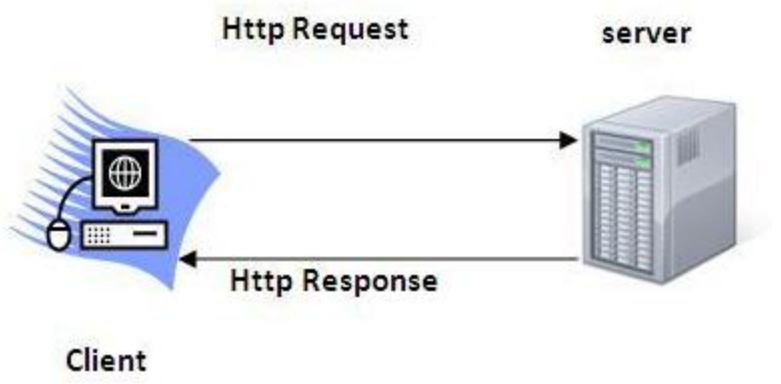
UNIT II JAVA NETWORKING FUNDAMENTALS

- Overview of Java Networking
  - TCP
  - UDP
  - InetAddress and Ports
- Socket Programming
- Working with URLs
- Internet Protocols simulation
  - HTTP
  - SMTP
  - POP
  - FTP
- Remote Method Invocation
- Multithreading Concepts.

Presented by,
B.Vijayalakshmi
Computer Centre
MIT Campus
Anna University

HTTP (Hyper Text Transfer Protocol)

HTTP (Hyper Text Transfer Protocol)



- The Hypertext Transfer Protocol (HTTP) is application-level protocol for collaborative, distributed, hypermedia information systems.
- It is the data communication protocol used to establish communication between client and server.
- HTTP is TCP/IP based communication protocol, which is used to deliver the data like image files, query results, HTML files etc on the World Wide Web (WWW) with the default port is TCP 80.
- It provides the standardized way for computers to communicate with each other

HttpURLConnection class

- It is an abstract class directly extending from URLConnection class .
- It includes all the functionality of its parent class with additional HTTP specific features
- It is mainly used for interacting with web servers
- **Constructor:**
- protected HttpURLConnection(URL u):

  Parameters :

  - u - the url Constructs the httpurlconnection to specified URL.

HttpURLConnection Methods

- **getResponseCode() :** Used to retrieve the response status from server.
  - 1xx : Information
  - 2xx : Success
  - 3xx : Redirection
  - 4xx : Client error
  - 5xx : Server error
- **setRequestMethod() :** Used to set the request method. Default is GET.

  Parameters:

  method : Must be any one of the following- GET, POST, HEAD, OPTIONS, PUT, DELETE, TRACE

  Throws : ProtocolException - If the method is not valid for http.
- **getResponseMessage()** :Retrieves the response message.

  Syntax : **public String getResponseMessage()**
  - 200 – OK
  - 404 - NOT FOUND

HttpURLConnection Methods

- **getHeaderField() :** Returns the nth header field, or null if it does not exist.It overrides getHeaderField method of URLConnection class.

  Syntax : **public String getHeaderField(int n)**

  Parameters: n : Index of the header field.

- **setFixedLengthStreamingMode() :** Used to set the length of content written on outputstream if it is known in advance.

  Syntax : **public void setFixedLengthStreamingMode(long n/int n) throws IllegalStateException**

  Parameters: n : Length of content to be written.

  Throws: IllegalStateException : If specified length of content is not written on outputstream.

- **setFollowRedirects() :** Sets whether a 3xx response code requests be redirected automatically or not.

  Syntax : **public void setFollowRedirects(boolean b)**

  Parameters: b : Must be true or false.

HttpURLConnection Methods Cont’d

- **getFollowRedirects() :** Returns true or false depending on automatic redirection or not.

  Syntax **: public static boolean getFollowRedirects()**
- **disconnect() :** Indicated that requests to server are highly unlikely in future.

  Syntax : **public void disconnect()**
- **usingProxy() :** Returns true if connection established using a proxy, else false.

  Syntax : **public boolean usingProxy()**
- **setChunkedStreamingMode() :** This mode is used when the content length is not known. Instead of creating a buffer of fixed length and writing it to server, content is broken into chunks and then written. Not all servers support this mode.

  Syntax : **public void setChunkedStreamingMode(int n) throws IllegalStateException**

  Parameters: n : length written in each chunk.

  Throws: IllegalStateException : If some different streaming mode is enabled.

HttpURLConnection Methods Cont’d

- **getPermission() :** Retrieves the permission required to connect to destination host and port.

  Syntax : **public Permission getPermission()**

- **getErrorStream() :** Gets the error stream if the server cannot be connected or some error occurred. It can contain information about how to fix the error from server.

  Syntax : **public InputStream getErrorStream()**

- **setInstanceFollowRedirects() :** Sets whether a 3xx response code requests be redirected automatically by this instance of httpURLconnection. It overrides the more generic setFollowRedirects parameter. If not specified, then the instance redirects based on setFollowRedirects().

  Syntax : **public void setFollowRedirects(boolean b)**

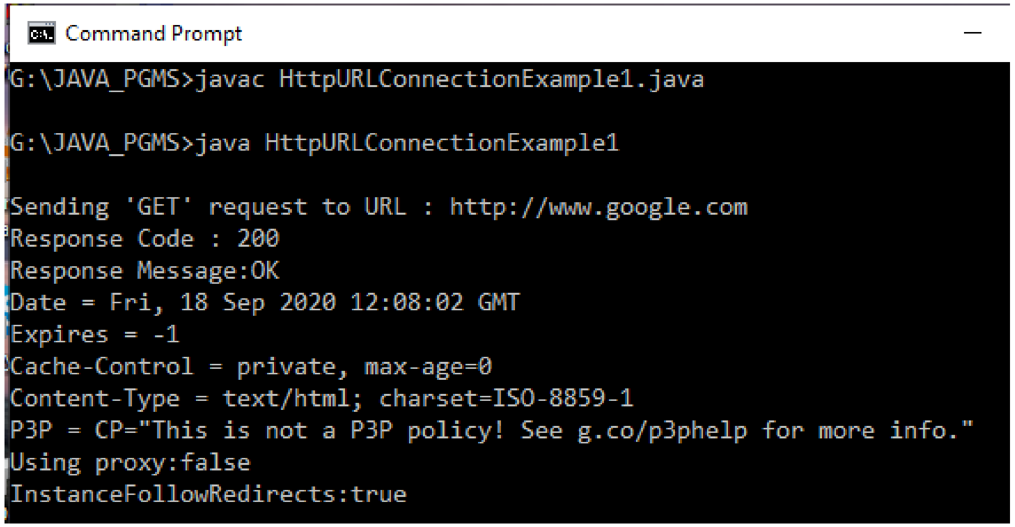
  Parameters**:** b : Must be true or false.

- **getInstanceFollowRedirects() :** Returns true or false depending on automatic instance redirection set or not.

