Summary

I show some examples of bad code and discuss how they fail to meet the Software Quality Guidelines.

Assignment

I gave this assignment in my Internet and Intranet Protocols and Applications course:

Build an Email Sender (ES) that will send an email message by forwarding the message to an SMTP mail server. ... you must write the program in Java.

Here are the ES requirements. On the command line, ES will input the following:

1. Source email address
2. Destination email address
3. SMTP server name
4. The name of a file containing a plain text email message

ES will connect to the SMTP server, read the message text from the file, and send the message. Then ES will exit.

To send the message, you will need to implement some of the SMTP protocol. Specifically, ES must implement the following SMTP commands:

<table>
<thead>
<tr>
<th>Command</th>
<th>OK reply code</th>
</tr>
</thead>
<tbody>
<tr>
<td>HELO</td>
<td>250</td>
</tr>
<tr>
<td>MAIL FROM</td>
<td>250</td>
</tr>
<tr>
<td>RCPT TO</td>
<td>250</td>
</tr>
<tr>
<td>DATA</td>
<td>354</td>
</tr>
<tr>
<td>QUIT</td>
<td>221</td>
</tr>
</tbody>
</table>

... To simplify your task, you may treat any reply code from the SMTP server other than the "OK reply code" in the table above as fatal error. That is, given a different reply code, write a message and throw an IOException. The message should say something like "Error: unexpected Reply Code. Sent command xxxx, received Reply yyyy" where xxxx is the Command name and yyyy is the reply received.

ES will send one email each time it runs. ...

Write ES directly to the Java sockets interface. Base it on Kurose Section 2.6, and the example client on p. 138. ...

Record and produce a complete a trace of the interaction between ES and the SMTP server. Output to a text log each send and each receive in the form:

SEND: xxxx
RECV: yyyy

Where, respectively, xxxx is the EXACT content (text) of the command ES sent to the SMTP server, and yyyy is the EXACT response (the entire text line) you received from the server.

I carefully read all 60+ email sender programs. I’ve picked example code at random, to illustrate problems with code that does not meet the Software Quality Guidelines.
Example A

This code does not meet guideline **Layout**. It has many lines without logical spaces, and weird indentation.

In your code you should indent properly, and when you indent use spaces, *not tabs*, because the width of tab stops can be changed.

```
1. //Open socket connection to port 25
2. emailSocket = new Socket(SMTPName, 25);
3. //create output object to send data to server
4. output = new PrintWriter(emailSocket.getOutputStream(), true);
5. input = new BufferedReader(new InputStreamReader
     (emailSocket.getInputStream()));
6. fileOutput.write("RECV: " + input.readLine() + "\n");
7. //Say hello to the SMTP Server
8. toServer = "HELO " + SMTPName;
9. output.println(toServer);
10. fileOutput.write("SEND: " + toServer + "\n");
11. //Receive reply from server
12. fromServer = input.readLine();
13. if (isOKCode(fromServer.substring(0, 3)) == true)
14.     fileOutput.write("RECV: " + fromServer + "\n");
15. else
16. {
17.     fileOutput.write("RECV: Error: unexpected Reply Code. Sent command " +
         toServer + ", received Reply " + fromServer + "\n");
18. }
19. //Set MAIL FROM address
20. toServer = "MAIL FROM: " + sourceEmail;
21. output.println(toServer);
22. fileOutput.write("SEND: " + toServer + "\n");
23. //Receive reply from server
24. fromServer = input.readLine();
25. if (isOKCode(fromServer.substring(0, 3)) == true)
26.     fileOutput.write("RECV: " + fromServer + "\n");
27. else
28. {
         toServer + ", received Reply " + fromServer + "\n");
30. }
31. //set SEND TO address
32. toServer = "RCPT TO: " + destEmail;
33. output.println(toServer);
34. fileOutput.write("SEND: " + toServer + "\n");
35. //Receive reply from server
36. fromServer = input.readLine();
37. if (isOKCode(fromServer.substring(0, 3)) == true)
38.     fileOutput.write("RECV: " + fromServer + "\n");
39. else
40. {
41.     fileOutput.write("RECV: Error: unexpected Reply Code. Sent command " +
         toServer + ", received Reply " + fromServer + "\n");
42. }
```

