Computer Network 1

LAB 4

Socket Programming in Java: Chat Application

Student Name: Alexandre Rousseau

Student ID: 1952001

Exercise 1:

```java
public class download {

  public static void DownloadWebPage(String webpage)
  {
    try {

      // Create URL object
      URL url = new URL(webpage);
      BufferedReader readr =
      new BufferedReader(new InputStreamReader(url.openStream()));

      // Enter filename in which you want to download
      BufferedWriter writer =
      new BufferedWriter(new FileWriter("Download.html"));

      // read each line from stream till end
      String line;
      while ((line = readr.readLine()) != null) {
        writer.write(line);
      }

      readr.close();
      writer.close();
      System.out.println("Successfully Downloaded.");
    }

    // Exceptions
    catch (MalformedURLException mue) {
      System.out.println("Malformed URL Exception raised");
    }
    catch (IOException ie) {
      System.out.println("IOException raised");
    }
  }
  public static void main(String args[])
    throws IOException
  {
    String url = "google.com";
    DownloadWebPage(url);
  }
}
```

Exercise 3:

```java

import java.io.*;
import java.util.*;
import java.net.*;

// Server class
public class Server
{

  // Vector to store active clients
  static Vector<ClientHandler> ar = new Vector<>();

  // counter for clients
  static int i = 0;

  public static void main(String[] args) throws IOException
  {
    // server is listening on port 1234
    ServerSocket ss = new ServerSocket(1234);

    Socket s;

    // running infinite loop for getting
    // client request
    while (true)
    {
      // Accept the incoming request
      s = ss.accept();

      System.out.println("New client request received : " + s);

      // obtain input and output streams
      DataInputStream dis = new DataInputStream(s.getInputStream());
      DataOutputStream dos = new DataOutputStream(s.getOutputStream());

      System.out.println("Creating a new handler for this client...");

      // Create a new handler object for handling this request.
      ClientHandler mtch = new ClientHandler(s,"client " + i, dis, dos);

      // Create a new Thread with this object.
      Thread t = new Thread(mtch);

      System.out.println("Adding this client to active client list");

      // add this client to active clients list
      ar.add(mtch);

      // start the thread.
      t.start();

      // increment i for new client.
      // i is used for naming only, and can be replaced
      // by any naming scheme
      i++;

    }
  }
}
```

```java
// ClientHandler class
class ClientHandler implements Runnable
{
  Scanner scn = new Scanner(System.in);
  private String name;
  final DataInputStream dis;
  final DataOutputStream dos;
  Socket s;
  boolean isloggedin;

  // constructor
  public ClientHandler(Socket s, String name,
            DataInputStream dis, DataOutputStream dos) {
    this.dis = dis;
    this.dos = dos;
    this.name = name;
    this.s = s;
    this.isloggedin=true;
  }

  @Override
  public void run() {

    String received;
    while (true)
    {
      try
      {
        // receive the string
        received = dis.readUTF();

        System.out.println(received);

        if(received.equals("logout")){
          this.isloggedin=false;
          this.s.close();
          break;
        }

        // break the string into message and recipient part
        StringTokenizer st = new StringTokenizer(received, "#");
        String MsgToSend = st.nextToken();
        String recipient = st.nextToken();

        // search for the recipient in the connected devices list.
        // ar is the vector storing client of active users
        for (ClientHandler mc : Server.ar)
        {
          // if the recipient is found, write on its
          // output stream
          if (mc.name.equals(recipient) && mc.isloggedin==true)
          {
            mc.dos.writeUTF(this.name+" : "+MsgToSend);
            break;
          }
        }
      } catch (IOException e) {

        e.printStackTrace();
      }

    }
    try
    {
      // closing resources
      this.dis.close();
      this.dos.close();

    }catch(IOException e){
      e.printStackTrace();
    }
  }
```