With the support of the Erasmus+ Programme of the European Union

EMJMD GENIAL

GREEN NETWORKING & CLOUD COMPUTING

UNIVERSITÉ DE LORRAINE

LEEDS BECKETT UNIVERSITY

LULEÅ UNIVERSITY OF TECHNOLOGY

Network programming and distributed applications (D7001D)

By

Md Abu Ahammed Babu

&

Mohammad Messbah Uddin

Luleå University of Technology

Department of Computer Science, Electrical and Space Engineering

Lab 2

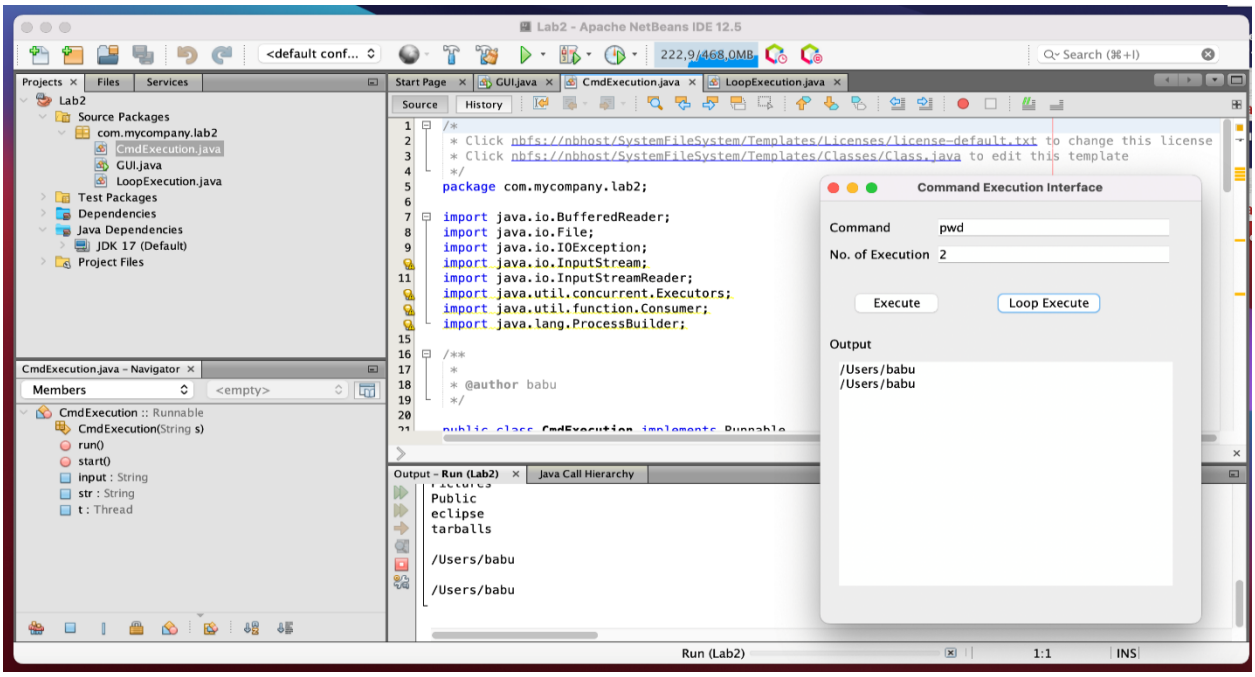
Lab Objective:

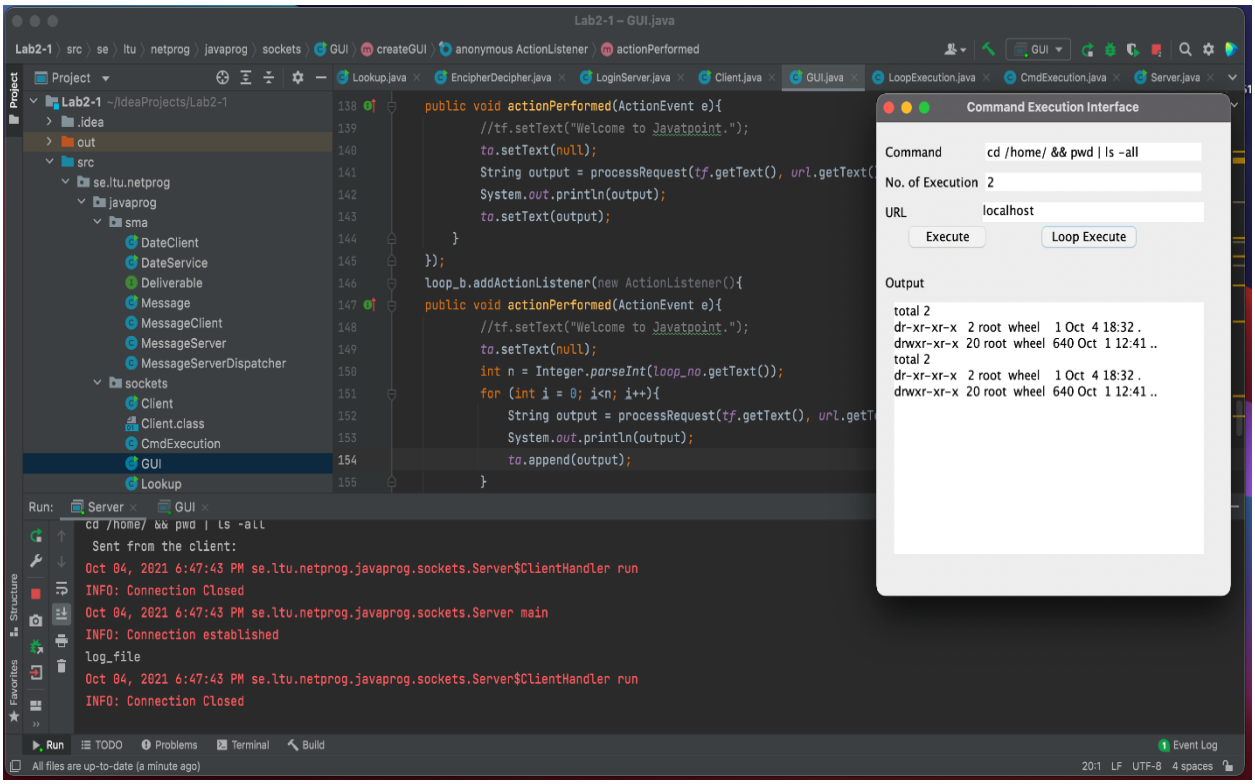
- Recall the main patterns of java programming in general and network programming in particular
- Write own threaded TCP server
- Experiment with an advanced messaging architecture

Part I – Java GUI (To be performed either on a local machine or on a AWS instance)

The task in this part is to write a simple java GUI with two fields and one button where the 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.

Observations:
We created three different classes for this part, one for GUI and two threaded class for Unix commands and Fibonacci series.

