

**应用基础实践（一）网络与JAVA**

**课程设计实验报告**

|  |  |
| --- | --- |
| 学生姓名 | **胡琛** |
| 学 号 | L208220432 |
| 专业班级 | **计算机与技术** |
| 指导教师 | **刘嫔** |
| 学 院 | 计算机学院 |
| 完成时间 | 2024-07-10 |

计算机学院

2024年7月

Contents

[**用JavaMail编程实现邮件客户端** 3](#_Toc171743095)

[**1.系统描述：Java邮件客户端** 3](#_Toc171743096)

[**2.系统内容** 3](#_Toc171743097)

[**2.1. 用户认证** 3](#_Toc171743098)

[**2.2. 邮件发送** 3](#_Toc171743099)

[**2.3 邮件接收** 4](#_Toc171743100)

[**2.4 附件处理** 4](#_Toc171743101)

[**2.5用户界面** 4](#_Toc171743102)

[**3.系统设计图** 5](#_Toc171743103)

[**3.1登录窗口设计图：** 5](#_Toc171743104)

[**3.2邮件处理窗口设计图：** 5](#_Toc171743105)

[**4.功能模块结构** 5](#_Toc171743106)

[**4.1功能模块划分** 5](#_Toc171743107)

[**4.2功能模块结构图** 6](#_Toc171743108)

[**5.各模块流程图** 7](#_Toc171743109)

[**5.1用户认证模块流程图**  **5.2邮件发送模块流程图** 7](#_Toc171743110)

[**5.3邮件接收模块流程图**  **5.4附件处理模块流程图** 8](#_Toc171743111)

[**5.5用户界面模块流程图** 9](#_Toc171743112)

[**5.6各模块功能描述** 9](#_Toc171743113)

[**6.主要模块的算法说明** 10](#_Toc171743114)

[**6.1 用户认证模块** 10](#_Toc171743115)

[**6.2 邮件发送模块** 11](#_Toc171743116)

[**6.3 邮件接收模块** 12](#_Toc171743117)

[**6.4 附件处理模块** 14](#_Toc171743118)

[**7.项目依赖库设置** 15](#_Toc171743119)

[**8.运行结果** 15](#_Toc171743120)

[**8,2输出结果：** 15](#_Toc171743121)

[**8.3.邮件发送功能** 16](#_Toc171743122)

[**8.4接收邮件：** 18](#_Toc171743123)

[**9.课程设计总结** 19](#_Toc171743124)

[**9.1.课程设计中遇到的问题** 19](#_Toc171743125)

[**9.2SMTP 和 POP3 服务器连接问题：** 19](#_Toc171743126)

[**9.3处理附件时出现问题：** 19](#_Toc171743127)

[**9.4 解决问题的过程和经验** 19](#_Toc171743128)

[**9.5课程设计的收获** 20](#_Toc171743129)

[**9.6. 对程序开发的理解和思考** 20](#_Toc171743130)

[**9.7总结** 20](#_Toc171743131)

[10.参考文献 20](#_Toc171743132)

[**附录：主要程序清单** 22](#_Toc171743133)

**用JavaMail编程实现邮件客户端**

**1.系统描述：Java邮件客户端**

**系统基本要求**

实践目的或任务：通过指导学生上机实践，对JavaMail编程、Java图形用户界面进行掌握。邮件客户端和服务器之间通信协议SMTP和POP3协议的规范。

实践基本要求：

1.了解实验目的及实验原理；

2.编写程序，并附上程序代码和结果图；

3.总结在编程过程中遇到的问题、解决办法和收获。

实践的内容或要求：

1. 编写图形用户界面，实现邮件客户端软件的设计与开发。用实际的邮箱账号密码进行登录。

2. 客户端要求可以完成邮件编辑。

3. 客户端要求可以配置与标准邮件服务器的连接。

4. 客户端可以发送邮件，并可以粘贴附件。

5. 客户端可以接受并且下载邮件、删除邮件。

6. 选择一特定的邮件服务器建立邮件账户，编程实现与服务器的通信建立、利用SMTP协议完成邮件发送，利用POP3完成邮件接收。

**2.系统内容**

**2.1. 用户认证**

用户认证模块负责验证用户的电子邮件地址和密码，并与邮件服务器建立连接。系统需要配置SMTP和POP3服务器的属性，使用用户提供的凭据进行认证。成功登录后，用户可以访问邮件处理功能。

