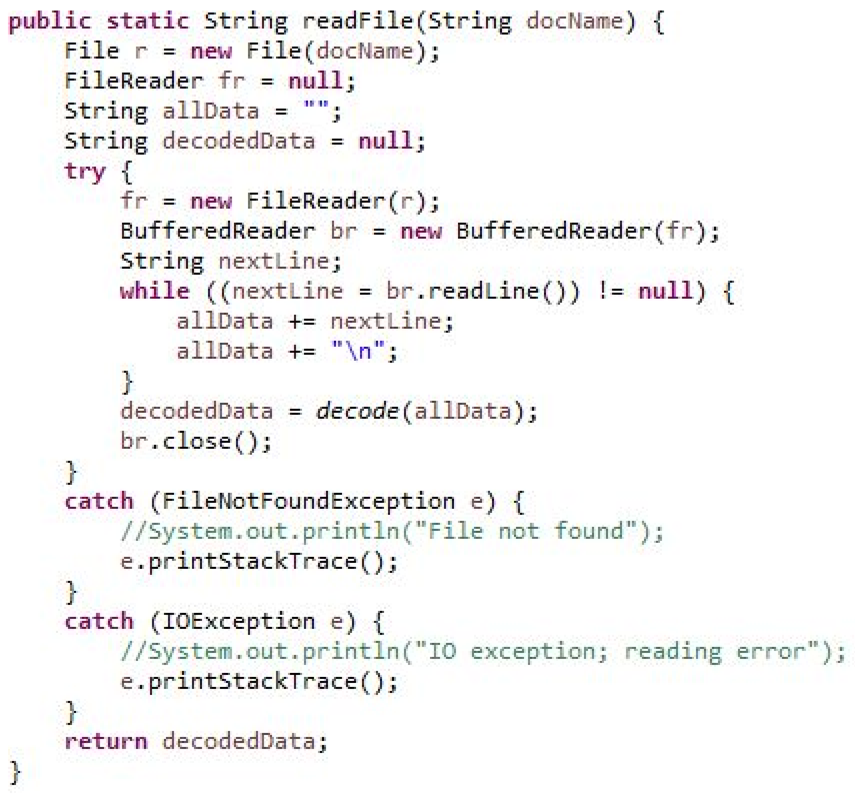


```java
public static void writeFile(String docName, String allData) {
    int slashIndex = docName.lastIndexOf('/');
    Path path = Paths.get(docName.substring(0, slashIndex+1));
    try {
        Files.createDirectories(path);
    } catch (IOException e1) {
        // TODO Auto-generated catch block
        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();
    }
}
```



```java
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();
    }
}
```

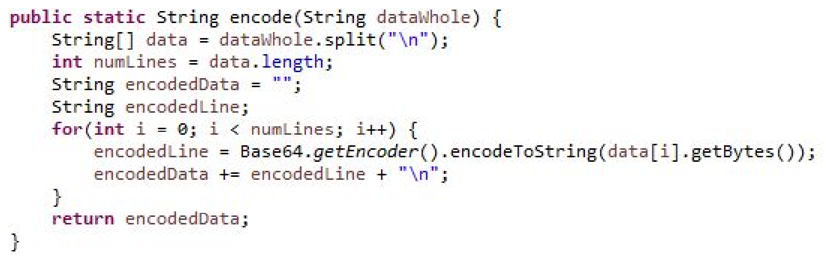
The readFile() method used in reading local announcement/login resource files is almost the “opposite” of the writing methods. A BufferedReader is used to read from the File created using the docName of the file to be read; all data in the file are then read and decoded using decode() method for file contents to be used.



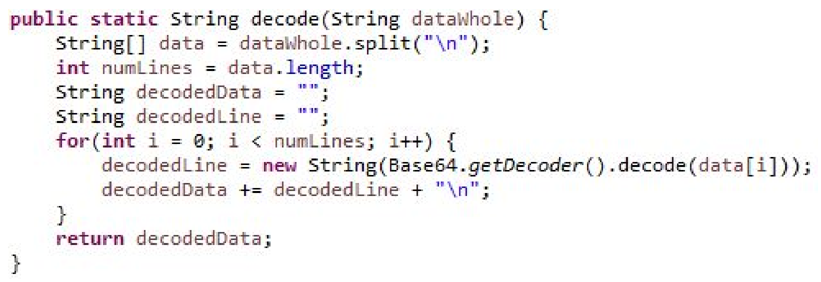
```java
public static String readFile(String docName) {
    File r = new File(docName);
    FileReader fr = null;
    String allData = "";
    String decodedData = null;
    try {
        fr = new FileReader(r);
        BufferedReader br = new BufferedReader(fr);
        String nextLine;
        while ((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;
}
```

<u>Encoding/Decoding Data (using java util Base64 package)</u>

To increase the security and privacy of the system and its files, all local data are encoded/decoded using the java util Base64 package. This ensures that the data is not easily understood by users if opened directly.

