Laboration 8 – TCP/IP, klientsidan

Syftet med labborationen är att du ska träna på att skriva program som kopplar upp mot server-applikationer.

På MAH kan du jobba med uppgifterna på skolans datorer eller via mobilt bredband. Eduroam blockerar portar i allmänhet och kan inte användas.

Uppgift 1 – Frågesport

En server som levererar frågor / bedömer svar på frågor finns på ip=195.178.227.53, port=1494.

Programmet ***DemoQuizClient1*** demonstrerar en klient vilken kopplar upp mot servern för att erhålla en fråga / få ett svar bedömt. Din uppgift är att studera programmet med hjälp av nedanstående kommentarer.

I ***run***-metoden får användaren mata in en sträng vilken består av två eller tre delar separerade med kolon:

”QUESTION,13” // Fråga 13
”ANSWER,13,1934” // Svar på fråga 13 är 1934

Strängen delas upp och en sträng-array skapas (metoden *split*):

”QUESTION,13” ger arrayen {”QUESTION”,”13”} och

”ANSWER,13,1934” ger arrayen {”ANSWER”,”13”,”1934”}

Därefter anropas antingen metoden *question()* eller metoden *answer()*.

Erhålla fråga

I metoden ***question()*** så kan du se hur:

Uppkoppling sker mot servern och hur kommunikationsströmmar skapas (*dis* resp *dos*)

Klienten tar emot en välkomsthälsning från servern (***dis.readUTF()***).

Klienten skickar element 0 i arrayen som sträng och element 1 i arrayen som int
dos.writeUTF(parts[0])
dos.writeInt(Integer.parseInt(parts[1]));
dos.flush();

Servern svarar med request (här ”QUESTION”), frågenummer och en fråga.

String request = dis.readUTF();
int nbr = dis.readInt();
String question = dis.readString();

Erhålla bedömning av svar

Metoden ***answer()*** liknar metoden question men har en skillnad. Efter att frågans nummer är skickad så skickas ytterligare en int – svaret på frågan (ett årtal).

dos.writeUTF(parts[0])
dos.writeInt(Integer.parseInt(parts[1]));
dos.writeInt(Integer.parseInt(parts[2]));
dos.flush();

Klienten tar sedan emot info på motsvarande sätt som i metoden question():

String request = dis.readUTF();
int nbr = dis.readInt();
String answer = dis.readString();

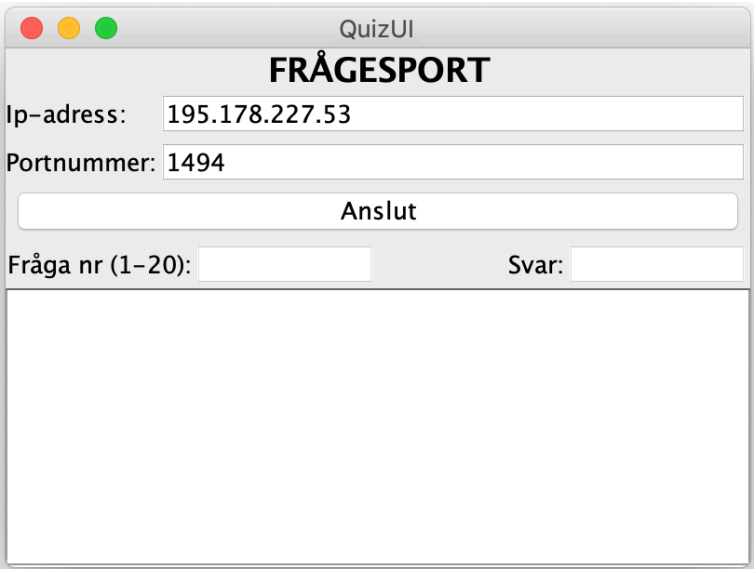
Glöm inte att testköra applikationen!

Uppgift 2 – Frågesport

I uppgift 1 sker uppkoppling mot servern vid varje förfrågan. Men det är inte nödvändigt, det går bra att koppla upp och sedan göra så många förfrågningar man önskar innan nerkoppling sker. Uppgift 2 går ut på att du ska skriva en klient som kopplar upp mot servern och som användaren kan använda så länge som önskas för att därefter kopplas ner.

