D7001D Network Programming and Distributed Applications – LAB2

Submitted By: Ahmed Afif Monrat, Syed Asif Iqbal, Daniyal Akthar

OBJECTIVES:

i. Recall the main patterns of java programming in general and network programming.

ii. Write own threaded TCP server.

iii. Experiment with an advanced messaging architecture.

PART-I (JAVA GUI)

Write a simple java GUI1 with two fields and one button. Where first field is used to input command and the second one is to display output. The application should be capable of executing the following Unix commands (or their counterparts in Windows):

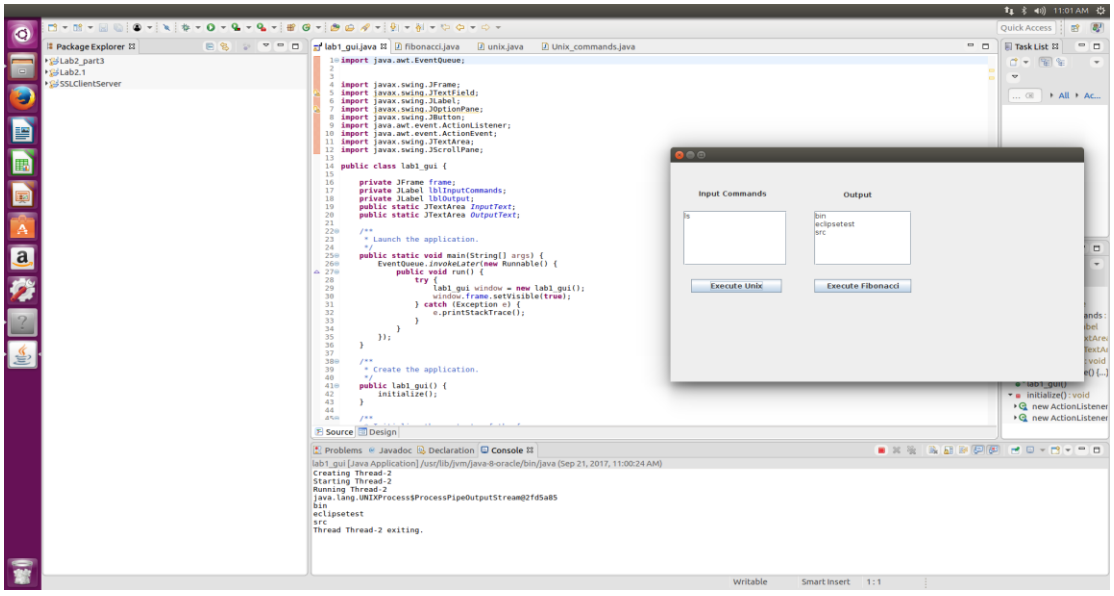
ls, pwd, uptime, cat file.txt, who, ps, grep, ifconfig

It should be possible to pipeline command. For example, this command cd /home/ && pwd | ls –al

Which should return a detailed list of files and directories under /home/ folder.

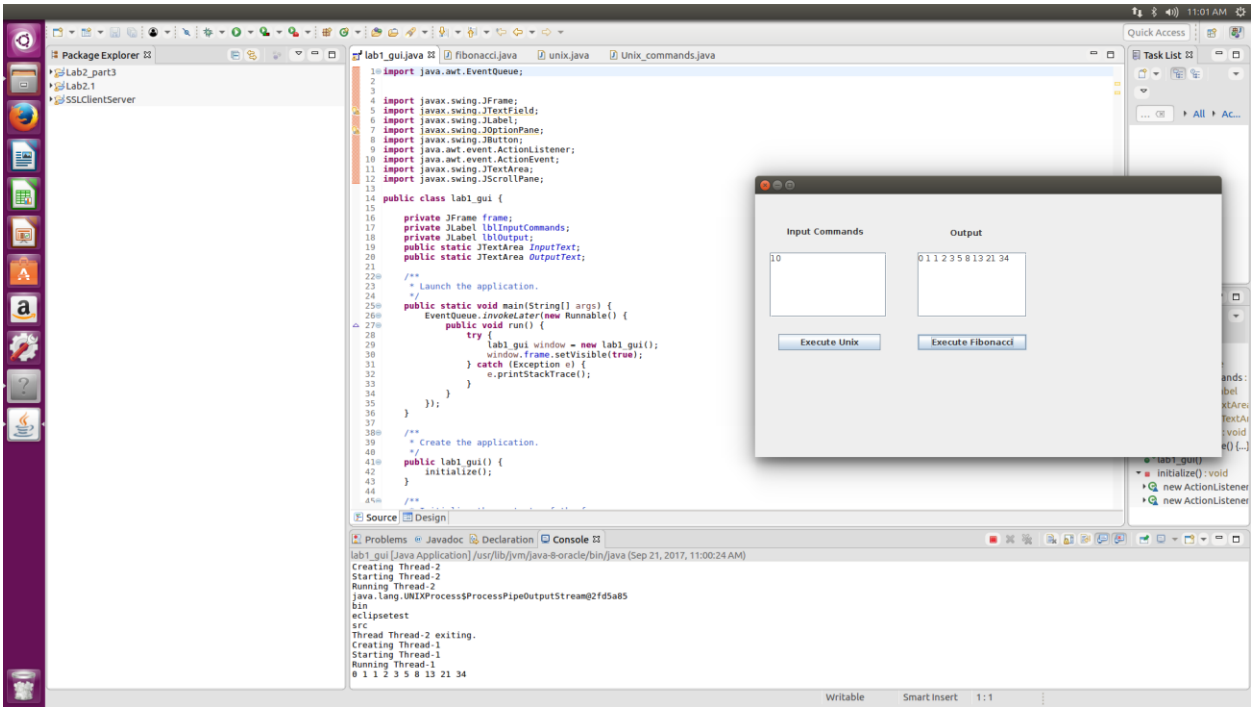
Now write another Java class, which will execute a Fibonacci Series. The program should take the number of Sequence as an input parameter, print the sequence in the output field. Launch this program from the GUI! The GUI and the command execution must be implemented in separate classes. Use threads to implement the command execution.

