ESERCITAZIONE 1

Socket in Java senza connessione

DAVIDE DI MOLFETTA

MIRKO LEGNINI

DANIELE NANNI CIRULLI

NATANAELE STAGNI

LORENZO VENERANDI

RS CLIENT

OBBIETTIVI:

- Comunicare al Discovery Server il nome del file di cui si vuole scambiare le righe.
- Ricevere il numero di porta (fornito dal Discovery Server) del Thread Swap Server a cui e' stato assegnato il file desiderato, per scambiare le righe.
- Comunicare le righe da scambiare a Swap Server e attendere l'esito dell'operazione.

RS CLIENT (SETUP)

Inizializzazione variabili

Controllo porte

Creazione DatagramSocket

```
DatagramSocket socket=null;
DatagramPacket packet=null;
byte[] buf=null;
InetAddress addr=null;
int port=-1;
ByteArrayInputStream biStream=null;
DataInputStream diStream = null;

try{
    if (args.length == 3){
        addr = InetAddress.getByName(args[0]);
        port = Integer.parseInt(args[1]);

        if(port<=1024 || port>=65536){
            System.out.println("Numero di porta non valido!\nSelezionare una porta compresa fra 1025 e 65535");
            System.exit(4);
        }
    }
    else{
        System.out.println("Utilizzo: java RSClient ipDS portDS fileName");
        System.exit(1);
    }

    buf=new byte[256];
    socket = new DatagramSocket();
    packet = new DatagramPacket(buf, buf.length, addr, port);
}
catch(UnknownHostException e){
    e.printStackTrace();
    System.exit(3);
}
catch (SocketException e) {
    e.printStackTrace();
    System.exit(2);
}
```

RS CLIENT (COMUNICAZIONE DS)

```
try{
    //Inzio codice Datagram
    BufferedReader stdIn = new BufferedReader(new InputStreamReader(System.in));

    // Inizializzazione Stream
    ByteArrayOutputStream boStream = new ByteArrayOutputStream();
    DataOutputStream doStream = new DataOutputStream(boStream);
    doStream.writeUTF(args[2]);

    // Invio del datagram al DiscoveryServer
    byte[] data = boStream.toByteArray();
    packet.setData(data);
    socket.send(packet);

    // Redirezione input
    packet.setData(buf);
    System.out.println("In attesa di risposta da DiscoveryServer ...");
    socket.receive(packet);

    biStream = new ByteArrayInputStream( packet.getData(),0,packet.getLength());
    diStream = new DataInputStream(biStream);

    // Reimpostazione packet
    if((port=diStream.readInt())<0){
        System.out.println("File non presente");
        System.exit(8);
    }

    packet.setPort(port);
```

RS CLIENT (COMUNICAZIONE RSS)

Wrapping

Esito operazione

```
String line;
System.out.println("Inserire l'indice delle righe da scambiare, separate da spazio\n^D(Unix)/^Z(W

while((line=stdIn.readLine())!=null){

    boStream=new ByteArrayOutputStream();
    doStream=new DataOutputStream(boStream);

    doStream.writeUTF(line.trim());
    data=boStream.toByteArray();

    packet.setData(data);
    socket.send(packet);

    // Ricezione conferma dal Serever
    packet.setData(buf);
    socket.receive(packet);

    biStream = new ByteArrayInputStream( packet.getData(),0,packet.getLength());
    diStream = new DataInputStream(biStream);

    if(diStream.readInt()<0){
        System.out.println("Operazione di swap fallita!");
    }
    else{
        System.out.println("Swap avvenuta con successo!");
    }
}
```

DISCOVERY SERVER

OBBIETTIVI:

- Creare dei Thread RowSwap associandoli ad un nome di file e ad un numero di porta.
- Mettersi in attesa di una richiesta dal Client.
- Comunicare al client il numero di porta del RowSwapServer associato al file indicato (dal client).

DS(SETUP)

Controllo porte

```java
RowSwapServer[] threadArray = new RowSwapServer[args.length/2];
int port=-1,tmp;

try{
    if(args.length%2==0){
        System.out.println("DiscoveryServer portaDiscoveryServer nomefile1 port1... \n(tutte coppie di argomenti fi
        System.exit(1);
    }

    port=Integer.parseInt(args[0]);
    if(port<=1024 || port>=65536){
        System.out.println("Numero di porta non valido!\nSelezionare una porta compresa fra 1025 e 65535");
        System.exit(2);
    }

    // Controllo porte duplicate
    for(int i=0;i<args.length;i+=2){
        for(int j=0;j<i;j+=2){
            if(args[i].equals(args[j])){
                System.out.println("Porte duplicate!");
                System.exit(3);
            }
        }
    }
```

```java
// Creazione thread
File tmpFile=null;
for(int i=1;i<args.length;i+=2){

    tmp=Integer.parseInt(args[i+1]);

    if((tmp<=1024 || tmp>=65536)){ // Controllo intervallo porte
        System.out.println("Numero di porta non valido!\nSelezionare una porta compresa fra 1025 e 65535");
        System.exit(2);
    }

    if(!((tmpFile=new File(args[i])).exists())){
        System.out.println("Impossibile aprire il file "+args[i]);  // Controllo apertura file
        System.exit(4);
    }
    threadArray[i/2]=new RowSwapServer(tmpFile,tmp);
}
for(RowSwapServer r:threadArray)    r.start();
```

