Introduction to Network Programming using Java
Starting Point

Development platform
- Unix/Linux/Windows available in the department or computing centre
  - More information http://www.tkk.fi/cc/computers/
- Using Sun JDK

Information sources
- Today’s slides and examples
- Sun Java Documentation
- Examples and tutorials available via search engines
- Send mail to assistants (if everything else has failed)
The Goals in this course assignments

- **Workable software**
  - Remember that you will need to build upon this later

- **Documentation**
  - Describe the concept shortly and document details inline with the code
    - 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**
  - Using IDE (Eclipse, NetBeans, JCreator ...)
  - Use existing libraries (Apache Commons ...)
  - Automate testing if possible
Some basic things...

- ... concerning Java programming in general
  - Parsing command line parameters
  - Handling Streams (java.io package)
  - Handling Channels (java.nio package)
  - Handling byte arrays
- ... concerning network programming
  - Resolving hostname
  - Handling address information
  - Creating Sockets
  - Sending and receiving data using blocking / non-blocking methods
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);
        }
    }
}
```

Or use the existing packages like:
- args4j [https://args4j.dev.java.net/](https://args4j.dev.java.net/)
- Apache Commons CLI [http://commons.apache.org/cli/](http://commons.apache.org/cli/)
Resolve hostname

- Transform a symbolic name into a protocol-specific address
- Select the most suitable implementation for the specific task
- InetAddress class for 32-bit and 128-bit IP addresses used for unicast or multicast
- InetSocketAddress class is implementation for IP address and port number pairs used by sockets for binding and connecting
- APIs
  - java.net.InetAddress
  - java.net.InetSocketAddress

- J2SE 1.5.0 API Documentation
  http://java.sun.com/j2se/1.5.0/docs/api/index.html
How to 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 isMulticastAddress()` Utility routine to check if the InetAddress is an IP multicast address.
  - `boolean isLinkLocalAddress()` Utility routine to check if the InetAddress is an link local address.
  - `boolean isLoopbackAddress()` Utility routine to check if the InetAddress is a loopback address.
  - `boolean isIPv4CompatibleAddress()` Utility routine to check if the InetAddress is an IPv4 compatible IPv6 address.
Socket Creation (blocking)

- `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.
Socket Creation (non-blocking)

- java.nio.channels.SocketChannel
- java.nio.channels.ServerSocketChannel

- Opening a socket channel

```java
InetSocketAddress isa
    = new InetSocketAddress(targetAddrs, targetPort);

// Connect
SocketChannel sChannel
    = SocketChannel.open();
sChannel.configureBlocking(false);
boolean connected = sChannel.connect(isa);

if(connected == false){
    sChannel.finishConnect();
}
```
Sending Data using Stream

- **Connection-oriented (TCP)**
  - `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.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 to the specified port number on the specified host.
  - `java.net.DatagramSocket.send(DatagramPacket p)`
    Sends a datagram packet from this socket.
Receiving Data using Stream

- Data reception (UDP) using DatagramSocket
  - `DatagramSocket.receive(DatagramPacket pPacket)`
    Receives a datagram packet from this socket. The DatagramPacket contains the bytes transmitted.

- Data reception (TCP) using Socket
  - `InputStream Socket.getInputStream()`
    Read InputStream using suitable classes

- To modify socket behaviour check the setter methods of the specified implementation
Sending Data using Channel

```java
//
// SocketChannel sChannel

try {
    String message = "NMPS course";
    ByteBuffer buf = ByteBuffer.wrap( message.getBytes() );
    sChannel.write(content);
} catch (IOException e) {
    // TODO Auto-generated catch block
    e.printStackTrace();
}
```
Receiving Data using Channel

```
//
// SocketChannel sChannel
// CharsetDecoder decoder

ByteBuffer dbuf = ByteBuffer.allocateDirect(1024);
CharBuffer cb = null;
int readCount = -1;
try {
    dbuf.clear();
    readCount = sChannel.read(dbuf);
    dbuf.flip();
    cb = decoder.decode(dbuf);
    dbuf.flip();
} catch (IOException e) {
    // TODO Auto-generated catch block
    e.printStackTrace();
}
```
Byte array operations

- Using byte array or java.nio.ByteBuffer

  ```java
  // array operations
  byte[] array = new byte[64];
  int arrayLength = array.length;
  byte[] content = new byte[arrayLength];
  System.arraycopy(array, 0, content, 0, arrayLength);

  // ByteBuffer
  String example = "Hello";
  ByteBuffer buffer = ByteBuffer.wrap(example.getBytes());
  ByteBuffer buffer2 = buffer.duplicate();
  buffer2.order(ByteOrder.BIG_ENDIAN);
  byte[] array2 = buffer2.array();
  ```

- Or use existing libraries like
Concurrency

- Learn how to use Threads or take the event base approach by using the new I/O package

```java
//
// ReceiverThread implements Runnable interface
ReceiverThread receiverConnection = new ReceiverThread();

receiver = new Thread(receiverConnection);
receiver.start();
```

- For the beginners read tutorials like
  - [http://java.sun.com/j2se/1.5.0/docs/guide/concurrency/index.html](http://java.sun.com/j2se/1.5.0/docs/guide/concurrency/index.html)
Hints (1)

.. about the structure of your implementation

- Try to keep your classes as simply as possible
  - group a certain set of functionalities into a specified class
- Use design 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
Hints (2)

... how to handle your connections

- 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)

… how to terminate program and release resources

- 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!!");
      }
    });
  }
  ```