需求：

输入框：用于输入电子邮件地址和密码。

登录按钮：点击后进行认证操作。

实现：

使用JavaMail API配置邮件服务器属性。

使用用户输入的凭据创建邮件会话。

成功登录后切换到邮件处理窗口。

**2.2. 邮件发送**

邮件发送模块允许用户输入邮件的收件人地址、主题和正文，并支持附件功能。用户可以通过点击发送按钮将邮件发送到指定的收件人。

需求：

输入框：用于输入收件人地址、主题和邮件正文。

附件按钮：用于选择本地文件作为附件。

发送按钮：点击后发送邮件。

实现：

创建MimeMessage对象，设置发件人、收件人、主题和正文。

如果有附件，创建附件部分并添加到邮件中。

使用Transport.send方法发送邮件。

**2.3 邮件接收**

邮件接收模块允许用户接收收件箱中的所有邮件，并显示每封邮件的详细信息。用户可以查看邮件的发件人、主题和内容，并能够删除邮件。

需求：

接收按钮：点击后接收所有邮件。

显示区域：用于显示邮件的发件人、主题和内容。

删除功能：接收邮件后将邮件从服务器上删除。

实现：

使用JavaMail API连接到POP3服务器。

获取收件箱中的所有邮件，逐一读取邮件的详细信息。

删除接收的邮件。

**2.4 附件处理**

附件处理模块允许用户选择本地文件作为邮件的附件，并支持移除已选择的附件。用户可以通过附件按钮选择文件，通过移除附件按钮删除已选择的文件。

需求：

附件按钮：用于选择本地文件。

移除附件按钮：用于删除已选择的附件。

显示标签：显示已选择的附件名称。

实现：

使用文件选择器选择本地文件，并将文件路径显示在标签上。

点击移除附件按钮后，清空附件选择并更新显示标签。

**2.5用户界面**

用户界面模块提供图形用户界面，使用户能够方便地进行各种操作。界面包括登录窗口和邮件处理窗口，布局合理，操作简便。

需求：

登录窗口：包含电子邮件地址、密码输入框和登录按钮。

邮件处理窗口：包含收件人地址、主题、正文输入框，附件按钮，发送和接收按钮。

实现：

使用Swing组件创建图形用户界面。

设置窗口布局和组件排列，使操作直观、简便。

登录成功后切换到邮件处理窗口。

**3.系统设计图**

**3.1登录窗口设计图：**

+-------------------------------+

|          登录窗口             |

+-------------------------------+

|  Email:      [            ]   |

|  Password:   [\*\*\*\*\*\*\*\*\*\*\*\*]   |

|                               |

|               [ Login ]       |

+-------------------------------+

## **3.2邮件处理窗口设计图：**

+----------------------------------------+

|      JavaMail Email Client             |

+----------------------------------------+

|  To:             [                ]    |

|  Subject:        [                ]    |

|  Message:        [                ]    |

|                 +------------------+   |

|                 |                  |   |

|                 |                  |   |

|                 +------------------+   |

|  Attachment:    [Attach File]          |

|  Selected File: [                 ]    |

|  [Remove Attachment]                   |

|                                        |

|  [Send Email]   [Receive Email]        |

+----------------------------------------+

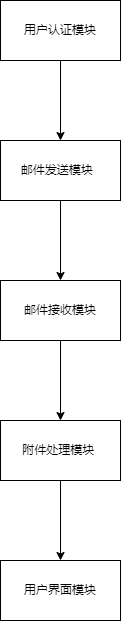
# **4.功能模块结构**

## **4.1功能模块划分**

1. 用户认证模块
2. 邮件发送模块
3. 邮件接收模块
4. 附件处理模块
5. 用户界面模块

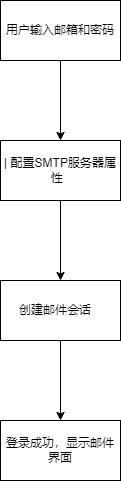
、

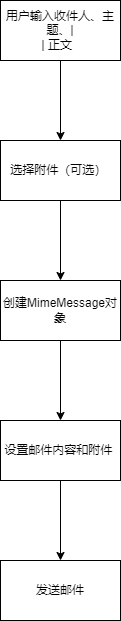
## **4.2功能模块结构图**

****

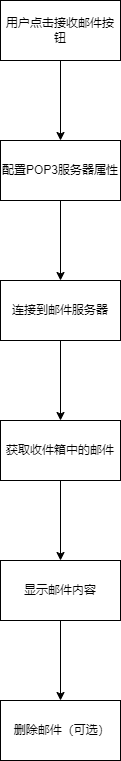
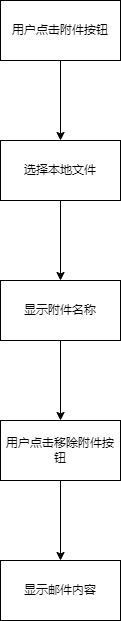
# **5.各模块流程图**

## **5.1用户认证模块流程图 5.2邮件发送模块流程图**

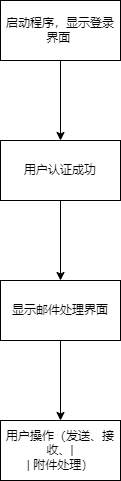
****

****

## **5.3邮件接收模块流程图 5.4附件处理模块流程图**

** **

## **5.5用户界面模块流程图**

****

## **5.6各模块功能描述**

**1.用户认证模块**

功能：验证用户的电子邮件地址和密码，并与SMTP服务器建立连接。

输入：电子邮件地址、密码。

输出：邮件会话，登录成功或失败提示。

**2.邮件发送模块**

功能：允许用户输入邮件的收件人地址、主题和正文，并支持附件发送。

输入：收件人地址、邮件主题、邮件正文、附件（可选）。

输出：发送邮件成功或失败提示。

**3.邮件接收模块**

功能：接收存储在服务器上的邮件，显示每封邮件的详细信息，并支持删除邮件。

输入：接收邮件指令。

输出：邮件列表及其详细信息，删除邮件功能。

**4.附件处理模块**

功能：允许用户选择本地文件作为邮件附件，并支持移除已选择的附件。

输入：附件文件、移除附件指令。

输出：已选择的附件名称，附件移除后的更新状态。

**5.用户界面模块**

功能：提供图形用户界面，使用户能够方便地进行邮件操作。

输入：用户操作（如输入邮件信息、选择附件、发送邮件、接收邮件等）。

输出：各功能模块的操作界面及相应的反馈信息。

# **6.主要模块的算法说明**

## **6.1 用户认证模块**

实现思路：

用户输入电子邮件地址和密码。

配置SMTP服务器属性，包括主机名、端口号、认证方式和TLS启用。

使用用户提供的电子邮件地址和密码创建一个邮件会话。

尝试连接SMTP服务器进行认证。

如果认证成功，则切换到邮件处理界面；如果失败，则提示用户重新输入。

**算法流程：**

-用户在GUI中输入电子邮件地址和密码。

-配置SMTP服务器属性：

-mail.smtp.auth 设置为 true 以启用认证。

-mail.smtp.starttls.enable 设置为 true 以启用TLS。

-mail.smtp.host 设置为SMTP服务器的主机名（如smtp.gmail.com）。

-mail.smtp.port 设置为服务器的端口号（如587）。

-创建一个新的Session实例，使用Authenticator对象提供用户的电子邮件地址和密码。

-尝试使用用户提供的凭据连接到SMTP服务器。

-如果连接成功，显示邮件处理界面；否则，显示错误消息。

private void login() {

    Properties properties = new Properties();

    properties.put("mail.smtp.auth", "true");

    properties.put("mail.smtp.starttls.enable", "true");

    properties.put("mail.smtp.host", "smtp.gmail.com");

    properties.put("mail.smtp.port", "587");