  Syntax **: public boolean getFollowRedirects()**

```
import java.io.BufferedReader;
import java.io.DataOutputStream;
import java.io.InputStreamReader;
import java.net.HttpURLConnection;
import java.net.URL;

public class HttpURLConnectionExample1
{
    public static void main(String[] args) throws Exception
     {
            String url = "http://www.google.com";
            URL obj = new URL(url);
            HttpURLConnection con = (HttpURLConnection) obj.openConnection();

            // optional default is GET
            con.setRequestMethod("GET");

            int responseCode = con.getResponseCode();
            System.out.println("\nSending 'GET' request to URL : " + url);
            System.out.println("Response Code : " + responseCode);
            System.out.println("Response Message:"+ con.getResponseMessage());
```

```
  for(int i=1;i<=5;i++)
 {
            System.out.println(con.getHeaderFieldKey(i)+" = "+con.getHeaderField(i));
  }
 System.out.println("Using proxy:" + con.usingProxy());
 System.out.println("InstanceFollowRedirects:"+ con.getInstanceFollowRedirects());
/*BufferedReader in = new BufferedReader(new
InputStreamReader(con.getInputStream()));
 String inputLine;
 StringBuffer response = new StringBuffer();
 while ((inputLine = in.readLine()) != null)
 {
            response.append(inputLine);
  }
 in.close();*/
//print result
//System.out.println(response.toString());
            }

}
```



Command Prompt

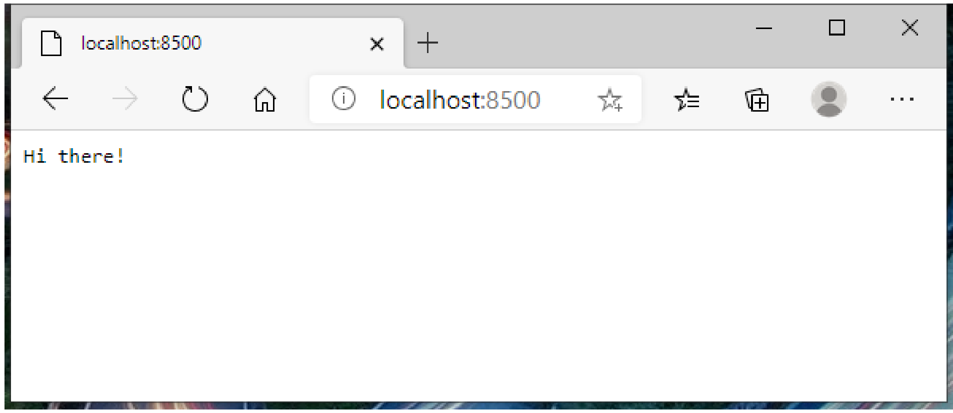
```
G:\JAVA_PGMS>javac HttpURLConnectionExample1.java

G:\JAVA_PGMS>java HttpURLConnectionExample1

Sending 'GET' request to URL : http://www.google.com
Response Code : 200
Response Message:OK
Date = Fri, 18 Sep 2020 12:08:02 GMT
Expires = -1
Cache-Control = private, max-age=0
Content-Type = text/html; charset=ISO-8859-1
P3P = CP="This is not a P3P policy! See g.co/p3phelp for more info."
Using proxy:false
InstanceFollowRedirects:true
```

HttpServer

com.sun.net.httpserver.HttpServer

- This class implements a simple HTTP server.
- A HttpServer is bound to an IP address and port number and listens for incoming TCP connections from clients on this address.
- The sub-class HttpsServer implements a server which handles HTTPS requests.
- **HttpServer**
- This class implements a simple HTTP server. It has factory methods create() to create its instance. We need to bind the server to an IP address and port number while initializing.
- **HttpContext**
- It represents a mapping between the root URI path to a HttpHandler.
- **HttpHandler**
- It is an interface, which needs to be implemented by the application to handle the Http requests. It has only one method:
- void handle(HttpExchange exchange) throws IOException
- **HttpExchange**
- An instance of this class is passed to HttpHandler#handle(). It has methods to access Http request information, and to prepare and send the response.

```
import com.sun.net.httpserver.HttpContext;
import com.sun.net.httpserver.HttpExchange;
import com.sun.net.httpserver.HttpServer;
import java.io.IOException;
import java.io.OutputStream;
import java.net.InetSocketAddress;
public class BasicHttpServerExample
{
 public static void main(String[] args) throws IOException
  {
    HttpServer server = HttpServer.create(new InetSocketAddress(8500), 0);
    HttpContext context = server.createContext("/");
    context.setHandler(BasicHttpServerExample::handleRequest);
    server.start();
 }
 private static void handleRequest(HttpExchange exchange) throws IOException
 {
    String response = "Hi there!";
   //response code and length
    exchange.sendResponseHeaders(200, response.getBytes().length);
    OutputStream os = exchange.getResponseBody();
    os.write(response.getBytes());
    os.close();
 }
}
```



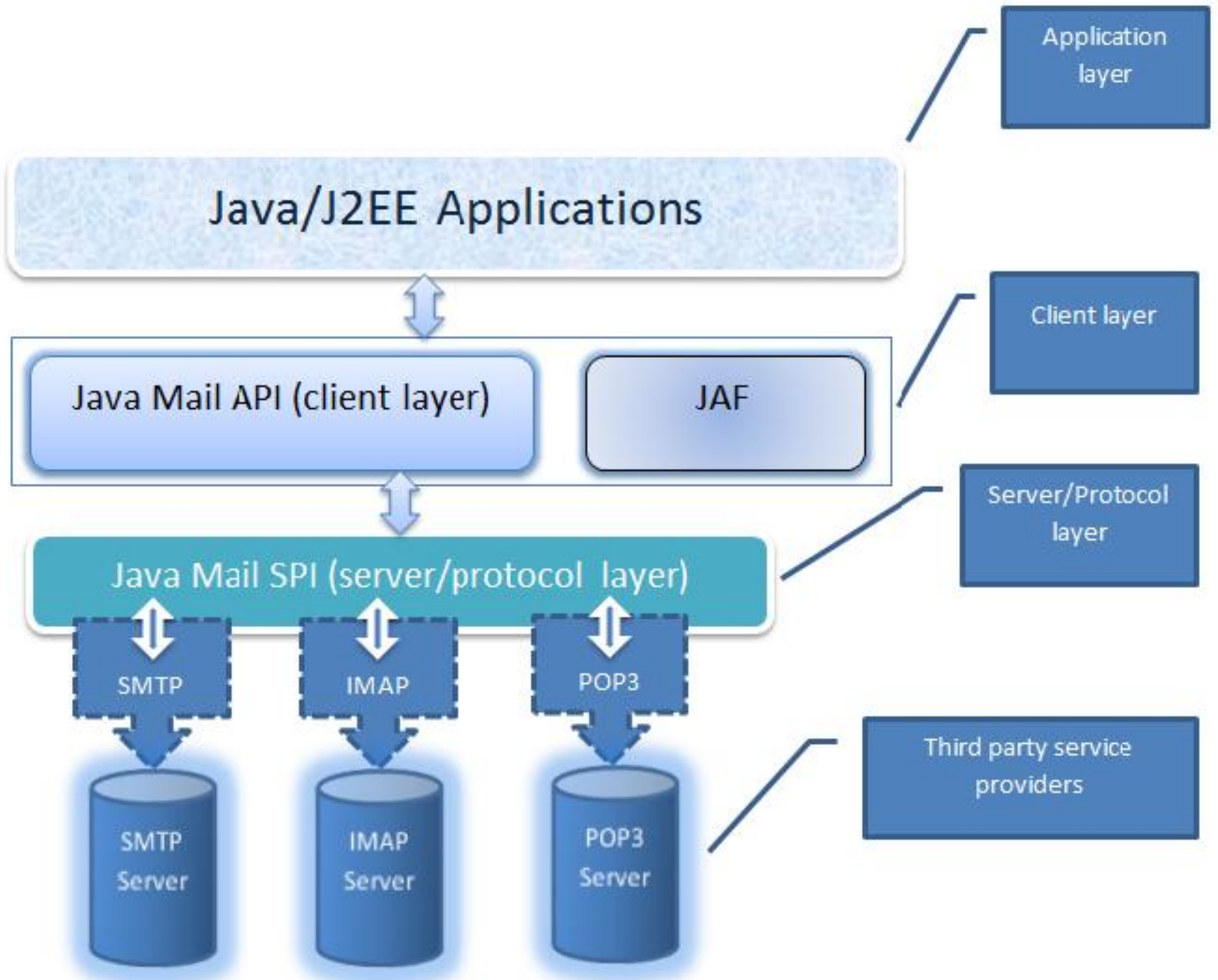
```
Hi there!
```

Simple Mail Transfer Protocol (SMTP)

Send email using Java Program

- 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,
  - JavaMail API
  - JavaBeans Activation Framework (JAF)
- Add **mail.jar** and **activation.jar** files in our CLASSPATH.

