Java Network Programming

Dave Price

Computer Science

University of Wales, Aberystwyth

TCP/UDP Socket Programming

- Classes etc. contained within java.net
- TCP connection support mostly provided by the class Socket and its methods.
- Socket also supported UDP connection but this is deprecated
- UDP support now provided mostly by Classes DatagramSocket and DatagramPacket

TCP Support

- Objects are created of the Class Socket to act as endpoints for your communication

```
Socket mySocket = null;
mySocket = new Socket("tigger",4000);
```

can then use the getInputStream() method of Socket to acquire an InputStream Object

Socket's Constructors

- Has multiple constructors, some now deprecated that were previously used with UDP sockets
- The easiest for us is
- public **Socket**(String *host*, int *port*) throws UnknownHostException, IOException;

Socket's Methods

- public InputStream **getInputStream**() throws IOException;
- public OutputStream **getOutputStream**() throws IOException;

Further Methods to access information about the connection

- public int **getPort**();
- public int **getLocalPort**();
- public InetAddress **getInetAddress**();
- public InetAddress **getLocalAddress**();

Methods to alter characteristics of the connection

- public void **setSoLinger**(boolean *on*, int *val*) throws SocketException;
- public int **getSoLinger**() throws SocketException;
- public void **setTcpNoDelay**(boolean *on*) throws SocketException;
- public boolean **getTCPNoDelay**() throws SocketException;

Blocking or non-Blocking I/O

- Can specify whether or not a read() method call on an InputStream associated with a socket will block for ever or return after a timeout if no data is available. After setting, a read() raises a java.io.InterruptedIOException if timeout expires

public synchronized void **setSoTimeout**(int *timeout*) throws SocketException;

- where the timeout is given in milli-seconds

public synchronized int **getSoTimeout**() throws SocketException

A Example Client

```
import java.io.*;
import java.net.*;

public class GetPoem {
    public static void main(String[] args)
        throws IOException {

        Socket mySocket = null;
        BufferedReader incoming = null;
```

```
try {
      mySocket = new Socket("stonkin", 4000);
      incoming = new BufferedReader(
            new InputStreamReader(
                  mySocket.getInputStream()));
} catch (UnknownHostException e) {
      System.err.println("Can't locate Host");
      System.exit(1);
} catch (IOException e) {
      System.err.println("IO exception
                  accessing Host");
      System.exit(1);
}
```

```
   String poemLine;

   while ( (poemLine = incoming.readLine())
                   != null) {
            System.out.println(poemLine);
   }

   incoming.close();
   mySocket.close();
  }
}
```

Some Exceptions

- running with no server on the port

```
moin% java GetPoem
IO exception accessing Host
moin%
```

- running with an incorrect hostname

```
moin% java GetPoem
Can't locate Host
moin%
```

A Successful Run

```
moin% java GetPoem
Lamb Poem service .
Mary had a little lamb
Its fleece was as white as snow
Everywhere that Mary went
The lamb was sure to go.
moin%
```

Server side Sockets

- Actually, in all the above examples the server that was running was written in "C" !
- But here is how one would write a simple poem server in Java

ServerSocket

- java.net provides a class called **ServerSocket** which is used to for server side sockets to which others will connect
- has various constructors, the mostly useful to us being

public **ServerSocket**(int *port*, int *backlog*) throws IOException;

Accepting incoming calls

- The ServerSocket class has a method called **accept** that allows one to wait for incoming connections. When it returns it gives you a **Socket** (not a ServerSocket) which is a normal communications endpoint.

```
public Socket accept() throws IOException;
```

Using the client socket

- accept has returned us a normal socket
- can therefore use any of the methods described earlier in conjunction with that socket
- in particular can use getInputStream() and getOutputStream()

Here's a Poem Server in Java

```
import java.io.*;
import java.net.*;

public class LambServer{
    public static void main(String[] args)
                 throws IOException {

        ServerSocket mySocket = null;
```

Creating our Server Socket

```
try {
  mySocket = new ServerSocket(4000,3);
} catch (IOException e) {
  System.err.println("IO exception on port");
  System.exit(1);
}

System.out.println("Have ServerSocket about
                     to wait for call");
```

Accepting An Incoming Call

```
Socket clientSocket = null;
try {
    clientSocket = mySocket.accept();
} catch (IOException e) {
    System.err.println("accept of incoming
                client call failed");
    System.exit(1);
}

System.out.println("Incoming Call
                Accepted");
```

Getting a PrintWriter

```
PrintWriter outgoing = null;

try {
   outgoing = new PrintWriter(
         clientSocket.getOutputStream(), true)
} catch (IOException e) {
  System.err.println(
         "getOutputStream failed");
  System.exit(1);
}
```

Sending the Poem

```
outgoing.println("Lamb Poem service .");
outgoing.println("Mary had a little lamb");
outgoing.println(
       "Its fleece was as white as snow");
outgoing.println("Everywhere that Mary went");
outgoing.println("The lamb was sure to go.");
```

And Tidying Up at the End

```
        outgoing.close();
        clientSocket.close();
        mySocket.close();
    }
}
```

Running the Java Client and Server

```
moin% java GetPoem
Lamb Poem service .
Mary had a little lamb
Its fleece was as white as snow
Everywhere that Mary went
The lamb was sure to go.
moin%

stonkin% java LambServer
Have ServerSocket about to wait for Call
Incoming Call Accepted
stonkin%
```