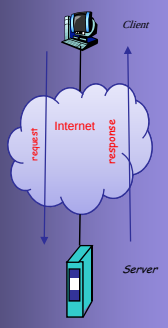

INTERNET PROTOCOLS AND CLIENT-SERVER PROGRAMMING SWE344

Fall Semester 2008-2009 (081)

Module 15: Network Programming using Java

Dr. El-Sayed El-Alfy
Computer Science Department
King Fahd University of Petroleum and Minerals
alfy@kfupm.edu.sa

Objectives

- Learn how to write java a program to resolve IP Addresses
- Learn how to write TCP C/S applications in java
- Lean how to handle Binary data in Java
- Learn how to write UDP C/S applications in java

Agenda

- Introduction
- Representing & Resolving IP Addresses
  - InetAddress, Inet4Address & Inet6Address classes
  - Example 1: IP Resolution
- Representing Socket EndPoint
  - SocketAddress & InetSocketAddress classes
- TCP C/S Programming
  - ServerSocket & Socket classes
  - Example 2: TCP Echo Server and Client
  - Exchanging Binary Data
  - Example 3: TCP File Server and Client
- UDP C/S Programming
  - DatagramSocket & DatagramPacket classes
  - Example 4: UDP Echo Server and Client

Introduction

- Java provides many classes in the java.net package for writing network applications using the underlying socket interface.
- Some of these classes are:

Address classes	Socket classes
InetAddress	ServerSocket
Inet4Address	Socket
Inet6Address	DatagramPacket
SocketAddress	DatagramSocket
InetSocketAddress	MulticastSocket

- This lecture shows how to use some of these classes to:
  - Represent and Resolve IP Addresses
  - Write TCP client-server application
  - Write UDP client-server application

Representing & Resolving IP Addresses

- The InetAddress class is a generic class for representing IP Addresses – can represent both IPv4 and IPv6.
- The subclasses, Inet4Address and Inet6Address are used if there is a need to use version-specific methods.
- InetAddress class also provides methods for resolving IP Addresses (through the DNS server).
- InetAddress has no constructor. Rather, InetAddress objects are obtained from factory methods.
- InetAddress represents both the hostname and the IP address. The toString() method of the class returns a string representation of the object as a pair in the form:

  www.microsoft.com/207.46.19.60

Representing & Resolving IP Addresses …

- Following are some of the classes of the InetAddress class:

byte[] getAddress()	Returns the raw IP address of this InetAddress object.
String getHostName()	Gets the host name for this IP address.
String getHostAddress()	Returns the IP address string in textual presentation. "hostname/IPAddress"
static InetAddress[] getAllByName(String host)	Given the name of a host, returns an array of its IP addresses
static InetAddress getByAddress(byte[] addr)	Returns an InetAddress object given the raw IP address.
static InetAddress getByAddress(String host, byte[] addr)	Creates an InetAddress based on the provided host name and IP address. No DNS is checked for the validity of the address.
static InetAddress getByName(String host)	Determines the IP address of a host, given the host's name.
String getCanonicalHostName ()	Gets the fully qualified domain name for this IP address.

Representing & Resolving IP Addresses …

static InetAddress getLocalHost()	Returns the local host.
boolean isAnyLocalAddress()	Checks if the InetAddress in a wildcard address.
boolean isLoopbackAddress()	Checks if the InetAddress is a loopback address.
boolean isMulticastAddress()	Checks if the InetAddress is an IP multicast address.

Example 1 : IP Resolution

```
import java.net.*;
import java.io.*;
public class IPAddressResolver {
  public static void main (String args[]) {
    try {
      BufferedReader stdin = new BufferedReader(new
                        InputStreamReader(System.in));
      InetAddress myself = InetAddress.getLocalHost();
      System.out.println("MyAddress is :"+myself);
      while (true) {
          System.out.print("Hostname to resolve - <exit> to quit: ");
          String input = stdin.readLine();
          if (input.equalsIgnoreCase("exit"))
                break;
          InetAddress address = InetAddress.getByName(input);
          System.out.println("IP Addrsss is: "+
              address.getHostAddress());
          System.out.println("Host name is: "+ address.getHostName());
          System.out.println("addresses for "+input+ " are: ");
          InetAddress[] addresses =
                    InetAddress.getAllByName(address.getHostName());
```