Creazione ed esecuzione dei thread

DS(COMUNICAZIONE RS CLIENT)

```
while(true){

    packet.setData(buffer);
    socket.receive(packet);
    i=0;

    biStream= new ByteArrayInputStream(packet.getData(),0,packet.getLength());
    diStream= new DataInputStream(biStream);

    nomeFile=diStream.readUTF();

    data=new byte[4];
    boStream=new ByteArrayOutputStream();
    doStream=new DataOutputStream(boStream);

    // Identifico il thread con il nome del file
    while(i<threadArray.length && !threadArray[i].getFileName().equals(nomeFile)){
        i++;
    }

    if(i == threadArray.length){
        doStream.writeInt(-1);  //file non trovato
    }
    else{
        doStream.writeInt(threadArray[i].getPortNumber()); //File identificato. redirezione su Thread
    }

    data=boStream.toByteArray();
    packet.setData(data);
    socket.send(packet);
}
```

ROW SWAP SERVER

OBBIETTIVI:

- Ricevere dal Client il numero associato alle righe da scambiare.
- Scambiare le righe indicate.
- Comunicare l'esito dell'operazione al Client.

ROW SWAP SERVER (COSTRUTTORE)

```
public RowSwapServer(File file,int port){
    this.port=port;
    this.file=file;

    bufferThread=new byte[256];
    data= new byte[4];

    try{
        packetThread=new DatagramPacket(bufferThread, bufferThread.length);
        socketThread=new DatagramSocket(port);

        boStream=new ByteArrayOutputStream( );
        doStream=new DataOutputStream(boStream);

    }
    catch(Exception e){
        e.printStackTrace( );
        System.exit(666);
    }
}
```

ROW SWAP SERVER(DAEMON)

Estrazione del numero di riga

Restituisce l'esito tramite un codice

```
try{
    while(true){

        packetThread.setData(bufferThread);
        socketThread.receive(packetThread);

        //initMillis=System.currentTimeMillis();

        biStream=new ByteArrayInputStream(packetThread.getData(),0,packetThread.getLength());
        diStream=new DataInputStream(biStream);

        String[] tmpThread =diStream.readUTF().trim().split(" ");

        row1=Integer.parseInt(tmpThread[0].trim());
        row2=Integer.parseInt(tmpThread[1].trim());
        System.out.println(row1+"\t"+row2);

        boStream=new ByteArrayOutputStream();
        doStream=new DataOutputStream(boStream);

        if(swap()){
            doStream.writeInt(1);
        }else{
            doStream.writeInt(-1);
        }

        data=boStream.toByteArray();
        packetThread.setData(data);
        socketThread.send(packetThread);

        finalMillis=System.currentTimeMillis();

        System.out.println("Tempo impiegato: "+(finalMillis-initMillis));
    }
}catch(Exception e){
    e.printStackTrace();
    System.exit(1);
}
```

ROW SWAP SERVER

```java
public boolean swap() throws EOFException,FileNotFoundException,IOException{

    if(row1==row2) return true;
    if(row1>row2){
        int tmp2=row1;  //swap row1 con row2
        row1=row2;
        row2=tmp2;
    }

    File fout=new File("out");
    BufferedReader br=new BufferedReader(new FileReader(file));
    String line;
    String swapLine=null;
    int i=0;
    PrintWriter pw=new PrintWriter(fout);
```

Scambia le linee tramite un file d'appoggio.

Non si creano file infiniti!

```java
    while((line=br.readLine())!=null){
        i++;
        if(i==row1){
            swapLine=line;
            BufferedReader br2 = new BufferedReader(new FileReader(file));
            int j=0;

            while(j != row2){
                j++;
                if((line = br2.readLine())==null){
                    br2.close();
                    fout.delete();
                    return false;
                }
            }
            br2.close();
        }
        else if (i==row2){
            line=swapLine;
        }
        pw.println(line);
    }
    pw.close();
    br.close();

    if(i<row1){
        fout.delete();        // Riga scelta non esistente
        return false;
    }

    Files.move(fout.toPath(),file.toPath(), StandardCopyOption.REPLACE_EXISTING);

    return true;
}
```

CONCLUSIONI

- Controllo degli argomenti eseguito nel Main per evitare errori nei Thread.
- Client realizzato come filtro per scambiare piú linee in una sola esecuzione.
- RowSwap tramite file d'appoggio per non allocare troppa memoria in variabili.
- I processi server sono stati realizzati in maniera sequenziale.