CS 580 Client-Server Programming
Spring Semester, 2006
Doc 16 Some Parsing
Mar 30, 2006

```
Socket connection = new Socket(server, port);
InputStream rawIn = connection.getInputStream();
BufferedReader in = new BufferedReader(
        new InputStreamReader(rawIn));
String answer = in.readLine();
```

Some low level Java Parsing

"cat;man;ran".split(";");

Returns an array of String [“cat”, “man”, “ran”];

StringTokenizer

```
parts = new java.util.StringTokenizer("cat,man;ran;,fan", ",;");
while (parts.hasMoreElements())
    {
    System.out.println( parts.nextToken());
    }
```

Output

```
cat
man
ran
fan
```

java.util.Scanner

```
String input = "1 fish 2 fish red fish blue fish";
Scanner s = new Scanner(input).useDelimiter("\\s*fish\\s*");
System.out.println(s.nextInt());
System.out.println(s.nextInt());
System.out.println(s.next());
 System.out.println(s.next());
 s.close();
```

Output

```
1
 2
red
blue
```

Ruby Streams

```
def send(text)
    connection = TCPSocket.new(@server, @port)
    connection.print(text)
    connection.flush
    answer = connection.gets("\n")
    connection.close
    answer
end
```

Java UpToReader?

```
Socket connection = new Socket(server, port);
InputStream rawIn = connection.getInputStream();
UpToReader in = new UpToReader(
        new InputStreamReader(rawIn));
String answer = in.upTo(';');
```

sdsu.io.ChunkReader

```
read = new sdsu.io.ChunkReader("catEOMmatEOM", "EOM")
while (read.hasMoreElements() )
    {
    System.out.println( read.readChunk());
    }
```

Output

```
cat
mat
```

Subclass FilterInputStream

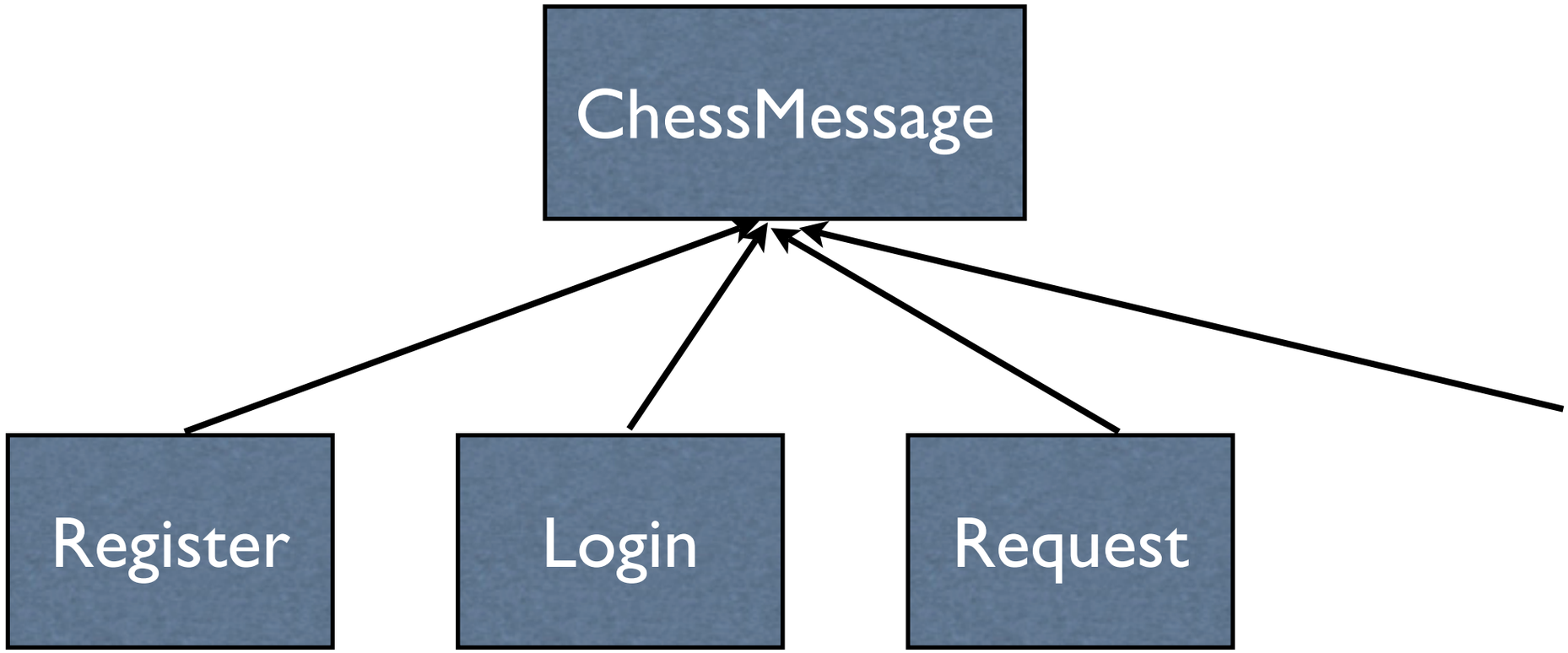
```
public class UpToInputStream extends FilterInputStream {
        public UpToInputStream(InputStream stream)
                { super(stream); }

        public byte[] upto(char end) throws IOException {
                int EOF = -1;
                ByteBuffer buffer = new ByteBuffer();
                int c;
                while (( c = super.read()) != EOF )  {
                        buffer.append( (byte)c);
                        if (c == end )
                                break;
                }
                if (c == EOF & (buffer.isEmpty()))
                        return new byte[0];

                return buffer.getBytes();
                }
}
```