Example 1 : IP Resolution

```
22.             for (int i = 0; i < addresses.length; i++)
23.                     System.out.println(addresses[i]);
24.         }
25.       }
26.       catch (UnknownHostException e) {
27.         System.out.println("Exception: "+e);
28.       }
29.       catch (Exception ex) {
30.         System.out.println("Exception: "+ex);
31.       }
32.     }
33.   }
```

```
MyAddress is :ics-bmghandi/196.1.65.143

Enter host name to resolve - <exit> to quit: www.microsoft.com
IP Addrsss is: 207.46.20.60
Host name is: www.microsoft.com

Other addresses for www.microsoft.com are:
www.microsoft.com/207.46.20.60
www.microsoft.com/207.46.198.30
www.microsoft.com/207.46.198.60
www.microsoft.com/207.46.199.30
www.microsoft.com/207.46.225.60
www.microsoft.com/207.46.19.30
www.microsoft.com/207.46.19.60
www.microsoft.com/207.46.20.30

Enter host name to resolve - <exit> to quit: _
```


Representing EndPoint

- The SocketAddress class is an abstract class for representing Socket endpoints.
- The InetSocketAddress is a subclass of SocketAddress that represents endpoint for a socket associated with Internet address family (IPv4 or IPv6).
- The Constructors of InetSocketAddress are:

InetSocketAddress(InetAddress addr, int port)	Creates a socket address from an IP address and a port number.
InetSocketAddress(int port)	Creates a socket address where the IP address is the wildcard address and the port number a specified value.
InetSocketAddress(String hostname, int port)	Creates a socket address from a hostname and a port number.

- InetSocketAddress also has the following accessor methods :
  InetAddress getAddress() , String getHostName() & int getPort()

TCP C/S Programming

- Java provides two TCP Socket classes, ServerSocket class for creating a TCP Server and Socket class for creating a client.
- The following are the constructors and some of the methods of the ServerSocket class:

ServerSocket()	Creates an unbound server socket
ServerSocket(int port)	Creates a server socket, bound to the specified port
ServerSocket(int port, int backlog)	Creates a server socket and binds it to the specified local port number, with the specified backlog
ServerSocket(int port, int backlog, InetAddress bindAddr)	Create a server with the specified port, listen backlog, and local IP address to bind to
Socket accept()	Listens for a connection to be made to this socket and accepts it

TCP C/S Programming …

void bind(SocketAddress endpoint) void bind(SocketAddress endpoint, int backlog)	Binds the ServerSocket to a specific endpoint. Second version specifies a backlog.
InetAddress getInetAddress()	Returns the local address of this server socket.
int getLocalPort()	Returns the port on which this socket is listening
SocketAddress getLocalSocketAddress()	Returns the address of the endpoint this socket is bound to, or null if it is not bound yet
int getReceiveBufferSize()	Gets the size of the receive buffer,
void setReceiveBufferSize(int size)	Sets the size of the receive buffer
int getSoTimeout()	Gets the socket timeout. The default value of zero means no timeout (infinity)
void setSoTimeout(int timeout)	Gets the socket timeout in milliseconds.
void close()	Closes the socket.

TCP C/S Programming …

- The following are the constructors and some of the methods of the Server class:

Socket()	Creates an unconnected socket
Socket(InetAddress address, int port)	Creates a stream socket and connects it to the specified port and IP address
Socket(String host, int port)	Creates a stream socket and connects it to the specified port number on the named host
Socket(String host, int port, InetAddress localAddr, int localPort)	Creates a socket on the given local endpoint and connects it to the specified remote endpoint
void bind(SocketAddress bindpoint)	Binds the socket to a local address
void connect(SocketAddress endpoint) void connect(SocketAddress endpoint, int timeout)	Connects this socket to the server. The second version specifies timeout.