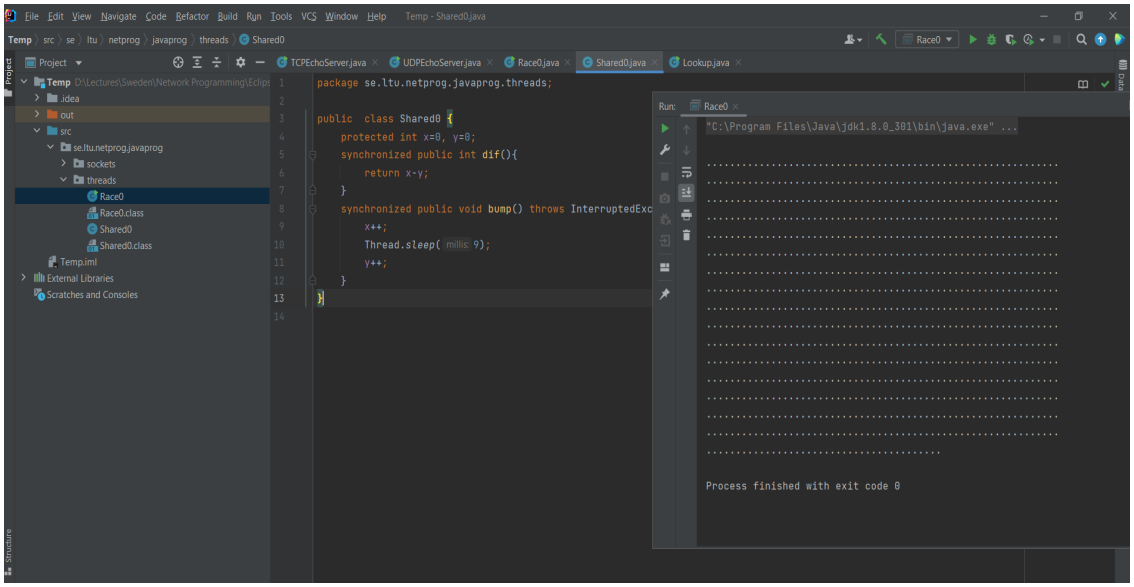

Lab2 - Apache NetBeans IDE 12.5
Command Execution Interface
Command pwd
No. of Execution 2
Execute
Loop Execute
Output
/Users/babu
/Users/babu


Lab2-1 – GUI.java
Command Execution Interface
Command cd /home/ && pwd | ls -all
No. of Execution 2
URL localhost
Execute
Loop Execute
Output
total 2
dr-xr-xr-x 2 root wheel 1 Oct 4 18:32 .
drwxr-xr-x 20 root wheel 640 Oct 1 12:41 ..
total 2
dr-xr-xr-x 2 root wheel 1 Oct 4 18:32 .
drwxr-xr-x 20 root wheel 640 Oct 1 12:41 ..

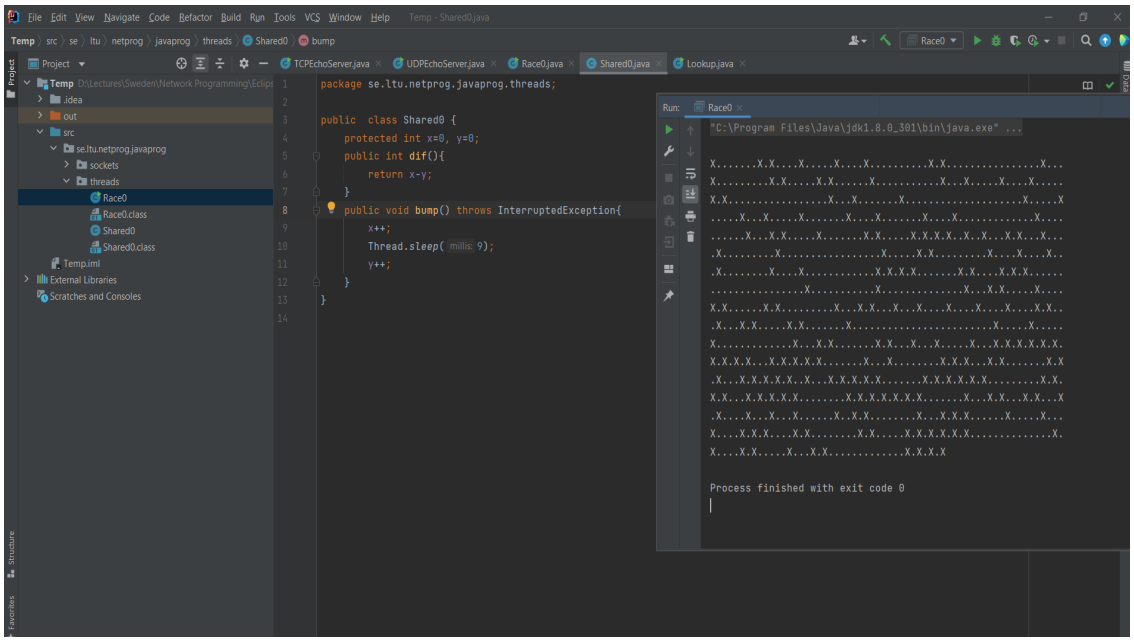
PART-II (Java netprog patterns – Sockets & Threads)

