Laboration 9 – TCP/IP, serversidan

Uppgift 1 – Tidsserver

Skriv en server, ***TimeServer***, som returnerar en sträng med datum och tid. Servern ska använda ***port 13***. Du kan använda dig av en instans av ***Date*** kombinerat med ***toString()-*** metoden för att få en sträng att skicka. Datum och tid kommer då vara inställningarna på din dator. Avsluta skrivningen av

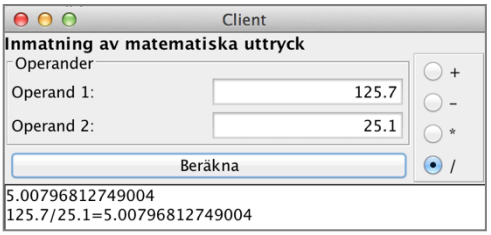
strängen med ett anrop till ***newLine();***
Servern ska vara en iterativ server vilken hanterar en klient åt gången.

Bl.a. följande klasser är användbara i din lösning:

ServerSocket, Socket, BufferedWriter, OutputStreamWriter, Date

Testa din lösning med programmet f8.Time (ändra ip till din dator).

Uppgift 2A – Kalkylera



I denna uppgift ska du skriva en server, ***CalcServerA***, som hjälper till med beräkningar. Klienten du skrev i Laboration 8 – Uppgift3A kan du använda då du testar servern.

- Servern ska lyssna efter anslutningar på port **721**
- Kommunikationen ska ske med klasserna ***Socket***, ***DataInputStream*** och ***DataOutputStream***.

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

1. Klienten kopplar upp sig mot servern.
2. Servern läser en sträng från klienten med readUTF. Strängen ska vara på formen `tal1,tal2,räknesätt` , t.ex. `12.5,3.45,+`
3. Servern delar upp strängen i delar (split), utför beräkning, och meddelar klienten med writeUTF.
   Meddelandet ska vara på formen: ”R\nT1OT2=R”, där R är resultatet av beräkningen, T1 är det första talet, T2 är det andra talet och O är räknesättet. Med värdena i punkt 2 ska svarssträngen vara: ”15.95\n12.5+3.45=15.95”
4. Koppla ner förbindelsen

Då du löser uppgiften kan du snegla på *LotteryServerB.java* från föreläsningen. Din lösning ska vara en *iterativ server*, dvs. en server med en endast en tråd.

Uppgift 2B – Kalkylera

Denna uppgift är i stort samma som uppgift 1A men i denna uppgift ska klienten koppla upp sig mot servern då programmet startar och hålla uppkopplingen under tiden programmet är igång. Klienten du skrev i Laboration 8 – Uppgift3B kan du använda då du testar servern.

- Servern ska lyssna efter anslutningar på port **722**
- Kommunikationen ska ske med klasserna ***Socket***, ***DataInputStream*** och ***DataOutputStream***.

Kommunikationen med klienten ska ske med hjälp av readUTF och writeUTF. Strängen som överförs till servern, och resultat-strängen som överförs till klienten, ska ha samma format som i Uppgift 1A.

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

Uppgift 2C – Kalkylera

Denna uppgift är i stort samma som uppgift 1B men kommunikationen med servern ska ske på ett lite annorlunda sätt. Klienten du skrev i Laboration 8 – Uppgift3C kan du använda då du testar servern.

- Servern ska lyssna efter anslutningar på port **723**
- Kommunikationen ska ske med klasserna ***Socket***, ***DataInputStream*** och ***DataOutputStream***.

Inläsning av tal och räknesätt från klienten ska ske med hjälp av anropen:

readDouble()
readDouble()
readChar(operation)

Svarssträngen ska samma format som i tidigare uppgifter. Då du löser uppgiften kan du snegla på *LotteryServerD* från föreläsningen. Din lösning ska vara en *flertrådad server* där varje klient hanteras av en nystartad tråd.

Uppgift 2D – Kalkylera

Denna uppgift är i stort samma som uppgift 1C men kommunikationen med klienten ska ske med hjälp av *objektströmmar*. Klienten du skrev i Laboration 8 – Uppgift 3D kan du använda då du testar servern. Filerna *Expresion.java* och *Calculation.java* ska du placera i paketet *laboration8*.

- Servern ska lyssna efter anslutningar på port **724**
- Kommunikationen ska ske med klasserna ***Socket***, ***ObjectInputStream*** och ***ObjectOutputStream***.

Servern ska läsa ett uttryck, av typen ***Expression***, från klienten med ***readObject()***. Ett objekt av typen ***Expression*** innehåller instansvariabler för två double och en char.

