Java Net Classes

Class	Description
DatagramPacket	This class represents a datagram packet.
DatagramSocket	This class represents a socket for sending and receiving datagram packets.
InetAddress	This class represents an Internet Protocol (IP) address.
MulticastSocket	The multicast datagram socket class is useful for sending and receiving IP multicast packets.
ServerSocket	This class implements server sockets.
Socket	This class implements client sockets (also called just "sockets").
URL	A pointer to a "resource" on the World Wide Web.
URLConnection	The superclass of all classes that represent a communications link between an application and a URL.

Class InetAddress

```
public boolean equals(Object obj);

public byte[] getAddress();
public static InetAddress[] getAllByName(String host);
public static InetAddress getByName(String host);
public String getHostName();
public static InetAddress getLocalHost();

public int hashCode();
public String toString();
```

This class represents an Internet Protocol (IP) address.

Applications should use the methods `getLocalHost()`, `getByName()`, or `getAllByName()` to create a new `InetAddress` instance.

HostInfo.java

```
import java.net.*;

public class HostInfo {
  public static void main( String args[] ) {
    InetAddress ipAddr;

    try {
      ipAddr = InetAddress.getLocalHost();
      System.out.println( "This is " + ipAddr );
    }
    catch ( UnknownHostException ex ) {
      System.out.println( "Unknown host" );
    }
}
```

Resolver.java

```
import java.net.*;

public class Resolver {
  public static void main( String args[] ) {
    InetAddress ipAddr;

    try {
      ipAddr = InetAddress.getByName( args[0] );
      System.out.print( "IP address = " + ipAddr + "\n " );
    }
    catch ( UnknownHostException ex ){
      System.out.println( "Unknown host " );
    }
  }
}
```

Daytime Service

Most UNIX servers run the daytime service on TCP port 13.

```
cobalt> telnet kiev.cs.rit.edu 13
Trying 129.21.38.145...
Connected to kiev.
Escape character is '^]'.
Fri Feb  6 08:33:44 1998
Connection closed by foreign host.
```

It is easy to write a Java daytime client. All the program needs to do is to establish a TCP connection on port 13 of a remote host.

A TCP style connection is made using the Socket class.

Class Socket

```
// Constructors (partial list)
public Socket()
public Socket(InetAddress address, int port);
public Socket(String host, int port);

// Methods (partial list)
public void close();

public InetAddress getInetAddress();
public int getLocalPort();

public InputStream getInputStream();
public OutputStream getOutputStream();

public int getPort();
public String toString();
```

Class Socket

- This class implements client sockets (also called just *sockets*). A socket is a end point for communication between two machines.
- The actual work of the socket is performed by an instance of the `SocketImpl` class.
- It is possible to modify some TCP parameters:
  - `SO_LINGER`
  - `SO_TIMEOUT`
  - `TCP_NODELAY`
  - Enable/disable `TCP_NODELAY` (disable/enable Nagle's algorithm).

DayTimeClient.java

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

public class DayTimeClient {
  static int dayTimePort = 13;

  public static void main(String args[]) {
    try {
      Socket sock = new Socket(args[0], dayTimePort);
      BufferedReader din = new BufferedReader(
        new InputStreamReader(sock.getInputStream()));
      String rTime = din.readLine();
      System.out.println(rTime);
      sock.close();
    }
    catch (exception e) {}
  }
}
```

A Java Daytime Server

- It is easy to create a daytime server in Java (the only real problem is that your Java server will not be able to use port 13).
- The server version of the program will use a `ServerSocket` to communicate with a client.
- A `ServerSocket` will open a TCP port and wait for a connection.
- Once a request is detected, a new port will be created, and the connection will be established between the client's source port and this new port.
- Most servers listen for requests on a particular port, and then service that request on a different port.
- This makes it easy for the server to accept and service requests at the same time.

Class ServerSocket

```
// Constructors (partial list)

public ServerSocket(int port);
public ServerSocket(int port, int count);

// Methods (partial list)

public Socket accept();
public void close();

public InetAddress getInetAddress();
public int getLocalPort();

public String toString();
```

Class ServerSocket

- A `ServerSocket` waits for requests to come in over the network. It performs some operation based on that request, and then possibly returns a result to the requester.
- The actual work of the `ServerSocket` is performed by an instance of the `SocketImpl` class.
- The abstract class `SocketImpl` is a common superclass of all classes that actually implement sockets. It is used to create both client and server sockets.
- A *plain* socket implements the SocketImpl methods exactly as described, without attempting to go through a firewall or proxy.

DayTimeServer

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

public class DayTimeServer {
  public static void main(String argv[]) {
    try {
      Date today = new Date();
      InetAddress localHost = InetAddress.getLocalHost();
      ServerSocket listen = new ServerSocket(0);
      System.out.println("Listening on port:  "+listen.getLocalPort());

      for(;;) {
        Socket clnt = listen.accept();
        System.out.println(clnt.toString());
        PrintWriter out = new PrintWriter(clnt.getOutputStream(), true);
        out.println(today);
        clnt.close();
      }
    } catch(Exception e) {}}}
```

