Rizvi College of Engineering

Department of Computer Engineering

SKL-OOP (JAVA) Mini Project Report

on

Email Sender Application

Submitted

by

Moin Mohammed Naik (29)

Saquib Mansoor Patel (33)

Rafat Muskan Shaikh (46)

Guide:

Prof. MOHD ASHFAQUE SHAIKH

University of Mumbai

2022-2023

CERTIFICATE

This is to certify that the mini-project entitled **"Email Sender Application"** is a bonafide work of "**Moin Naik (29), Saquib Patel (33) & Rafat Muskan Shaikh (46)"** submitted to the University of Mumbai in partial fulfillment of the requirement for the Mini-Project 1/2 for Second of the Bachelor of Engineering in **"Computer Engineering"**.

Prof. MOHD ASHFAQUE SHAIKH
Guide

_____

Prof. Shiburaj Pappu
Head of Department

_____

Dr. Varsha Shah
Principal

Abstract

In any software application, sending and receiving electronic messages, more specifically e-mails are an essential part. Emails are a medium of communication between different parties who are using the application. In most of the standard programming languages, email APIs are available for communication, and Java is also not an exception. Java provides e-mail APIs which are platform and protocol independent. The mail management framework consists of various abstract classes for defining an e-mail communication system. There are various ways to send email using JavaMail API like my SMTP server is mail.javatpoint.com or other companies e.g. gmail etc.

Project Description

Sending Email is a basic requirement regardless of which platform you are working on like JAVA, JavaEE, Python etc. Sending Email may be required to send error alerts or confirmation of registration or signup. Java provides the facility to send emails by writing java programs. To send emails using Java, you need three things:

- JavaMail API: These are the Java interfaces to send and receive e-mails. This layer is completely independent of the underlying protocols.
- Java Activation Framework (JAF): This framework is used to manage mail contents like URL, attachments, mail extensions etc.
- SMTP server details: *S*imple *M*ail *T*ransfer *P*rotocol is used for sending e-mails.

➢ How to send and receive e-mails?

Send Email: By using Java mail API and SMTP server.

Following are the steps to be followed. Setup 'From' and 'To' address along with the user id and password. Setup SMTP host Setup properties values Create session object Form the message details Send the message by using Transport object.

The application consists of four main classes:

- EmailUtility.java: implements a function for sending an e-mail message including attachments.
- ConfigUtility.java: implements two functions for reading and writing SMTP settings from/to the smtp.properties file as described above.
- SettingsDialog.java: implements a user interface that allows the user to update SMTP settings.
- SwingEmailSender.java: is the main entry of the application, it builds the main user interface that displays e-mail sending form and connects all the

above pieces together. To allow the user to add a file as attachment, this class uses the JFilePicker class.

- Protocols used in JavaMail API

There are some protocols that are used in JavaMail API.

- SMTP
- POP
- IMAP
- MIME
- NNTP and others

- POP:-

POP is an acronym for Post Office Protocol, also known as POP3. It provides a mechanism to receive the email. It provides support for single mail box for each user.

- SMPT:-

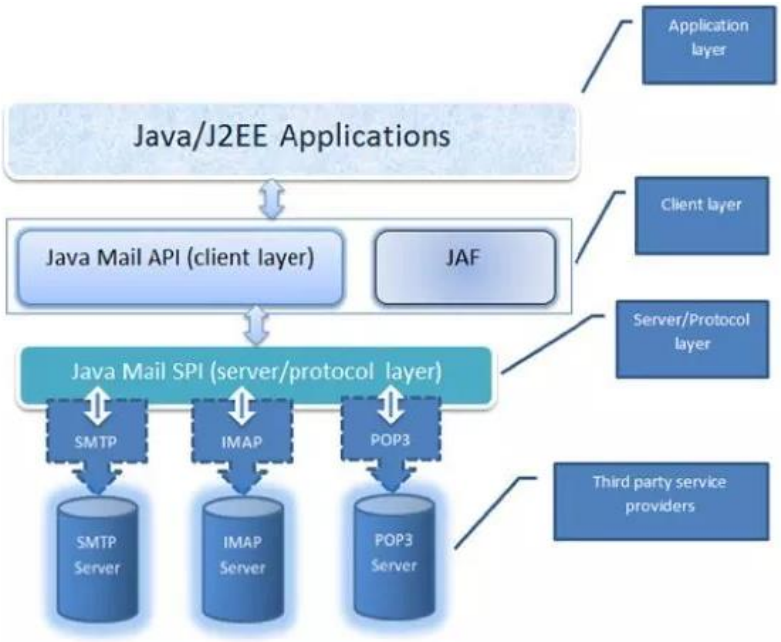
SMTP is an acronym for Simple Mail Transfer Protocol. It provides a mechanism to deliver the email. We can use Apache James server, Postcast server, cmail server etc. as an SMTP server. But if we purchase the host space, an SMTP server is bydefault provided by the host provider. For example, my smtp server is mail.javatpoint.com. If we use the SMTP server provided by the host provider, authentication is required for sending and receiving emails.