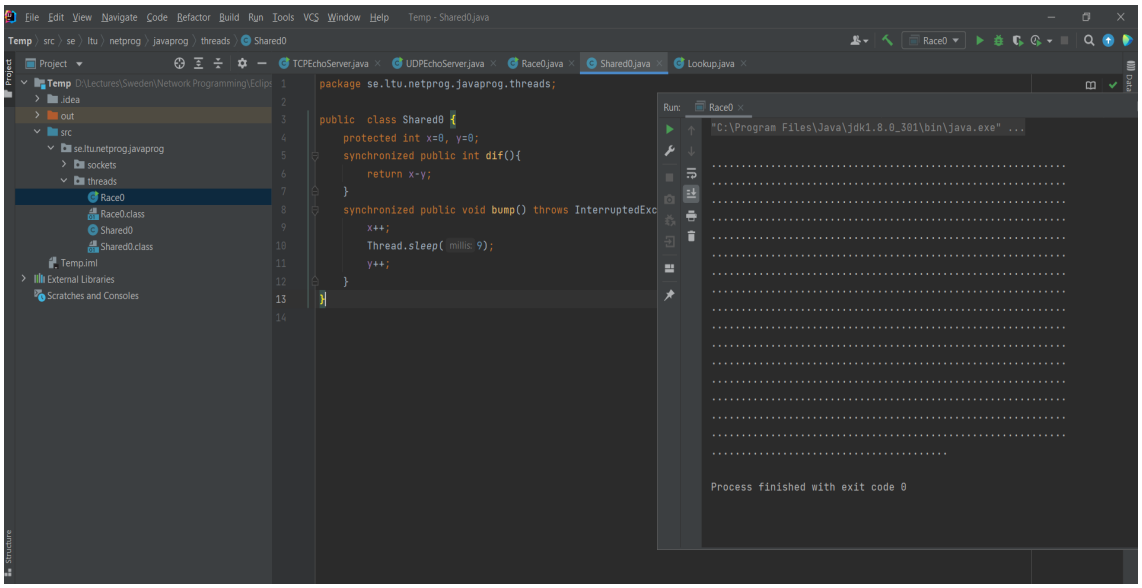
The tasks in this part include:

1. Compile and debug the Lookup, TCPEchoServer and UDPEchoServer classes. **(Done)**
2. Be able to describe the details of the implementation of each class (to be tested during individual assessment by Evgeny).
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. Compile the class Race0 in the threads part of the project.
   i) What kind of behavior do you observe?
      (1) Both the threads are working simultaneously but one after another due to thread.sleep(). When one thread goes to sleep the other executes and so on.
   ii) Make now both dif() and bump() methods synchronized. Compile the classes and run.



   iii) How is the behavior 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.