JavaMail architecture



- **Java Mail APIs:** These are the Java interfaces to send and receive e-mails. This layer is completely independent of the underlying protocols.
- **JavaBeans Activation Framework (JAF):** This framework is used to manage mail contents like URL, attachments, mail extensions etc.
- **Service Provider Interfaces(SPI):** This layer sits between the protocol implementers and the Java applications, more specifically Java mail APIs. SPIs understand the protocol languages and hence create the bridge between the two sides.
- **Service protocol implementers:** These are the third party service providers who implements different protocols like ***SMTP, POP3*** and ***IMAP*** etc.

JavaMail API - Overview

- This API provides a platform-independent and protocol-independent framework to build mail and messaging applications.
- It provides a set of abstract classes defining objects that comprise a mail system
- It provides elements that are used to construct an interface to a messaging system, including system components and interfaces.
- Following are some of the protocols supported in JavaMail API:
  - **SMTP** (Simple Mail Transfer Protocol)
  - **POP** (Post Office Protocol)
  - **IMAP** (Internet Message Access Protocol)
  - **MIME** (Multipurpose Internet Mail Extensions)
  - **NNTP** (Network News Transfer Protocol)

JavaMail API Cont'd

- **SMTP**: Acronym for **Simple Mail Transfer Protocol**. It provides a mechanism to deliver email.
- **POP**: Acronym for **Post Office Protocol**. POP is the mechanism most people on the Internet use to get their mail. It defines support for a single mailbox for each user. RFC 1939 defines this protocol.
- **IMAP**: Acronym for **Internet Message Access Protocol**. It is an advanced protocol for receiving messages. It provides support for multiple mailbox for each user, in addition to, mailbox can be shared by multiple users. It is defined in RFC 2060.
- **MIME**: Acronym for **Multipurpose Internet Mail Extensions**. . It is not a mail transfer protocol. Instead, it defines the content of what is transferred: the format of the messages, attachments, and so on. There are many different documents that take effect here: RFC 822, RFC 2045, RFC 2046, and RFC 2047. As a user of the JavaMail API, we usually don't need to worry about these formats. However, these formats do exist and are used by our programs.
- **NNTP and Others**: There are many protocols that are provided by third-party providers. Some of them are Network News Transfer Protocol (NNTP), Secure Multipurpose Internet Mail Extensions (S/MIME) etc.

javaMail API Core Classes

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

SMTP server

- To send emails, we must have SMTP server that is responsible to send mails.
- We can use one of the following techniques to get the SMTP server:
  - Install and use any SMTP server such as Postfix server (for Ubuntu), Apache James server (Java Apache Mail Enterprise Server)etc. (or)
  - Use the SMTP server provided by the host provider for eg: free SMTP provide by JangoSMTP site is *relay.jangosmtp.net* (or)
  - Use the SMTP Server provided by companies e.g. gmail, yahoo, etc

Steps to send email using JavaMail API (SMTP)

- There are following three steps to send email using JavaMail. They are,
  - **Get the session object** that stores all the information of host like host name, username, password etc.
  - **compose the message**
  - **send the message**

Get the session object (javax.mail.Session)

- The *Session* class is the primary class of the JavaMail API and it is not subclassed.
- The *Session* object acts as the connection factory for the JavaMail API, which handles both configuration setting and authentication.
- *Session* object can be created in the following ways:
  - By looking up the administered object stored in the JNDI (Java Naming and Directory Interface) service.
  - **Programmatic approach in which we can use a *java.util.Properties* object to override some of the default information, such as the mail server name, username, password, and other information that can be shared across our entire application.**

- The **javax.mail.Session class** provides two methods to get the object of session,
  - Session.getDefaultInstance()
  - Session.getInstance().

You can use any method to get the session object.

Method of Session class

No	Method	Description
1	public static Session getDefaultInstance(Properties props)	returns the default session.
2	public static Session getDefaultInstance(Properties props,Authenticator auth)	returns the default session.
3	public static Session getInstance(Properties props)	returns the new session.
4	public static Session getInstance(Properties props,Authenticator auth)	returns the new session.

Example of getDefaultInstance() method

```
Properties properties=new Properties();
//fill all the information like host name etc.
Session session=Session.getDefaultInstance(properties,null);

PasswordAuthentication   auth=new
   PasswordAuthentication(“name”,”psw”)
Session sess=Session.getDefaultInstance(props,auth)
```

Example of getInstance() method

```
Properties properties=new Properties();
//fill all the information like host name etc.
Session session=Session.getInstance(properties,null);
```

Compose the message (javax.mail.Message)

- This class provides methods to compose the message.
- But it is an abstract class so its subclass javax.mail.internet.MimeMessage class is mostly used.
- To create the message, we need to pass session object in MimeMessage class constructor.

  For example:

  ```
  MimeMessage message=new MimeMessage(session);
  ```

- Now message object has been created but to store information in this object MimeMessage class provides many methods.

Commonly used methods of MimeMessage class

No	Method	Description
1	public void setFrom(Address address)	is used to set the from header field.
2	public void addRecipient(Message.RecipientType type, Address address)	is used to add the given address to the recipient type.
3	public void addRecipients(Message.RecipientType type, Address[] addresses)	is used to add the given addresses to the recipient type.
4	public void setSubject(String subject)	is used to set the subject header field.
5	public void setText(String textmessage)	is used to set the text as the message content using text/plain MIME type.
6	public void setContent(Object msg, String contentType)	is used to set the content as the message content using given MIME type.

Example to compose the message:

```
MimeMessage message=new MimeMessage(session);
message.setFrom(new InternetAddress(“emailID"));
message.addRecipient(Message.RecipientType.To,
new InternetAddress(“Receiver ID"));
message.setHeader("Hi, everyone");

message.setText("Hi, This mail is to inform you...");
```

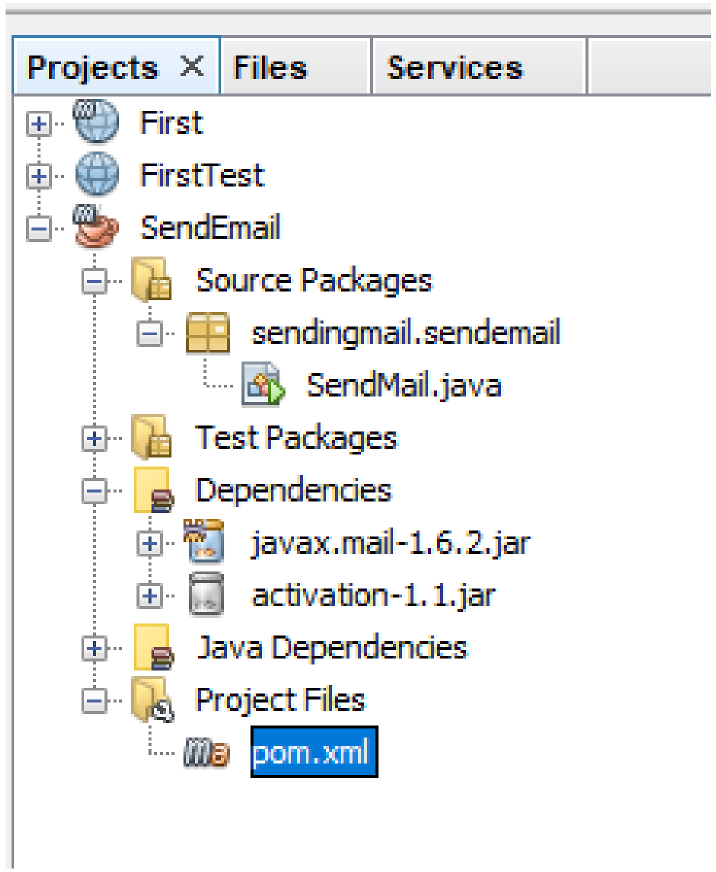
Send the message (javax.mail.Transport)

- It is used as a message transport mechanism.
- This class normally uses the SMTP protocol to send a message.
- It is an abstract class.
- **Commonly used methods of Transport class**

No.	Method	Description
1	public static void send(Message message)	is used send the message.
2	public static void send(Message message, Address[] address)	is used send the message to the given addresses.