- MIME:-

Multiple Internet Mail Extension (MIME) tells the browser what is being sent e.g. attachment, format of the messages etc. It is not known as mail transfer protocol but it is used by your mail program.

- JavaMail Architecture

The java application uses JavaMail API to compose, send and receive emails. The JavaMail API uses SPI (Service Provider Interfaces) that provides the intermediatory services to the java application to deal with the different protocols. Let's understand it with the figure given below:

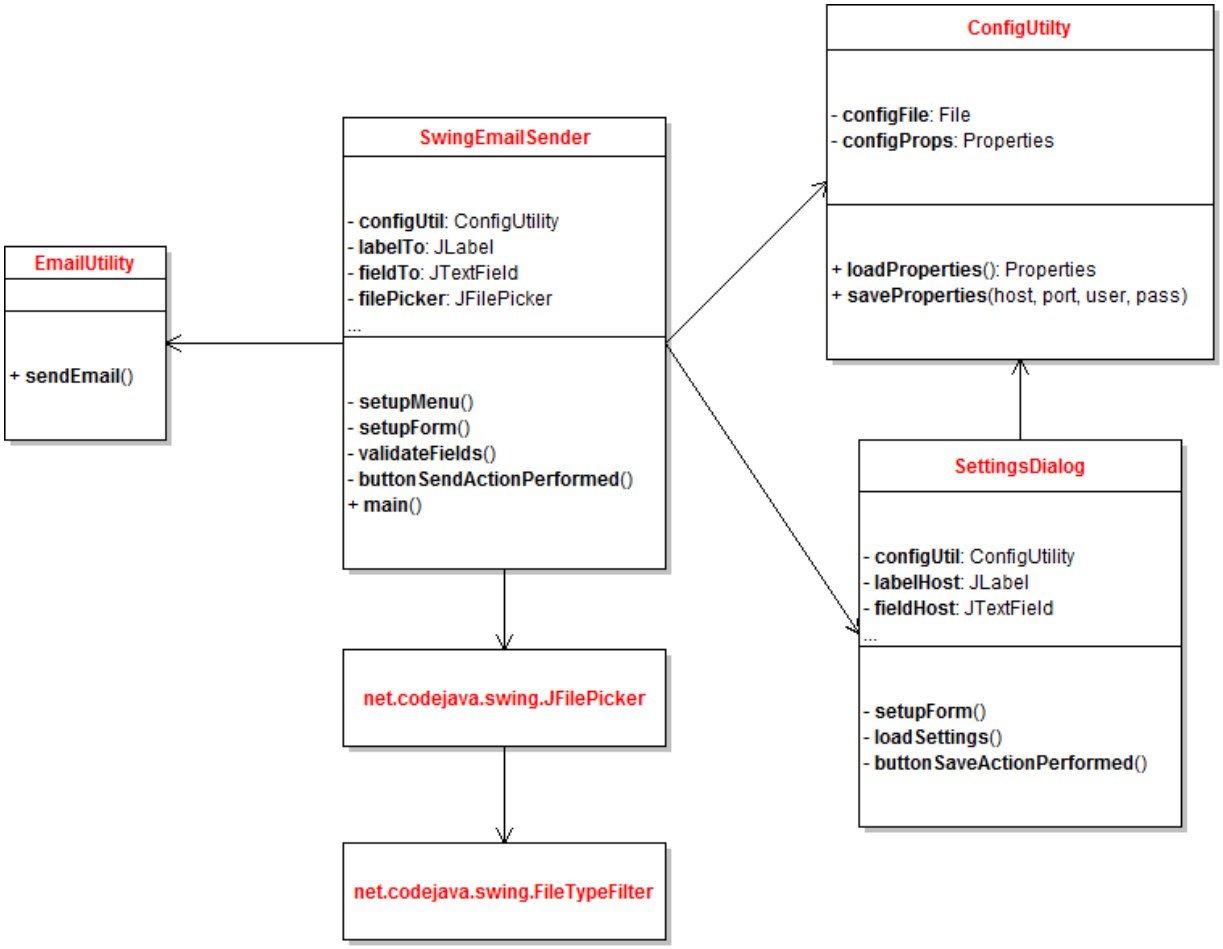


JavaMail API Core Classes

There are two packages that are used in Java Mail API: javax.mail and javax.mail.internet package. These packages contains many classes for Java Mail API. They are:

- javax.mail.Session class
- javax.mail.Message class
- javax.mail.internet.MimeMessage class
- javax.mail.Address class
- javax.mail.internet.InternetAddress class
- javax.mail.Authenticator class
- javax.mail.PasswordAuthentication class
- javax.mail.Transport class
- javax.mail.Store class
- javax.mail.Folder class etc.

Class Diagram



Program:

```
package com.rms.send_email;


import javax.swing.*;

import java.awt.*;

import java.awt.event.*;

import java.io.File;

import java.util.Properties;

import javax.mail.Authenticator;

import javax.mail.Message;

import javax.mail.PasswordAuthentication;

import javax.mail.Session;

import javax.mail.Transport;

import javax.mail.internet.InternetAddress;

import javax.mail.internet.MimeBodyPart;

import javax.mail.internet.MimeMessage;

import javax.mail.internet.MimeMultipart;


public class App extends JFrame implements ActionListener      {

        JButton send, reset;

        JOptionPane pop = new JOptionPane();

        JFrame frame = new JFrame("Email Sender Application");

        JLabel label = new JLabel("Email Sender Application");

        JLabel to = new JLabel("To: ");

        JLabel subject= new JLabel("Subject: ");

        JLabel file = new JLabel("Attachment: ");

        JLabel msg = new JLabel("Message: ");

        JLabel from = new JLabel("From: ");

        JLabel password = new JLabel("Password: ");
```

```
        JTextField textTo = new JTextField();
        JTextField textSubject = new JTextField();
        JTextField textFile = new JTextField();
        JTextField textFrom = new JTextField();
        JTextArea textMsg = new JTextArea();
        JPasswordField textPassword = new JPasswordField();

        public App()    {