TCP C/S Programming …

InputStream getInputStream()	Returns an input stream for this socket
OutputStream getOutputStream()	Returns an output stream for this socket
int getLocalPort()	Returns the port on which this socket is listening
SocketAddress getLocalSocketAddress()	Returns the address of the endpoint this socket is bound to, or null if it is not bound yet
SocketAddress getRemoteSocketAddress()	Returns the address of the endpoint this socket is connected to or null if not connected
int getReceiveBufferSize()	Gets the size of the receive buffer,
void setReceiveBufferSize(int size)	Sets the size of the receive buffer
int getSoTimeout()	Gets the socket timeout. The default value of zero means no timeout (infinity)
void setSoTimeout(int timeout)	Gets the socket timeout in milliseconds.
void shutdownInput() void shutdownOutput()	Disables the input/output stream for this socket.
void close()	Closes the socket.

Example 2 : TCP Echo C/S …

```
import java.net.*;
import java.io.*;
import java.util.*;
public class TcpEchoServer {
  public static void main(String[] args) {
    int port = 9090;
    try {
      ServerSocket server = new ServerSocket(port);
      while(true) {
          System.out.println("Waiting for clients on port " + port);
          Socket client = server.accept();
          System.out.println("Got connection from " +
                    client.getInetAddress()+":"+client.getPort());
          BufferedReader reader = new BufferedReader(new
                   InputStreamReader(client.getInputStream()));
          PrintWriter writer = new PrintWriter(
                                  client.getOutputStream());
          writer.println("Welcome to my server");
          writer.flush();
          String  message = reader.readLine();
```


Example 2 : TCP Echo C/S

```
          while (!(message == null ||
                    message.equalsIgnoreCase("exit"))) {
                  System.out.println("MessageReceived: "+message);
                  writer.println(message);
                  writer.flush();

                  message = reader.readLine();
          }
          client.close();
      }
    } catch(Exception ex) {
      System.out.println("Connection error: "+ex);
    }
  }
}
```

Example 2 : TCP Echo C/S …

```
1.  import java.net.*;
2.  import java.io.*;
3.  import java.util.*;
4.  public class TcpEchoClient {
5.    public static void main(String[] args) {
6.      int port = 9090;
7.      try {
8.        //assumes server is local
9.        String host = InetAddress.getLocalHost().getHostName();
10.       Socket client = new Socket(host, port);
          PrintWriter writer = new PrintWriter(client.getOutputStream());
11.       BufferedReader reader = new BufferedReader(new
12.                          InputStreamReader(client.getInputStream()));
13.       BufferedReader stdin = new BufferedReader(new
14.                         InputStreamReader(System.in));

15.       //read welcome message and prints it
16.       System.out.println(reader.readLine());
17.       String message;
```


Example 2 : TCP Echo C/S …

```
18.       while (true) {
19.            System.out.print("Enter message to echo or Exit to end : ");
20.            message = stdin.readLine();
21.            if (message == null || message.equalsIgnoreCase("exit"))
22.                   break;
23.            writer.println(message);
24.            writer.flush();
25.            System.out.println("Echo from server: "+reader.readLine());
26.       }
27.       client.close();
28.     }catch (Exception ex) {
29.       System.out.println("Exception: "+ex);
30.     }
31.   }
32. }
```

Example 3 : TCP File C/S

- The Following example shows how to exchange binary data.

```
import java.net.*;
import java.io.*;
import java.util.*;
public class FileServer {
  public static void main(String[] args) {
    int port = 9070;
    BufferedReader reader;
    PrintWriter writer;
    InputStream inStream;
    OutputStream outStream;
    try {
      ServerSocket server = new ServerSocket(port);
      while(true) {
          System.out.println("Waiting for clients on port " + port);
          Socket client = server.accept();
          inStream = client.getInputStream();
          outStream = client.getOutputStream();
          reader = new BufferedReader(new
                          InputStreamReader(inStream));
          writer = new PrintWriter(outStream);
```


Example 3 : TCP File C/S ...

```
          writer.println("Welcome to my file server");
          writer.flush();
          String filename = reader.readLine();
          File file = new File(filename);
          if (!file.exists()){
                  writer.println("ERROR");
                  writer.flush();
          }
          else {
                  FileInputStream fileInStream = new
                           FileInputStream(filename);
                  writer.println(""+file.length());
                  writer.flush();
                  sendFile(fileInStream, file.length(), outStream);
                  fileInStream.close();
          }
          client.close();
      }
    } catch(Exception ex) {
      System.out.println("Connection error: "+ex);
    }
  }
```