Transport.send(message);

Project tab in IDE

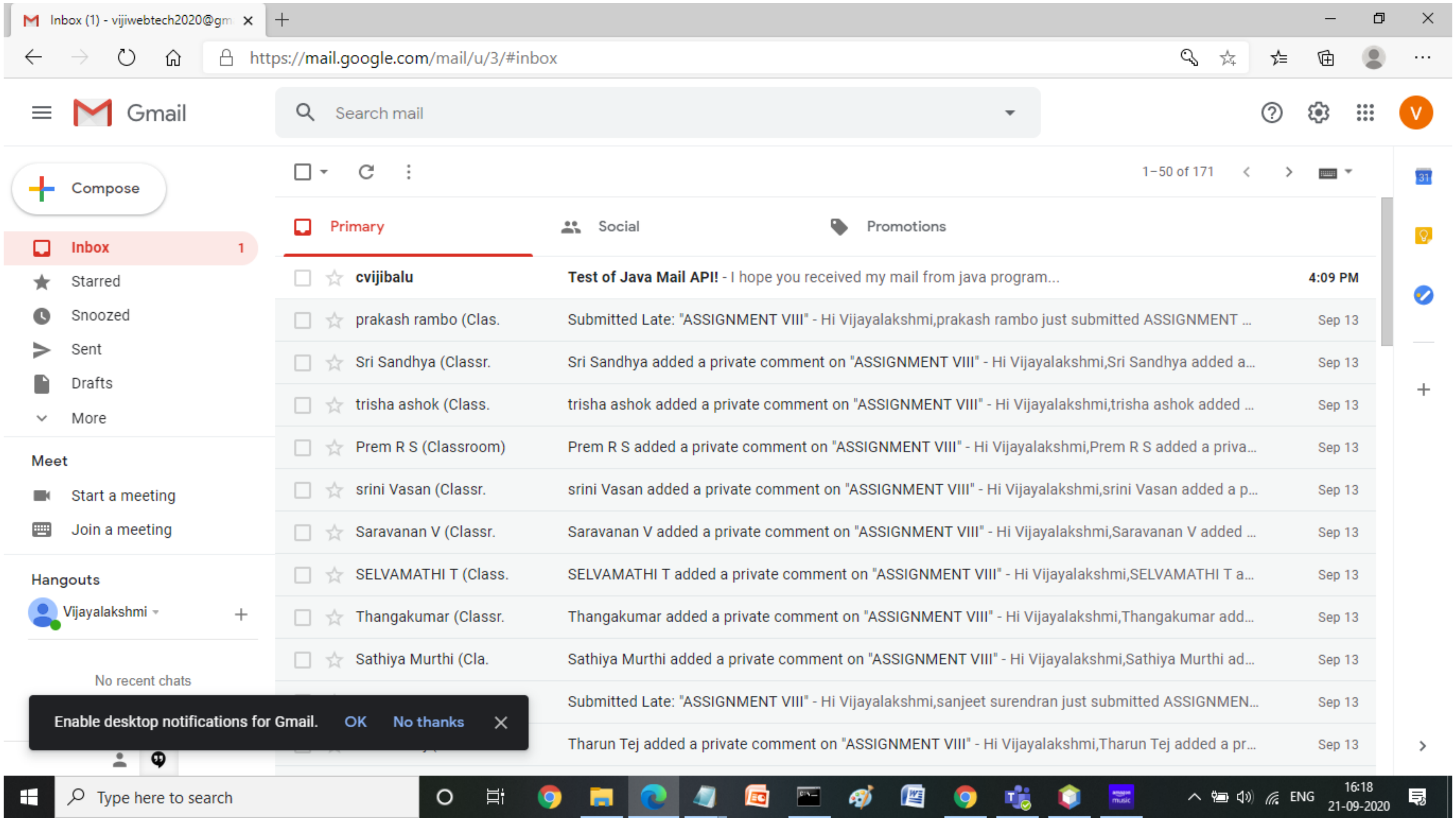


```
First
FirstTest
SendEmail
    Source Packages
        sendingmail.sendemail
            SendMail.java
    Test Packages
    Dependencies
        javax.mail-1.6.2.jar
        activation-1.1.jar
    Java Dependencies
    Project Files
        pom.xml
```

pom.xml

```
<?xml version="1.0" encoding="UTF-8"?>
<project xmlns="http://maven.apache.org/POM/4.0.0"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:schemaLocation="http://maven.apache.org/POM/4.0.0
http://maven.apache.org/xsd/maven-4.0.0.xsd">
  <modelVersion>4.0.0</modelVersion>
  <groupId>sendingMail</groupId>
  <artifactId>SendEmail</artifactId>
  <version>1.0-SNAPSHOT</version>
  <packaging>jar</packaging>
  <properties>
    <project.build.sourceEncoding>UTF-8</project.build.sourceEncoding>
    <maven.compiler.source>14</maven.compiler.source>
    <maven.compiler.target>14</maven.compiler.target>
  </properties>
  <dependencies>
  <dependency>
      <groupId>com.sun.mail</groupId>
      <artifactId>javax.mail</artifactId>
      <version>1.6.2</version>
    </dependency>
  </dependencies>
</project>
```

Send a mail from Java Application



Inbox (1) - vijiwebtech2020@gm

https://mail.google.com/mail/u/3/#inbox

Gmail

Search mail

Compose

Inbox 1

Starred

Snoozed

Sent

Drafts

More

Meet

Start a meeting

Join a meeting

Hangouts

Vijayalakshmi

No recent chats

1–50 of 171

Primary | Social | Promotions

cvijibalu	**Test of Java Mail API!** - I hope you received my mail from java program...	4:09 PM
prakash rambo (Clas.	Submitted Late: "ASSIGNMENT VIII" - Hi Vijayalakshmi,prakash rambo just submitted ASSIGNMENT ...	Sep 13
Sri Sandhya (Classr.	Sri Sandhya added a private comment on "ASSIGNMENT VIII" - Hi Vijayalakshmi,Sri Sandhya added a...	Sep 13
trisha ashok (Class.	trisha ashok added a private comment on "ASSIGNMENT VIII" - Hi Vijayalakshmi,trisha ashok added ...	Sep 13
Prem R S (Classroom)	Prem R S added a private comment on "ASSIGNMENT VIII" - Hi Vijayalakshmi,Prem R S added a priva...	Sep 13
srini Vasan (Classr.	srini Vasan added a private comment on "ASSIGNMENT VIII" - Hi Vijayalakshmi,srini Vasan added a p...	Sep 13
Saravanan V (Classr.	Saravanan V added a private comment on "ASSIGNMENT VIII" - Hi Vijayalakshmi,Saravanan V added ...	Sep 13
SELVAMATHI T (Class.	SELVAMATHI T added a private comment on "ASSIGNMENT VIII" - Hi Vijayalakshmi,SELVAMATHI T a...	Sep 13
Thangakumar (Classr.	Thangakumar added a private comment on "ASSIGNMENT VIII" - Hi Vijayalakshmi,Thangakumar add...	Sep 13
Sathiya Murthi (Cla.	Sathiya Murthi added a private comment on "ASSIGNMENT VIII" - Hi Vijayalakshmi,Sathiya Murthi ad...	Sep 13
	Submitted Late: "ASSIGNMENT VIII" - Hi Vijayalakshmi,sanjeet surendran just submitted ASSIGNMEN...	Sep 13
	Tharun Tej added a private comment on "ASSIGNMENT VIII" - Hi Vijayalakshmi,Tharun Tej added a pr...	Sep 13