                String image = "C:\\Users\\Moin MN\\OneDrive\\Pictures\\Saved
Pictures\\avatar.jpg";
                Font myFontHead = new Font("Times New Roman",Font.PLAIN,24);
                Font myFont = new Font("Times New Roman",Font.PLAIN,20);
                Font myFontText = new Font("Times New Roman",Font.PLAIN,16);
                frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
                frame.setSize(820, 690);          //display window size
                frame.setLayout(null);
                ImageIcon icon = new ImageIcon(image);
                frame.setLocationRelativeTo(null);
                frame.setIconImage(icon.getImage());
                //adding heading
                label.setFont(myFontHead);
                label.setBounds(275, 20, 300, 50);
                frame.add(label);
                //"From: " text
                from.setFont(myFont);
                from.setLayout(null);
                from.setBounds(40, 100, 150, 50);
                frame.add(from);
                //From textField
```

```
frame.setLayout(null);
textFrom.setBounds(165, 100, 500, 40);
textFrom.setFont(myFontText);
frame.add(textFrom);
//"password: " text
password.setFont(myFont);
password.setLayout(null);
password.setBounds(40, 150, 150, 50);
frame.add(password);
//password text field
frame.setLayout(null);
textPassword.setBounds(165, 150, 500, 40);
textPassword.setFont(myFontText);
frame.add(textPassword);
//to text
to.setFont(myFont);
to.setLayout(null);
to.setBounds(40, 200, 150, 50);
frame.add(to);
//to text field
frame.setLayout(null);
textTo.setBounds(165, 200, 500, 40);
textTo.setFont(myFontText);
frame.add(textTo);
//subject text
subject.setFont(myFont);
subject.setLayout(null);
subject.setBounds(40, 250, 150, 50);
frame.add(subject);
```

```
//subject text field
frame.setLayout(null);
textSubject.setBounds(165, 250, 500, 40);
textSubject.setFont(myFontText);
frame.add(textSubject);
//attachment text
file.setFont(myFont);
file.setLayout(null);
file.setBounds(40, 300, 150, 50);
frame.add(file);
//attachment textfield
textFile.setBounds(165, 300, 500, 40);
textFile.setLayout(null);
textFile.setFont(myFontText);
frame.add(textFile);
//message text
msg.setFont(myFont);
msg.setLayout(null);
msg.setBounds(40, 350, 150, 50);
frame.add(msg);
//message text area
textMsg.setBounds(165, 350, 500, 200);
textMsg.setLineWrap(true);
textMsg.setFont(myFontText);
frame.add(textMsg);
//submit button
send = new JButton("Send");
send.setBounds(165, 590, 90, 35);
frame.add(send);
```

