

Lab 1 – Threads and Sockets

1. Simple Threads

- a) Write a class called `Counter` that **extends** `Thread`. The `Counter` should count from **10** to **500**.
- b) Write a class called `SleepyCounter` that implements the `Runnable` interface. The class should behave like `Counter`, except that after each output it should sleep for **0.05** seconds. [Hint: Remember that `Thread.sleep()` can throw an `InterruptedException` that must be handled.]
- c) Write a class called `CounterApp` with a `main()` method, create one `Counter` thread and one `SleepyCounter` thread, and start both of them. Run the program several times and observe its output.
- d) **Questions:**
 - What does this (i.e. the outputs observed) tell you about *thread scheduling*?
 - What advantage does implementing `Runnable` have over extending `Thread`?
 - What are *runnable* and *running*? Are they the same? Explain.

2. Synchronization

- a) Read the examples `Buffer.java`, `Producer.java`, and `Consumer.java` from the course lecture notes. Make sure you fully understand these programs¹.
- b) Modify the `Buffer` class, using an integer array instead of the character array to represent `Buffer`.
- c) Modify the `Producer` class and the `Consumer` class to reflect the change above. The `Producer` class should send integers **0** to **50** to the `Buffer`. The `Consumer` class should receive the **50** integers from the `Buffer` and output them to the console. [Note: You can use `System.out.println()`.]
- d) Write a class called `BufferApp` with a `main()` method, create a `Buffer` with size **50**, create one `Producer` thread and one `Consumer` thread and start both of them. Run the program.

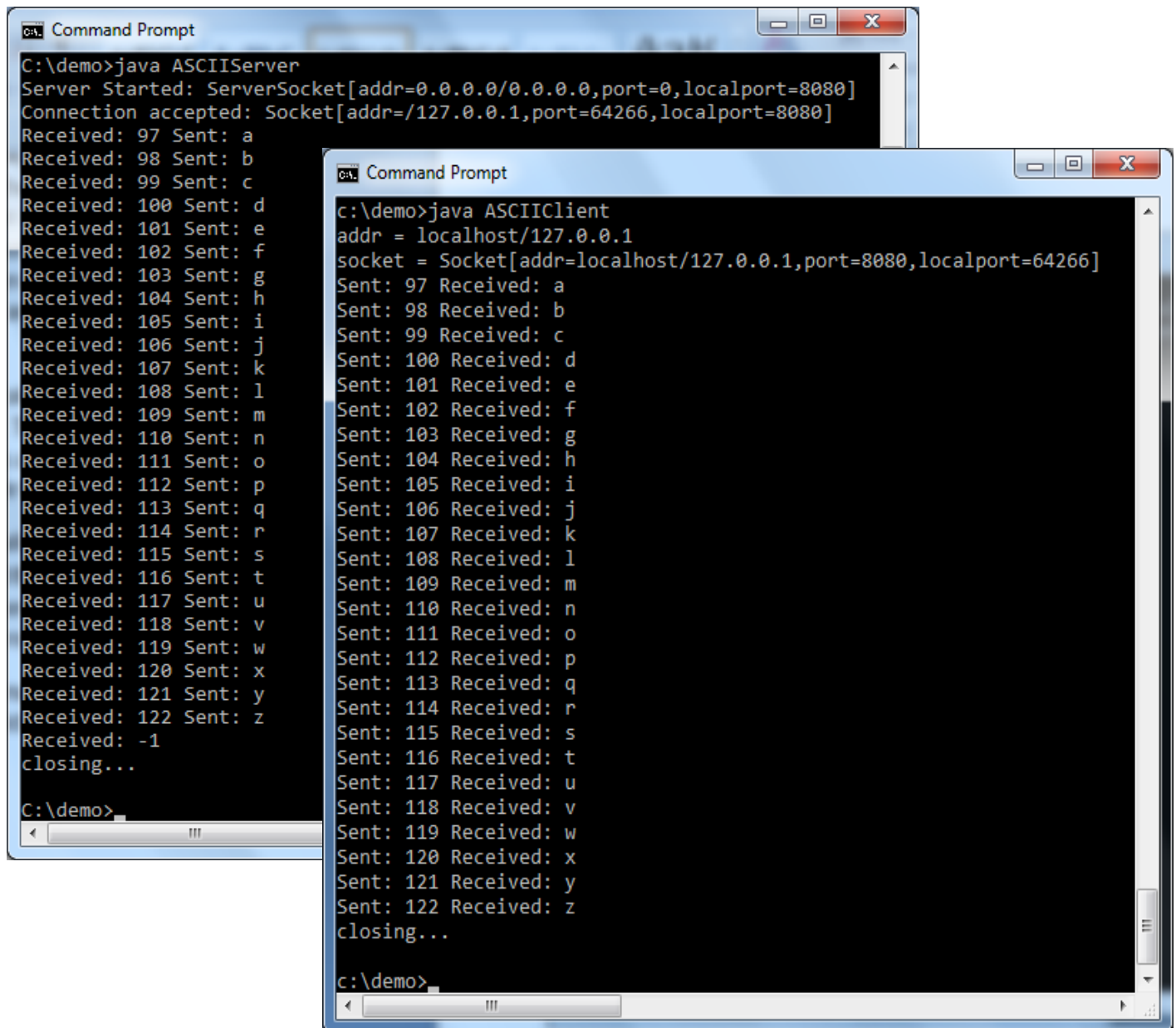
¹ They were covered in the “Threads” course topic.