This repetitious code also does not meet guideline **Non-repetitive**. It repeats this pattern several times:

```
1. //Set MAIL FROM address
2. toServer = "MAIL FROM: " + sourceEmail;
3. output.println(toServer);
```
fileOutput.write("SEND: " + toServer + "\n");
//Receive reply from server
fromServer=input.readLine();
if (isOKCode(fromServer.substring(0,3))==true)
    fileOutput.write("RECV: " +fromServer + "\n");
else
{
    fileOutput.write("RECV: Error: unexpected Reply Code. Sent command " + toServer + ", received Reply " + fromServer + "\n");
}

We simplify and clarify it by defining this method:
// assume that the following parameters are defined and
// initialized:
// BufferedWriter fileOutput;
// BufferedReader input;
// PrintWriter output ;
/**
 * interactWithSMTPServer
 * sends to and receives from the SMTP server
 * @param command the command to send to the server
 */
public void interactWithSMTPServer( String command )
{
    String fromServer;
    output.println( command );
    fileOutput.write("SEND: " + command + "\n");
    fromServer=input.readLine();
    if (isOKCode(fromServer.substring(0,3)) == true)
        fileOutput.write("RECV: " +fromServer + "\n");
    else
    {
        fileOutput.write("RECV: Error: unexpected Reply Code. Sent command " + command + ", received Reply " + fromServer + "\n");
    }
}

Then we can rewrite Example A:
//Open socket connection to port 25
emailSocket=new Socket(SMTPName, 25);
//create output object to send data to server
output = new PrintWriter(emailSocket.getOutputStream(), true);
input = new BufferedReader(new InputStreamReader
(emailSocket.getInputStream()));
fileOutput.write("RECV: " + input.readLine() + "\n");

//Say hello to the SMTP Server
interactWithSMTPServer("HELO " + SMTPName);

//Set MAIL FROM address
interactWithSMTPServer("MAIL FROM: " + sourceEmail);

//set SEND TO address
interactWithSMTPServer( "RCPT TO: " + destEmail );

(This code is still buggy. It doesn’t check the error codes properly.) Avoiding repetition makes code shorter, easier to read and understand, and easier to change.

Example B

```java
1. public class emailSender
2. {
3.     public static void main(String[] args)
4.     {
5.         String sendersEmailID;
6.         String receiversEmailID;
7.         String SMTPServerName;
8.         String dot;
9.         String stringData;
10.        int    data;
11.       Character characterData;
12.       if( args.length == 4 )
13.           {
14.               sendersEmailID   = new String( args[0] );
15.               receiversEmailID = new String( args[1] );
16.               SMTPServerName   = new String( args[2] );
17.               dot              = new String( "." );
18.               }
19.         else
20.             {
21.                 System.out.println( "ERROR: Usage - emailSender SourceEmailAddress DestinationEmailAddress SMPTServerName MessageFile\n" );
22.                 return;
23.             }
24.         try
25.             {
26.                 FileWriter SMPTtraceLog            = new FileWriter( "SMPTtrace.log" );
27.                 FileReader messageFile             = new FileReader( args[3] );
28.                 Socket clientMailSenderSocket      = new Socket( SMTPServerName, 25 );
29.                 DataOutputStream outToMailReceiver =
30.                     new DataOutputStream( clientMailSenderSocket.getOutputStream() );
31.                 BufferedReader inFromMailReceiver =
32.                     new BufferedReader(
33.                         new InputStreamReader( clientMailSenderSocket.getInputStream() ) );
34.                 // Receive Response to Connection
35.                 String receivedMessage = inFromMailReceiver.readLine();
36.                 String replyCode = receivedMessage.substring( 0, 3 );
37.                 SMPTtraceLog.write( "SEND: " + "griffin.cs.nyu.edu\n" );
38.                 SMPTtraceLog.write( "RECV: " + receivedMessage + '\n' );
39.                 SMPTtraceLog.flush();
40.                 if( replyCode.compareTo( "220" ) == 0 )
41.                     {
42.                         // Sending HELO
43.                         String sendingMessage = new String( "HELO cs.nyu.edu" );
44.                         outToMailReceiver.writeBytes( sendingMessage + '\n' );
45.                         SMPTtraceLog.write( "SEND: " + sendingMessage + '\n' );
46.                         // Receive Response to HELO
47.                         receivedMessage = inFromMailReceiver.readLine();
48.                         replyCode = receivedMessage.substring( 0, 3 );
49.                         SMPTtraceLog.write( "RECV: " + receivedMessage + '\n' );
50.                         SMPTtraceLog.flush();
51.                         if( replyCode.compareTo( "250" ) == 0 )
52.                             {
53.                                 // Sending MAIL FROM
54.                                 sendingMessage = new String( "MAIL FROM: " + sendersEmailID );
55.                                 outToMailReceiver.writeBytes( sendingMessage + '\n' );
56.                                 SMPTtraceLog.write( "SEND: " + sendingMessage + '\n' );
57.                                 // Receive Response to MAIL FROM
58.                                 receivedMessage = inFromMailReceiver.readLine();
59.                                 replyCode = receivedMessage.substring( 0, 3 );
```