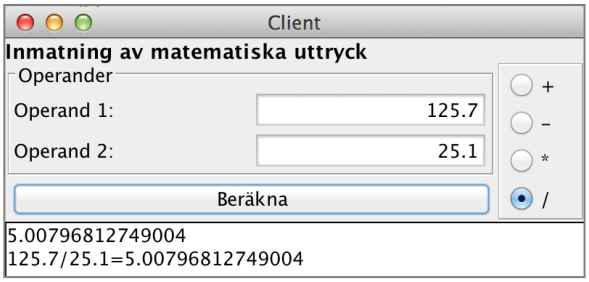
Du kan bygga en menybaserad klient där användaren kan välja det som ska ske:

Om du vill bygga en klient med ett grafiskt användargränssnitt kan du utgå från ***QuizUI.java*** eller göra ett eget UI. I QuizUI kopplas JButton-komponenten mot en instans av *ConnectListener*. JTextField-komponenten för frågenummer kopplas mot en instans av *Question* (*actionPerformed* anropas om komponenten har fokus och användaren klickar på ENTER) och JTextField-komponenten för årtal kopplas mot en instans av *Answer* (*actionPerformed* anropas om komponenten har fokus och användaren klickar på ENTER)



Uppgift 3A – Kalkylera

I denna uppgift ska du koppla upp mot en server som hjälper till med beräkningar.



- IP-adressen till servern – **195.178.227.53**
- Porten som du ska ansluta på – **721**
- Kommunikationen ska ske med klasserna ***Socket***, ***DataInputStream*** och ***DataOutputStream***.

Kommunikationen med servern ska ske på följande sätt då en beräkning ska utföras.

1. Koppla upp mot servern via Socket.
2. Överför en sträng till servern med writeUTF. Strängen ska vara på formen `tal1,tal2,räknesätt` , t.ex. `12.5,3.45,+`
3. Läs serverns svar med readUTF och visa resultatet på lämpligt sätt.
4. Koppla ner förbindelsen

Till din hjälp då du löser uppgiften har du ***CalcUI.java*** (så du slipper bygga ett GUI) och dessutom kan du snegla på *ClientA* från föreläsningen.

Uppgift 3B – Kalkylera

Denna uppgift är i stort samma som uppgift 3A men i denna uppgift ska klienten koppla upp sig mot servern så programmet startar och hålla uppkopplingen under tiden programmet är igång.

- IP-adressen till servern – **195.178.232.7**
- Porten som du ska ansluta på – **722**
- Kommunikationen ska ske med klasserna ***Socket***, ***DataInputStream*** och ***DataOutputStream***.

Kommunikationen med servern ska ske med hjälp av writeUTF och readUTF och strängen som överförs till servern ska ha samma format som i uppgift 2A.

I din lösning ska du ha en separat tråd för att lyssna efter svar från servern (readUTF).

Då du löser uppgiften kan du snegla på *ClientB* från föreläsningen.

Uppgift 3C – Kalkylera

Denna uppgift är i stort samma som uppgift 3B men kommunikationen med servern ska ske på ett lite annorlunda sätt.

- IP-adressen till servern – **195.178.232.7**
- Porten som du ska ansluta på – **723**
- Kommunikationen ska ske med klasserna ***Socket***, ***DataInputStream*** och ***DataOutputStream***.

Kommunikationen med servern ska ske med hjälp av anropen:

writeDouble(nbr1)
writeDouble(nbr2)
writeChar(operation)

En separate tråd ska lyssna efter svar från server (readUTF).

Då du löser uppgiften kan du snegla på *ClientC* från föreläsningen.

Uppgift 3D – Kalkylera

Denna uppgift är i stort samma som uppgift 3C men kommunikationen med servern ska ske på ett lite annorlunda sätt. I din lösning ska du använda *objektströmmar*.

- IP-adressen till servern – **195.178.232.7**
- Porten som du ska ansluta på – **724**
- Kommunikationen ska ske med klasserna ***Socket***, ***ObjectInputStream*** och ***ObjectOutputStream***.

Kommunikationen med servern ska ske med hjälp av ***writeObject(expression)***. Ett objekt av typen ***laboration8.Expression*** ska överföras till servern (innehåller instansvariabler för två double och en char). OBS! Du måste använda klassen som bifogas med laborationen.

En separat tråd ska lyssna efter ett svar med med metoden ***readObjekt()***. Resultatet vid läsningen är ett objekt av typen ***laboration8.Calculation*** (krävs typkonvertering).

```
Calculation calc;
:
calc = (Calculation)oos.readObject(); // Kan kasta ClassNotfoundException, try-sats behövs
```

OBS! Du måste använda klassen som bifogas med laborationen.

Då du löser uppgiften kan du snegla på *ClientD* från föreläsningen.