Enable desktop notifications for Gmail. OK No thanks

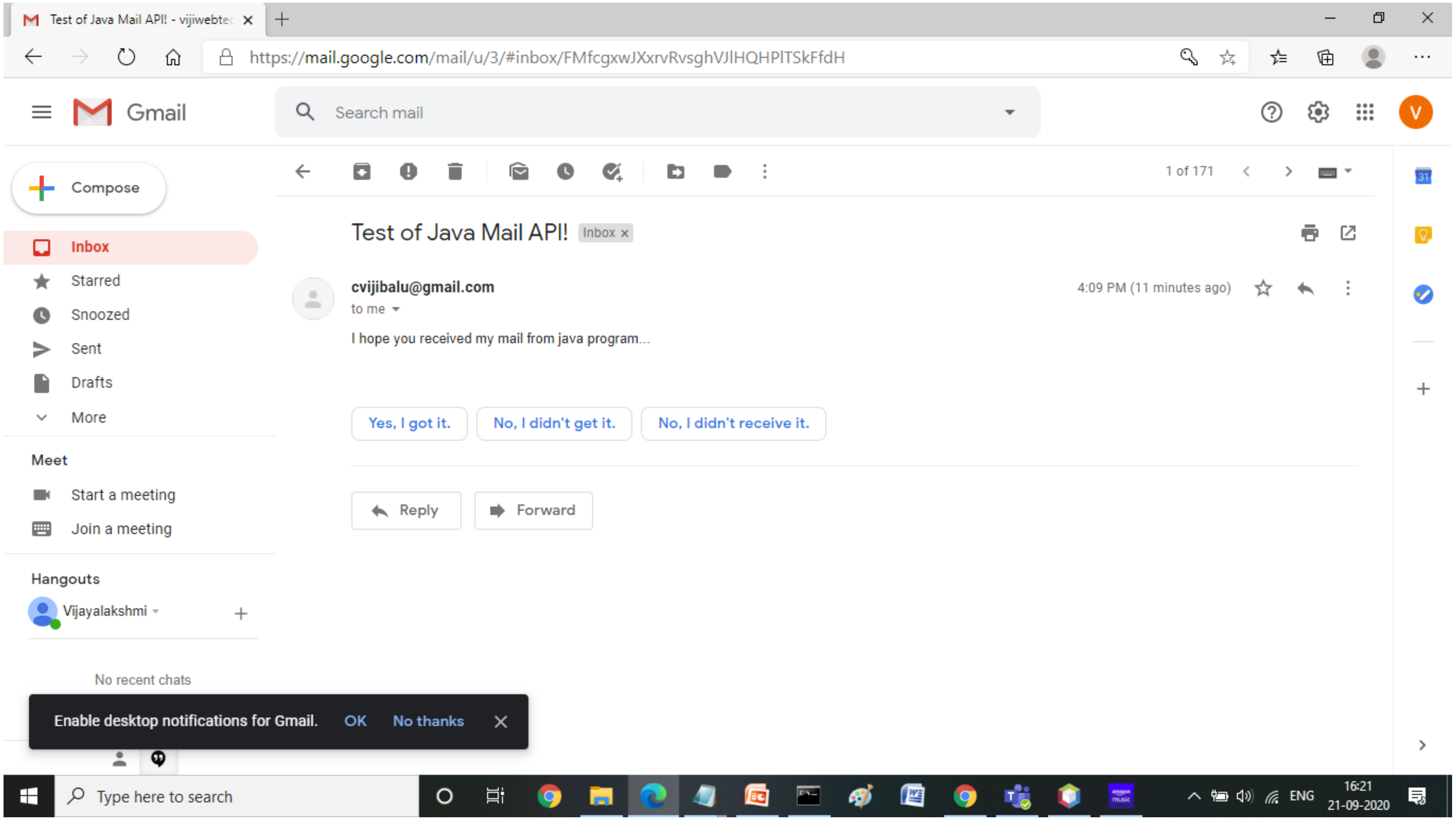
Type here to search

ENG 16:18 21-09-2020

```
import java.util.Properties;

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;

public class SendMail
{
   public static void main(String[] args)
   {
      // Recipient's email ID needs to be mentioned.
      String to = "Receivers email id";
      // Sender's email ID needs to be mentioned
      String from = "sender email id";
      // Assuming you are sending email from through gmails smtp
      String host = "smtp.gmail.com";
```

```
// Get system properties
      Properties properties = System.getProperties();

      // Setup mail server
      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");

// Get the Session object.// and pass username and password

      Session session = Session.getInstance(properties, new javax.mail.Authenticator() {
         protected PasswordAuthentication getPasswordAuthentication() {
            return new PasswordAuthentication(from, "*******"); // password of the
sender
         }
      });
```

```
// Used to debug SMTP issues
      session.setDebug(true);

      try {
        // Create a default MimeMessage object.
        MimeMessage message = new MimeMessage(session);
        // Set From: header field of the header.
        message.setFrom(new InternetAddress(from));
        // Set To: header field of the header.
        message.addRecipient(Message.RecipientType.TO, new InternetAddress(to));
        // Set Subject: header field
        message.setSubject("Test of Java Mail API!");
        // Now set the actual message
        message.setText("I hope you received my mail from java program...");
        System.out.println("sending...");
        // Send message
        Transport.send(message);
        System.out.println("Sent message successfully....");
      } catch (MessagingException mex) {
        mex.printStackTrace();
      }
    }

}
```



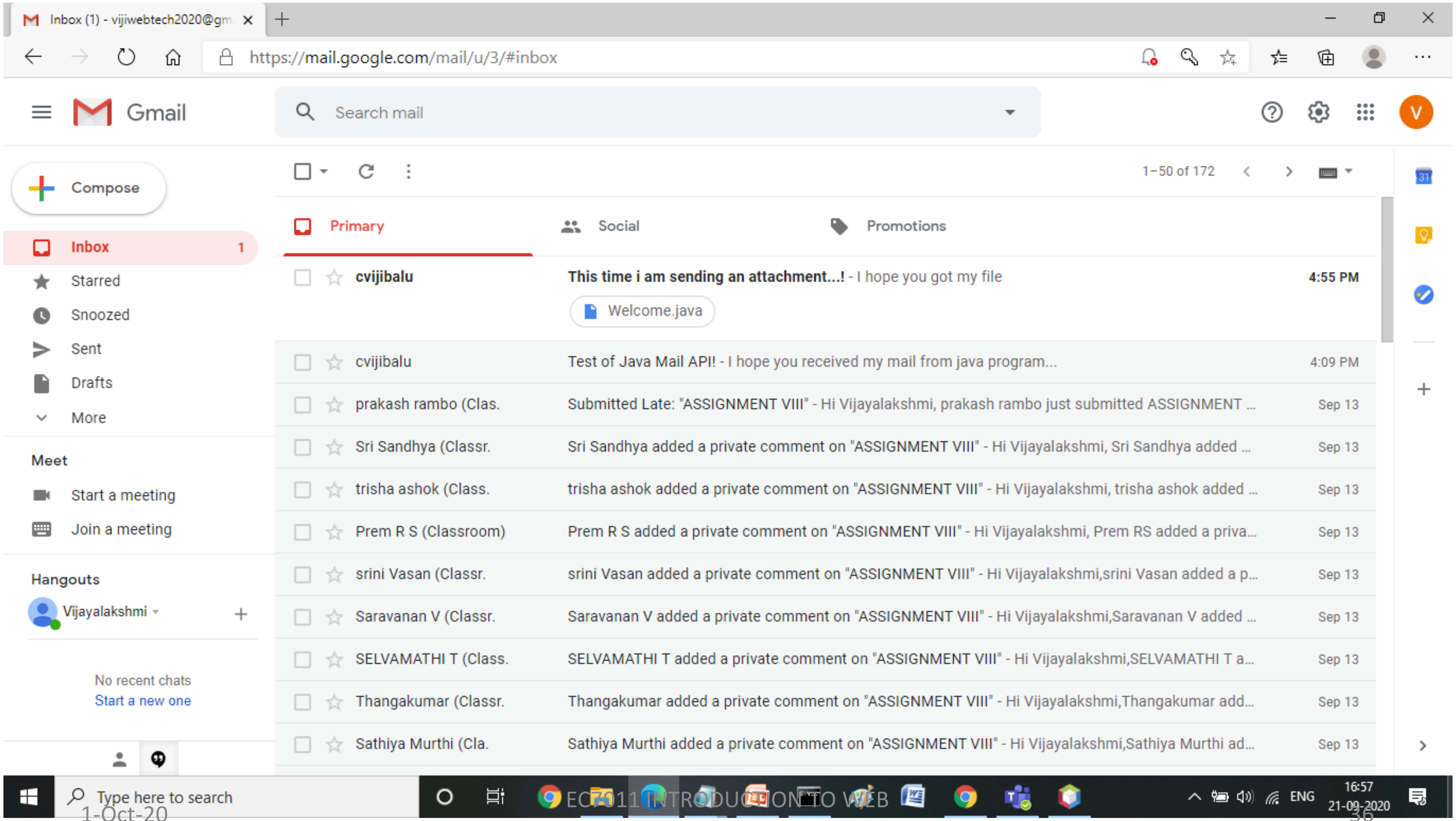
Test of Java Mail API!

cvijibalu@gmail.com
to me

4:09 PM (11 minutes ago)

I hope you received my mail from java program...