4
SMPTtracelog.write("RECV: " + receivedMessage + '\n');
SMPTtracelog.flush();

if( replyCode.compareTo("250") == 0 )
{
   // Sending RCPT TO
   sendingMessage = new String("RCPT TO: " + receiversEmailID);
   outToMailReceiver.writeBytes( sendingMessage + '\n');
   SMPTtracelog.write("SEND: " + sendingMessage + '\n');

   // Receive Response to RCPT TO
   receivedMessage = inFromMailReceiver.readLine();
   replyCode = receivedMessage.substring(0, 3);
   SMPTtracelog.write("RECV: " + receivedMessage + '\n');
   SMPTtracelog.flush();

   if( replyCode.compareTo("250") == 0 )
   {
      // Sending DATA
      sendingMessage = new String("DATA");
      outToMailReceiver.writeBytes( sendingMessage + '\n');
      SMPTtracelog.write("SEND: " + sendingMessage + '\n');

      // Receive Response to DATA
      receivedMessage = inFromMailReceiver.readLine();
      replyCode = receivedMessage.substring(0, 3);
      SMPTtracelog.write("RECV: " + receivedMessage + '\n');
      SMPTtracelog.flush();

      if( replyCode.compareTo("354") == 0 )
      {
         // Sending mail data
         data = messageFile.read();
         stringData = new StringBuilder().append((char) data).toString();
         while( data != -1 )
         {
            data = messageFile.read();
            if( data != -1 )
            {
               characterData = new Character((char) data);
               stringData = stringData.append(characterData.toString());
            }
         }

         sendingMessage = stringData.toString();

         outToMailReceiver.writeBytes(sendingMessage + '\n');
         SMPTtracelog.write("SEND: " + sendingMessage + '\n');

         // Sending mail dot
         sendingMessage = new String(".");
         outToMailReceiver.writeBytes(sendingMessage + '\n');
         SMPTtracelog.write("SEND: " + sendingMessage + '\n');

         // Receive Response to DOT
         receivedMessage = inFromMailReceiver.readLine();
         replyCode = receivedMessage.substring(0, 3);
         SMPTtracelog.write("RECV: " + receivedMessage + '\n');
         SMPTtracelog.flush();

         if( replyCode.compareTo("250") == 0 )
         {
            // Sending QUIT
            sendingMessage = new String("QUIT");
            outToMailReceiver.writeBytes(sendingMessage + '\n');
            SMPTtracelog.write("SEND: " + sendingMessage + '\n');

            // Receive Response to QUIT
            receivedMessage = inFromMailReceiver.readLine();
            replyCode = receivedMessage.substring(0, 3);
            SMPTtracelog.write("RECV: " + receivedMessage + '\n');
            SMPTtracelog.flush();

            if( replyCode.compareTo("221") != 0 )
String command = new String( "QUIT" );
String reply = receivedMessage.substring( 3 );
IOException exception = new IOException( new String( "ERROR: Unexpected Reply Code. Sent Command " +command +", Received Reply " + reply ) );
throw( exception );
}