Lösningar

Uppgift 2, meny-baserad

```
public class QuizClient2 {
    private Socket socket;
    private DataInputStream dis;
    private DataOutputStream dos;
    private String ip;
    private int port;
    private boolean connected;

    public QuizClient2(String ip, int port) {
        this.ip = ip;
        this.port = port;
        new Send().start();
    }

    private void connect() {
        if(!connected) {
            try {
                socket = new Socket(ip,port);
                dis = new DataInputStream(socket.getInputStream());
                dos = new DataOutputStream(socket.getOutputStream());
                new Listener().start();
                connected = true;
            } catch(IOException e) {
                System.out.println("connect(): " + e);
            }
        }

    }

    private void disconnect() {
        if(connected) {
            try {
                socket.close();
            } catch(Exception e) {}
            connected = false;
        }
    }

    private void question() {
        int nbr;
        if(connected) {
            try {
                nbr = getInt("Ange frågans nummer (1-20)");
                dos.writeUTF("QUESTION");
                dos.writeInt(nbr);
                dos.flush();
                System.out.println("QUESTION skickad: " + nbr);
            } catch(Exception e) {
                System.out.println(e);
            }
        }
    }

    private void answer() {
        int nbr, answer;
        if(connected) {
            try {
                nbr = getInt("Ange frågans nummer (1-20)");
                answer = getInt("Ange årtal");
                dos.writeUTF("ANSWER");
                dos.writeInt(nbr);
                dos.writeInt(answer);
                dos.flush();
                System.out.println("ANSWER skickad: " + nbr + ", " + answer);
            } catch(Exception e) {
                System.out.println(e);
            }
        }
```

```
    }

    private class Send extends Thread {
        public void run() {
            String menu = "1. Koppla upp\n2. Koppla ner\n3. Hämta fråga\n4. Lämna
svar\n\n0. Avsluta";
            int choice = getInt(menu);
            while(choice!=0) {
                switch(choice) {
                case 1 : connect(); break;
                case 2 : disconnect(); break;
                case 3 : question(); break;
                case 4 : answer(); break;
                }
                choice = getInt(menu);
            }
            disconnect();
            System.exit(0);
        }
    }

    private int getInt(String txt) {
        int nbr = 0;
        boolean nbrOk = false;
        do {
            try {
                nbr = Integer.parseInt(JOptionPane.showInputDialog(txt));
                nbrOk = true;
            }catch(Exception e){}
        }while(!nbrOk);
        return nbr;
    }

    private class Listener extends Thread {
        public void run() {
            String request, response;
            int nbr;
            try {
                System.out.println("Listening");
                request = dis.readUTF();
                System.out.println(request);
                while(true) {
                    request = dis.readUTF();
                    nbr = dis.readInt();
                    response = dis.readUTF();
                    System.out.println(request+", fråga "+nbr+"\n"+response);
                }
            } catch(IOException e) {
                System.out.println(e);
            }
            System.out.println("Not listening");
            disconnect();
        }
    }

    public static void main(String[] args) {
        new QuizClient2("127.0.0.1",1494);
    }
}
```