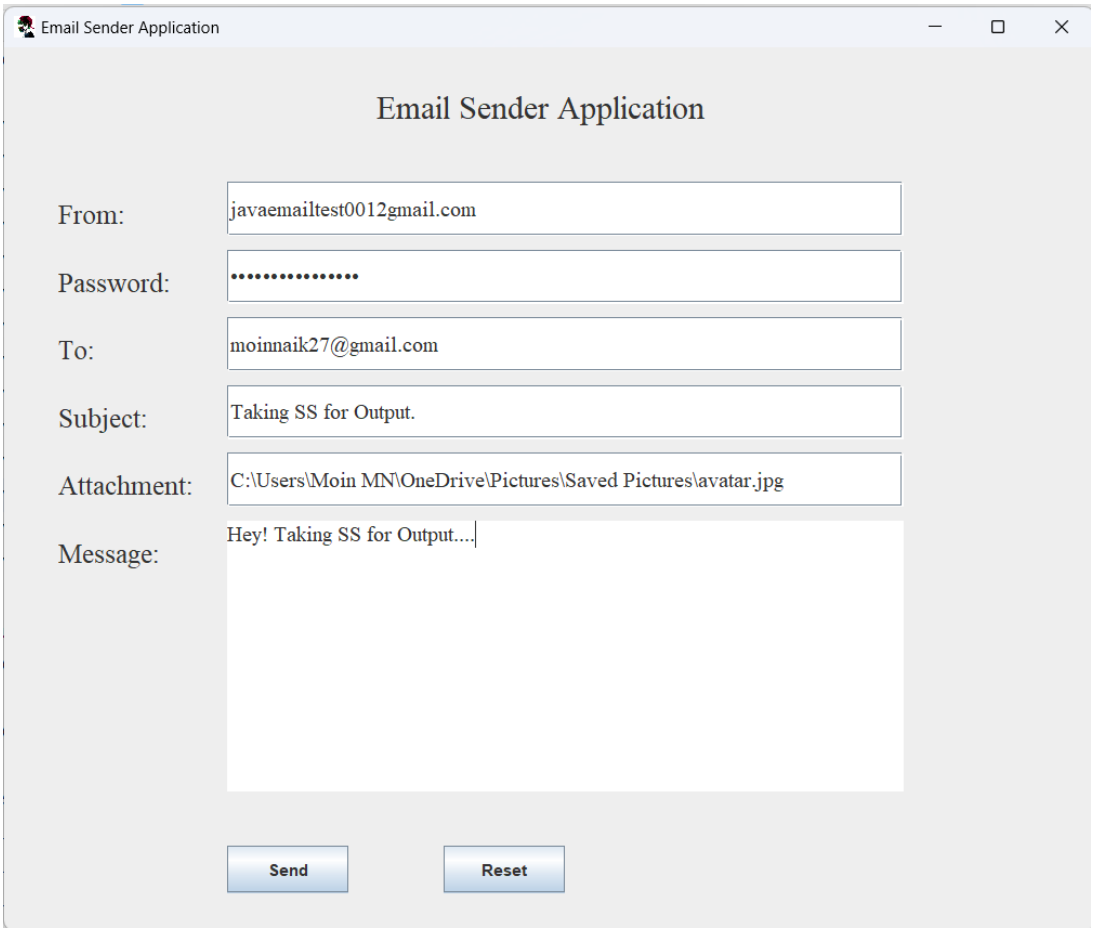
```
send.addActionListener(this);
//reset button
reset = new JButton("Reset");
reset.setBounds(325, 590, 90, 35);
frame.add(reset);
reset.addActionListener(this);
frame.setVisible(true);
//when reset button is clicked
reset.addActionListener(new ActionListener() {
        public void actionPerformed(ActionEvent e) {
                textTo.setText("");
                textMsg.setText("");
                textSubject.setText("");
                textFile.setText("");
                textFrom.setText("");
                textPassword.setText("");
        }
});
//when send button is clicked
send.addActionListener(new ActionListener() {
        public void actionPerformed(ActionEvent e)      {
                //reading input from text field
                String to = textTo.getText();
                String subject = textSubject.getText();
                String message = textMsg.getText();
                String path = textFile.getText();
                String from = textFrom.getText();
                String password = textPassword.getText();
                //sending info sendAttach method
```