    String email = emailField.getText();

    String password = new String(passwordField.getPassword());

    session = Session.getInstance(properties, new Authenticator() {

        protected PasswordAuthentication getPasswordAuthentication() {

            return new PasswordAuthentication(email, password);

        }

    });

    try {

        Transport transport = session.getTransport("smtp");

        transport.connect();

        JOptionPane.showMessageDialog(loginFrame, "Login successful!");

        loginFrame.setVisible(false);

        emailFrame.setVisible(true);

    } catch (MessagingException e) {

        JOptionPane.showMessageDialog(loginFrame, "Login failed: " + e.getMessage());

    }

}

## **6.2 邮件发送模块**

**实现思路：**

用户在GUI中输入收件人地址、邮件主题和邮件正文。

如果用户选择了附件，则将附件添加到邮件中。

创建一个MimeMessage对象，并设置邮件的发件人、收件人、主题和内容。

使用Transport.send方法发送邮件。

显示发送成功或失败的提示信息。

**算法流程：**

用户在GUI中输入收件人地址、邮件主题和邮件正文。

创建一个新的MimeMessage对象，使用当前会话。

设置邮件的发件人、收件人和主题。

创建一个MimeBodyPart对象，设置邮件正文。

创建一个Multipart对象，将正文部分添加到Multipart中。

如果用户选择了附件，创建一个MimeBodyPart对象并将附件添加到Multipart中。

将Multipart对象设置为邮件内容。

使用Transport.send方法发送邮件。

显示发送成功或失败的提示信息。

private void sendEmail() {

    try {

        String to = toField.getText();

        String subject = subjectField.getText();

        String messageText = messageArea.getText();

        Message message = new MimeMessage(session);

        message.setFrom(new InternetAddress(emailField.getText()));

        message.setRecipients(Message.RecipientType.TO, InternetAddress.parse(to));

        message.setSubject(subject);

        MimeBodyPart textPart = new MimeBodyPart();

        textPart.setText(messageText);

        Multipart multipart = new MimeMultipart();

        multipart.addBodyPart(textPart);

        if (selectedFile != null) {

            MimeBodyPart attachmentPart = new MimeBodyPart();

            attachmentPart.attachFile(selectedFile);

            multipart.addBodyPart(attachmentPart);

        }

        message.setContent(multipart);

        Transport.send(message);

        JOptionPane.showMessageDialog(emailFrame, "Email sent successfully!");

    } catch (MessagingException | IOException e) {

        JOptionPane.showMessageDialog(emailFrame, "Error sending email: " + e.getMessage());

    }

}

## **6.3 邮件接收模块**

**实现思路：**

用户点击接收邮件按钮。

配置POP3服务器属性，包括主机名、端口号和TLS启用。

使用用户提供的电子邮件地址和密码连接到POP3服务器。

获取收件箱中的所有邮件。

逐一读取邮件的发件人、主题和内容，并显示在GUI中。

提供删除功能，允许用户选择删除邮件。

**算法流程：**

用户点击接收邮件按钮。

配置POP3服务器属性：

mail.pop3.host 设置为POP3服务器的主机名（如pop.gmail.com）。

mail.pop3.port 设置为服务器的端口号（如995）。

mail.pop3.starttls.enable 设置为 true 以启用TLS。

创建一个新的Session实例。

使用用户提供的凭据连接到POP3服务器。

打开收件箱文件夹，并获取所有邮件。

逐一读取每封邮件的发件人、主题和内容。

显示邮件详细信息。

如果用户选择删除邮件，将邮件标记为删除，并从服务器上移除。

private void receiveEmail() {

    try {

        Properties properties = new Properties();

        properties.put("mail.pop3.host", "pop.gmail.com");

        properties.put("mail.pop3.port", "995");

        properties.put("mail.pop3.starttls.enable", "true");

        String email = emailField.getText();

        String password = new String(passwordField.getPassword());

        Session emailSession = Session.getDefaultInstance(properties);

        Store store = emailSession.getStore("pop3s");

        store.connect("pop.gmail.com", email, password);

        Folder emailFolder = store.getFolder("INBOX");

        emailFolder.open(Folder.READ\_WRITE); // 需要删除邮件

        Message[] messages = emailFolder.getMessages();

        for (Message message : messages) {

            System.out.println("Subject: " + message.getSubject());

            System.out.println("From: " + message.getFrom()[0]);

            Object content = message.getContent();

            if (content instanceof String) {

                System.out.println("Text: " + content);

            } else if (content instanceof Multipart) {

                Multipart multipart = (Multipart) content;

                for (int i = 0; i < multipart.getCount(); i++) {

                    BodyPart bodyPart = multipart.getBodyPart(i);

                    if (bodyPart.isMimeType("text/plain")) {

                        System.out.println("Text: " + bodyPart.getContent());

                    } else if (bodyPart.isMimeType("text/html")) {

                        String html = (String) bodyPart.getContent();

                        System.out.println("HTML: " + html);

                    }

                }

            }

            message.setFlag(Flags.Flag.DELETED, true);

        }

        emailFolder.close(true); // true表示删除已标记为删除的邮件

        store.close();

        JOptionPane.showMessageDialog(emailFrame, "Emails received and deleted successfully!");

    } catch (NoSuchProviderException e) {

        JOptionPane.showMessageDialog(emailFrame, "No provider: " + e.getMessage());

    } catch (MessagingException e) {

        JOptionPane.showMessageDialog(emailFrame, "Messaging exception: " + e.getMessage());

