Names

Select names that help the reader understand the code

Use standard naming conventions

```java
public class client { blah
Socket s = null;
BufferedReader kbd;
Socket sktServer;
```
Comments

Write comments to help the reader understand the code

Don’t waste readers time repeating the code

//defining server port
int port = 7777;

while ((c = in.read()) != 13) //stop when read cr= 13
while ((c = in.read()) != CARRAGE_RETURN)
Socket client;
server = new ServerSocket(port);
System.out.println("Server port: "+ server.getLocalPort())

while (true)
{
    client = server.accept();
}

server = new ServerSocket(port);
System.out.println("Server port: "+ server.getLocalPort())

while (true)
{
    Socket client = server.accept();
}
Println()

Don’t use

Can cause problems between platforms
DataOutputSteam & DataInputStream

Used to read/write binary versions of data types

Avoid in Client/Server

Both sides must use same format for messages
**When are we at the end of a message?**

Read while characters are available?

This can cause problems

- Multiple message may be sent at the same time
- Messages may be broken into small packets

```java
while (available() > 0)
    buffer.append(in.read());
```

Read one buffer full?

Some packets may be in transit

```java
byte[] buffer = new byte[1024];
while (in.read(buffer) == 0)
```

Read to EOF?

Only works after other end close connection

```java
while ((c = in.read()) != -1)
```

Just read a message!

Protocol defines syntax for a message

Length of a message

Special token terminates message
Mixing UI and Network Code

Don’t!

```java
System.out.println("Select one: Date or Time?");
request = console.readLine();
if (request = "Date")
    toServer.write( "date" + CR);
    response = fromServer.read();
```
public class TimeDateClient
{
    private static final char CARRIAGE_RETURN = (char) 13;
    private static final char LINE_FEED = (char) 10;

    String server;
    int serverPort;

    public TimeDateClient(String serverNameOrIP, int port)
    {
        server = serverNameOrIP;
        serverPort = port;
    }

    public String date() throws IOException
    {
        return sendMessage("date");
    }

    public String time() throws IOException
    {
        return sendMessage("time");
    }

    public String sendMessage(String message) throws IOException
    {
        Socket serverConnection = new Socket(server, serverPort);
        writeMessage(message, serverConnection);
        byte[] result = readMessage(serverConnection);
        serverConnection.close();
        return new String(result);
    }
}
private byte[] readMessage(Socket serverConnection) throws IOException
{
    UpToFilterInputStream in = new UpToFilterInputStream(
        new BufferedInputStream(serverConnection.getInputStream()));
    byte[] result = in.upTo(LINE_FEED);
    return result;
}

private void writeMessage(String message, Socket serverConnection)
    throws IOException
{
    OutputStream out = new BufferedOutputStream(
        serverConnection.getOutputStream());
    out.write((message + CARRIAGE_RETURN).getBytes());
    out.flush();
}