Why not read Message Objects?

```
InputStream rawIn = connection.getInputStream();
ChessMessageReader in = new ChessMessageReader(
        rawIn);
Message answer = in.next();
```


ChessMessage
Register
Login
Request

Client Side

```
Socket connection = new Socket(server, port);
OutputStream rawOut = connection.getOutputStream();
PrintStream out = new PrintStream(new BufferedOutputStream(rawOut));
InputStream rawIn = connection.getInputStream();
ChessMessageReader in = new ChessMessageReader(rawIn);
```

```
LoginMessage login = new LoginMessage(“whitney”, “topSecret”);
out.print(login.toString());
ChessMessage result = in.next();

if (result.isError() ) then
      deal with bad login
else
      deal with successful login
```

Server-Side

```
ChessMessage request = in.next();

if (request.isLogin() ) then
      LoginMessage login = (LoginMessage) request;
      String userName = login.userName();
      String password = login.password();
      booelan isValidUser = processLogin(userName, password);
      if (isValidUser) {
            etc
      }
else
      blah
```

Message Responsibilities

Hide all message syntax

Read message and convert to object

```
ChessMessage message =
    LoginMessage.from("LOGIN\r\nUserName:whitney\r\nPassword:foo\r\n\r\n");
```

Create message from values

```
LoginMessage login = new LoginMessage("whitney", "topSecret");
```

Access information about message

```
message.isLogin()
message.userName()
```

Testing

```
public void testLogin() {
        LoginMessage login = new LoginMessage(“whitney”, “topSecret”);
        assertTrue( login.toString() =
                “LOGIN\r\nUserName:whitney\r\nPassword:foo\r\n\r\n”
```

Those Pesky If Statements

```
ChessMessage request = in.next();

if (request.isLogin() ) then
      do login
else if (request.isRegister() )
      do registration
else if (request.isAccept() )
      blah
else if (request.isMessage() )
      blah
```

Use Polymorphism

In Server

```
ChessMessage request = in.next();
ChessMessage result = request.execute( this);
out.print( result.toString());
```

```
class LoginMessage {
public ChessMessage execute(Server serverReference) {
        booelan isValidUser = serverReference.processLogin(userName, password);
        if (isValidUser) {
                etc
```

The Other If Statement

```
ChessMessage message;
String commandName = in.upTo(“\r\n”);
if (commandName == “LOGIN” ) then
      message = new Login();
else if (commandName  = “REGISTRATION” )
      message = new Registration();
etc.
```