Adventist University of Central Africa
P.O. Box 2461 Kigali, Rwanda | www.auca.ac.rw | info@auca.ac.rw

Faculty of Information Technology
Final Exam, 2019/2020 (1)
Course (Code) Name: COSC 313 TCP/IP Programming
Instructor: NIZEYIMANA PIERRE CELESTIN Exam Duration: 3 hrs
Group: Evening. Date: ……/……../2019

INSTRUCTIONS:

- This paper comprises 5 questions
- Read the questions carefully and provide precise answers

TCP/IP Programming / 40 marks

<u>Question1 / 15 marks</u>

The program below prints the broadcast address of the IP address of the host machine's "eth0". You are requested to re-write the program with minimum changes to it so that:

a) It prints both the IPv4 and IPV6 addresses of the Network Interface "eth0". / **5 marks**
b) It only prints the subnet mask of the IP v6 address of the Network Interface "eth0". / **5 marks**
c) It only prints the IP v4 associated with the "LocalHost" InetAddress object. / **5 marks**

```
public class Question1
{ public static void main(String[] args) throws Exception
  { NetworkInterface nif = NetworkInterface.getByName("eth0");
List<InterfaceAddress> list = nif.getInterfaceAddresses();
 for (InterfaceAddress iaddr : list)
     { InetAddress iaddress = iaddr.getAddress ();
if (iaddress instanceof Inet4Address)
System.out.println(iaddr.getBroadcast()); }}
```

NB: For each of the above 3 sub questions write a separate program.

```
import java.net.*;


public class Question1a {

  public static void main(String[] args) throws Exception {

    NetworkInterface nif = NetworkInterface.getByName("eth0");

    List<InterfaceAddress> list = nif.getInterfaceAddresses();
```

```
        for (InterfaceAddress iaddr : list) {

           InetAddress iaddress = iaddr.getAddress();

           if (iaddress instanceof Inet4Address) {

              System.out.println("IPv4 Broadcast Address: " + iaddr.getBroadcast());

           } else if (iaddress instanceof Inet6Address) {

              System.out.println("IPv6 Address: " + iaddress.getHostAddress());

           }

        }

    }

}
```

<u>Question2 / 5 marks</u>

The program below prints the entire list of all network interfaces of the local host. You are requested to re-write the program with minimum changes to it so that:

a) It only prints the Display Name of the 5th Network Interface on the list / **5 marks**

```
public class Question2 {
   public static void main(String[] args) throws SocketException {
ArrayList <NetworkInterface> interfaces = Collections
            .list(NetworkInterface.getNetworkInterfaces());
for(int i=0;i<=interfaces.size();i++)
System.out.println(interfaces.get(i));
}}
```

```
public class Question2 {
   public static void main(String[] args) throws SocketException {
      ArrayList<NetworkInterface> interfaces =
Collections.list(NetworkInterface.getNetworkInterfaces());
      if(interfaces.size() >= 5) {
         System.out.println(interfaces.get(4).getDisplayName());
      }
   }
}
```

<u>Question3 / 5 marks</u>

The program below prints an enumeration of InetAddresses associated with the 2nd Network Interface of the local host. You are requested to re-write the program with minimum changes to it so that:

a) It prints the list of interface addresses associated to the 2nd Network interface of the local host.

```
public class Question3 {
   public static void main(String[] args) throws SocketException {
ArrayList <NetworkInterface> interfaces = Collections
            .list(NetworkInterface.getNetworkInterfaces())
Enumeration <InetAddress> inet = interfaces.get(2).getInetAddresses();
while (inet.hasMoreElements())
System.out.println(inet.nextElement());}}
```

```
import java.net.*;
import java.util.*;

public class Question3 {
   public static void main(String[] args) throws SocketException {
      ArrayList<NetworkInterface> interfaces =
Collections.list(NetworkInterface.getNetworkInterfaces());
      NetworkInterface ni = interfaces.get(2);
      Enumeration<InetAddress> inetAddresses = ni.getInetAddresses();
      ArrayList<InetAddress> interfaceAddresses = new ArrayList<>();
      while (inetAddresses.hasMoreElements()) {
         InetAddress inetAddress = inetAddresses.nextElement();
         interfaceAddresses.add(inetAddress);
      }
      System.out.println("Interface Addresses of " + ni.getName() + ":");
      for (InetAddress addr : interfaceAddresses) {
         System.out.println(addr.getHostAddress());
      }
   }
}
```

Question4 / 5 marks

Assuming "**din**" is the input stream and "**dout**" is the output stream, explain the role of the server which runs the following java code fragment.

```
String a = null;
while (true)
{
a = din.readUTF ();
System.out.println ("Client:" + a);
dout.writeUTF (a);
System.out.println (); }
```

The Java code fragment represents a server-side implementation for a simple client-server communication protocol. The server waits for incoming messages from the client through the

input stream "din" using the "readUTF()" method, which reads the data from the input stream as a sequence of Unicode characters in the modified UTF-8 format.

Once the server receives the message, it prints the message to the console with the prefix "Client:" using the "println()" method. After that, the server writes the same message back to the client through the output stream "dout" using the "writeUTF()" method. This sends the message to the client in the same modified UTF-8 format as it was received.

Overall, the server's role is to receive messages from the client, display them on the console, and send them back to the client, acting as a mediator between the two ends of the communication channel. The server runs the code fragment in an infinite loop, ensuring that it continuously listens for incoming messages from the client.

Question 5 / 10 marks

The following statement instantiate a Date Object:
Date date = new Date ();
The date and time can be displayed using toString () method as follows:
System.out.println (date.toString ());

Write a java program whereby the client connects to the server and as soon as the connection is established the server sends to the client the current "date and time" . Write both the client and the server sides.

```
import java.io.IOException;
import java.io.OutputStream;
import java.net.ServerSocket;
import java.net.Socket;
import java.util.Date;

public class Server {
  public static void main(String[] args) {
    try {
      ServerSocket serverSocket = new ServerSocket(5000);
      System.out.println("Server is running and waiting for a client to connect...");
      Socket clientSocket = serverSocket.accept();
      System.out.println("Client connected from " +
clientSocket.getInetAddress().getHostName());

      // Send current date and time to the client
      OutputStream outputStream = clientSocket.getOutputStream();
      Date currentDate = new Date();
```

```
            outputStream.write(currentDate.toString().getBytes());
            outputStream.flush();
            outputStream.close();
            clientSocket.close();
            serverSocket.close();
        } catch (IOException e) {
            e.printStackTrace();
        }
    }
}
```

```
import java.io.IOException;
import java.io.InputStream;
import java.net.Socket;

public class Client {
    public static void main(String[] args) {
        try {
            Socket socket = new Socket("localhost", 5000);
            InputStream inputStream = socket.getInputStream();
            byte[] buffer = new byte[1024];
            int bytesRead = inputStream.read(buffer);
            String currentDateAndTime = new String(buffer, 0, bytesRead);
            System.out.println("Current date and time received from server: " + currentDateAndTime);
            inputStream.close();
            socket.close();
        } catch (IOException e) {
            e.printStackTrace();
        }
    }
}
```

<<Best of Luck>>