Sedan ska servern skriva ett objekt av typen ***Calculation*** som svar. Detta ska ske med ***writeObject(Calculation)***.

```
Calculation response = new Calculation(…);
:
oos.writeObject();
```

Då du löser uppgiften kan du snegla på *ClientD* från föreläsning 8 och din lösning på Laboration 8 – Uppgift 3D. Men nu är det serversidan du ska skriva.

Du kan själv välja typ av server (iterativ/flertrådad).

Lösningar

Uppgift 1

```
public class TimeServer extends Thread {
    public void run() {
        try (ServerSocket serverSocket = new ServerSocket(13)) {
            while(true) {
                try ( Socket socket = serverSocket.accept();
                        BufferedWriter bw = new BufferedWriter(new
OutputStreamWriter(socket.getOutputStream()))) {
                    bw.write(new Date().toString());
                    bw.newLine();
                    bw.flush();
                } catch(IOException e) {
                    System.out.println(e);
                }
            }
        } catch(IOException e) {
            System.out.println(e);
        }
    }

    public static void main(String[] args) {
        new TimeServer().start();
    }
}
```

Uppgift 2

Klassen Calculator används av samtliga servrar.

```
public class Calculator {
    public double calculate(double nbr1, double nbr2, char operation) {
        double res=Double.NaN;
        switch(operation) {
        case '+' : res = nbr1+nbr2; break;
        case '-' : res = nbr1-nbr2; break;
        case '*' : res = nbr1*nbr2; break;
        case '/' : res = nbr1/nbr2; break;
        }
        return res;
    }
}
```

Uppgift 2A

```
public class CalcServerA implements Runnable {
    private Calculator calculator;
    private ServerSocket serverSocket;

    public CalcServerA(Calculator calculator, int port) {
        try {
            this.calculator = calculator;
            serverSocket = new ServerSocket(port);
            new Thread(this).start();
        }catch(IOException e) {}
    }

    public void run() {
        String[] parts;
        String request, response="";
        double nbr1, nbr2, answer;
        char operation;

        System.out.println("Server startad");
        while(true) {
            try( Socket socket = serverSocket.accept();
                    DataInputStream dis = new
DataInputStream(socket.getInputStream());
                    DataOutputStream dos = new
DataOutputStream(socket.getOutputStream()) ) {
                request = dis.readUTF();
```

```
                parts = request.split(",");
                if(parts.length==3) {
                    try {
                        nbr1 = Double.parseDouble(parts[0]);
                        nbr2 = Double.parseDouble(parts[1]);
                        operation = parts[2].charAt(0);
                        answer = calculator.calculate(nbr1,nbr2,operation);
                        response = answer + "\n" +
                                parts[0] + parts[2] + parts[1] + "=" + answer;
                    } catch(Exception e) {
                        response = e.toString() + "\n" + parts[0] + parts[2] +
parts[1];
                    }
                } else {
                    response = "Fel antal argument: " + parts.length;
                }
                dos.writeUTF(response);
                dos.flush();
            } catch(IOException e) {
                System.err.println(e);
            }
        }
    }

    public static void main(String[] args) {
        new CalcServerA(new Calculator(),721);
    }
}
```

Uppgift 2B

```
public class CalcServerB {
    private Calculator calculator;

    public CalcServerB(Calculator calculator, int port) {
        this.calculator = calculator;
        new Connection(port).start();
    }

    private class Connection extends Thread {
        private int port;

        public Connection(int port) {
            this.port = port;
        }
        public void run() {
            Socket socket = null;
            System.out.println("Server startad");
            try (ServerSocket serverSocket = new ServerSocket(port)) {
                while(true) {
                    try {
                        socket = serverSocket.accept();
                        new ClientHandler(socket);
                    } catch(IOException e) {
                        System.err.println(e);
                        if(socket!=null)
                            socket.close();
                    }
                }
            } catch(IOException e) {
                System.err.println(e);
            }
            System.out.println("Server stoppad");
        }
    }

    private class ClientHandler extends Thread {
        private Socket socket;
        private DataInputStream dis;
        private DataOutputStream dos;

        public ClientHandler(Socket socket) throws IOException {
            this.socket = socket;
```

```
            dis = new DataInputStream(new BufferedInputStream( w
DataOutputStream(new BufferedOutputStream( socket.getOutputStream()));
            start();
        }

        public void run() {
            String[] parts;
            String request, response;
            double nbr1, nbr2, answer;
            char operation;
            try {
                while(true) {
                    request = dis.readUTF();
                    parts = request.split(",");
                    if(parts.length==3) {
                        try {
                            nbr1 = Double.parseDouble(parts[0]);
                            nbr2 = Double.parseDouble(parts[1]);
                            operation = parts[2].charAt(0);
                            answer = calculator.calculate(nbr1,nbr2,operation);
                            response = answer + "\n" +
                                    parts[0] + parts[2] + parts[1] + "=" + answer;
                        } catch(Exception e) {
                            response = e.toString() + "\n" + parts[0] + parts[2] +
parts[1];
                        }
                    } else {
                        response = "Fel antal argument: " + parts.length;
                    }
                    dos.writeUTF(response);
                    dos.flush();
                }
            } catch(IOException e) {
                try {
                    socket.close();
                } catch(Exception e2) {}
            }
            System.out.println("Klient nerkopplad");
        }
    }

    public static void main(String[] args) {
        new CalcServerB(new Calculator(),722);
    }
}
```