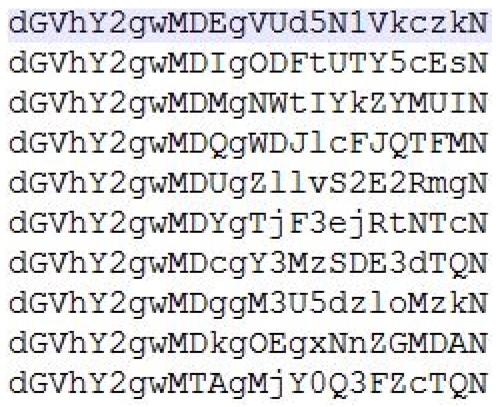


```java
public static String encode(String dataWhole) {
    String[] data = dataWhole.split("\n");
    int numLines = data.length;
    String encodedData = "";
    String encodedLine;
    for(int i = 0; i < numLines; i++) {
        encodedLine = Base64.getEncoder().encodeToString(data[i].getBytes());
        encodedData += encodedLine + "\n";
    }
    return encodedData;
}
```



```java
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;
}
```

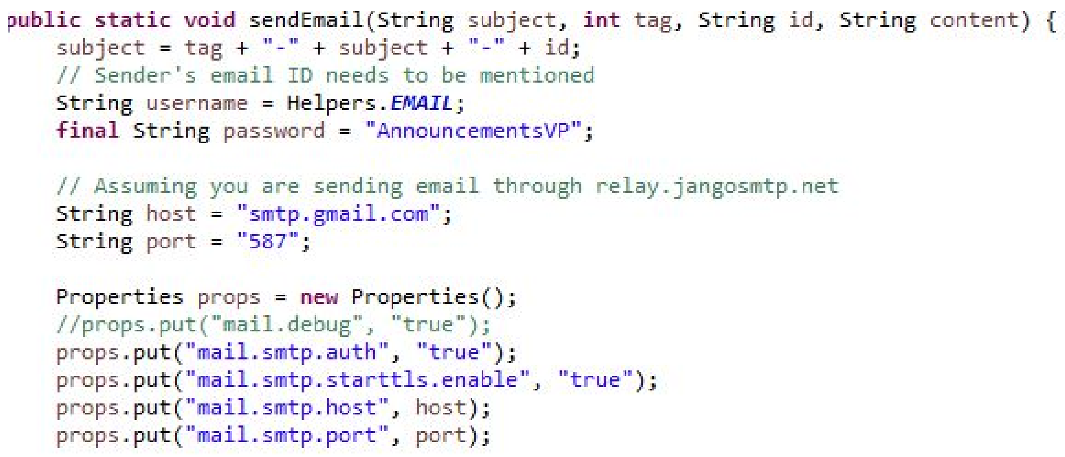
Below is an extract of the login resources file after encryption opened with notepad.



```
dGVhY2gwMDEgVUd5N1VkczkN
dGVhY2gwMDIgODFtUTY5cEsN
dGVhY2gwMDMgNWtIYkZYMUIN
dGVhY2gwMDQgWDJlcFJQTFMN
dGVhY2gwMDUgZllvS2E2RmgN
dGVhY2gwMDYgTjF3ejRtNTcN
dGVhY2gwMDcgY3MzSDE3dTQN
dGVhY2gwMDggM3U5dzloMzkN
dGVhY2gwMDkgOEgxNnZGMDAN
dGVhY2gwMTAgMjY0Q3FZcTQN
```

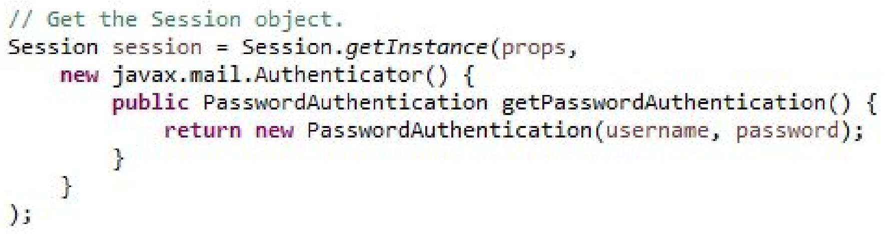
<u>Sending Email (using external JavaMail/javax mail and activation packages)</u>

To send the announcements via email, external packages of javax mail and activation packages (all by Oracle; the .jar files are downloaded for usage) are used to send and receive email via SMTP (Simple Mail Transfer Protocol). In the method sendEmail() in Helpers class, the specific host and port for gmail is initialized.