DayTimeServer in Action

The output from the daytime server looks like this:

```
kiev> java DayTimeServer
Listening on port:  36109
Socket[addr=cobalt/129.21.37.176,port=32875,localport=36109]
Socket[addr=localhost/127.0.0.1,port=36112,localport=36109]
```

The client output looks like this:

```
cobalt> telnet kiev 36109
Trying 129.21.38.145...
Connected to kiev.
Escape character is '^]'.
Fri Feb 06 09:53:00 EST 1998
Connection closed by foreign host.
```

Multi-Threaded Servers

- It is quite easy, and natural in Java, to make a server multi-threaded.
- In a multi-threaded server a new thread is created to handle each request.
- Clearly for a server such as the daytime server this is not necessary, but for an FTP server this is almost required.
- The code for the multi-threaded version of the server consists of a new class called Connection.
- An instance of this class handles the clients request.

Connection.java

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

class Connection extends Thread {
  protected Socket clnt;
  public Connection(Socket sock) {
    clnt = sock;
    this.start();
  }

  public void run() {
    Date today = new Date();
    try {
      PrintWriter out = new PrintWriter(clnt.getOutputStream(), true);
      out.println(today);
      client.close();
    } catch (IOException e) {}}}
```

TDayTimeServer.java

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

public class TDayTimeServer {
  public static void main(String argv[]) {
    try {
      InetAddress localHost = InetAddress.getLocalHost();
      ServerSocket listen = new ServerSocket(0);
      System.out.println("Listening on:  "+listen.getLocalPort());

      for(;;) {
        Socket clnt = listen.accept();
        System.out.println(clnt.toString());
        Connection c = new Connection(client);
      }
    }
    catch(Exception e) { System.out.println("Server terminated"); }
  }
}
```

UDP Based Applications

- UDP provides an unreliable packet based delivery service. An application that uses UDP must deal with the errors that can arise during communication.
- The basic unit of transfer is called a *Datagram*. Datagrams are small, fixed-length messages.
- Datagram based services do have some advantages:
  – Speed
  – Message-oriented service.

Datagrams

- Datagram packets are used to implement a connectionless, packet based, delivery service.
- Each message is routed from one machine to another based solely on information contained within that packet.
- Multiple packets sent from one machine to another might be route d differently, and might arrive in any order.
- Packets may be lost or duplicated during transit.
- The class `DatagramPacket` represents a datagram in Java.

Class DatagramPacket

```
//Constructors
public DatagramPacket(byte ibuf[], int ilength);
public DatagramPacket(
  byte ibuf[], int ilength, InetAddress iaddr, int iport);

// Methods
public synchronized InetAddress getAddress();
public synchronized int getPort();
public synchornized byte[] getData();
int getLength();

void setAddress(InetAddress iaddr);
void setPort(int iport);
void setData(byte ibuf[]);
void setLength(int ilength);
```

Class DatagramSocket

- This class represents a socket for sending and receiving datagram packets.
- Addressing information for outgoing packets is contained in the packet header.
- A socket that is used to read incoming packets must be bound to an address (sockets that are used for sending must be bound as well, but in most cases it is done automatically).
- There is no special datagram server socket class.
- Since packets can be lost, the ability to set timeouts is important.

Class DatagramSocket

```
// Constructors
DatagramSocket()
DatagramSocket(int port)
DatagramSocket(int port, InetAddress iaddr)

// Methods
void close()
InetAddress getLocalAddress()
int getLocalPort()
int getSoTimeout()
void receive(DatagramPacket p)
void send(DatagramPacket p)
setSoTimeout(int timeout)
```

Echo Services

- A common network service is an echo server
- An echo server simply sends packets back to the sender
- A client creates a packet, sends it to the server, and waits for a response.
- Echo services can be used to test network connectivity and performance.
- There are typically different levels of echo services. Each provided by a different layer in the protocol stack.

UDPEchoClient.java

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

public class UDPEchoClient {
  static int echoPort = 7; static int msgLen = 16; static int timeOut=1000;

  public static void main(String argv[]) {
    try {
      DatagramSocket sock = new DatagramSocket();
      DatagramPacket pak;
      byte msg[] = new byte[msgLen];

      InetAddress echoHost = InetAddress.getByName(argv[0]);
      pak = new DatagramPacket(msg,msgLen,echoHost,echoPort);

      sock.send(pak);
      sock.setSoTimeout(timeOut);
      sock.receive(pak);
    }
    catch (InterruptedIOException e) {System.out.println("Timeout");}
    catch (Exception e) {}
  }}
```