else
{
String command = new String( "DOT" );
String reply = receivedMessage.substring( 3 );
IOException exception = new IOException( new String( "ERROR: Unexpected Reply Code. Sent Command " +command +", Received Reply " + reply ) );
throw( exception );
}
else
{
String command = new String( "DATA" );
String reply = receivedMessage.substring( 3 );
IOException exception = new IOException( new String( "ERROR: Unexpected Reply Code. Sent Command " +command +", Received Reply " + reply ) );
throw( exception );
}
else
{
String command = new String( "RCPT TO" );
String reply = receivedMessage.substring( 3 );
IOException exception = new IOException( new String( "ERROR: Unexpected Reply Code. Sent Command " +command +", Received Reply " + reply ) );
throw( exception );
}
else
{
String command = new String( "MAIL FROM" );
String reply = receivedMessage.substring( 3 );
IOException exception = new IOException( new String( "ERROR: Unexpected Reply Code. Sent Command " +command +", Received Reply " + reply ) );
throw( exception );
}
else
{
String command = new String( "HELO" );
String reply = receivedMessage.substring( 3 );
IOException exception = new IOException( new String( "ERROR: Unexpected Reply Code. Sent Command " +command +", Received Reply " + reply ) );
throw( exception );
}
else
{
String reply = receivedMessage.substring( 3 );
IOException exception = new IOException( new String( "ERROR: Unexpected Reply Code. Sent Command Open Connection, Received Reply " + reply ) );
throw( exception );
}
catch( IOException e )
{
System.out.println( "Successfully sent the message - Thank you\n" );
close();
SMTPtraceLog.close();
messageFile.close();
System.out.println( "G
" );
catch( IOException e )
{
System.out.println( e.getMessage() );
return ;
}
7 deep! This violates guideline **Simplicity**. They’re embedded unnecessarily, in fact, because when an exception is thrown control flow branches to the catch() statement! This could be straight line code. Make use of return and exception to avoid nested if-then-else statements.

**Example C**

It’s good to make an email address checker method. But example C violates guidelines **Semantics**, **Simplicity** and **Well Commented**:  

```
public static boolean isValidEmail(String anEmailAddr) {
    if (anEmailAddr == null) {
        return false;
    } else if (anEmailAddr.equals("")) {
        return false;
    } else if (anEmailAddr.indexOf("@") == -1) {
        return false;
    } else if (anEmailAddr.indexOf(".") == -1) {
        return false;
    } else if (anEmailAddr.lastIndexOf("." < anEmailAddr.lastIndexOf("@") {
        return false;
    } else if ((anEmailAddr.lastIndexOf(".") - anEmailAddr.lastIndexOf("@") == 1) {
        return false;
    } else if ((anEmailAddr.length() - anEmailAddr.lastIndexOf(".") > 5) {
        return false;
    } else if ((anEmailAddr.length() - anEmailAddr.lastIndexOf(".") < 2) {
        return false;
    }
    return true;
}
```

First, this code violates the specification for an acceptable email address in RFC 821  
http://www.ietf.org/rfc/rfc0821.txt (**Semantics**). For example, a valid email address might not have a ‘.’ (dot), so the third ‘if’ is wrong. The 6th ‘if’ rejects the legal address

```
userid@localhost
```

For more, see this comment from my example code:

```
/**
 * Check email address syntax.
 * this should check that mailbox satisfies the BNF for <mailbox> in
 * RFC 821 Section 4.1.2 p. 30,
 * <mailbox> ::= <local-part> "@" <domain>
 * <local-part> ::= <dot-string> | <quoted-string>
 * <dot-string> ::= <string> | <string> "." <dot-string>
 * <string> ::= <char> | <char> <string>
 * <char> ::= <c> | "\" <x>
 * <c> ::= any one of the 128 ASCII characters, but not any
 * <special> or <SP>
 * <quoted-string> ::= ""<qtext> ""
 * <domain> ::= <element> | <element> "." <domain>
 * etc.
 * but doing so really involves building a grammer
 * so this code only checks that an @ is present.
 * @param mailbox the email address
 */
```

Second, the structure could be simplified since the else branches are not needed after the return statements as in this example:

```
public static boolean isValidEmail(String anEmailAddr) {
    if (anEmailAddr == null) {
        return false;
    }
```
if (anEmailAddr.equals("")) {
    return false;
} else if (anEmailAddr.indexOf("@") == -1) {
    return false;
} else if (anEmailAddr.indexOf(".") == -1) {
    return false;
} // code left out o o o
return true;

Third, documentation (Well Commented) is necessary. You should write comments that describe the architecture of the program, and the structure of your algorithms.

You Try It

How would you improve these example SMTP senders?