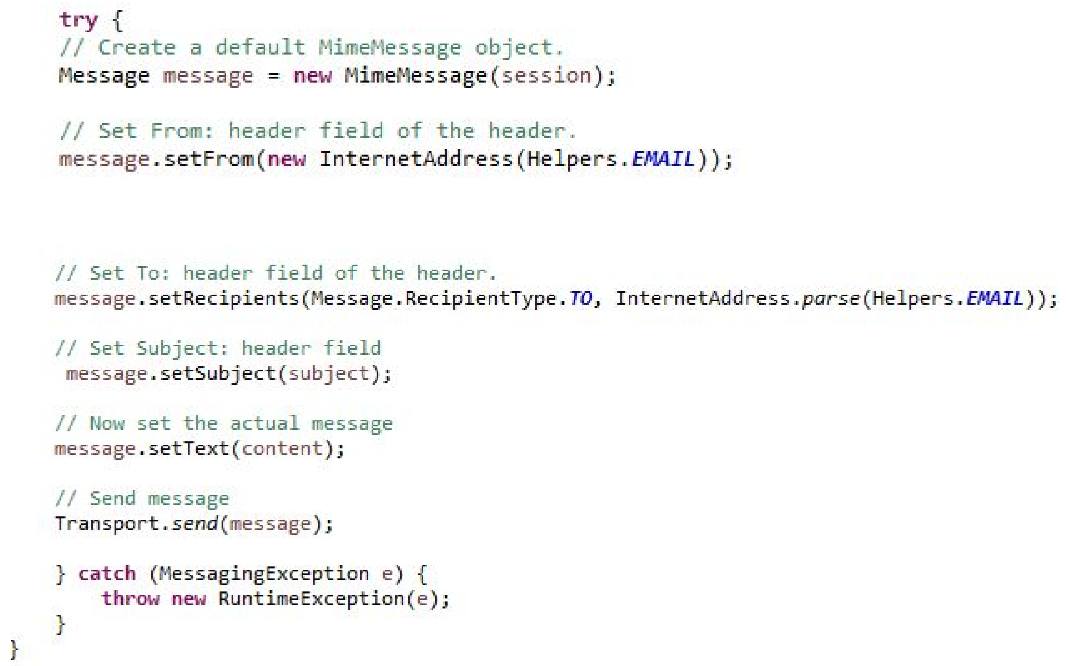
```java
public static void sendEmail(String subject, int tag, String id, String content) {
    subject = tag + "-" + subject + "-" + id;
    // Sender's email ID needs to be mentioned
    String username = Helpers.EMAIL;
    final String password = "AnnouncementsVP";

    // 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);
```

Then, the authentication of the user (since Gmail is used to send) is done via session class in the external package.



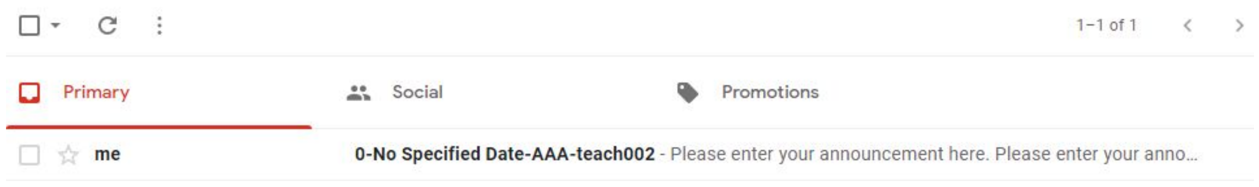
```java
    // Get the Session object.
    Session session = Session.getInstance(props,
        new javax.mail.Authenticator() {
            public PasswordAuthentication getPasswordAuthentication() {
                return new PasswordAuthentication(username, password);
            }
        }
    );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

A MimeMessage instance is then created to contain the sender, receiver, subject and text information of the actual email containing the announcement. The message is then sent using Transport.send() from the external JavaMail package.