Uppgift 2C

```
public class CalcServerC {
    private Calculator calculator;

    public CalcServerC(Calculator calculator, int port) {
        this.calculator = calculator;
        new Connection(port).start();
    }

    private class Connection extends Thread {
        private int port;

        public Connection(int port) {
            this.port = port;
        }
        public void run() {
            Socket socket = null;
            System.out.println("Server startad");
            try (ServerSocket serverSocket = new ServerSocket(port)) {
                while(true) {
                    try {
                        socket = serverSocket.accept();
                        new ClientHandler(socket);
                    } catch(IOException e) {
                        System.err.println(e);
                        if(socket!=null)
```

```
                            socket.close();
                        }
                    }
                } catch(IOException e) {
                    System.err.println(e);
                }
                System.out.println("Server stoppad");
            }
        }

        private class ClientHandler extends Thread {
            private Socket socket;
            private DataInputStream dis;
            private DataOutputStream dos;

            public ClientHandler(Socket socket) throws IOException {
                this.socket = socket;
                dis = new DataInputStream(new BufferedInputStream(
socket.getInputStream()));
                dos = new DataOutputStream(new BufferedOutputStream(
socket.getOutputStream()));
                start();
            }

            public void run() {
                String request, response;
                double nbr1, nbr2, answer;
                char operation;
                try {
                    while(true) {
                        nbr1 = dis.readDouble();
                        nbr2 = dis.readDouble();
                        operation = dis.readChar();
                        answer = calculator.calculate(nbr1,nbr2,operation);
                        response = answer + "\n" + nbr1 + operation + nbr2 + "=" +
answer;
                        dos.writeUTF(response);
                        dos.flush();
                    }
                } catch(IOException e) {
                    try {
                        socket.close();
                    } catch(Exception e2) {}
                }
                System.out.println("Klient nerkopplad");
            }
        }

        public static void main(String[] args) {
            new CalcServerC(new Calculator(),723);
        }
}
```

Uppgift 2D

```
public class CalcServerD {
    private Calculator calculator;

    public CalcServerD(Calculator calculator, int port) {
        this.calculator = calculator;
        new Connection(port).start();
    }

    private class Connection extends Thread {
        private int port;

        public Connection(int port) {
            this.port = port;
        }
        public void run() {
            Socket socket = null;
            System.out.println("Server startad");
            try (ServerSocket serverSocket = new ServerSocket(port)) {
```

```
                while(true) {
                    try {
                        socket = serverSocket.accept();
                        new ClientHandler(socket);
                    } catch(IOException e) {
                        System.err.println(e);
                        if(socket!=null)
                            socket.close();
                    }
                }
            } catch(IOException e) {
                System.err.println(e);
            }
            System.out.println("Server stoppad");
        }
    }

    private class ClientHandler extends Thread {
        private Socket socket;
        private ObjectInputStream ois;
        private ObjectOutputStream oos;

        public ClientHandler(Socket socket) throws IOException {
            this.socket = socket;
            ois = new ObjectInputStream(socket.getInputStream());
            oos = new ObjectOutputStream(socket.getOutputStream());
            start();
        }

        public void run() {
            Expression exp;
            double answer;
            try {
                while(true) {
                    try {
                        exp = (Expression)ois.readObject();
                        answer =
calculator.calculate(exp.getNbr1(),exp.getNbr2(),exp.getOperation());
                        oos.writeObject(new Calculation(answer,exp));
                        oos.flush();
                    } catch (ClassNotFoundException e) {}
                }
            } catch(IOException e) {
                try {
                    socket.close();
                } catch(Exception e2) {}
            }
            System.out.println("Klient nerkopplad");
        }
    }

    public static void main(String[] args) {
        new CalcServerD(new Calculator(),724);
    }
}
```