    } catch (IOException e) {

        JOptionPane.showMessageDialog(emailFrame, "IO exception: " + e.getMessage());

    }

}

## **6.4 附件处理模块**

实现思路：

用户点击附件按钮，选择本地文件作为邮件附件。

系统使用文件选择器打开文件对话框，允许用户选择文件。

选择文件后，显示文件名称。

用户可以点击移除附件按钮，移除已选择的附件。

**算法流程：**

用户点击附件按钮。

打开文件选择器对话框，允许用户选择本地文件。

用户选择文件后，获取文件路径并显示文件名称。

用户点击移除附件按钮，清空附件选择并更新显示标签。

attachButton.addActionListener(new ActionListener() {

    public void actionPerformed(ActionEvent e) {

        JFileChooser fileChooser = new JFileChooser();

        int returnValue = fileChooser.showOpenDialog(null);

        if (returnValue == JFileChooser.APPROVE\_OPTION) {

            selectedFile = fileChooser.getSelectedFile();

            attachmentLabel.setText(selectedFile.getName());

        }

    }

});

removeAttachmentButton.addActionListener(new ActionListener() {

    public void actionPerformed(ActionEvent e) {

        selectedFile = null;

        attachmentLabel.setText("");

    }

});

# **7.项目依赖库设置**

在开发Java邮件客户端项目时，需要引入JavaMail库和相关的依赖库，以便实现邮件的发送和接收功能。以下是依赖库设置说明。

依赖库文件

在项目中，lib文件夹下包含了三个主要的JAR文件：

activation-1.1.1.jar

javax.mail-1.6.2.jar

javax.mail.jar

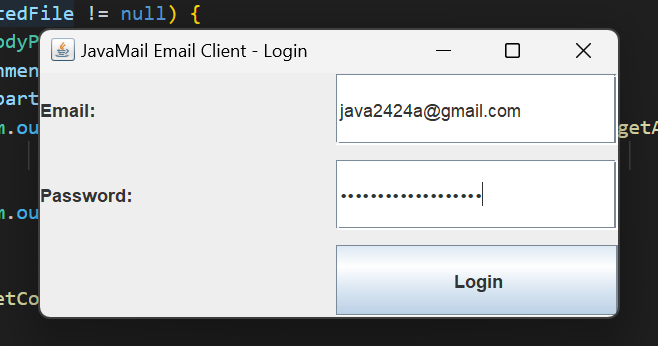
这些JAR文件分别提供了JavaMail API和激活框架的支持。

# **8.运行结果**

**8.1 登录界面 输入数据：**

**Email: example@gmail.com**

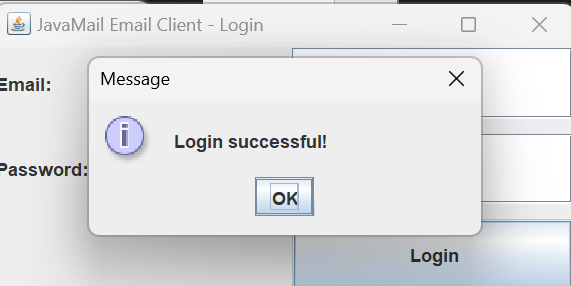
**Password: \*\*\*\*\*\*\*\***



## **8,2输出结果：**

成功登录后显示消息 "Login successful!" 并切换到邮件处理界面。

登录失败显示消息 "Login failed: [错误信息]"