Example 3 : TCP File C/S ...

```
43.     public static void sendFile(FileInputStream file, long size,
44.                                 OutputStream outStream) {
45.       byte[] buffer = new byte[1024];
46.       int sofar = 0;
47.
48.       try {
49.
50.         while (sofar < size) {
51.               int read = file.read(buffer, 0, buffer.length);
52.               sofar+=read;
53.               outStream.write(buffer, 0, read);
54.               outStream.flush();
55.         }
56.       }
57.       catch (Exception ex) {
58.         System.out.println("Exception: "+ex);
59.       }
60.     }
61.   }
```


Example 3 : TCP File C/S

- The Following is the receiveFile method on the client side.

```
1.   public void receiveFile(String filename, long size,
2.                InputStream inStream) {
3.     try {
4.       FileOutputStream file = new FileOutputStream(filename);
5.       byte[] buffer = new byte[1024];
6.       long sofar = 0;
7.
8.       int read = inStream.read(buffer, 0, buffer.length);
9.       while (sofar < size ) {
10.        file.write(buffer, 0, read);
11.        sofar+=read;
12.        read = inStream.read(buffer, 0, buffer.length);
13.      }
14.      file.flush();
15.      file.close();
16.    }
17.    catch(Exception ex) {
18.      System.out.println("Exception: "+ex);
19.    }
20.  }
```

UDP C/S Programming

- Java provides the class, DatagramSocket for creating UDP Sockets.
- The class, DatagramPacket, is used to represent data as Datagrams before sending it using the DatagramSocket.
- The following are the constructors and some of the methods of the DatagramSocket class:

DatagramSocket()	Constructs a datagram socket and binds it to any available port on the local machine
DatagramSocket(int port)	Constructs a datagram socket and binds it to the specified port on the local host machine
DatagramSocket(int port, InetAddress laddr)	Creates a datagram socket, bound to the specified local address
DatagramSocket(SocketAddress bindaddr)	Creates a datagram socket, bound to the specified local socket address
void bind(SocketAddress addr)	Binds this DatagramSocket to a specific local address & port

UDP C/S Programming …

void connect(InetAddress address, int port) void connect(SocketAddress addr)	Connects the socket to a remote address for this socket
void disconnect()	Disconnects the socket
SocketAddress getRemoteSocketAddress()	Returns the socket address to which this socket is connected to.
SocketAddress getLocalSocketAddress()	Returns the socket address to which this socket is bound to
void receive(DatagramPacket p)	Receives a datagram packet from this socket
void send(DatagramPacket p)	Sends a datagram packet from this socket
int getReceiveBufferSize()	Gets the size of the receive buffer,
void setReceiveBufferSize(int size)	Sets the size of the receive buffer
boolean getBroadcast()	Tests if broadcast is enabled
void setBroadcast(boolean on)	Enables/disables Socket broadcast
void close()	Closes the socket.

UDP C/S Programming

- The following are the constructors and some of the methods of the DatagramPacket class:

DatagramPacket(byte[] buf, int length)	Constructs a DatagramPacket for **receiving** packets of given length
DatagramPacket(byte[] buf, int offset, int length)	Constructs a DatagramPacket for **receiving** packets of length length, specifying an offset into the buffer
DatagramPacket(byte[] buf, int length, InetAddress address, int port) DatagramPacket(byte[] buf, int length, SocketAddress address)	Constructs a datagram packet for **sending** packets of given length to the specified endpoint
DatagramPacket(byte[] buf, int offset, int length, InetAddress address, int port) DatagramPacket(byte[] buf, int offset, int length, SocketAddress address)	Constructs a datagram packet for **sending** packets of given length with offset ioffset to the specified endpoint.