```
    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);
    }
}
```

As shown below, the sending of announcements are successful.

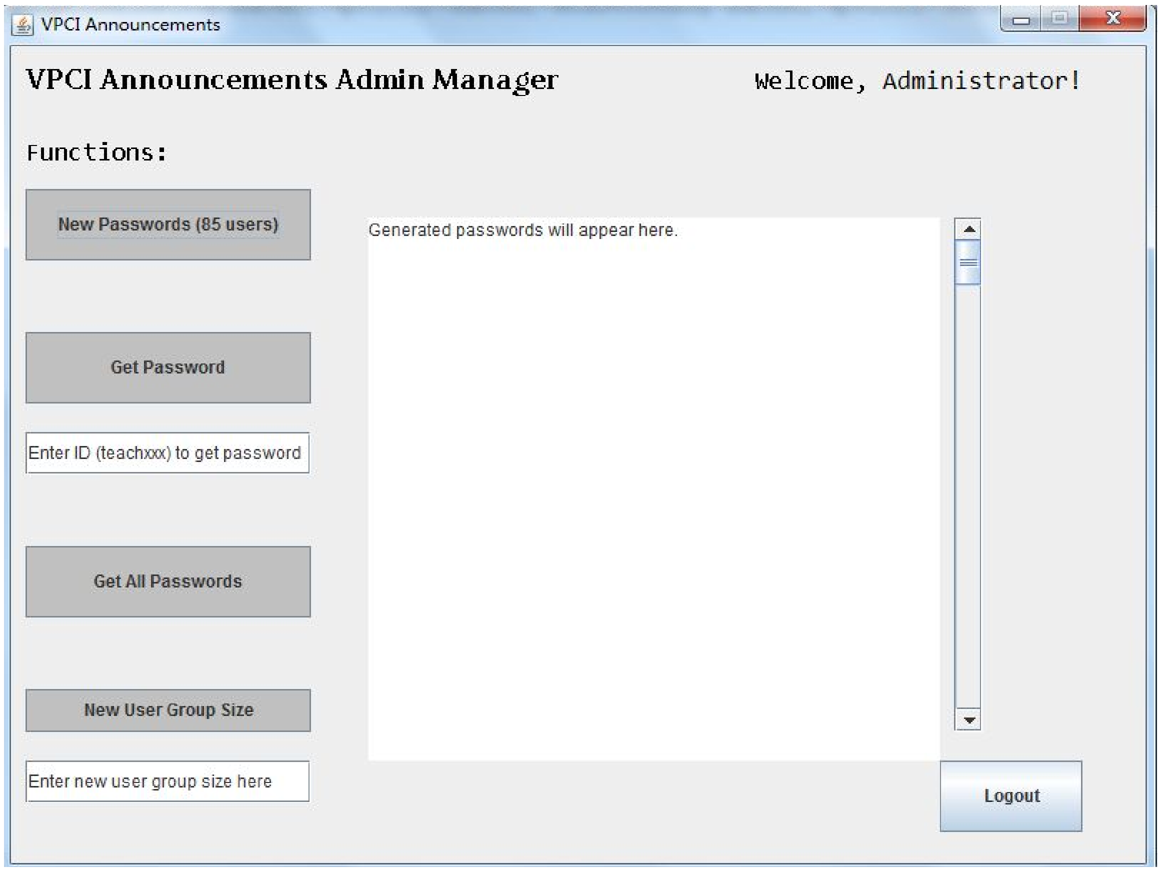


me 0-No Specified Date-AAA-teach002 - Please enter your announcement here. Please enter your anno...

Administrator Control JFrame

If the user “teach001” successfully logged in, an AdminGUI instance, inhering JFrame and other aspects from UserGUI class, is created and the adminGUI JFrame is displayed with run() method.

```java
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();
}
```

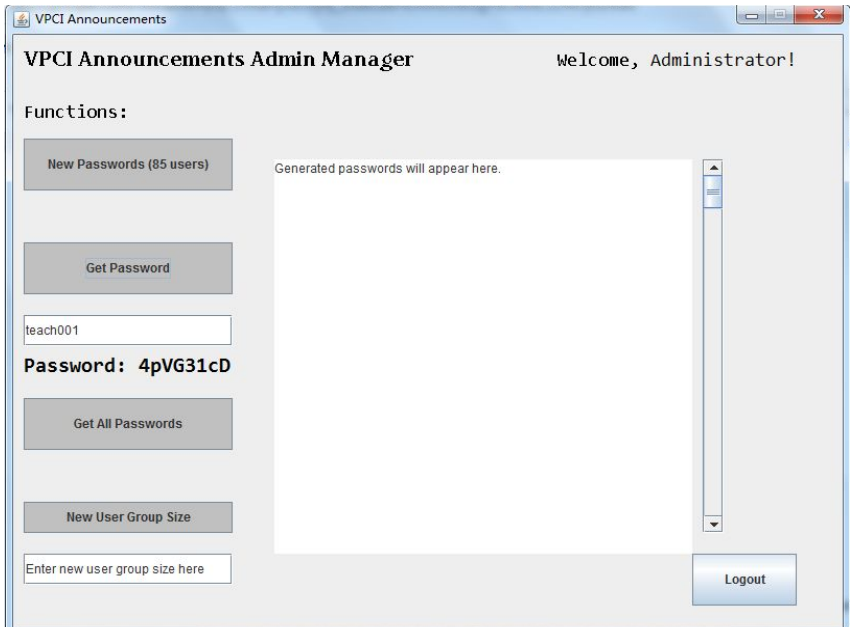

VPCI Announcements
VPCI Announcements Admin Manager
Welcome, Administrator!
Functions:
New Passwords (85 users)
Get Password
Enter ID (teachxxx) to get password
Get All Passwords
New User Group Size
Enter new user group size here
Generated passwords will appear here.
Logout

Get Password and Get All Passwords Functions

These two buttons, as their name suggests, get the password of a specific id entered or get all passwords.

Get password button checks if the function id entered is valid; a message of the password or error due to invalid id is displayed.