## **8.3.邮件发送功能**

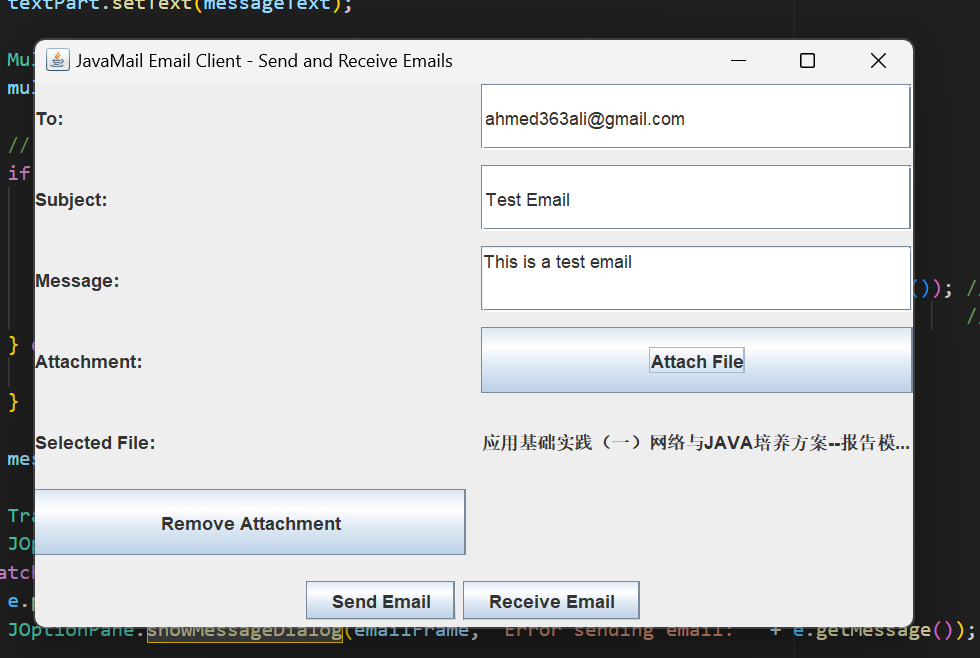
**输入数据：**

**To: ahmed363ali@gmail.com**

**Subject: Test Email**

**Message: This is a test email.**

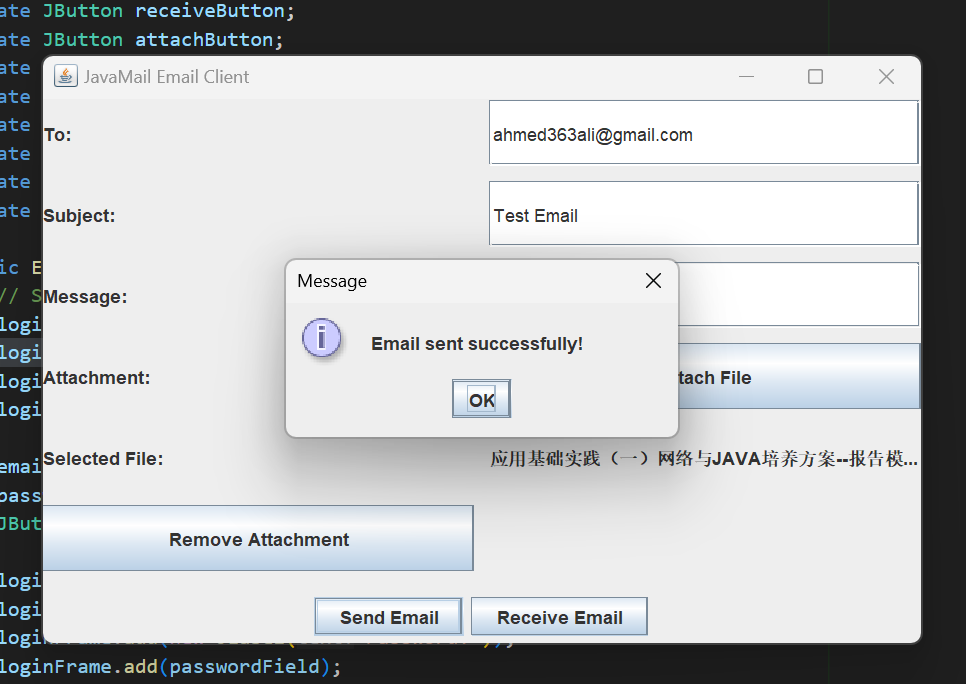
**Attachment: C:\path\to\file.txt.[ (可选) 如果需要，点击 "Attach File" 按钮选择一个文件.作为附件。点击 "Send Email" 按钮发送邮件。**