Observations:



For this scenario, we have created three separate classes where a class will feature GUI components and rest of the classes are representing two thread class for generating Fibonacci series and Unix commands.

Here we have used Windows builder in Eclipse IDE for GUI part and threads have been implemented for executing commands.



PART-II (Java netprog patterns – Sockets & Threads)

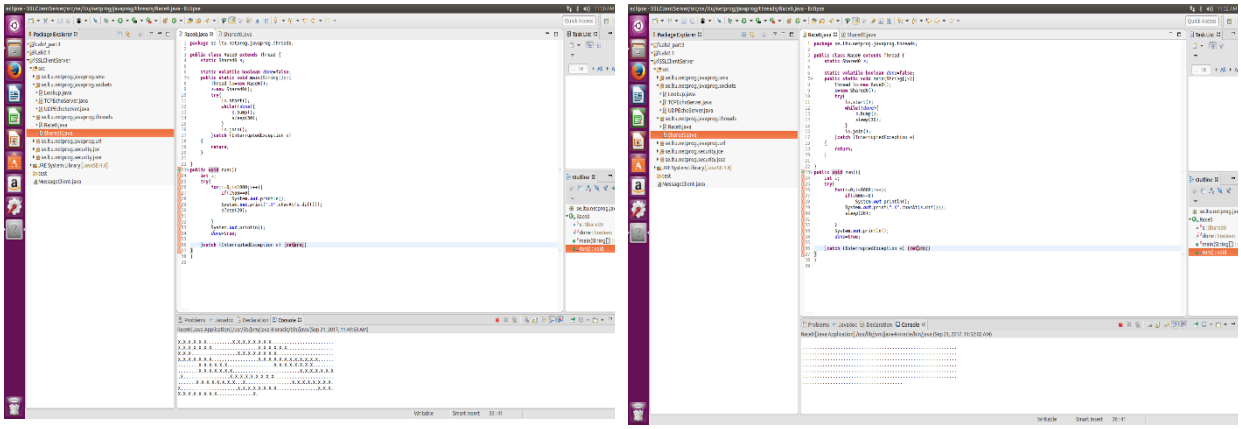
1. Compile and debug the Lookup, TCPEchoServer and UDPEchoServer classes.
Compiled successfully

2. Be able to describe the details of the implementation of each class.

3. Modify the Lookup class so, that it outputs your name in addition to the input parameters that you supply at runtime
Done
4. Modify both the TCPEchoServer and UDPEchoServer classes so that in addition to echoed input symbols the server would send back your name.
Done
5. 5. Compile the class Race0 in the threads part of the project.
a. What kind of behavior do you observe?
Initially both of the threads were working simultaneously as when one thread was suspending execution due to Thread.sleep, other one was activating automatically. That's why, it was producing unforeseen result due to concurrency issues.
b. Make now both dif() and bump() methods synchronized. Compile the classes
and run
Done

c. How the behavior is changed now?

After the change, when a thread has entered a synchronized instance method then no other thread can enter any of synchronized instance methods of the same instance.



Part III: Client-server application

1. The GUI runs as a client-side application. Add an additional field to your GUI where the user can enter the URL of the server-side application.