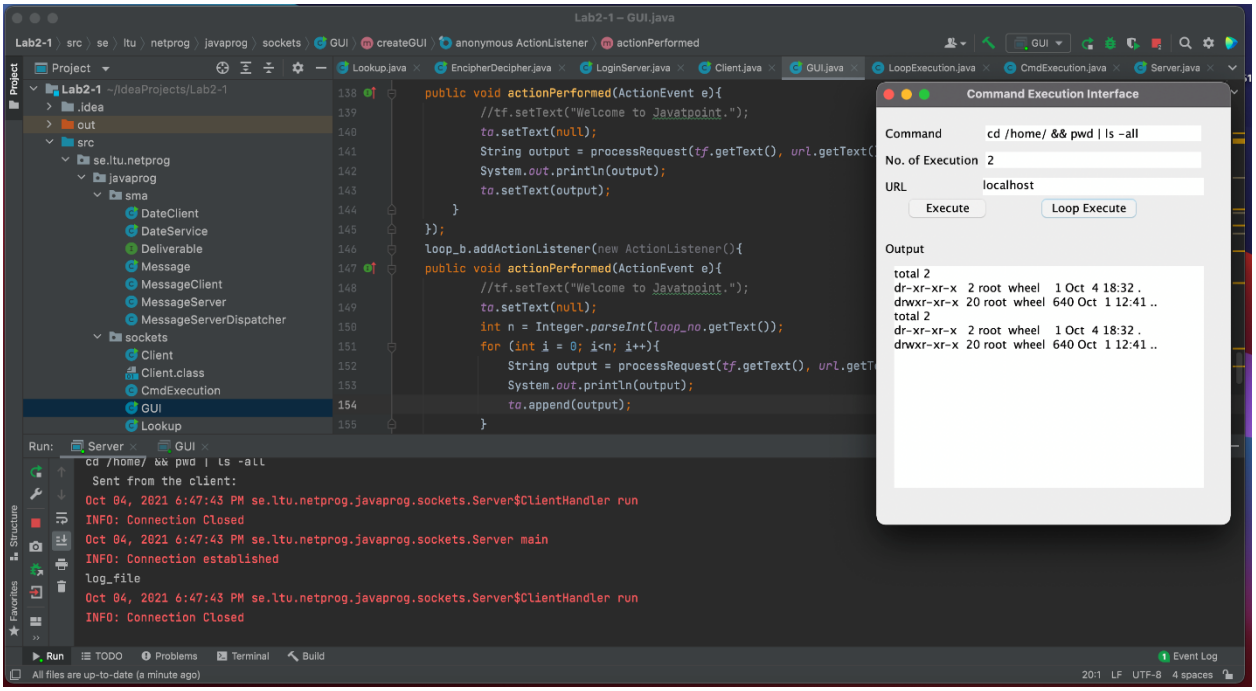


Part III : Client-server application :

The task of this part is to modify Part I of this Lab 2 so that:

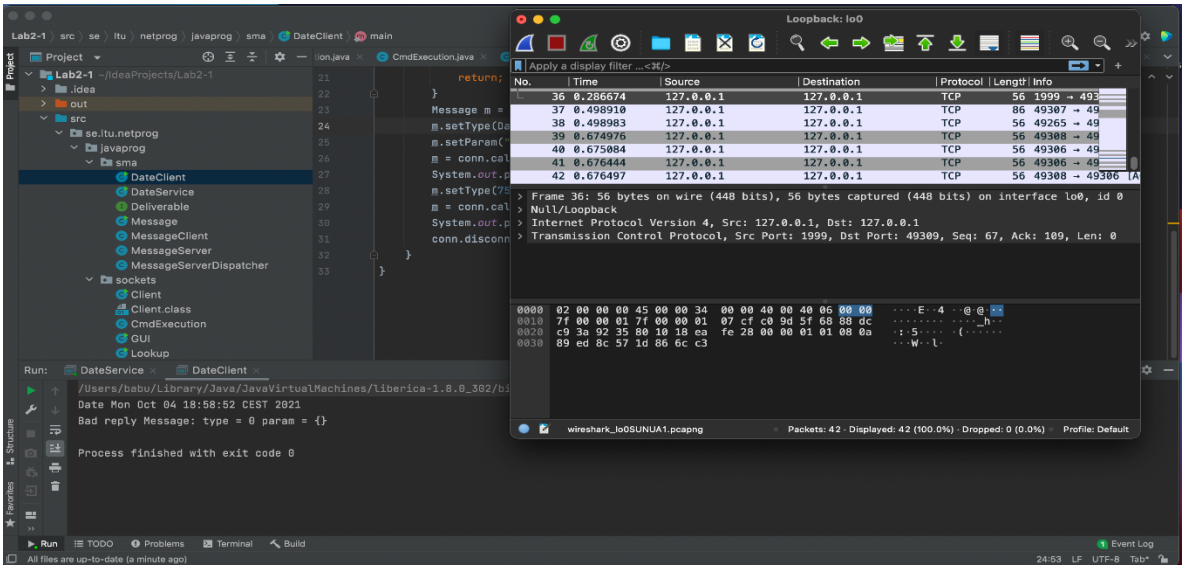
1. The GUI runs as a client-side application. Add an additional field to your GUI where the user can enter the URL of the server-side application. **(Done)**
2. The entered commands should be executed and run as the server-side application. You can implement these assignments in either of the following 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 capability 3,4 to the server-side application. **(Done)**

4. The application should take an argument defining which level of debug should be logged (warning, info, error, debug). **(Done)**
5. When adding different log levels to functions you must explain why you add this level of logging to the specific function. Write this explanation as a comment in your source code. **(Done)**



Part IV – Java netprog patterns – Simple Messaging Architecture:

1. Compile and install the classes in the SMA module of the project. **(Done)**
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.