Introduction to Network Programming using Java
Starting Point Using Java

IDE
Unix/Linux available in the department
Alternative: MS Windows workstations
Using Sun JDK

Information sources
Today’s slides and examples
Details on the web page
javadoc, Google
Send mail to assistants (if everything else has failed)
The Goals in the assignments

Workable software
   Remember that you will need to build upon this later
   Compiled and tested on the department workstations (Unix/Linux)
   Learning: how to get there
   Functionality: to actually arrive at a working solution

Documentation
   Inline
   Shows that you understood the problem and the solutions
   Helps you to remember what you were thinking today in two months from now
   Helps us to understand what you meant to do
   → There should be no “wrong” solutions (only malfunctioning ones)

Working with development tools
   ant, javac, svn
   Using IDE (Eclipse, NetBeans, JCreator ...)
Program Structure

Initialization
- Parse the command line arguments
- Resolve hostnames
- Prepare Socket instances

Main thread
- Manage Socket instances
- Read data from receiving sockets
- Handle received input

Clean-up
- Close all Socket instances
- Terminate threads
Parse Command Line in Java

```java
public static void main(String[] args) {
    // String array containing the program arguments
    // Example iterating through array
    for (int i = 0; i < args.length; i++) {
        String type = args[i++];
        String value = args[i];
        if (type.equalsIgnoreCase("-l")) {
            // use value
            setExampleParameter(value);
        }
    }
}
```
Resolve hostname

Transform a symbolic name into a protocol-specific address
Attention: different address formats and lengths!
Select the most suitable implementation for the specific task

APIs

java.net.InetAddress
public static InetAddress getByName(String host)
public static InetAddress getByAddress(byte[] addr)
java.net.InetSocketAddress

J2SE 1.5.0 API Documentation
http://java.sun.com/j2se/1.5.0/docs/api/index.html
Get Detailed Address Info

Get detailed address info using `java.net.InetAddress` subclasses `java.net.Inet4Address` or `java.net.Inet6Address` for example following methods are available:

- `boolean isMCGlobal()`
  Utility routine to check if the multicast address has global scope.

- `boolean isMCLinkLocal()`
  Utility routine to check if the multicast address has link scope.

- `boolean isMCNodeLocal()`
  Utility routine to check if the multicast address has node scope.

- `boolean isMCOrgLocal()`
  Utility routine to check if the multicast address has organization scope.

- `boolean isMCSiteLocal()`
  Utility routine to check if the multicast address has site scope.

- `boolean isMulticastAddress()`
  Utility routine to check if the InetAddress is an IP multicast address.
Socket Creation

```java
java.net.Socket
java.net.ServerSocket
java.net.DatagramSocket
java.net.MulticastSocket
```

`java.net.Socket()`

Creates an unconnected socket, with the system-default type of SocketImpl.

`java.net.Socket(InetAddress address, int port)`

Creates a stream socket and connects it to the specified port number at the specified IP address.

`java.net.ServerSocket()`

Creates an unbound server socket.

`java.net.ServerSocket(int port)`

Creates a server socket, bound to the specified port.

`java.net.ServerSocket(int port, int backlog, InetAddress bindAddr)`

Create a server with the specified port, listen backlog, and local IP address to bind to.
Sending Data

Connection-oriented (TCP)

```java
java.net.Socket(InetAddress address, int port)
  Creates a stream socket and connects it to the specified port number at the specified IP address.
java.net.Socket.getOutputStream()
  Write into OutputStream using suitable classes
```

Connectionless (UDP)

```java
java.net.DatagramSocket(int port)
  Constructs a datagram socket and binds it to the specified port on the local host machine.
java.net.DatagramPacket(byte[] buf, int length, InetAddress address, int port)
  Constructs a datagram packet for sending packets of length length to the specified port number on the specified host.
java.net.DatagramSocket.send(DatagramPacket p)
  Sends a datagram packet from this socket.
```
Receiving Data

Data reception (UDP) using DatagramSocket

\[ \text{DatagramSocket.receive(DatagramPacket pPacket)} \]
Receives a datagram packet from this socket. The DatagramPacket contains the bytes transmitted.

Data reception (TCP) using Socket

\[ \text{InputStream Socket.getInputStream()} \]
Read InputStream using suitable classes

To modify socket behaviour check the setter methods of the specified implementation
Multicast reception

Joining the multicast group
try {
    java.net.MulticastSocket msocket =
        new java.net.MulticastSocket(port);
    java.net.InetAddress group =
        java.net.InetAddress.getByName(groupName);
    msocket.joinGroup(group);
} catch (IOException e) {
}

Leaving the multicast group
try {
    msocket.leaveGroup(group);
} catch (IOException e) {
}
Hints (1)

Try to group a certain set of functionalities into a specified class.
Use designing patterns to get a controlled structure for your program.

For example, Observer – Observable pattern can be used to deliver the received data for multiple users.

Use the `java.io` with `java.net` to achieve simpler program structure than by using the `java.nio` package.

The lower performance of `java.io` package isn't an issue here.
Hints (2)

Use worker threads to receive multiple connections for a single server socket

```java
while(serverIsRunning){
    // ConnectionHandler is own class implementing the Runnable interface
    ConnectionHandler worker;
    try{
        // server.accept returns a client connection
        worker = new ConnectionHandler(server.accept());
        Thread t = new Thread(worker);
        t.start();
    } catch (IOException e) {
        // handle the exceptions
    }
}
```
Hints (3)

To handle shutdown signal use addShutdownHook() method for Runtime class

```java
Runtime.getRuntime().addShutdownHook(new Thread() {
    public void run() {
        System.out.println("Called at shutdown.");
    }
});
```

Other alternative is to use handle() method in sun.misc.Signal class to catch signals

```java
public static void main(String[] args) throws Exception {
    Signal.handle(new Signal("INT"), new SignalHandler () {
        public void handle(Signal sig) {
            System.out.println("Received a interrupt!!");
        }
    });
    //
}
```