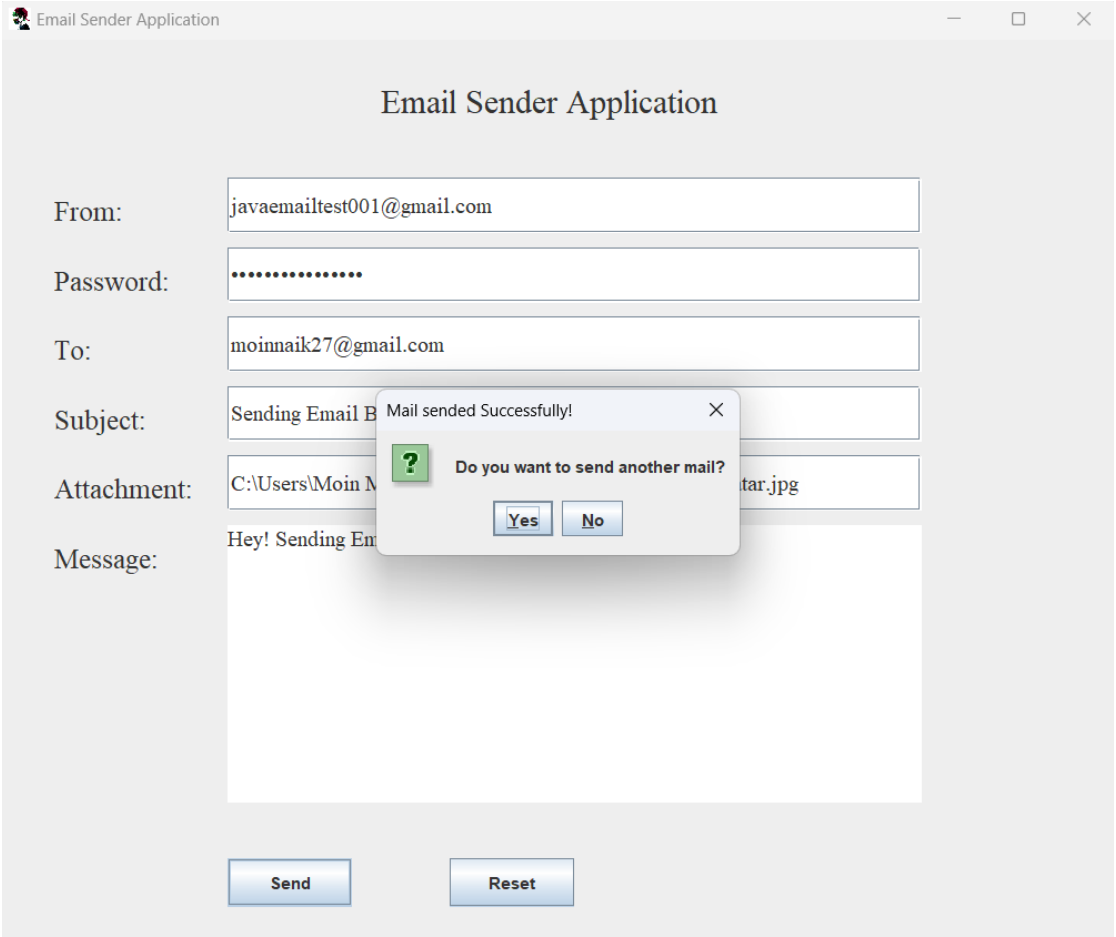
```
                sendAttach(message, subject, to, from, path, password);
            }
            private void sendAttach(String message, String subject, String to, final String
from, String path, final String password) {
                //variable for mails
                String host = "smtp.gmail.com";
                //get the system properties
                Properties properties = System.getProperties();
                //setting important to properties object
                //host set
                properties.put("mail.smtp.host", host);
                properties.put("mail.smtp.port", "465");
                properties.put("mail.smtp.ssl.enable", "true");
                properties.put("mail.smtp.auth", "true");
                //to get the session object...

                Session session = Session.getInstance(properties, new
Authenticator() {
                    @Override
                    protected PasswordAuthentication
getPasswordAuthentication() {
                        return new PasswordAuthentication(from,
password);
                    }
                });
                //session.setDebug(true);          to display the debuging
                MimeMessage m = new MimeMessage(session);
                try     {
                    //from mail
                    m.setFrom(from);
                    //adding recipient to message
```