e) Questions:

- What is the purpose of the `wait()` and `notify()` methods?
- When should you *synchronize* all methods in a class? What might be a disadvantage of doing this?
- There is also another problem that can occur with multi-threaded applications. What is it called? How can you fix that problem?

3. Sockets

- a)** Read the `MyServer.java` and `MyClient.java` files from the course lecture notes. Make sure you fully understand the socket programs. Based on this example, write a *Client-Server socket program*: the *Client* should send the integers from **97** to **122** to the *Server*, the *Server* then convert the integers to ASCII characters (a-z) and return them to the *Client*. The *Client* then displays the returned characters to the console. Please write the *Server* program and the *Client* program according to the specification in parts **b)** and **c)** below.
- b)** Write the *Server* class, which is called `ASCIIServer`. The class uses the IP address **127.0.0.1** and port number **8080**. This class should receive integers from the *Client*, convert the integers to characters and send the characters back to the *Client*, until it receives the integer **-1**, to indicate the end of the sending information.
- c)** Write the *Client* class, which is called `ASCIIClient`. The class should send integers from **97** to **122** to the *Server*, and then send **-1**, to indicate the end of the sending information. The *Client* should then receive the response from the *Server* and display the received characters to the console.
- d)** Test your programs in the same machine. Start the *Server* program, and then start the *Client* program. The output should be similar to the screen shots in **Figure 1**.
- e) Questions:**
- How can you modify the *Server* program to accept multiple *Clients*?
 - If possible, test the programs in two machines, one for the *Server* program, and one for the *Client* program. What changes do you need to make in the source code?
 - What is a *Socket*?
 - What is a *host machine*?



```
C:\demo>java ASCIIserver
Server Started: ServerSocket[addr=0.0.0.0/0.0.0.0,port=0,localport=8080]
Connection accepted: Socket[addr=/127.0.0.1,port=64266,localport=8080]
Received: 97 Sent: a
Received: 98 Sent: b
Received: 99 Sent: c
Received: 100 Sent: d
Received: 101 Sent: e
Received: 102 Sent: f
Received: 103 Sent: g
Received: 104 Sent: h
Received: 105 Sent: i
Received: 106 Sent: j
Received: 107 Sent: k
Received: 108 Sent: l
Received: 109 Sent: m
Received: 110 Sent: n
Received: 111 Sent: o
Received: 112 Sent: p
Received: 113 Sent: q
Received: 114 Sent: r
Received: 115 Sent: s
Received: 116 Sent: t
Received: 117 Sent: u
Received: 118 Sent: v
Received: 119 Sent: w
Received: 120 Sent: x
Received: 121 Sent: y
Received: 122 Sent: z
Received: -1
closing...

C:\demo>

c:\demo>java ASCIIclient
addr = localhost/127.0.0.1
socket = Socket[addr=localhost/127.0.0.1,port=8080,localport=64266]
Sent: 97 Received: a
Sent: 98 Received: b
Sent: 99 Received: c
Sent: 100 Received: d
Sent: 101 Received: e
Sent: 102 Received: f
Sent: 103 Received: g
Sent: 104 Received: h
Sent: 105 Received: i
Sent: 106 Received: j
Sent: 107 Received: k
Sent: 108 Received: l
Sent: 109 Received: m
Sent: 110 Received: n
Sent: 111 Received: o
Sent: 112 Received: p
Sent: 113 Received: q
Sent: 114 Received: r
Sent: 115 Received: s
Sent: 116 Received: t
Sent: 117 Received: u
Sent: 118 Received: v
Sent: 119 Received: w
Sent: 120 Received: x
Sent: 121 Received: y
Sent: 122 Received: z
closing...

c:\demo>
```

Figure 1

END of Lab 1: “Threads and Sockets”