UDP C/S Programming

InetAddress getAddress() void setAddress(InetAddress iaddr)	Gets/Setss the IP address of the machine to which this datagram is being sent or from which the datagram was received
byte[] getData() void setData(byte[] buf) void setData(byte[] buf, int offset, int length)	Gets/Sets the data buffer
int getLength() void setLength(int length)	Gets/Sets the length of the data to be sent or the length of the data received.
int getOffset() void setPort(int iport)	Gets/Sets the offset of the data to be sent or the offset of the data received
int getPort() void setPort(int iport)	Gets/Sets the port number on the remote host to which this datagram is being sent or from which the datagram was received.
SocketAddress getSocketAddress() void setSocketAddress(SocketAddress address)	Gets/Sets the SocketAddress of the remote host that this packet is being sent to or is coming from.

Example 4 : UDP Echo C/S

```
1.  import java.net.*;
2.  import java.io.*;
3.  public class UdpEchoServer {
4.    static final int port = 9999;
5.    static final int packetSize = 1024;
6.    public static void main(String args[]) throws SocketException{
7.      DatagramPacket packet;
8.      DatagramSocket socket;
9.      byte[] data;
10.     int clientPort;
11.     InetAddress address;
12.     String str;
13.     int recvSize;
14.     socket = new DatagramSocket(port);
15.     while(true){
16.       data = new byte[packetSize];
17.       // Create packets to receive the message
18.       packet = new DatagramPacket(data,packetSize);
19.       System.out.println("to receive the packets or port: "+port);
```


Example 4 : UDP Echo C/S …

```
20.       try{
21.            socket.receive(packet);
22.       }catch(IOException ie){
23.            System.out.println(" Could not Receive:"+ie.getMessage());
24.            System.exit(0);
25.       }
26.       // get data about client in order to echo data back
          address = packet.getAddress();
27.       clientPort = packet.getPort();
28.       recvSize = packet.getLength();
29.       str = new String(data,0,recvSize);
30.       System.out.println("Message  from "+ address+":"+clientPort+": "
31.                  +str.trim());
32.       // echo data back to the client
33.       packet = new DatagramPacket(data,recvSize,address,clientPort);
34.       try{
35.            socket.send(packet);
36.       }catch(IOException ex){
37.            System.out.println("Could not Send "+ex.getMessage());
38.       }
39.     }
40.   }}
```

Example 4 : UDP Echo C/S …

```
1.  import java.net.*;
2.  import java.io.*;
3.  public class UdpEchoClient {
4.     static final int packetSize = 1024;
5.     static BufferedReader stdin = new BufferedReader(new
6.                                       InputStreamReader(System.in));
7.     public static void main(String args[]) throws UnknownHostException,
8.                          SocketException{
9.       DatagramSocket socket;
10.      DatagramPacket packet;
11.      InetAddress address;
12.      String messageSend;
13.      String messageReturn;
14.      byte[] data;
15.      int port;
16.      try {
17.        System.out.print("Enter server name: ");
18.        address = InetAddress.getByName(stdin.readLine());
19.        System.out.print("Enter server port: ");
20.        port = Integer.parseInt(stdin.readLine());
```


Example 4 : UDP Echo C/S …

```
21.        while (true) {
22.             System.out.print("Type message for server enter to exit:
23.             messageSend = stdin.readLine();
24.             if(messageSend.length() == 0){
25.                     break;
26.             }
27.             socket = new DatagramSocket();
28.             data = messageSend.getBytes();
29.             packet = new DatagramPacket(data,data.length,address,port);
30.             socket.send(packet);
31.             //packet is reinitialized to use it for recieving
32.             data = new byte[packetSize];
33.             packet = new DatagramPacket(data,data.length);
34.             socket.receive(packet);
35.             messageReturn = new String(data,0,packet.getLength());
36.             System.out.println("Message Returned:"+ messageReturn);
37.        }
38.      }catch(IOException iee){
39.        System.out.println("Could not receive :+iee.getMessage() );
40.      }
41.    }
42. }
```

Resources

- http://java.sun.com/docs/books/tutorial/networking/TOC.html
- Lecture notes of previous offerings of SWE344 and ICS343
- Some other web sites and books; check the course website at
  - http://faculty.kfupm.edu.sa/ics/alfy/files/teaching/swe344/index.htm