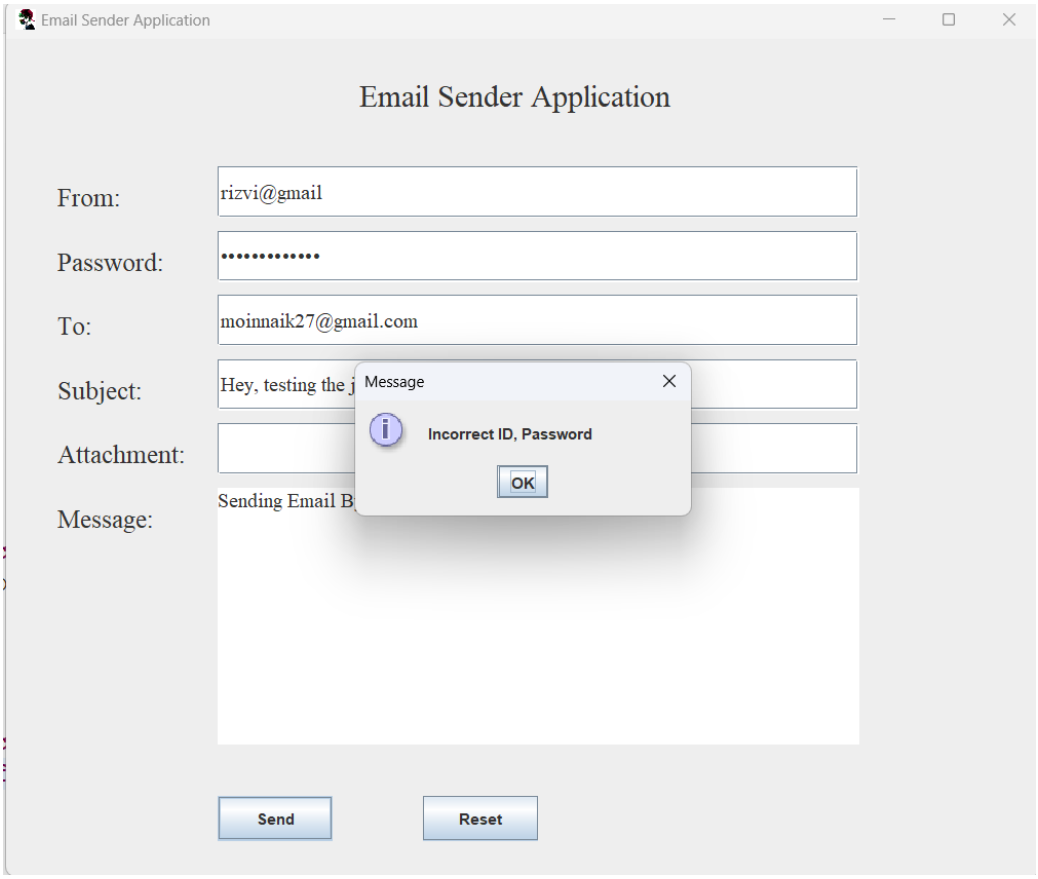
```
                m.addRecipient(Message.RecipientType.TO, new
InternetAddress(to));
                //adding subject to message
                m.setSubject(subject);
                //adding text to message
                m.setText(message);
                MimeMultipart mimeMultipart = new MimeMultipart();
                MimeBodyPart textMime = new MimeBodyPart();
                MimeBodyPart fileMime = new MimeBodyPart();
                try     {
                        textMime.setText(message);      //adding message
                        File file = new File(path);
                        fileMime.attachFile(file);
                        mimeMultipart.addBodyPart(textMime);
                        mimeMultipart.addBodyPart(fileMime);
                }       catch(Exception e)      {
                        JOptionPane.showMessageDialog(null, "Wrong
details entered!");
                }
                if(path.isBlank())          {
                        //checking weather attachment is enter or not
                }
                else
                        m.setContent(mimeMultipart);
                //send message using transport class
                Transport.send(m);
                //pop up
                int response = JOptionPane.showConfirmDialog(null, "Do
you want to send another mail?", "Mail sended Successfully!",
                        JOptionPane.YES_NO_OPTION,
JOptionPane.QUESTION_MESSAGE);
```

```
                                        if (response == JOptionPane.NO_OPTION) {
                                            System.exit(0);
                                        } else if (response == JOptionPane.YES_OPTION) {
                                            textTo.setText("");
                                            textSubject.setText("");
                                            textMsg.setText("");