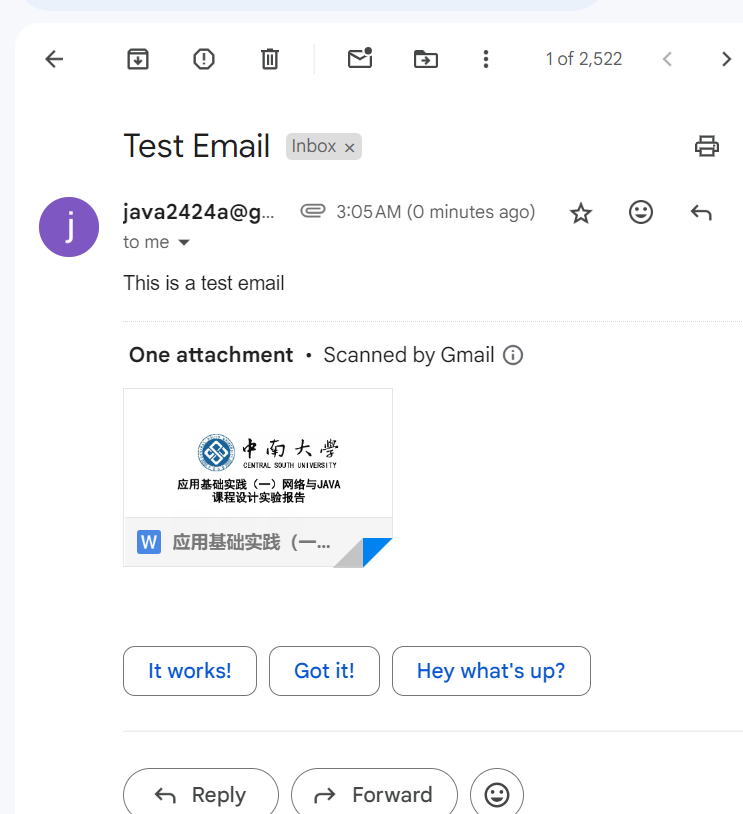
**输出结果：**

成功发送邮件后显示消息 "Email sent successfully!"。

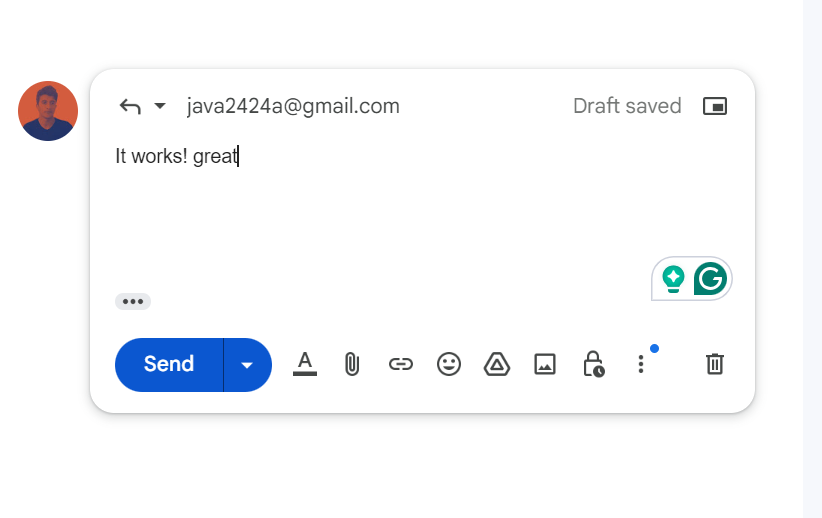
发送失败显示消息 "Error sending email: [错误信息]"。



**Email sent successfully!"**



回email消息 it’s work great！

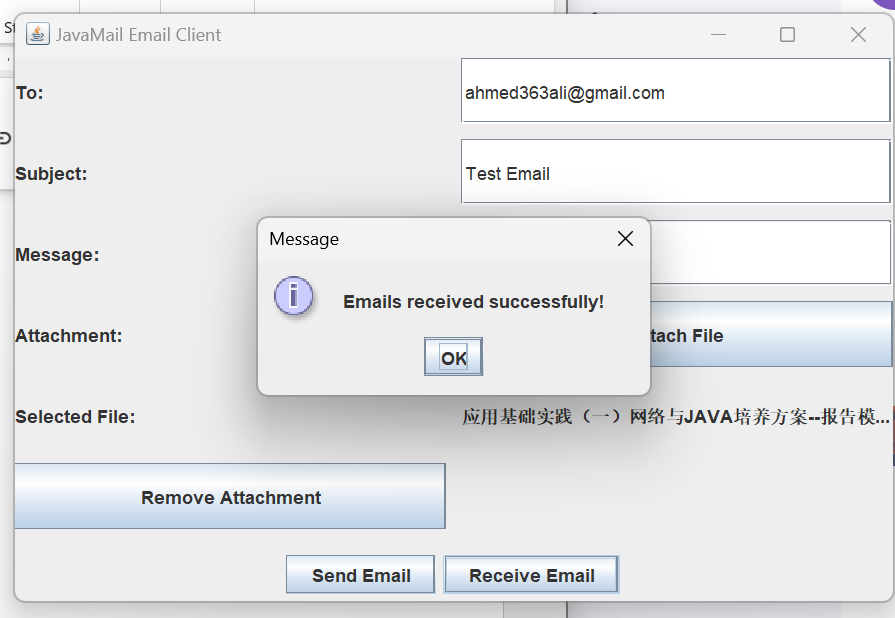


## **8.4接收邮件：**

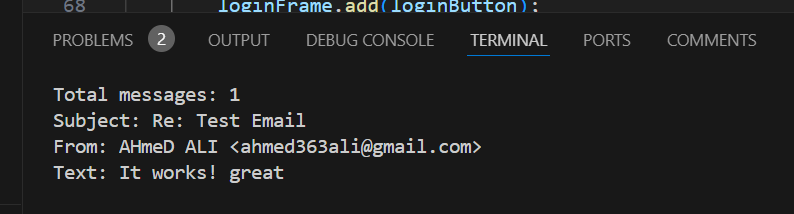
点击 "Receive Email" 按钮。

显示收到的邮件列表及详细信息。

显示 "Emails received successfully!"。



**Email 内容**



通过上述运行结果示例，可以看到Java邮件客户端在用户输入有效凭据后能够成功登录，用户能够发送带附件的邮件，并能够接收和删除收件箱中的邮件。

# **9.课程设计总结**

## **9.1.课程设计中遇到的问题**

**依赖库配置问题：**

**问题描述**： 在项目之初，我在配置依赖于JavaMail API的库时遇到了一些困难。虽然在项目的lib文件夹下添加了JAR文件，但是编译的时候报错，提示找不到相关的类。

**解决方案**：通过参考IntelliJ IDEA的帮助文档，您了解了如何正确将外部JAR文件添加到项目依赖中，并确保它们在编译和运行时被正确识别。具体步骤包括在项目设置中添加 JAR 文件并设置其编译范围。

## **9.2SMTP 和 POP3 服务器连接问题：**

**问题描述**：在执行用户身份验证和电子邮件接收功能时，与 SMTP 和 POP3 服务器的连接有时会失败，并且错误消息不明确。

**解决方案：**通过详细阅读错误信息，并结合JavaMail API官方文档，发现是由于服务器地址或端口号配置错误、TLS/SSL配置错误等导致的。通过正确配置服务器地址、端口号并启用 TLS/SSL，连接问题最终得到解决。

## **9.3处理附件时出现问题：**

**问题描述：**发送带有附件的电子邮件时，某些电子邮件客户端无法正确显示附件。

**解决方案**：通过分析发送邮件的代码，发现问题出在邮件内容的MIME类型设置上。通过正确设置Multipart和MimeBodyPart对象，并确保邮件内容和附件的MIME类型正确，解决附件显示问题。

## **9.4 解决问题的过程和经验**

在解决依赖库配置问题时，通过阅读IDE的帮助文档和在线资源，我逐渐掌握了如何正确配置外部依赖。关于服务器连接问题，我通过阅读JavaMail API官方文档和在线社区的讨论，学习了如何调试和解决网络连接问题。在处理附件问题时，通过反复测试和代码调优，我终于明白了电子邮件MIME类型的重要性。解决这些问题的过程让我深刻认识到：

**阅读文档的重要性：**官方文档和帮助文档是解决技术问题的重要资源，可以提供可靠、详细的指导。

**掌握调试技巧：**通过阅读错误信息和调试代码，可以快速确定问题的根本原因。

**利用社区资源**：在线社区和论坛是宝贵的资源，可以提供其他开发人员的专业知识和解决方案。

## **9.5课程设计的收获**

**掌握JavaMail API：**通过设计本课程，我深入学习并掌握了JavaMail API的使用，包括如何发送和接收电子邮件以及处理电子邮件附件。

**提高GUI编程能力：**在设计和实现GUI时，更加熟悉Swing组件的使用和布局管理，提高了GUI编程能力。

**提高项目管理和调试能力：**在项目开发过程中，我学习了如何配置和管理项目依赖项，以及如何通过调试解决各种技术问题。

## **9.6. 对程序开发的理解和思考**

**需求分析的重要性：**在开发一个完整的软件之前，详细的需求分析和功能模块的划分非常重要。这可以帮助我们明确各个模块的职责和实现思路，避免开发过程中出现混乱。

**良好的编程习惯：**我保持代码干净易读，并编写注释和文档以帮助后续维护和扩展。良好的编程习惯在团队合作中尤为重要。

**不断学习和自我完善**：技术在不断发展，作为一名开发人员，我需要不断学习和掌握新技术和工具。通过设计这门课程，我认识到艺术学习是一个积累和不断提高的过程。

## **9.7总结**

通过设计Java Mail Client课程，我不仅掌握了JavaMail API的使用，还提高了我的GUI编程和项目管理技能。在解决问题的过程中，我学会了如何利用文档、社区资源和调试技术。这些收获和经历将对我今后的程序开发工作产生深远的影响。