Uppgift 2, UI-baserad

```
public class QuizController {
    private QuizUI ui = new QuizUI(this);
    private QuizClient client;

    public QuizController() {
        SwingUtilities.invokeLater(new Runnable() {
            public void run() {
                JFrame frame = new JFrame();
                frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
                frame.add(ui);
```

```
                frame.pack();
                frame.setVisible(true);
            }
        });
        client = new QuizClient(this);
    }

    // Metoder som QuizClient anropar ----------------------------
    public void newQuestion(String txt, int nbr) {
        ui.setText("Fråga: " + nbr + "\n" + txt);
        ui.enableAnswer();
    }

    public void newResponse(String txt, int nbr) {
        ui.appendText("\nResultat fråga " + nbr + ": " + txt);
        ui.enableQuestion();
    }

    public void connected(String txt) {
        ui.appendText("\n" + txt);
        ui.enableQuestion();
        ui.connected();
    }

    public void disconnected(String txt) {
        ui.setText(txt);
        disconnect();
    }
    // Metoder som QuizClient anropar ----------------------------

    // Metoder som QuizUI anropar ----------------------------
    public void question(String nbrStr) {
        int nbr;
        try {
            nbr = Integer.parseInt(nbrStr);
            client.question(nbr);
            ui.waitingForResponse();
        } catch(NumberFormatException e1) {
            ui.setText("Inmatning ska vara heltal (fråga nr): " + nbrStr);
        } catch(IOException e2) {
            ui.setText(e2.toString());
            disconnect();
        }
    }

    public void answer(String nbrStr, String answerStr) {
        try {
            int nbr = Integer.parseInt(nbrStr);
            int answer = Integer.parseInt(answerStr);
            client.answer(nbr,answer);
            ui.waitingForResponse();
        } catch (NumberFormatException e) {
            ui.appendText("\nInmatning ska vara heltal, Fråga nr: " + nbrStr + ",
Årtal: " + answerStr);
        } catch (IOException e) {
            ui.setText(e.toString());
            disconnect();
        }
    }

    public void connect(String ip, String port) {
        try {
            client.connect(ip,Integer.parseInt(port));
            ui.setText("Ansluter till: ip="+ip+", port="+port+"\n");
            ui.connected();
        } catch(Exception ex) {
            ui.setText(ex.toString());
        }
    }

    public void disconnect() {
        try {
```

```
            ui.disconnected();
            client.disconnect();
        } catch(IOException ex) {}
    }
    // Metoder som QuizUI anropar ----------------------------

    public static void main(String[] args) {
        SwingUtilities.invokeLater( new Runnable() {
            public void run() {
                QuizController prog = new QuizController();
            }
        });
    }
}


public class QuizClient {
    private QuizController controller;
    private Socket socket;
    private DataInputStream dis;
    private DataOutputStream dos;

    public QuizClient(QuizController controller) {
        this.controller = controller;
    }

    public void connect(String ip, int port) throws IOException {
        socket = new Socket(ip,port);
        dis = new DataInputStream(new
BufferedInputStream(socket.getInputStream()));
        dos = new DataOutputStream(new
BufferedOutputStream(socket.getOutputStream()));
        new Listener().start();
    }

    public void disconnect() throws IOException {
        if(socket!=null)
            socket.close();
    }

    public void question(int nbr) throws IOException {
        dos.writeUTF("QUESTION");
        dos.writeInt(nbr);
        dos.flush();
    }

    public void answer(int nbr, int answer) throws IOException {
        dos.writeUTF("ANSWER");
        dos.writeInt(nbr);
        dos.writeInt(answer);
        dos.flush();
    }

    private class Listener extends Thread {
        public void run() {
            String request;
            int nbr;
            String response;
            try {
                response = dis.readUTF(); // Välkomstmeddelande
                controller.connected(response);
                while(true) {
                    request = dis.readUTF();
                    nbr = dis.readInt();
                    response = dis.readUTF();
                    if("QUESTION".equals(request.toUpperCase())) {
                        controller.newQuestion(response,nbr);
                    } else {
                        controller.newResponse(response,nbr);
                    }
                }
            } catch(Exception e) {
                controller.disconnected(e.toString());
```

```
            }

        }
    }
}
```

Uppgift 3A

```
public class CalcController {
    private CalcClient client;
    private CalcUI ui = new CalcUI(this);

    private void showCalcUI() {
        SwingUtilities.invokeLater(new Runnable() {
            public void run() {
                JFrame frame = new JFrame("Client");
                frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
                frame.add(ui);
                frame.pack();
                frame.setVisible(true);
            }
        });
    }

    public CalcController(CalcClient client) {
        this.client = client;
        client.setCalcController(this);
        showCalcUI();
    }
    public void newCalculation(String nbr1, String nbr2, String operation) {
        try {
            client.newCalculation(nbr1+","+nbr2+","+operation);
        } catch (IOException e) {
            e.printStackTrace();
        }
    }
    public void newResponse(final String response) {
        SwingUtilities.invokeLater(new Runnable() {
            public void run() {
                ui.setResult(response);
            }
        });
    }
    public static void main(String[] args) {
        try {
            CalcClient clientA = new CalcClientA("195.178.227.53",721);
            new CalcController(clientA);
//            CalcClient clientB = new CalcClientB("195.178.227.53",722);
//            new CalcController(clientB);
//            CalcClient clientC = new CalcClientC("195.178.227.53",723);
//            new CalcController(clientC);
//            CalcClient clientD = new CalcClientD("195.178.227.53",724);
//            new CalcController(clientD);
        } catch (IOException e) {
            e.printStackTrace();
        }
    }
}

public interface CalcClient {
    public void newCalculation(String expression) throws IOException;
    public void setCalcController(CalcController calcController);
}

public class CalcClientA implements CalcClient {
    private CalcController controller;
    private String ip;
    private int port;
    private Socket socket;
    private DataInputStream dis;
```

```
    private DataOutputStream dos;
    public CalcClientA(String ip, int port) throws IOException {
        this.ip = ip;
        this.port = port;
    }

    public void setCalcController(CalcController controller) {
        this.controller = controller;
    }

    private void connect() throws IOException {
        socket = new Socket(ip,port);
        socket.setSoTimeout(5000);
        dis = new DataInputStream(new
BufferedInputStream(socket.getInputStream()));
        dos = new DataOutputStream(new
BufferedOutputStream(socket.getOutputStream()));
    }

    private void disconnect() throws IOException {
        socket.close();
    }
    public void newCalculation(String expression) throws IOException {
        connect();
        dos.writeUTF(expression);
        dos.flush();
        String response = dis.readUTF();
        controller.newResponse(response);
        disconnect();
    }
}
```