```java
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();
        }
    }
});
```

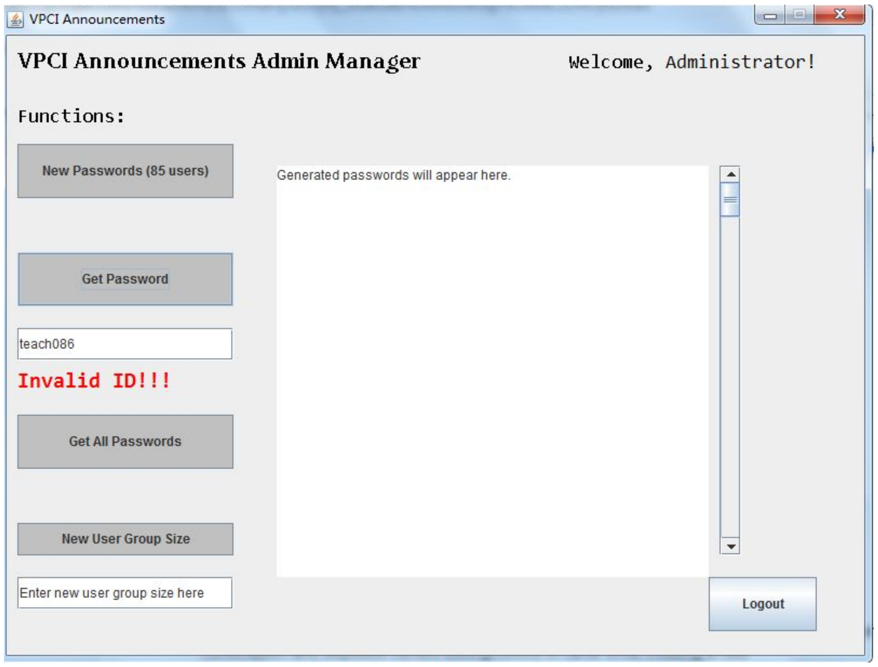


teach001

Password: 4pVG31cD

Generated passwords will appear here.

Enter new user group size here


VPCI Announcements

VPCI Announcements Admin Manager

Welcome, Administrator!

Functions:

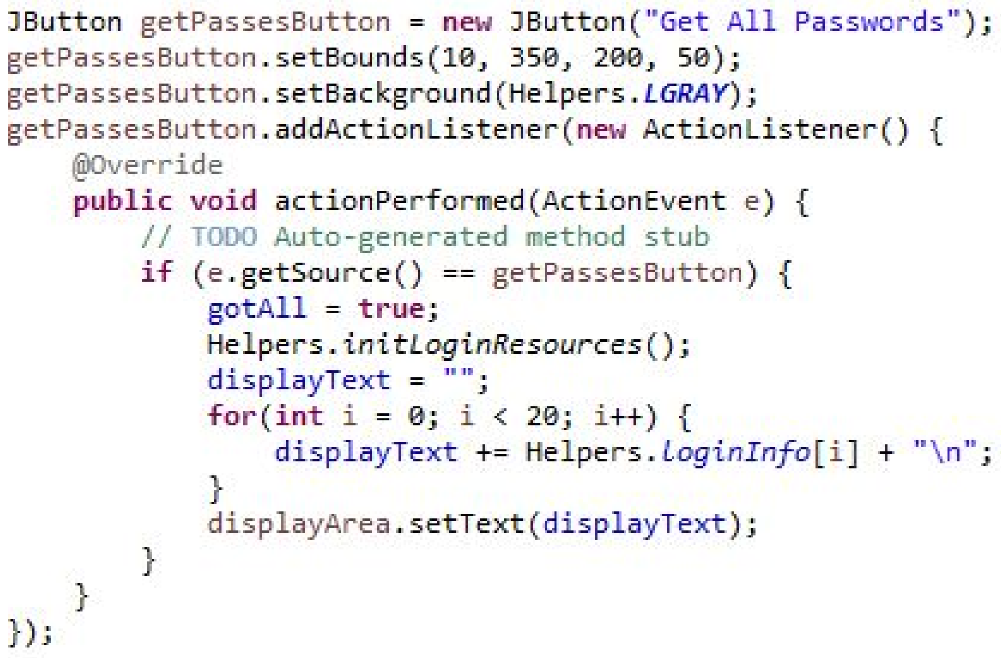
teach086

Invalid ID!!!

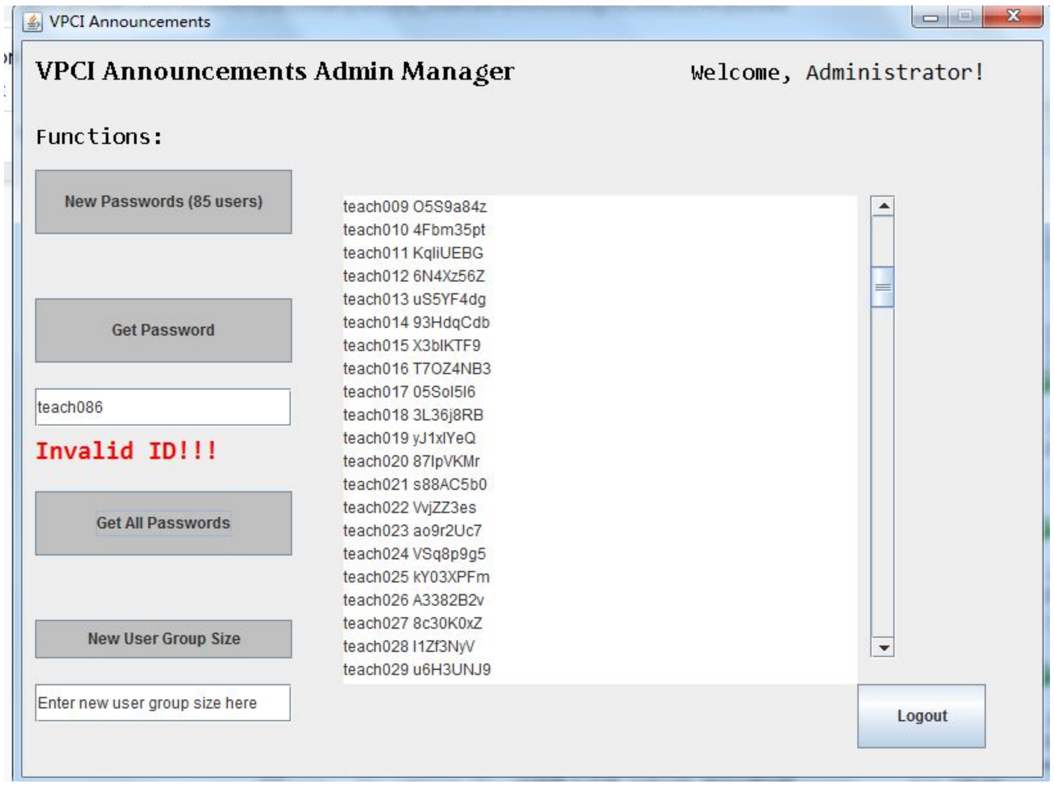
Generated passwords will appear here.

Enter new user group size here

Get all passwords button simply outputs all login resources into the JTextarea on the right. A JScrollBar is added to move the text up and down for reading.


