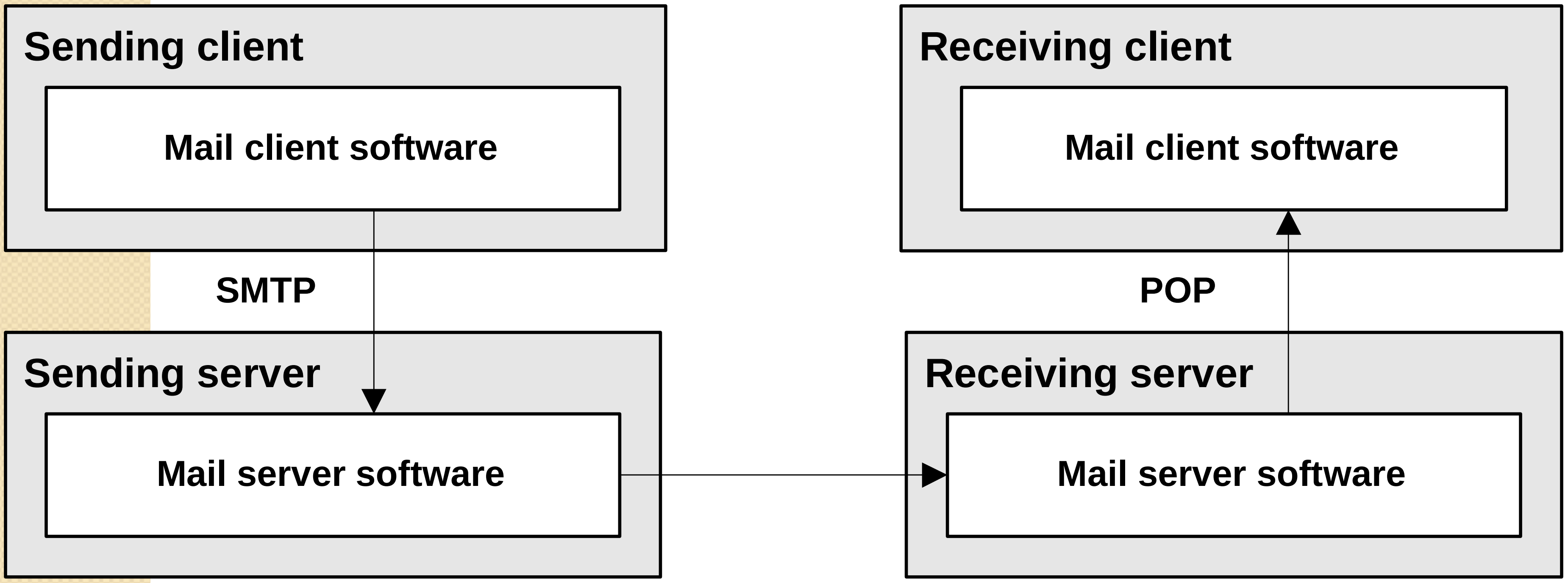
Course Name:

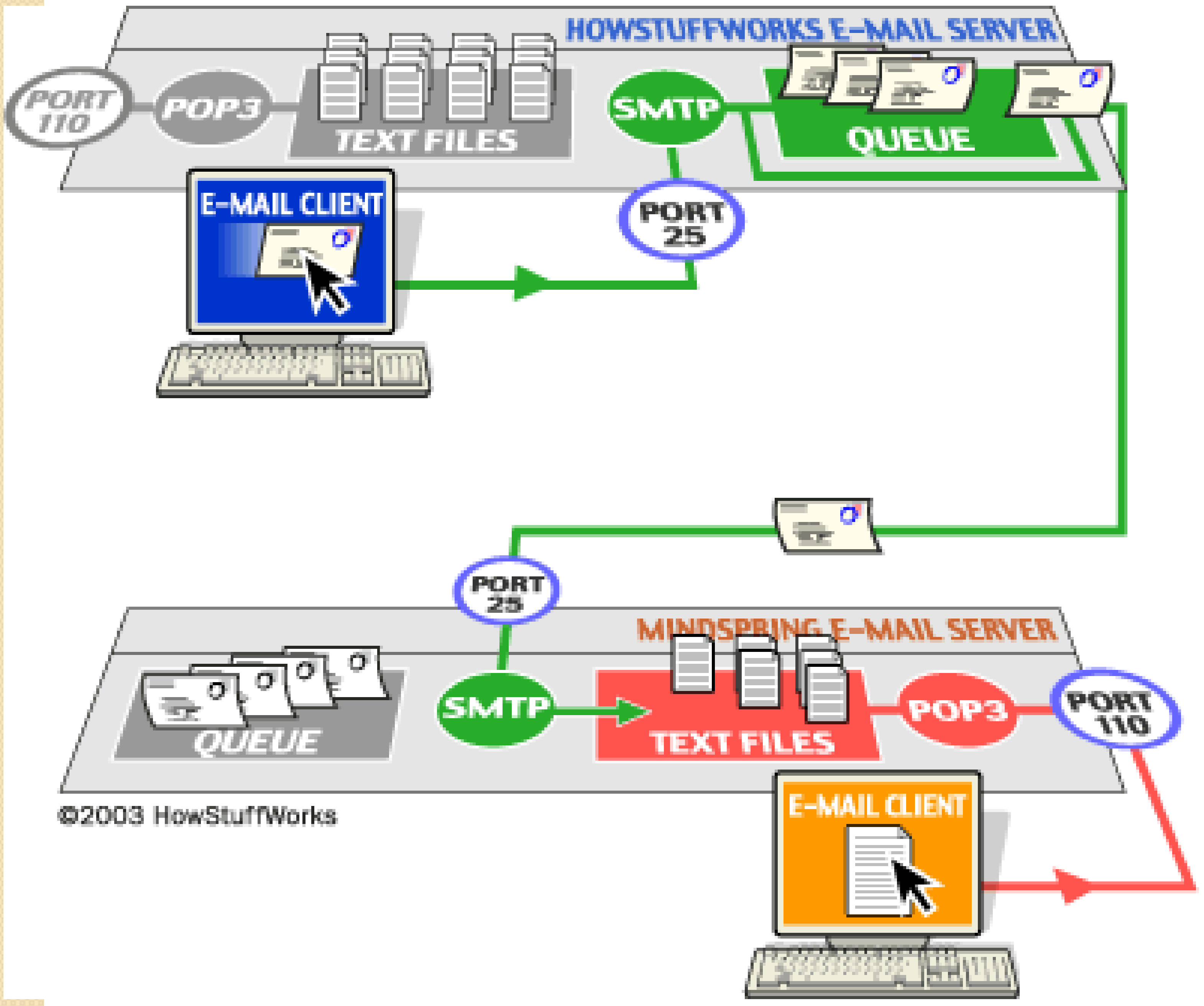
Advanced Java

Lecture 12
Topics to be covered

- Sending E-Mail

How email works




HOWSTUFFWORKS E-MAIL SERVER
PORT 110
POP3
TEXT FILES
SMTP
QUEUE
E-MAIL CLIENT
PORT 25
PORT 25
MINDSPRING E-MAIL SERVER
QUEUE
SMTP
TEXT FILES
POP3
PORT 110
E-MAIL CLIENT
©2003 HowStuffWorks

Three protocols for sending and retrieving email messages

Protocol	Description
SMTP	*Simple Mail Transfer Protocol* is used to send a message from one mail server to another.
POP	Post Office Protocol is used to retrieve messages from a mail server. This protocol transfers all messages from the mail server to the mail client. Currently, POP is in version 3 and is known as POP3.
IMAP	Internet Message Access Protocol is used by web-based mail services such as Hotmail and Yahoo. This protocol allows a web browser to read messages that are stored in the directories of the mail server. Currently, IMAP is in version 4 and is known as IMAP4.

Protocol	Description
MIME	The Multipurpose Internet Message Extension type, or MIME type, specifies the type of content that can be sent as a message or attachment.

An introduction to the JavaMail API

- When an email message is sent, it goes from the sender’s *mail client* to its *mail server* to the receiver’s mail server to the receiver’s mail client.
- SMTP, POP, and IMAP are the protocols that are commonly used for sending and receiving email messages.
- The *JavaMail API* is a high level API that allows you to use a mail protocol to communicate with a mail server.
- The JavaMail API depends upon another API known as the JavaBeans Activation Framework API, or the JAF API.

mail.jar	Contains the Java classes for the JavaMail API.
activation.jar	Contains the Java classes for the JavaBean Activation Framework. These classes are necessary for the JavaMail API to run.

Code that uses the JavaMail API to send an email

```
// 1 - get the mail session
Properties props = new Properties();
props.put("mail.smtp.host", "localhost");
Session session = Session.getDefaultInstance(props);

// 2 - create the message
MimeMessage message = new MimeMessage(session);
message.setSubject("Order Confirmation");
message.setText("Thanks for your order!");

// 3 – address the message
InternetAddress addressFrom = new
InternetAddress("av@cvsoftech.com");
message.setFrom(addressFrom);
InternetAddress addressTo = new
InternetAddress("abhimeenu2001@gmail.com");
message.setRecipient(Message.RecipientType.TO,addressTo);

// 4 - send the message
Transport.send(message);
```

A few standard properties that can be set for a Session object

Property name	Description
mail.smtp.host	Specifies the default outgoing host for SMTP protocol.
mail.from	Specifies the default return email address.
mail.user	Specifies the default username to use when connecting to the mail server.