Send a mail with attachment



Inbox (1) - vijiwebtech2020@gm

https://mail.google.com/mail/u/3/#inbox

1–50 of 172

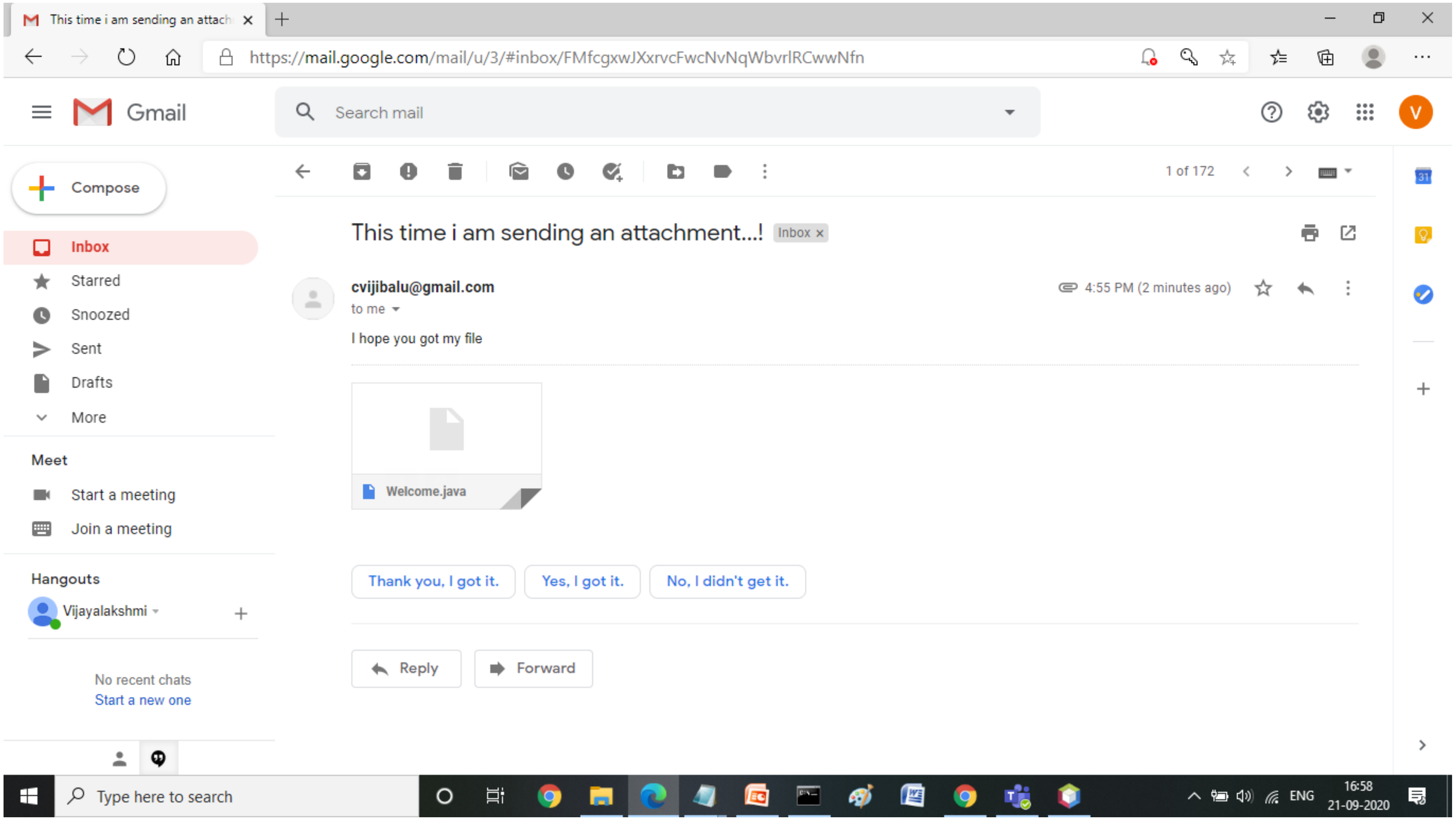
cvijibalu	This time i am sending an attachment...! - I hope you got my file Welcome.java	4:55 PM
cvijibalu	Test of Java Mail API! - I hope you received my mail from java program...	4:09 PM
prakash rambo (Clas.	Submitted Late: "ASSIGNMENT VIII" - Hi Vijayalakshmi, prakash rambo just submitted ASSIGNMENT ...	Sep 13
Sri Sandhya (Classr.	Sri Sandhya added a private comment on "ASSIGNMENT VIII" - Hi Vijayalakshmi, Sri Sandhya added ...	Sep 13
trisha ashok (Class.	trisha ashok added a private comment on "ASSIGNMENT VIII" - Hi Vijayalakshmi, trisha ashok added ...	Sep 13
Prem R S (Classroom)	Prem R S added a private comment on "ASSIGNMENT VIII" - Hi Vijayalakshmi, Prem RS added a priva...	Sep 13
srini Vasan (Classr.	srini Vasan added a private comment on "ASSIGNMENT VIII" - Hi Vijayalakshmi,srini Vasan added a p...	Sep 13
Saravanan V (Classr.	Saravanan V added a private comment on "ASSIGNMENT VIII" - Hi Vijayalakshmi,Saravanan V added ...	Sep 13
SELVAMATHI T (Class.	SELVAMATHI T added a private comment on "ASSIGNMENT VIII" - Hi Vijayalakshmi,SELVAMATHI T a...	Sep 13
Thangakumar (Classr.	Thangakumar added a private comment on "ASSIGNMENT VIII" - Hi Vijayalakshmi,Thangakumar add...	Sep 13
Sathiya Murthi (Cla.	Sathiya Murthi added a private comment on "ASSIGNMENT VIII" - Hi Vijayalakshmi,Sathiya Murthi ad...	Sep 13

To add attachments in your mail

- Add few more classes as given below…
  - ➢import java.io.File;
  - ➢import java.io.IOException;
  - ➢import javax.mail.internet.MimeBodyPart;
  - ➢import javax.mail.internet.MimeMessage;
  - ➢import javax.mail.internet.MimeMultipart;
- **Create Multipart message and send the message**

Mail with attachments

```
// Set Subject: header field
         message.setSubject("This time i am sending an attachment...!");
         MimeMultipart multipart = new MimeMultipart();
         MimeBodyPart attachmentPart = new MimeBodyPart();
         MimeBodyPart textPart = new MimeBodyPart();
         try {
            File f =new File("G:\\Java\\Welcome.java");
            attachmentPart.attachFile(f);
            textPart.setText("I hope you got my file");
            multipart.addBodyPart(textPart);
            multipart.addBodyPart(attachmentPart);

         } catch (IOException e) {
            e.printStackTrace();
         }
         message.setContent(multipart);
         System.out.println("sending...");
         // Send message
         Transport.send(message);
         System.out.println("Sent message successfully....");
      } catch (MessagingException mex) {
         mex.printStackTrace();
      }
```



This time i am sending an attachment...! Inbox

cvijibalu@gmail.com

to me

4:55 PM (2 minutes ago)

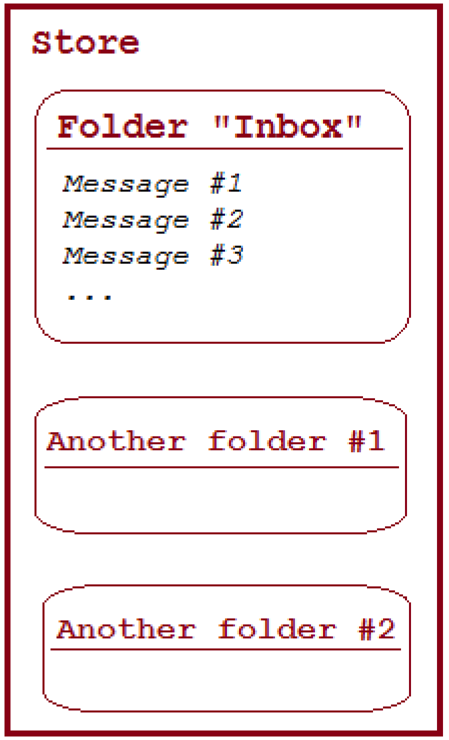
I hope you got my file

Welcome.java

Post Office Protocol (POP)

Receiving email in Java

Message stored in mail server



- The picture depicts how messages are logically stored on the server.
- we can see, for each user account, the server has a store which is the storage of user's messages.
- The store is divided into folders, and the "inbox" folder is the primarily folder which contains e-mail messages.
- A folder can contain both messages and sub-folders.

