|  |  |
| --- | --- |
| **Grade:** | X / 100 |
| **Instructor’s Comments:** | To be filled by the instructor |

This lab template is developed per data collection requirement for ABET accreditation process. Your cooperation of using it is highly appreciated. ***Please do not delete this page!***

**Grading Criteria**:

Each lab is worth 100 points. The first 20 points are toward your submission of required documents, and the rest 80 points are for the problem of each lab assignment.

|  |  |  |
| --- | --- | --- |
| **Item** | **Description** | **Points** |
| Required documents | 1) The lab report (this document)  **2) Any source code files (Don’t forget to compress them into a “.zip” file!)** | 20 |
| For all exercises/problems | Completeness of your work  1) Complete solution  2) Inclusion of screenshot of executing your code – you must include at least one whether your code works or not (if applicable)  Correctness of your logic/solution  Coding style (below are common mistakes)  1) Proper alignment of your code  2) Proper naming convention  3) Meaningful naming | 80 |

**ITS 330 – Advanced Operating Systems**

**Lab 04**

**Due: February 25th, 2021**

**Laquon Hamilton**

(***Create a table of contents before your solution***)

Contents

[Problem 1. 3](#_Toc65172166)

[Problem 2. 9](#_Toc65172167)

[Problem 3. 9](#_Toc65172168)

[Problem 4. 10](#_Toc65172169)

[Problem 5. 10](#_Toc65172170)

[Problem 6. 10](#_Toc65172171)

[Problem 7. 10](#_Toc65172172)

[Problem 8. 10](#_Toc65172173)

[Problem 9. 11](#_Toc65172174)

# Problem 1.

Code:

Part 1

/\*

 ITS-330

 Lab 04

 Problem 1

 2/25/2021

 Laquon Hamilton

\*/

public class ClientTest

{

  //confirming the messages sent between server and client

  public void TestClient()

  {

    System.out.print("\nMessage to send to server: ");

  }

  public void TestServer()

  {

    System.out.print("\nMessage received from client: ");

  }

}

Part 2

/\*

 ITS-330

 Lab 04

 Problem 1

 2/25/2021

 Laquon Hamilton

\*/

import java.net.\*;

import java.io.\*;

public class PrintServer extends ClientTest //Server program, run before PrintClient

{

//initialize socket and input stream + port and max connection variables

private Socket socket = null;

private ServerSocket server = null;

private DataInputStream in = null;

private int port;

  private int MaxConnections;

  public int getPort()

  {

    return port;

  }

  public void setPort(int newPort)

  {

    this.port = newPort;

  }

  public int getMaxConnections()

  {

    return MaxConnections;

  }

  public void setMaxConnections(int newMax)

  {

    this.MaxConnections = newMax;

  }

public PrintServer(int port)

{

// starts the server and waits for a connection

try

{

server = new ServerSocket(port);

System.out.println("Server started.");

System.out.println("\nListening for a client ...");

socket = server.accept();

System.out.println("\nNew Client found.");

// takes input from the client socket

in = new DataInputStream(new BufferedInputStream(socket.getInputStream()));

String line = "";

// reads message from client until "exit" is sent

while (!line.equals("exit"))

{

try

{

  super.TestServer();

line = in.readUTF();

System.out.println(line);

}

catch(IOException i)

{

System.out.println(i);

}

}

System.out.println("\nClosing connection...");

// close connection and stream

socket.close();

in.close();

}

catch(IOException i)

{

System.out.println(i);

}

}

public static void main(String args[])

{

@SuppressWarnings("unused") //object not needed inside main method

    PrintServer server = new PrintServer(5000);

}

}

Part 3

/\*

 ITS-330

 Lab 04

 Problem 1

 2/25/2021

 Laquon Hamilton

\*/

import java.net.\*;

import java.io.\*;

import java.util.Scanner;

public class PrintClient extends ClientTest //Client program, run after PrintServer

{

// initialize socket and input output streams + host variable

private Socket socket = null;

private Scanner input = null;

private DataOutputStream out = null;

private String host;

  public String getHost()

  {

    return host;

  }

  public void setHost(String newHost)

  {

    this.host = newHost;

  }

public PrintClient(String address, int port)

{

// create connection

try

{

socket = new Socket(address, port);

System.out.println("Connection successful.");

// takes input from terminal

input = new Scanner(System.in);

// sends output to the socket

out = new DataOutputStream(socket.getOutputStream());

}

catch(UnknownHostException u)

{

System.out.println(u);

}

catch(IOException i)

{

System.out.println(i);

}

// string to read message from input

String line = "";

// keep reading until "Over" is input

while (!line.equals("exit"))

{

try

{

  super.TestClient();

line = input.nextLine();

System.out.println("\nMessage sent.");

out.writeUTF(line);

}

catch(IOException i)

{

System.out.println(i);

}

}

// close connections and streams

System.out.println("\nClosing connection...");

try

{

input.close();

out.close();

socket.close();

}

catch(IOException i)

{

System.out.println(i);

}

}

public static void main(String args[])

{

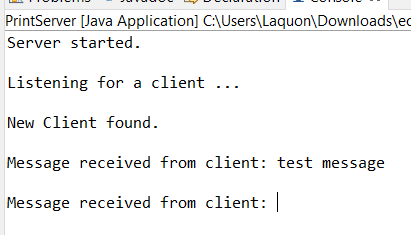
@SuppressWarnings("unused") //object not needed in main method

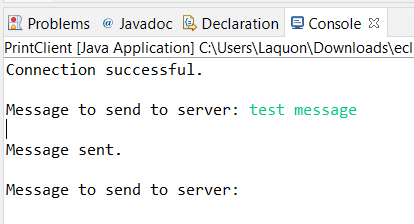
    PrintClient client = new PrintClient("127.0.0.1", 5000);

}

}

Screenshot





# Problem 2.

Code

Screenshot

# Problem 3.

Code:

Screenshot

# Problem 4.

Code

Screenshot

# Problem 5.

Code

Screenshot

# Problem 6.

Code

Screenshot

# Problem 7.

Code

Screenshot

# Problem 8.

Code

Screenshot

# Problem 9.

Code

Screenshot