Chapter 2

Practical assignment 2

2.1 Background

A Telnet client is available for all well-known operating systems. To activate it, one simply enters the command
`telnet <computer name>`
or, sometimes,
`telnet <computer name> <port number>`

The Telnet client is in the first place meant for communication with a Telnet server. One may, for example, use Telnet to work on a Unix computer located in any place in the world as if one were working from a terminal directly connected to the computer. Telnet may, however, communicate with any server. Essentially it simply sends each character that is entered on the keyboard to the server — and displays each character that the server sends to the client on the client's screen.

In the basic configuration even the characters typed on the keyboard are not displayed, but sent to the server which 'echoes' each character so that one sees what one types. This has the advantage that one knows while you are communicating with the server since each character that one sees has been sent to the server and returned by it.

In contrast, the default setting of some Telnet clients are to display whatever is typed, and then *not* display the character when (and if) it is echoed. This is determined by a property known as *localecho*, if *localecho* is on, then the Telnet client will locally 'echo' everything that is typed, without waiting for it to be echoed by the server.

In a number of instances we will use Telnet to interact with a server that was never intended to be used via Telnet. In those cases one wants *localecho* to be on, since the server has no reason to echo any input it receives. If the default setting of your client is *localecho off*, it needs to be set to on. To do this, establish the con-

nection with your server. Then press `^]` (hold `Ctrl` and press ']'). This causes you to 'escape' into the client shell, and you will be presented with a > prompt. Then enter

```
set localecho
```

and you will be informed that "Local echo [is now] on." If you want to turn it off, use the command

```
unset localecho
```

To "escape" from the client shell, enter the command `quit` and you will be communicating with the connected server (rather than the local client) again.

For this assignment you are expected to write a tiny, special-purpose Telnet server. Ideally it should work with a Telnet client irrespective of the setting of *localecho*. Hence it is advisable to echo characters that the user is supposed to see. However, test it with a client where *localecho* is on: you do not want to see every character twice — once when it is locally echoed and once when the server echoes it...

2.2 Your assignment

You are expected to write a server that maintains a 'database' of your appointments (date, time, with whom, etc). All the usual operations such as searching, addition and deletion must be available. After your server has been activated, ¡b¿all¡/b¿ interaction with it has to occur through Telnet. Your server is also responsible to echo everything that is entered. Note that the 'database' may be an array that is read from or written to an ordinary file.

To make your program more visually pleasing, you may use ANSI escape sequences that are supported by ANSI and VT100 (and other) emulations. Two of the more useful ANSI escape sequences are:

- `ESC[2J` to clear the screen; and
- `ESC[y;xH` to move the cursor to position `(x,y)` on the screen;

Here `ESC` is ASCII character 27_{10}; `x` and `y` are numbers in string format. If `screen` is a suitable Java stream object, then

```
screen.write( 27 );
screen.print( "[20;5HHello" );
```

will display the message `Hello` in line 5 from position 20 on the screen. Experiment to ensure that you understand the concept before you write your program.

(Naturally the best solution is to write your own class and methods to hide this level of detail from the rest of your program.)

Remember that your program is a server that is to be used via the network. After you have activated the server, all interaction with the server has to occur via Telnet: during development it may be a good idea to build a server that outputs debugging information in the server window; however, once it is demonstrated, the server will not output any values on its window or display.

2.3 Assessment

A working program (that uses screen control) will earn 8 out of 10. To earn higher marks your program will be expected to do more than just the basics, such as to allow more than one user to use your program simultaneously or to simply use colour. However, since colour has now been mentioned, it is no longer an original idea and won't earn many additional marks.