Receiving email in Java

- Speaking of JavaMail API's language, it provides **three corresponding classes: Store, Folder and Message.**
- A Store object can be obtained from the current session by invoking the method **getStore(String protocol)** of Session class.
- Connecting to the Store by calling its method **connect(String user, String pass),** disconnecting by calling its **close()** method.

- A Folder object can be obtained from the store by invoking the **getFolder(String folderName)** method.
- For a regular mail box, the folder name must be "inbox" (case-insensitive).
- The most important methods of the Folder class are:
  - **open(int mode):** Opens the folder either in READ_ONLY mode or READ_WRITE mode.
  - **getMessages():** Retrieves an array of Message objects which are marked un-read in the folder. A Message object may be a lightweight reference, and its detailed content will be filled up on demand.
  - **close(boolean expunge):** Closes the folder, and permanently remove all messages which are marked delete, if expunge is true.

Receiving email in Java

- A Message object represents an e-mail message.
- To get detailed attributes of a message, the following methods can be called on the Message object:
  - **Address[] getFrom():** returns a list of senders in From attribute of the message.
  - **Address[] getRecipients(Message.RecipientType type):** gets recipient addresses of the message, type can be either TO or CC.
  - **String getSubject():** gets subject of the message.
  - **Date getSentDate():** gets date and time when the message was sent.
  - **Object getContent():** gets content of the message.

Steps to do to download email

- Setup properties for the mail session.
- Creates a javax.mail.Authenticator object.
- Creating mail session.
- Get the POP3 store provider and connect to the store.
- Get folder and open the INBOX folder in the store.
- Retrieve the messages from the folder.
- Close folder and close store.

```
import javax.mail.*;
import java.util.Properties;
public class ReceiveMail
{
   public static final String USERNAME = “mailid";
   public static final String PASSWORD = “password";
   public static void main(String[] args) throws Exception
   {
      // 1. Setup properties for the mail session.
      Properties props = new Properties();
      props.put("mail.pop3.socketFactory.class", "javax.net.ssl.SSLSocketFactory");
      props.put("mail.pop3.socketFactory.fallback", "false");
      props.put("mail.pop3.socketFactory.port", "995");
      props.put("mail.pop3.port", "995");
      props.put("mail.pop3.host", "pop.gmail.com");
      props.put("mail.pop3.user", ReceiveMail.USERNAME);
      props.put("mail.store.protocol", "pop3");
      // 2. Creates a javax.mail.Authenticator object.
      Authenticator auth = new Authenticator() {
       protected PasswordAuthentication getPasswordAuthentication() {
            return new PasswordAuthentication(ReceiveMail.USERNAME,
ReceiveMail.PASSWORD);
         }
      };
```

```
// 3. Creating mail session.
     Session session = Session.getDefaultInstance(props, auth);

// 4. Get the POP3 store provider and connect to the store.
     Store store = session.getStore("pop3");
     store.connect("pop.gmail.com", ReceiveMail.USERNAME, ReceiveMail.PASSWORD);

 // 5. Get folder and open the INBOX folder in the store.
     Folder inbox = store.getFolder("INBOX");
     inbox.open(Folder.READ_ONLY);

 // 6. Retrieve the messages from the folder.
Message[] messages = inbox.getMessages();

   if (messages.length == 0)
            System.out.println("No messages found.");
```

```
for (int i = 0; i < messages.length; i++)
  {
    // stop after listing ten messages
    if (i > 10)
     {
      System.exit(0);
      inbox.close(true);
      store.close();
     }
    System.out.println("Message " + (i + 1));
    System.out.println("From : " + messages[i].getFrom()[0]);
    System.out.println("Subject : " + messages[i].getSubject());
    System.out.println("Sent Date : " + messages[i].getSentDate());
    System.out.println();
  }
   // 7. Close folder and close store.
      inbox.close(false);
      store.close();
}
}
```

File Transfer Protocol (FTP)

File Transfer Protocol (FTP)

- It is the commonly used protocol for exchanging files over the Internet.
- It uses the Internet's TCP/IP protocols to enable data transfer.
- It uses a client-server architecture, often secured with SSL/TLS.
- It promotes sharing of files via remote computers with reliable and efficient data transfer

FTPServer

```
import java.io.*;
import java.net.*;
class FTPServer
{
  public static void main(String args[])throws IOException
  {
     int c;
     String s,inp;
     //CONNECTION ESTABLISHMENT PHASE
     System.out.println("Binding to port...");
     ServerSocket ss=new ServerSocket(6666);
     System.out.println("Waiting for Client...\n\n");
     Socket soc=ss.accept();
     System.out.println(soc+" AND SERVER CONNECTED!!!");

    //DATA TRANSFER PHASE
    DataInputStream dis=new DataInputStream(soc.getInputStream());
    DataOutputStream dout=new DataOutputStream(soc.getOutputStream());

    getFilelist(dis,dout);
    inp=dis.readUTF();  //readUTF reads in Unicode Text Format hence

     FileInputStream fin = new FileInputStream("F:/"+inp);
```

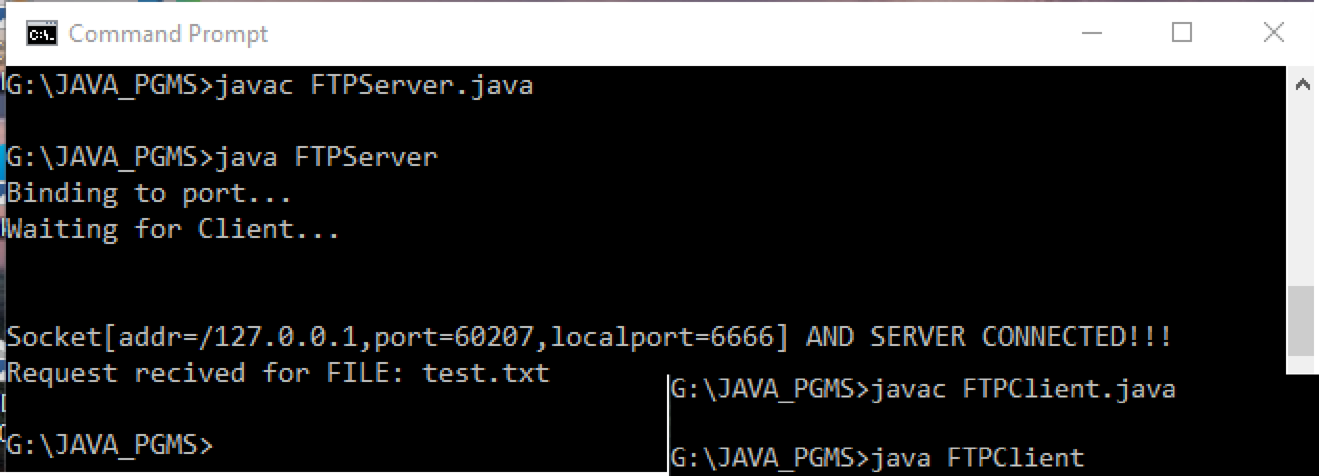
```
 System.out.println("Request recived for FILE: "+inp);
 while((c=fin.read())!=-1)
 {
             dout.write(c);
             dout.flush();
 }
 fin.close();
}
public static void getFilelist(DataInputStream din,DataOutputStream dout) throws
IOException
{
  int p=0;
  File folder = new File("F:/");
  File listOfFiles[] = folder.listFiles();
  for (int i=0;i<listOfFiles.length;i++)
  {
     //listFiles() returns both files as well as directories so display only Files
     if (listOfFiles[i].isFile())
      {
             p++;
     }
  }
 dout.write(p);  dout.flush();
```

```
for (int i=0;i<listOfFiles.length;i++)
{
//listFiles() returns both files as well as directories so display only Files
if (listOfFiles[i].isFile())
  {
                dout.writeUTF(listOfFiles[i].getName());
                dout.flush();
  }
 }
}
}
```