```java
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);
        }
    }
});
```

```java
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();
            }

        }
    }
});
```

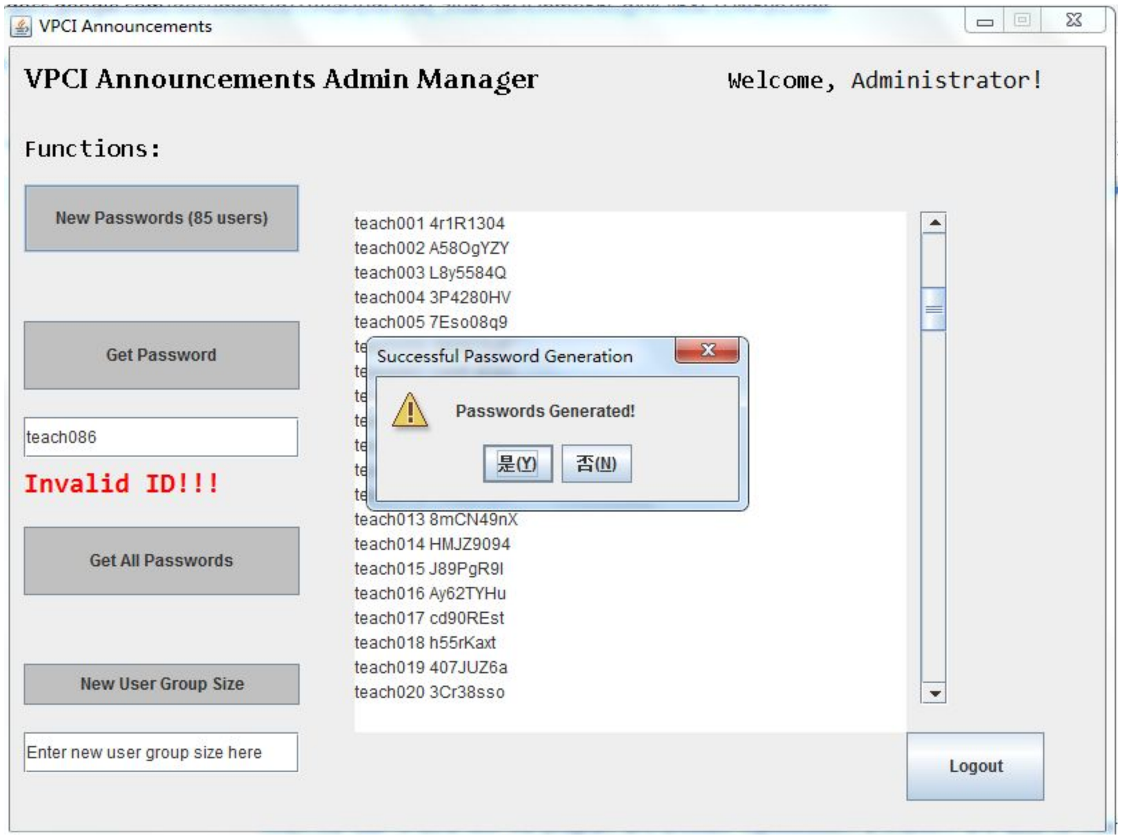


```
teach009 O5S9a84z
teach010 4Fbm35pt
teach011 KqliUEBG
teach012 6N4Xz56Z
teach013 uS5YF4dg
teach014 93HdqCdb
teach015 X3blKTF9
teach016 T7OZ4NB3
teach017 05Sol5l6
teach018 3L36j8RB
teach019 yJ1xlYeQ
teach020 87IpVKMr
teach021 s88AC5b0
teach022 VvjZZ3es
teach023 ao9r2Uc7
teach024 VSq8p9g5
teach025 kY03XPFm
