Programming Assignment using the Sockets API

This is a group assignment. **Total points: 50**

Use the Echo client and Echo server as your template to complete this programming assignment. The Echo client can be found at: https://docs.oracle.com/javase/tutorial/displayCode.html?code=https://docs.oracle.com/javase/tutorial/networking/sockets/examples/EchoClient.java
and the Echo server can be found at:
https://docs.oracle.com/javase/tutorial/displayCode.html?code=https://docs.oracle.com/javase/tutorial/networking/sockets/examples/EchoServer.java

1. Get the Echo client and sever code to run and communicate with each other. Use the Java IDE of your choice (e.g., Eclipse for Java at https://www.eclipse.org/downloads/packages/)
   Use localhost or 127.0.0.1 for the server's IP address that you seek to connect to from the client.

   For e.g., the client-side code seeks to connect to the server by creating a socket as:

   Socket socket = new Socket("127.0.0.1", 4000)

   The first argument – IP address of Server that the client wants to connect to (127.0.0.1 is the IP address of localhost, where code will run on the single stand-alone machine).
   The second argument – TCP Port. (Just a number representing which application to run on a server.) For example, HTTP runs on port 80. Select any port number between 2000 – 4000 for the server to run on and the client then seeks to connect to that port number as in the code above.

2. Now modify the client code to send 3 separate requests to the server for information as in:
   a. "Date" (server will respond with the current date/time by using the "date" command)
   b. "Uptime" (server will respond its uptime information by using the "uptime" command)
   c. "Memory" (server will respond with its available memory information by using the "free" command)

   In each case:

- Send that command request to the server on the host
- Get response back from server
- Display response at the client side

The client code can look similar to:

```
BufferedReader input = new BufferedReader(new
InputStreamReader(clientSocket.getInputStream()));

PrintWriter out = new PrintWriter(clientSocket.getOutputStream(), true);
System.out.println("Uptime request sent to server");
out.println("uptime");

String answer;
while((answer = input.readLine()) != null)
        System.out.println(answer);
```

You will also need to modify the server code to send a response for each of the 3 requests.

As an example, to send the uptime from the server to the client for part *b* you could something similar to this code:

```
String uptimeServer = "uptime";
Process uptimeStat = Runtime.getRuntime().exec(uptimeServer);
BufferedReader readingUptime = new BufferedReader(new
InputStreamReader(uptimeStat.getInputStream()));
//Make the string and send it back
String uptimeReading = "";
uptimeReading = readingUptime.readLine();
writer.println(uptimeReading); //send to the client, where //writer is the name of your
socket on the server side
```

Or you can use this code on the server side:

```
static DataInputStream inputFromClient;
static DataOutputStream outputToClient;

ServerSocket server;

Socket clientConnection;
int portNum = 5000; // Preset port number
BufferedReader input;

server = new ServerSocket(portNum);

System.out.print("Waiting for connection");
clientConnection = server.accept();
inputFromClient = new DataInputStream(clientConnection.getInputStream());
outputToClient = new DataOutputStream(clientConnection.getOutputStream());
```

```
Process process;
String output;
BufferedReader br;
process = Runtime.getRuntime().exec("uptime");
br = new BufferedReader(new InputStreamReader(process.getInputStream()));
while ((output = br.readLine()) != null)
        outputToClient.writeChars(output);
```

Do the same thing using command “date” for part *a*, and “free” for part *c*.

3. Provide a screen shot of the client-side display in each of the 3 cases. Include those 3 screen shots and your client code and server code in your submission in Canvas.