Done Successfully

2. The entered commands should be executed and run as the server-side application. You can implement this assignments in either of the following two ways

a. A CGI program running at the remote webserver2 (you would get minimum points sufficient to pass this task, however).

b. **Implement your own multi-threaded TCP server (higher points will be granted).** ✓

3. Add logging capability3,4 to the server-side application.

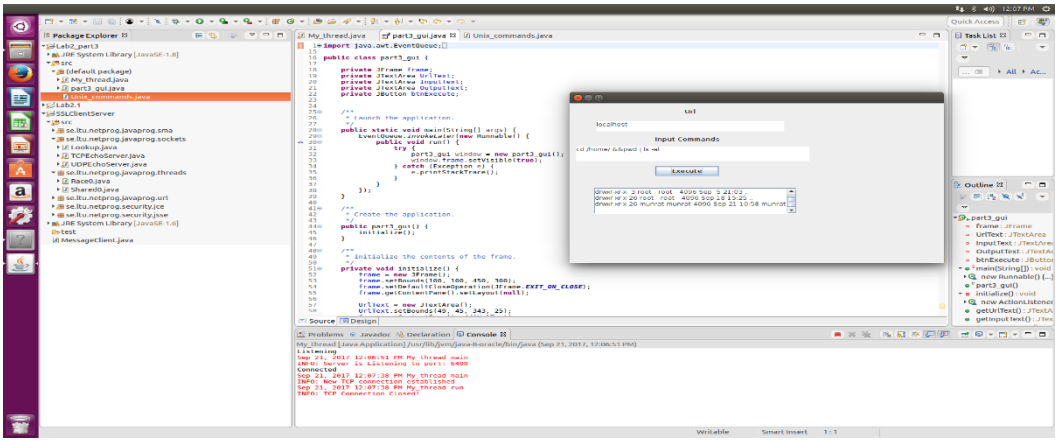
Done Successfully

4. The application should take an argument defining which level of debug should be logged (warning, info, error, debug).

Done Successfully

5. When adding different log levels to functions you have to explain why you add this level of logging to the specific function. Write this explanation as a comment in your source code.

Done Successfully

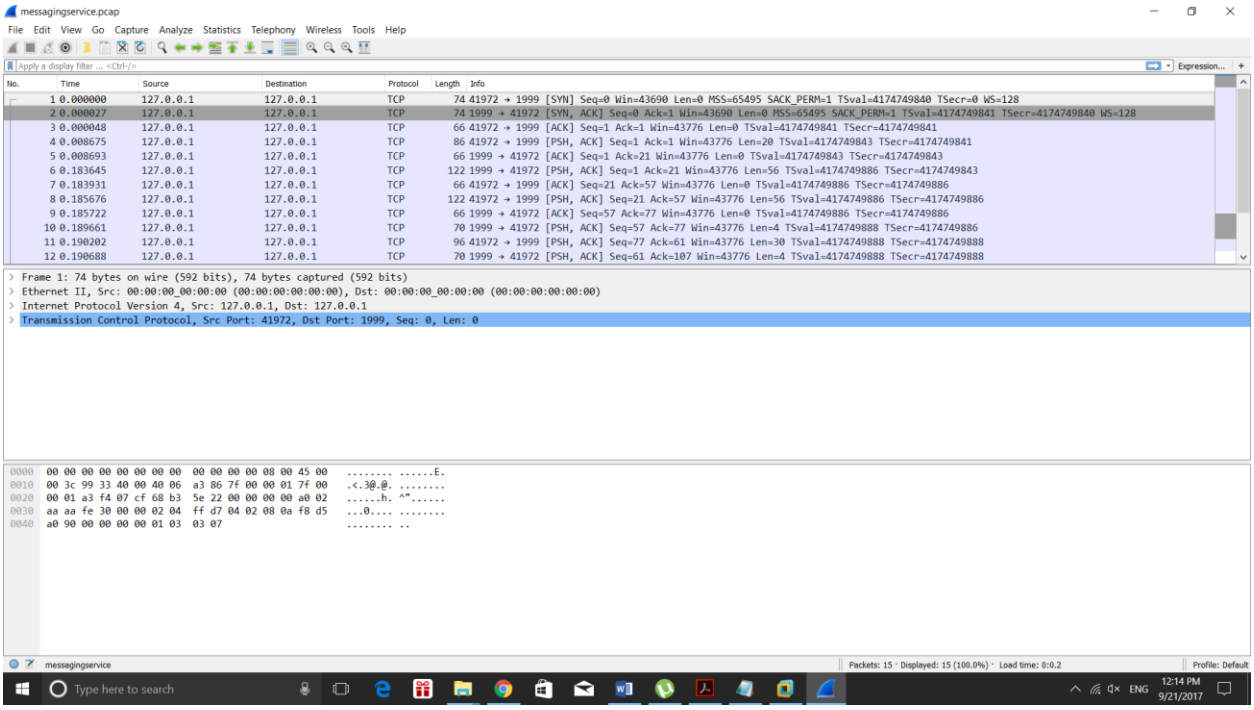


Part IV – Java netprog patterns – Simple Messaging Architecture

1. Compile and install the classes in the SMA module of the project.

Done Successfully

2. Demonstrate the functioning classes during the lab assessment. Use Wireshark to capture the traffic of the SMA session. You should be able to explain the details of the implementation during the individual assessment.



Part V – Java netprog patterns – security

1. Compile and install (in the cloud) the classes in the security module of the project (JCE and JSSE).
2. Demonstrate the functioning classes during the lab assessment. You should be able to explain the details of the implementation during the individual assessment.