teach026 A3382B2v
teach027 8c30K0xZ
teach028 I1Zf3NyV
teach029 u6H3UNJ9
```

<u>New Passwords and New User Group Size Functions</u>

If the user wants to generate new passwords for current user group size or for a new group size, these two buttons use the generatePasswords() method in Helpers and displays all passwords in the JTextarea after JOptionPane prompt/message.

```java
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);
        }
    }
});
```



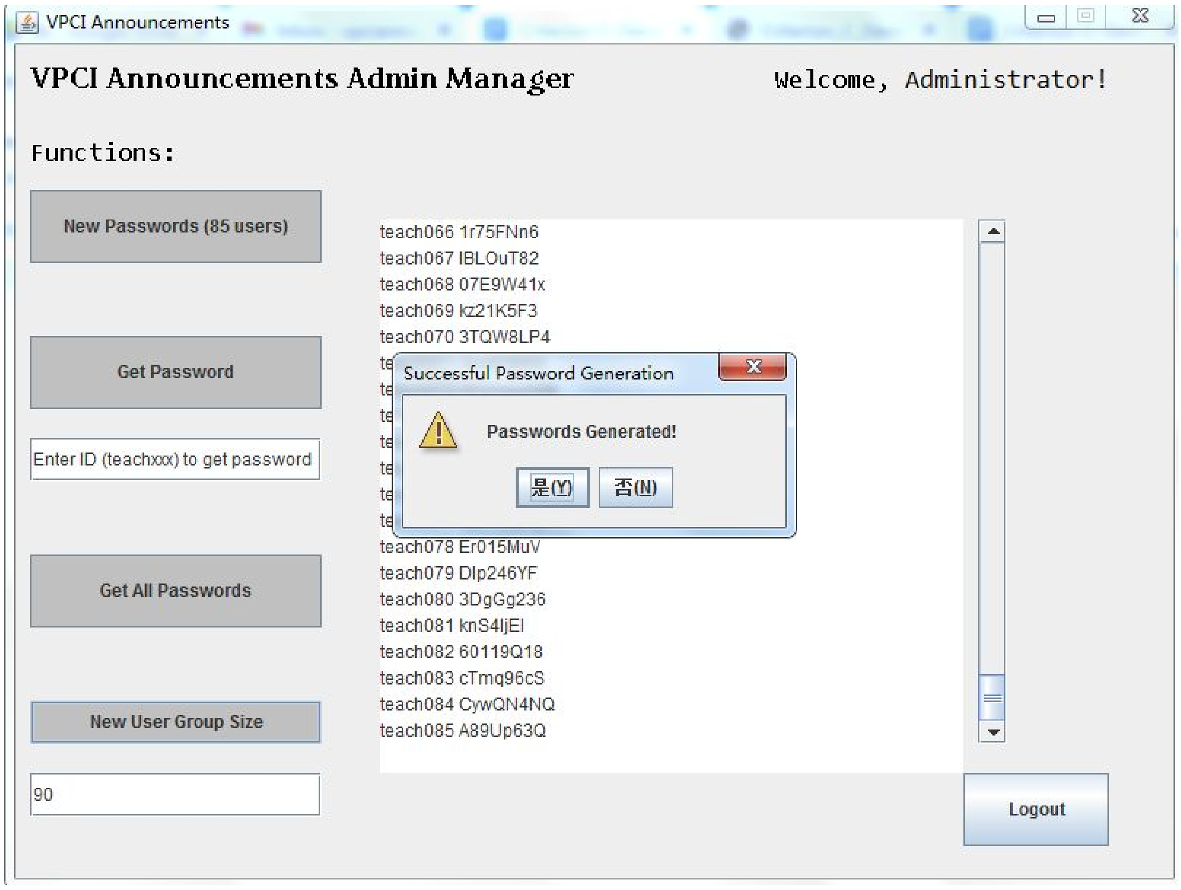
teach086

Invalid ID!!!