                                            textFile.setText("");
                                            textFrom.setText("");
                                            textPassword.setText("");
                                        } else if (response == JOptionPane.CLOSED_OPTION) {
                                            System.exit(0);
                                        }
                                    } catch(Exception e) {
                                        JOptionPane.showMessageDialog(null, "Something went
wrong!");
                                    }
                                }
                            });
        }
    public static void main( String[] args ) {
        App a = new App();
    }
        public void actionPerformed(ActionEvent e) {
            if(textSubject.getText().isEmpty() || textMsg.getText().isEmpty() ||
textTo.getText().isEmpty() || textFrom.getText().isEmpty() || textPassword.getText().isEmpty())
                JOptionPane.showMessageDialog(null, "The Field is Empty.");
        }
}
```

Output:


Email Sender Application
Email Sender Application
From:
javaemailtest0012gmail.com
Password:
Password:
To:
moinnaik27@gmail.com
Subject:
Taking SS for Output.
Attachment:
C:\Users\Moin MN\OneDrive\Pictures\Saved Pictures\avatar.jpg
Message:
Hey! Taking SS for Output....
Send
Reset


Email Sender Application
Email Sender Application
From:
javaemailtest001@gmail.com
Password:
Password:
To:
moinnaik27@gmail.com
Subject:
Sending Email B
Attachment:
C:\Users\Moin M
tar.jpg
Message:
Hey! Sending Em
Mail sended Successfully!
Do you want to send another mail?
Yes
No
Send
Reset



References:

- https://www.codejava.net/coding/swing-application-for-sending-e-mail-with-attachments
- https://kodejava.org/how-do-i-create-a-simple-mail-client-program-in-swing/
- https://www.youtube.com/watch?v=l0J-Edn76js
- https://www.tutorialspoint.com/java/java_sending_email.htm