Example X

1. import java.net.*;
2. import java.io.*;
3. 4. class socketInclude{
5.   protected socketInclude(String socketname,int port,String logName)
6.     throws Exception{
7.         clientSocket = new Socket(socketname,port);
8.         outToServer = new DataOutputStream(clientSocket.getOutputStream());
9.         inFromServer = new BufferedReader(new InputStreamReader
10.            (clientSocket.getInputStream()));
11.         outLog = new FileWriter(logName);
12.     }
13.   protected void disconnectSocket() throws Exception{
14.     }
15.   protected String read() throws Exception{
16.     return inFromServer.readLine();
17.   }
18.   protected void write(String in) throws Exception{
19.     outToServer.writeBytes(in);
20.   }  
21.   protected void writeLog(String in,boolean close) throws Exception{
22.     outLog.write(in.toCharArray());
23.     if(close)
24.         outLog.close();
25.   }  
26.   private Socket clientSocket;
27.   private DataOutputStream outToServer;
28.   private BufferedReader inFromServer;
29.   private FileWriter outLog;
30. }
31. public class emailSender extends socketInclude{
32.   public emailSender(String sn,String sa,String da,String fp)
33.     throws Exception{
34.         super(sn,25,"SMTPtrace.log");
35.         inFromServer = new BufferedReader(new FileReader(fp));
36.         filein = new StringBuffer("");  
37.         String line;
38.         while((line = br.readLine()) != null){
39.             if(line.length()>0 && line.charAt(0) == '.'){
line = "." + line;
filein.append(line).append('\n');
}
br.close();
data = new String[4];
data[0] = sa.substring(sa.indexOf('@') + 1);
data[1] = sa;
data[2] = da;
data[3] = filein.toString() + ".";
}
public void sendMail() throws Exception{
String msg;
for(int i=-1;i<5;++i){
switch(i){
case -1:
writeLog("RECV: " + read() + "\n",false);
break;
case 3:
write(command[i] + "\n");
getReplyCode(msg = read(),replyCode[i],i);
writeLog("SEND: " + command[i] + "\n" +
"RECV: " + msg + "\n",false);
write(data[i] + "\n");
writeLog("SEND: " + data[i] + "\n" +
"RECV: " + read() + "\n",false);
break;
case 4:
write(command[i] + "\n");
getReplyCode(msg = read(),replyCode[i],i);
writeLog("SEND: " + command[i] + "\n" +
"RECV: " + msg + "\n",true);
break;
default:
write(command[i] + data[i] + "\n");
getReplyCode(msg = read(),replyCode[i],i);
writeLog("SEND: " + command[i] + data[i] + "\n" +
"RECV: " + msg + "\n",false);
break;
}
}
disconnectSocket();
private void getReplyCode(String in,int normalCode,int commandIndex)
throws Exception{
if(Integer.valueOf(in.substring(0,3)).intValue() != normalCode)
throw new Exception("Error: unexpected Reply Code. Send command " +
cmd[commandIndex] + data[commandIndex] +
", received Reply "+ in);
}
private int [] replyCode = {250,250,250,354,221};
private String [] command = {"HELO ","MAIL FROM: ","RCPT TO: ",
"DATA","QUIT"};
public static void main(String[] args){
String socketName,sourceAddr,destinationAddr,filePath;
BufferedReader inFromUser = new BufferedReader(new InputStreamReader
(System.in));
try{
System.out.println("Please enter your email address");
sourceAddr = inFromUser.readLine();
System.out.println("Please enter the address you want to mail");
destinationAddr = inFromUser.readLine();
System.out.println("Please enter the SMTP server you want");
socketName = inFromUser.readLine();
System.out.println("Please enter the path of your mail body");
}
Example Y

Assignment: analyze this example. Which code quality guidelines does it violate? Document the violations.

Also, find a 2 to 4 page sample of Java code that you wrote and analyze it. Which code quality guidelines did you violate? Document your violations.