```
teach001 4r1R1304
teach002 A58OgYZY
teach003 L8y5584Q
teach004 3P4280HV
teach005 7Eso08q9
te
te
te
te
te
te
te
teach013 8mCN49nX
teach014 HMJZ9094
teach015 J89PgR9I
teach016 Ay62TYHu
teach017 cd90REst
teach018 h55rKaxt
teach019 407JUZ6a
teach020 3Cr38sso
```

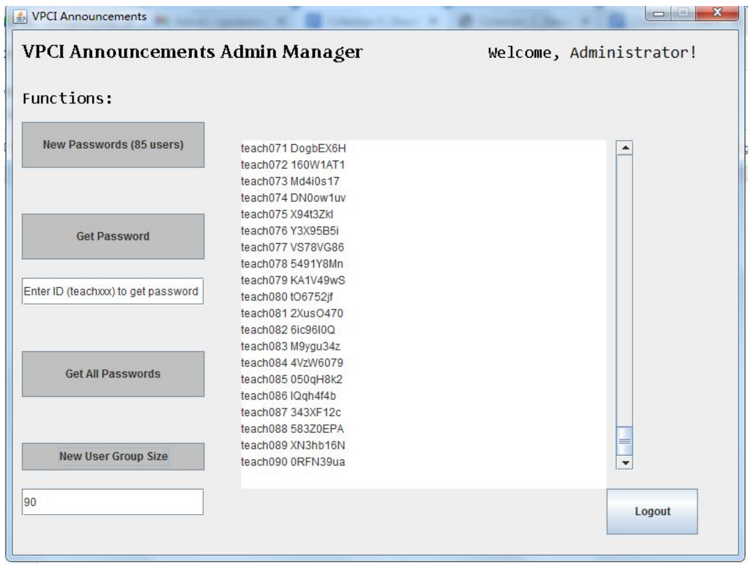
Passwords Generated!

```
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!");
            }

        }
    }
});
```



Passwords Generated!

```
teach066 1r75FNn6
teach067 IBLOuT82
teach068 07E9W41x
teach069 kz21K5F3
teach070 3TQW8LP4
te
te
te
te
te
te
te
teach078 Er015MuV
teach079 DIp246YF
teach080 3DgGg236
teach081 knS4IjEI
teach082 60119Q18
teach083 cTmq96cS
teach084 CywQN4NQ
teach085 A89Up63Q
```



```
teach071 DogbEX6H
teach072 160W1AT1
teach073 Md4i0s17
teach074 DN0ow1uv
teach075 X94t3Zkl
teach076 Y3X95B5i
teach077 VS78VG86
teach078 5491Y8Mn
teach079 KA1V49wS
teach080 tO6752jf
teach081 2XusO470
teach082 6ic96I0Q
teach083 M9ygu34z
teach084 4VzW6079
teach085 050qH8k2
teach086 lQqh4f4b
teach087 343XF12c
teach088 583Z0EPA
teach089 XN3hb16N
teach090 0RFN39ua
```

<u>Logout Function</u>

When the user is done with the program and presses logoutButton, a JOptionPane asks for their confirmation and disposes current editing/control JFrame while creating a new AnnouncementsProgram to restart the process (run() sets the login JFrame visible since it is private).

```java
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();
        }
    }
});
```

Word count: 1207 words

External API (Application Programming Interface) Citation (not included in word count)

Oracle Corporation (2021). JavaMail API (1.6.5). Retrieved from https://javaee.github.io/javamail/.