How to create a mail session

- A Session object contains information about the *mail session*. For example, it contains information about the host and protocol for the mail server, the return address, the username, and so on.
- The getDefaultInstance method of the Session class returns the default Session object for the application.
- To supply default values for the properties of a Properties object, you can create a Properties object and use the put method to specify each property name and value.
- To specify the SMTP server for a session, you can use the mail.smtp.host property to specify the host name of the SMTP server.
- If the Java application is running on the same server as the mail server, use the localhost keyword to specify the host address.
- If the Java application isn’t running on the same server as the mail server, contact your network administrator or ISP.

How to create a message

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

How to set the subject line of a message

```
message.setSubject("Order Confirmation");
```

How to set the body of a plain text message

```
message.setText("Thanks for your order!");
```

How to set the body of an HTML message

```
message.setContent("<H1>Thanks for your order!</H1>",
                   "text/html");
```

How to create a message

- You can use the MimeMessage class that's stored in the javax.mail.internet package to create a message. This message extends the Message class that's stored in the java.mail package.
- To create a MimeMessage object, you supply a valid Session object to the MimeMessage constructor.
- Once you've created a MimeMessage object, you can use the setSubject and setText methods to set the subject line and body of the email message. This automatically sets the MIME type to text/plain.
- You can use the setContent method to include an HTML document as the body of the message. To do that, the first argument specifies a string for the HTML document, and the second argument specifies text/html as the MIME type.

How to set the From address

```
InternetAddress fromAddress = new InternetAddress("av@cvsoftech.com");
message.setFrom(fromAddress);
```

How to set the To address

```
InternetAddress toAddress = new
InternetAddress("abhimeenu2001@gmail.com");
message.setRecipient(Message.RecipientType.TO,toAddress);
```

How to set the CC address

```
InternetAddress ccAddress = new
InternetAddress("info@cvsoftech.com");
message.setRecipient(Message.RecipientType.CC,ccAddress);
```

How to set the BCC address

```
InternetAddress bccAddress = new
InternetAddress("admin@cvsoftech.com");
message.setRecipient(Message.RecipientType.BCC,bccAddress);
```

How to address a message

- To define an email address, you can use the InternetAddress class that’s stored in the javax.mail.internet package.
- You can use the setFrom method of the MimeMessage object to set the From address.
- You can use the setRecipient and setRecipients methods of the MimeMessage object to set the To, CC (*carbon copy*), and BCC (*blind carbon copy*) addresses.
- To include a name that’s associated with an email address, you can add a second argument to the InternetAddress constructor.

How to send a message

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

Notes about this method

- The send method throws a SendFailedException object when a message can’t be sent.
- If the SMTP host is incorrect in the session object, the send method will throw a SendFailedException object.
- The SendFailedException class inherits the MessagingException class. As a result, you can handle both of these exceptions by handling the MessagingException.

A helper class with a method that sends an email

```
package util;
import javax.mail.*;
import javax.mail.internet.*;
import java.util.*;

public class MailUtil{
    public static void sendMail(String to, String from,
               String subject, String messageText)
               throws MessagingException{

        // 1 - get a mail session
        Properties props = new Properties();
        props.put("mail.smtp.host", "localhost");
        Session session = Session.getDefaultInstance(props);
        // 2 - create a message
        MimeMessage message = new MimeMessage(session);
        message.setSubject(subject);
        message.setText(messageText);
        // 3 - address the message
        InternetAddress fromAddress = new InternetAddress(from);
        InternetAddress toAddress = new
              InternetAddress(to);
        message.setFrom(fromAddress);
        message.setRecipient(Message.RecipientType.TO,toAddress);
        // 4 - send the message
        Transport.send(message);
    }
}
```

A servlet that sends an email

```
import java.io.*;
import javax.servlet.*;
import javax.servlet.http.*;
import javax.mail.*;
import business.User;
import data.UserIO;
import util.MailUtil;

public class EmailServlet extends HttpServlet{

    public void doGet(HttpServletRequest request,
                      HttpServletResponse response)
                      throws IOException, ServletException{

       String firstName = request.getParameter("firstName")
       String lastName = request.getParameter("lastName");
       String emailAddress =
              request.getParameter("emailAddress");
```

The servlet (continued)

```
        User user = new User(firstName, lastName,
                             emailAddress);
        UserIO.addRecord(user,
            "../webapps/murach/WEB-INF/etc/UserEmail.txt");

        String to = emailAddress;
        String from = "emaillist@murach.com";
        String subject = "Welcome to our email list";
        String message = "Dear " + firstName + ",\n" +
            "Thanks for joining our email list. We'll make "
          + "sure to send you announcements about new "
          + "products and promotions.\n Have a great day "
          + "and thanks again!";

        try{
            MailUtil.sendMail(to, from, subject, message);
        }
        catch (MessagingException me){
            log("MessagingException: " + emailAddress);
            log(me.toString());
        }
        RequestDispatcher dispatcher =
            getServletContext().getRequestDispatcher(
            "/email12/show_email_entry.jsp");
        dispatcher.forward(request, response);
    }
}
```