# 10.参考文献

1. JavaMail API Documentation:

- [Oracle JavaMail API Documentation](<https://javaee.github.io/javamail/>)

- 描述: 官方文档提供了JavaMail API的

2. Java Platform, Standard Edition (Java SE) Documentation:

- (<https://docs.oracle.com/en/java/javase/>)

3. Java Swing Tutorial:

(<https://docs.oracle.com/javase/tutorial/uiswing/>)

- 描述: 详细介绍了Java Swing的使用方法，包括各种GUI组件和布局管理器，为本项目的图形用户界面设计提供了指导。

4. Stack Overflow:

(<https://stackoverflow.com/>)

- 描述: 提供了大量开发者的经验和解决方案，通过搜索相关问题，我在解决依赖库配置和网络连接问题时受到了很大帮助。

5. SMTP and POP3 Protocols:

- (<https://tools.ietf.org/html/rfc5321>)

, [POP3 Protocol](https://tools.ietf.org/html/rfc1939)

- 描述: 详细介绍了SMTP和POP3协议的规范和工作原理，为实现邮件发送和接收功能提供了理论基础。

6. Java Activation Framework (JAF) (<https://javaee.github.io/activation/>)

-(<https://www.jetbrains.com/idea/documentation/>)

- 描述: 详细介绍了IntelliJ IDEA的使用方法和功能，通过阅读文档，学会了如何在项目中添加和管理依赖库。

7. GitHub Repositories:

- (<https://github.com/>)

# **附录：主要程序清单**

import java.awt.BorderLayout;

import java.awt.GridLayout;

import java.awt.event.ActionEvent;

import java.awt.event.ActionListener;

import java.io.File;

import java.io.IOException;

import java.util.Properties;

import javax.mail.Authenticator;

import javax.mail.BodyPart;

import javax.mail.Flags;

import javax.mail.Folder;

import javax.mail.Message;

import javax.mail.MessagingException;

import javax.mail.Multipart;

import javax.mail.NoSuchProviderException;

import javax.mail.PasswordAuthentication;

import javax.mail.Session;

import javax.mail.Store;

import javax.mail.Transport;

import javax.mail.internet.InternetAddress;

import javax.mail.internet.MimeBodyPart;

import javax.mail.internet.MimeMessage;

import javax.mail.internet.MimeMultipart;

import javax.swing.JButton;

import javax.swing.JFileChooser;

import javax.swing.JFrame;

import javax.swing.JLabel;

import javax.swing.JOptionPane;

import javax.swing.JPanel;

import javax.swing.JPasswordField;

import javax.swing.JScrollPane;

import javax.swing.JTextArea;

import javax.swing.JTextField;

public class EmailClient extends JFrame {

    private JTextField emailField;  // 邮箱地址输入框

    private JPasswordField passwordField;  // 密码输入框

    private JTextField toField;  // 收件人地址输入框

    private JTextField subjectField;  // 邮件主题输入框

    private JTextArea messageArea;  // 邮件正文输入区域

    private JButton sendButton;  // 发送邮件按钮

    private JButton receiveButton;  // 接收邮件按钮

    private JButton attachButton;  // 附件按钮

    private JButton removeAttachmentButton;  // 移除附件按钮

    private JLabel attachmentLabel;  // 附件标签

    private JFrame loginFrame;  // 登录窗口

    private JFrame emailFrame;  // 邮件处理窗口

    private Session session;  // 邮件会话

    private File selectedFile;  // 已选择的附件文件

    public EmailClient() {

        // 初始化登录界面

        loginFrame = new JFrame("JavaMail Email Client - Login");

        loginFrame.setSize(400, 200);

        loginFrame.setDefaultCloseOperation(EXIT\_ON\_CLOSE);

        loginFrame.setLayout(new GridLayout(3, 2, 10, 10));

        emailField = new JTextField();

        passwordField = new JPasswordField();

        JButton loginButton = new JButton("Login");

        loginFrame.add(new JLabel("Email:"));

        loginFrame.add(emailField);

        loginFrame.add(new JLabel("Password:"));

        loginFrame.add(passwordField);

        loginFrame.add(new JPanel()); // 空白面板用于间隔

        loginFrame.add(loginButton);

        loginButton.addActionListener(new ActionListener() {

            public void actionPerformed(ActionEvent e) {

                login();

            }

        });

        loginFrame.setLocationRelativeTo(null); // 居中显示登录窗口

        loginFrame.setVisible(true);

        // 初始化邮件处理界面

        emailFrame = new JFrame("JavaMail Email Client");

        emailFrame.setSize(600, 400);

        emailFrame.setDefaultCloseOperation(EXIT\_ON\_CLOSE);

        emailFrame.setLayout(new BorderLayout(10, 10));

        JPanel emailPanel = new JPanel(new GridLayout(6, 2, 10, 10));

        toField = new JTextField();

        subjectField = new JTextField();

        messageArea = new JTextArea();

        sendButton = new JButton("Send Email");

        receiveButton = new JButton("Receive Email");

        attachButton = new JButton("Attach File");

        removeAttachmentButton = new JButton("Remove Attachment");

        attachmentLabel = new JLabel();

        emailPanel.add(new JLabel("To:"));

        emailPanel.add(toField);

        emailPanel.add(new JLabel("Subject:"));

        emailPanel.add(subjectField);

        emailPanel.add(new JLabel("Message:"));

        emailPanel.add(new JScrollPane(messageArea));

        emailPanel.add(new JLabel("Attachment:"));

        emailPanel.add(attachButton);

        emailPanel.add(new JLabel("Selected File:"));

        emailPanel.add(attachmentLabel);

        emailPanel.add(removeAttachmentButton);