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

public class emailSender {
    public static void main(String[] args) throws IOException{
        String sourceEmailAdd = args[0];
        String destinationEmailAdd = args[1];
        String SMTPServer = args[2];
        String emailFile = args[3];
        Socket SMTPSocket = null;
        DataOutputStream out = null;
        BufferedReader in = null;
        String fromServer;
        String fromUser;
        String lastCommand = null;
        String fileLine;
        BufferedReader file = new BufferedReader(new FileReader(emailFile));
        FileWriter writeLog = new FileWriter("SMTPTrace.log");
        try {
            SMTPSocket = new Socket(SMTPServer, 25);
            out = new DataOutputStream(SMTPSocket.getOutputStream());
            in = new BufferedReader(new InputStreamReader(SMTPSocket.getInputStream()));
            while ((fromServer = in.readLine()) != null) {
                StringTokenizer inCode = new StringTokenizer(fromServer);
                System.out.println("RECV: " + fromServer);
                writeLog.write("RECV: "+fromServer+"\n");
                String firstCode = inCode.nextToken();
```

108.     filePath = inFromUser.readLine();
109.     emailSender es = new emailSender(socketName,sourceAddr,
110.            destinationAddr,filePath);
111.     es.sendMail();
112. }
113. }
114. catch(Exception e){
115.     System.out.println(e.toString());
116. }
117. }
118. }
```

1. import java.io.*;
2. import java.net.*;
3. import java.util.*;
4. public class emailSender {
5.     public static void main(String[] args) throws IOException{
6.         String sourceEmailAdd = args[0];
7.         String destinationEmailAdd = args[1];
8.         String SMTPServer = args[2];
9.         String emailFile = args[3];
10.        Socket SMTPSocket = null;
11.        DataOutputStream out = null;
12.        BufferedReader in = null;
13.        String fromServer;
14.        String fromUser;
15.        String lastCommand = null;
16.        String fileLine;
17.        BufferedReader file = new BufferedReader(new FileReader(emailFile));
18.        FileWriter writeLog = new FileWriter("SMTPTrace.log");
19.        try {
20.            SMTPSocket = new Socket(SMTPServer, 25);
21.            out = new DataOutputStream(SMTPSocket.getOutputStream());
22.            in = new BufferedReader(new InputStreamReader(SMTPSocket.getInputStream()));
23.            while ((fromServer = in.readLine()) != null) {
24.                StringTokenizer inCode = new StringTokenizer(fromServer);
25.                System.out.println("RECV: " + fromServer);
26.                writeLog.write("RECV: "+fromServer+"\n");
27.                String firstCode = inCode.nextToken();
```
if (firstCode.equals("220")) {
    out.writeBytes("HELO "+SMTPServer+"\r\n");
    writeLog.write("SEND: HELO "+SMTPServer+"\r\n");
    System.out.println("SEND: HELO "+SMTPServer+"\r\n");
    lastCommand = "HELO";
}
else if (firstCode.equals("250") && lastCommand.equals("HELO")) {
    System.out.println("SEND: MAIL FROM: <"+sourceEmailAdd+">\r\n");
    out.writeBytes("MAIL FROM: <"+sourceEmailAdd+">\r\n");
    writeLog.write("SEND: MAIL FROM: <"+sourceEmailAdd+">\r\n");
    lastCommand = "MAIL FROM";
}
else if (firstCode.equals("250") && lastCommand.equals("MAIL FROM")) {
    System.out.println("SEND: RCPT TO: <"+destinationEmailAdd+">\r\n");
    out.writeBytes("RCPT TO: <"+destinationEmailAdd+">\r\n");
    writeLog.write("SEND: RCPT TO: <"+destinationEmailAdd+">\r\n");
    lastCommand = "RCPT TO";
}
else if (firstCode.equals("250") && lastCommand.equals("RCPT TO")) {
    System.out.println("SEND: DATA\r\n");
    out.writeBytes("DATA\r\n");
    writeLog.write("SEND: DATA\r\n");
    lastCommand = "DATA";
}
else if (firstCode.equals("354")) {
    writeLog.write("SEND: MAIL BODY\r\nSEND: .\r\n");
    while (fileLine = file.readLine() != null) {
        out.writeBytes(fileLine+"\r\n");
        System.out.println("SEND: "+fileLine);
    }
    out.writeBytes("\r\n.\r\n");
    System.out.println("SEND: .\r\n");
}
else if (firstCode.equals("250") && lastCommand.equals("DATA")) {
    System.out.println("SEND: Quit\r\n");
    out.writeBytes("QUIT\r\n");
    writeLog.write("SEND: Quit\r\n");
}
else if (firstCode.equals("221")) {
    out.close();
    in.close();
    SMTPSocket.close();
    writeLog.close();
    break;
}
else {
    System.out.println("error:");
    System.out.println("Send: "+lastCommand);
    System.out.println("Recv: "+fromServer);
    break;
}
}