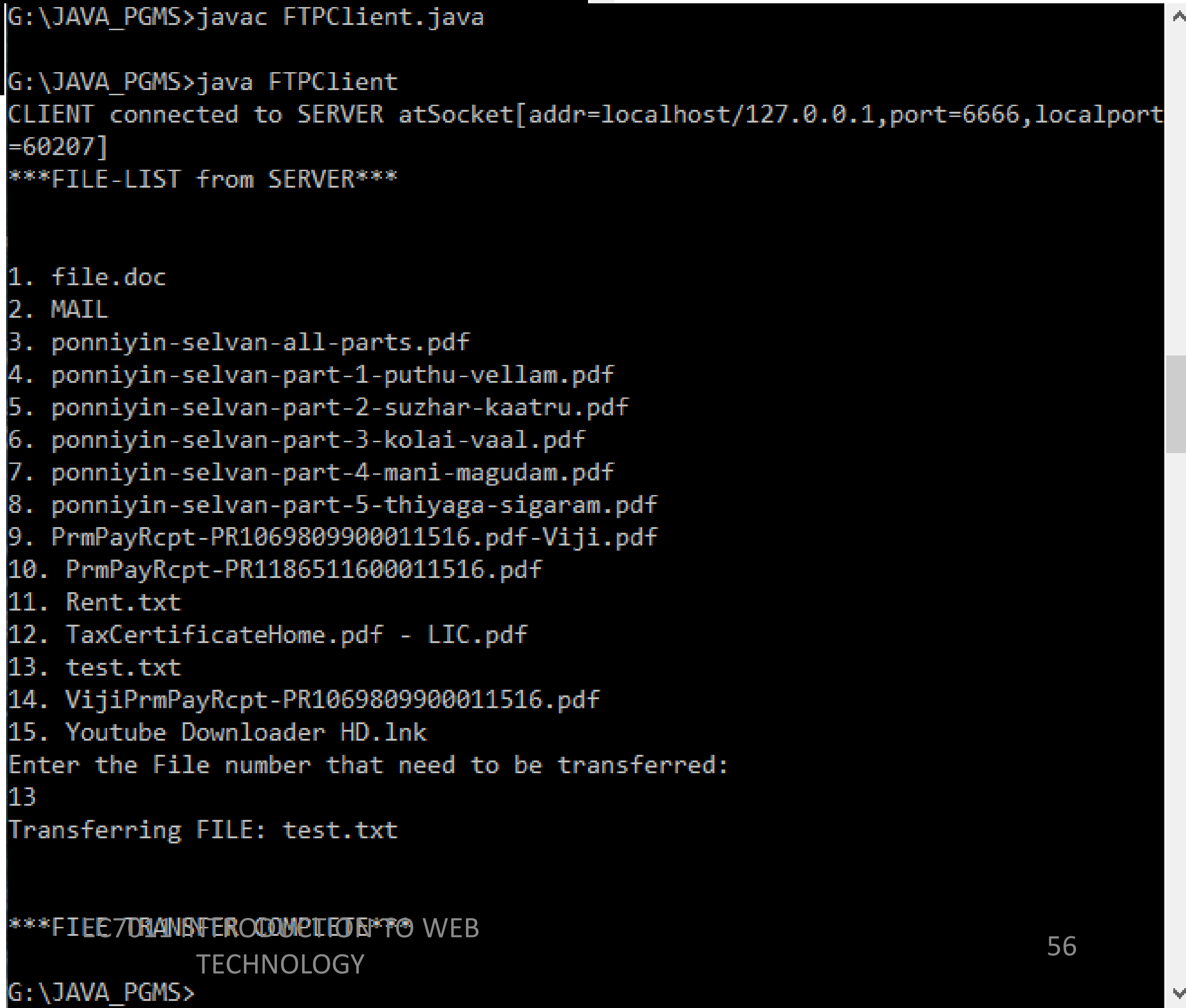
FTPClient

```
import java.io.*;
import java.net.*;
import java.util.*;

class FTPClient
{
 public static void main(String args[])throws IOException
 {

		String list[]=new String[20];
		int p=0,ch;
		Socket s=new Socket(InetAddress.getByName("localhost"),6666);
		System.out.println("CLIENT connected to SERVER at"+ s);
		DataInputStream dis=new DataInputStream(s.getInputStream());
		DataOutputStream dout=new DataOutputStream(s.getOutputStream());

		ch=getFile(dis,dout,list);
FileOutputStream fout = new FileOutputStream("F:/Download/"+list[ch-1]);
          try
          {
                    System.out.println("Transferring FILE: "+list[ch-1]+"\n\n");
                    do
                    {
                              ch=dis.read();
                              fout.write(ch);
                    }while(ch!=-1);
          }catch(SocketException e) //Exception indicates FILE TRANSFER complete.
          {
                    System.out.println("***FILE TRANSFER COMPLETE***");
          }

  }
```

```
 public static int getFile(DataInputStream dis,DataOutputStream dout,String flist[]) throws
IOException
   {
            Scanner sc=new Scanner(System.in);
            int i=0,ch=0;
            System.out.println("***FILE-LIST from SERVER***\n\n");
            ch=dis.read();
            for(i=0;i<ch;i++)
            {
                        flist[i]=dis.readUTF();
                        System.out.println((i+1)+". "+flist[i]);
            }
            System.out.println("Enter the File number that need to be transferred:");
            ch=sc.nextInt();
            dout.writeUTF(flist[(ch-1)]);
            dout.flush();
            return(ch);
   }
}
```



```
Command Prompt
G:\JAVA_PGMS>javac FTPServer.java

G:\JAVA_PGMS>java FTPServer
Binding to port...
Waiting for Client...


Socket[addr=/127.0.0.1,port=60207,localport=6666] AND SERVER CONNECTED!!!
Request recived for FILE: test.txt

G:\JAVA_PGMS>
```



```
G:\JAVA_PGMS>javac FTPClient.java

G:\JAVA_PGMS>java FTPClient
CLIENT connected to SERVER atSocket[addr=localhost/127.0.0.1,port=6666,localport
=60207]
***FILE-LIST from SERVER***


1. file.doc
2. MAIL
3. ponniyin-selvan-all-parts.pdf
4. ponniyin-selvan-part-1-puthu-vellam.pdf
5. ponniyin-selvan-part-2-suzhar-kaatru.pdf
6. ponniyin-selvan-part-3-kolai-vaal.pdf
7. ponniyin-selvan-part-4-mani-magudam.pdf
8. ponniyin-selvan-part-5-thiyaga-sigaram.pdf
9. PrmPayRcpt-PR1069809900011516.pdf-Viji.pdf
10. PrmPayRcpt-PR1186511600011516.pdf
11. Rent.txt
12. TaxCertificateHome.pdf - LIC.pdf
13. test.txt
14. VijiPrmPayRcpt-PR1069809900011516.pdf
15. Youtube Downloader HD.lnk
Enter the File number that need to be transferred:
13
Transferring FILE: test.txt


***FILE TRANSFER COMPLETE***

G:\JAVA_PGMS>
```