Basic Functionality

- Essentially everyone got these
  - Put your name and student ID in the comments at the beginning.
  - Call the class emailSender and store it in emailSender.java
  - Use Java.
  - Write ES directly to the Java sockets interface
- On command line, read
  - 1. Source email address
  - 2. Destination email address
  - 3. SMTP server name
  - 4. The name of email file
- Connect to the SMTP server on port 25
- Send the message
- Given a bad reply code, write a properly formatted message
- Record and produce a complete trace of the interaction
- Pass in the log from a successful execution
Check email address syntax, at least a little

- Difficult to do right

  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.

  really involves building a grammer

  My code only checks that an @ is present

  Other easy INCOMPLETE checks:
  `<local-part>` is either a `<quoted-string>` or contains no unescaped `<SP>`
  `<domain>` contains no `<SP>`
The argument field of HELO

• From 821: The argument field contains the host name of the sender-SMTP.
  – I misspoke and suggested contains the host name of the sender email address.

• Right way:

  // get the local host name
  String myHostName =
      aSocket.getLocalAddress().getHostName();
SMTP commands are terminated by <CRLF>

- 821 is very specific about <CRLF> termination
- Unfortunately, some SMTP servers are lax, so your programs worked (beware of working code!)
- In fact, 2821, which replaced 821, says

In addition, the appearance of "bare" "CR" or "LF" characters in text (i.e., either without the other) has a long history of causing problems in mail implementations and applications that use the mail system as a tool. SMTP client implementations MUST NOT transmit these characters except when they are intended as line terminators and then MUST, as indicated above, transmit them only as a <CRLF> sequence.
Check All 7 Success Reply Codes

• They are
  – CONNECT, "220"
  – HELO, "250"
  – MAIL_FROM, "250"
  – RCPT_TO, "250"
  – DATA, "354"
  – END_OF_MAIL, "250"
  – QUIT, "221"

• Good to put in a data structure
  – Consider extending the code: full SMTP has 21 codes, each with # and string

I didn’t accept full credit for

```python
if( ReturnCode in [ 220, 250, 354, 221 ] )
```

because it doesn’t check that the current reply has its correct code (the wrong reply might be arriving)
Given a Bad Reply Code Throw an IOException

- I’m not sure this was such a good design decision because some code still has to print the error and exit
- But it's not entirely unreasonable
Handle Multiline Responses

- RFC 821 Section 4.2 p. 34
  Only the EXPN and HELP commands are expected to result in multiline replies in normal circumstances, however multiline replies are allowed for any command.

- I didn’t expect anyone to do this—and nobody did (except me):
  ```java
  while( true ) {
      line = fromSMTPserver.readLine();
      toLogFile.println( "RECV: " + line  );
      if( line.charAt(3) == SP )
      {
          break;
      }
  }
  ```
Successfully send email containing lines that start with ‘.’

- This is a critical piece of the SMTP spec.
  - RFC 821 Section 4.5.2. TRANSPARENCY p. 41

Without some provision for data transparency the character sequence "<CRLF>.<CRLF>" ends the mail text and cannot be sent by the user. In general, users are not aware of such "forbidden" sequences. To allow all user composed text to be transmitted transparently the following procedures are used.

1. Before sending a line of mail text the sender-SMTP checks the first character of the line. If it is a period, one additional period is inserted at the beginning of the line.
Send email containing lines that start with ‘.’

- One possible implementation

```java
while( true ) {
    String line = inEmailFile.readLine();  // read the data
    if( line == null ) break;  // EOF
    if( line.length() == 0 ) {
        toSMTPserver.writeBytes( CRLF );
        continue;
    }
    if( line.charAt(0) == '.' ) {
        // send email_message, escape the '.
        toSMTPserver.writeBytes( '.' + line + CRLF );
    } else {
        toSMTPserver.writeBytes( line + CRLF );
    }
}
```
Some special cases
(see code on previous page)

- Successfully send email containing blank (zero length) lines.
- Successfully send an email containing nothing.