        JPanel buttonPanel = new JPanel();

        buttonPanel.add(sendButton);

        buttonPanel.add(receiveButton);

        emailFrame.add(emailPanel, BorderLayout.CENTER);

        emailFrame.add(buttonPanel, BorderLayout.SOUTH);

        sendButton.addActionListener(new ActionListener() {

            public void actionPerformed(ActionEvent e) {

                sendEmail();

            }

        });

        receiveButton.addActionListener(new ActionListener() {

            public void actionPerformed(ActionEvent e) {

                receiveEmail();

            }

        });

        attachButton.addActionListener(new ActionListener() {

            public void actionPerformed(ActionEvent e) {

                JFileChooser fileChooser = new JFileChooser();

                int returnValue = fileChooser.showOpenDialog(null);

                if (returnValue == JFileChooser.APPROVE\_OPTION) {

                    selectedFile = fileChooser.getSelectedFile();

                    attachmentLabel.setText(selectedFile.getName());

                }

            }

        });

        removeAttachmentButton.addActionListener(new ActionListener() {

            public void actionPerformed(ActionEvent e) {

                selectedFile = null;

                attachmentLabel.setText("");

            }

        });

        emailFrame.setLocationRelativeTo(null); // 居中显示邮件窗口

    }

    private void login() {

        Properties properties = new Properties();

        properties.put("mail.smtp.auth", "true");

        properties.put("mail.smtp.starttls.enable", "true");

        properties.put("mail.smtp.host", "smtp.gmail.com");

        properties.put("mail.smtp.port", "587");

        String email = emailField.getText();

        String password = new String(passwordField.getPassword());

        session = Session.getInstance(properties, new Authenticator() {

            protected PasswordAuthentication getPasswordAuthentication() {

                return new PasswordAuthentication(email, password);

            }

        });

        JOptionPane.showMessageDialog(loginFrame, "Login successful!");

        loginFrame.setVisible(false);

        emailFrame.setVisible(true);

    }

    private void sendEmail() {

        try {

            String to = toField.getText();

            String subject = subjectField.getText();

            String messageText = messageArea.getText();

            Message message = new MimeMessage(session);

            message.setFrom(new InternetAddress(emailField.getText()));

            message.setRecipients(Message.RecipientType.TO, InternetAddress.parse(to));

            message.setSubject(subject);

            MimeBodyPart textPart = new MimeBodyPart();

            textPart.setText(messageText);

            Multipart multipart = new MimeMultipart();

            multipart.addBodyPart(textPart);

            // 添加附件部分

            if (selectedFile != null) {

                MimeBodyPart attachmentPart = new MimeBodyPart();

                attachmentPart.attachFile(selectedFile);

                multipart.addBodyPart(attachmentPart);

                System.out.println("Attachment file path: " + selectedFile.getAbsolutePath()); // 打印附件路径以供调试

            } else {

                System.out.println("No file attached.");

            }

            message.setContent(multipart);

            Transport.send(message);

            JOptionPane.showMessageDialog(emailFrame, "Email sent successfully!");

        } catch (MessagingException | IOException e) {

            e.printStackTrace();

            JOptionPane.showMessageDialog(emailFrame, "Error sending email: " + e.getMessage());

        }

    }

    private void receiveEmail() {

        try {

            Properties properties = new Properties();

            properties.put("mail.pop3.host", "pop.gmail.com");

            properties.put("mail.pop3.port", "995");

            properties.put("mail.pop3.starttls.enable", "true");

            String email = emailField.getText();

            String password = new String(passwordField.getPassword());

            Session emailSession = Session.getDefaultInstance(properties);

            Store store = emailSession.getStore("pop3s");

            store.connect("pop.gmail.com", email, password);

            Folder emailFolder = store.getFolder("INBOX");

            emailFolder.open(Folder.READ\_WRITE); // 需要删除邮件

            Message[] messages = emailFolder.getMessages();

            System.out.println("Total messages: " + messages.length);

            for (Message message : messages) {

                System.out.println("Subject: " + message.getSubject());

                System.out.println("From: " + message.getFrom()[0]);

                Object content = message.getContent();

                if (content instanceof String) {

                    System.out.println("Text: " + content);

                } else if (content instanceof Multipart) {

                    Multipart multipart = (Multipart) content;

                    for (int i = 0; i < multipart.getCount(); i++) {

                        BodyPart bodyPart = multipart.getBodyPart(i);

                        if (bodyPart.isMimeType("text/plain")) {

                            System.out.println("Text: " + bodyPart.getContent());

                        } else if (bodyPart.isMimeType("text/html")) {

                            String html = (String) bodyPart.getContent();

                            System.out.println("HTML: " + html);

                        }

                    }

                }

                // 删除邮件

                message.setFlag(Flags.Flag.DELETED, true);

            }

            emailFolder.close(true); // true表示删除已标记为删除的邮件

            store.close();

            JOptionPane.showMessageDialog(emailFrame, "Emails received successfully!");

        } catch (NoSuchProviderException e) {

            e.printStackTrace();

            JOptionPane.showMessageDialog(emailFrame, "No provider: " + e.getMessage());

        } catch (MessagingException e) {

            e.printStackTrace();

            JOptionPane.showMessageDialog(emailFrame, "Messaging exception: " + e.getMessage());

        } catch (IOException e) {

            e.printStackTrace();

            JOptionPane.showMessageDialog(emailFrame, "IO exception: " + e.getMessage());

        }

    }

    public static void main(String[] args) {

        EmailClient client = new EmailClient();

        client.setVisible(true);

    }

}