Uppgift 3B

```
public class CalcClientB implements CalcClient {
    private CalcController controller;
    private Socket socket;
    private DataInputStream dis;
    private DataOutputStream dos;
    public CalcClientB(String ip, int port) throws IOException {
        socket = new Socket(ip,port);
        dis = new DataInputStream(new
BufferedInputStream(socket.getInputStream()));
        dos = new DataOutputStream(new
BufferedOutputStream(socket.getOutputStream()));
        new Listener().start();
    }
    public void setCalcController(CalcController controller) {
        this.controller = controller;
    }
    public void newCalculation(String expression) throws IOException {
        dos.writeUTF(expression);
        dos.flush();
    }
    private class Listener extends Thread {
        public void run() {
            String response;
            try {
                while(true) {
                    response = dis.readUTF();
                    controller.newResponse(response);
                }
            } catch(IOException e) {}
            try {
                socket.close();
            } catch(Exception e) {}
            controller.newResponse("Klient kopplar ner");
        }
    }
}
```

Uppgift 3C

```
public class CalcClientC implements CalcClient {
    private CalcController controller;
    private Socket socket;
    private DataInputStream dis;
    private DataOutputStream dos;
    public CalcClientC(String ip, int port) throws IOException {
        socket = new Socket(ip,port);
        dis = new DataInputStream(new
BufferedInputStream(socket.getInputStream()));
        dos = new DataOutputStream(new
BufferedOutputStream(socket.getOutputStream()));
        new Listener().start();
    }
    public void setCalcController(CalcController controller) {
        this.controller = controller;
    }
    public void newCalculation(String expression) throws IOException {
        double nbr1=Double.NaN, nbr2=Double.NaN;
        char operation=' ';
        String[] parts = expression.split(",");
        try {
            nbr1 = Double.parseDouble(parts[0]);
            nbr2 = Double.parseDouble(parts[1]);
            operation = parts[2].charAt(0);
        } catch(Exception e) {
            controller.newResponse("Bad arguments: " + parts[0] + parts[2] +
parts[1]);
            return;
        }
        dos.writeDouble(nbr1);
        dos.writeDouble(nbr2);
        dos.writeChar(operation);
        dos.flush();
    }
    private class Listener extends Thread {
        public void run() {
            String response;
            try {
                while(true) {
                    response = dis.readUTF();
                    controller.newResponse(response);
                }
            } catch(IOException e) {}
            try {
                socket.close();
            } catch(Exception e) {}
            controller.newResponse("Klient kopplar ner");
        }
    }
}
```

Uppgift 3D

```
public class CalcClientD implements CalcClient {
    private CalcController controller;
    private Socket socket;
    private ObjectInputStream ois;
    private ObjectOutputStream oos;
    public CalcClientD(String ip, int port) throws IOException {
        socket = new Socket(ip,port);
        oos = new ObjectOutputStream(socket.getOutputStream());
        ois = new ObjectInputStream(socket.getInputStream());
        new Listener().start();
    }
    public void setCalcController(CalcController controller) {
        this.controller = controller;
    }
    public void newCalculation(String expression) throws IOException {
        double nbr1=Double.NaN, nbr2=Double.NaN;
        char operation=' ';
```

```
        String[] parts = expression.split(",");
        try {
            nbr1 = Double.parseDouble(parts[0]);
            nbr2 = Double.parseDouble(parts[1]);
            operation = parts[2].charAt(0);
        } catch(Exception e) {
            controller.newResponse("Bad arguments: " + parts[0] + parts[2] +
parts[1]);
            return;
        }
        oos.writeObject(new Expression(nbr1,nbr2,operation));
        oos.flush();
    }
    private class Listener extends Thread {
        public void run() {
            Calculation response;
            try {
                while(true) {
                    response = (Calculation)ois.readObject();
                    controller.newResponse(response.getResult()+"\n"+response);
                }
            } catch(Exception e) {}
            try {
                socket.close();
            } catch(Exception e) {}
            controller.newResponse("Klient kopplar ner");
        }
    }
}
```