G51PRG:
Introduction to Programming
Second semester
Lecture 12

Natasha Alechina
School of Computer Science & IT
`nza@cs.nott.ac.uk`

Previous lecture: threads

- What is a thread
- Why use multiple threads
- Issues and problems involved
- Java threads

This lecture:networking

How to make your program to make network connections (java.net package)

- Connecting via URLs
- Connecting via ports and sockets

URL

- URL is the acronym for Uniform Resource Locator. It is a reference (an address) to a resource on the Internet.
- A URL takes the form of a string that describes how to find a resource on the Internet. URLs have two main components: the protocol needed to access the resource and the location of the resource. For example, http://www.cs.nott.ac.uk/
- Java programs can use a class called URL in the java.net package to represent a URL address.

URL class

- The simplest constructor:

`URL(String url)`

throws a **`MalformedURLException`**.

- Example:

```
URL school = new
URL("http://www.cs.nott.ac.uk/");
```

URL class: constructors

- More complicated constructor:

```
public URL(String protocol,
           String host,
           int port,
           String file)
throws MalformedURLException.
```

Example: **http://www.ncsa.uiuc.edu:8080/demoweb/url-primer.html**

```
URL u = new
URL("http","www.ncsa.uiuc.edu", 8080,
  "demoweb/url-primer.html");
```

URL class: methods

- **`String getFile()`** - returns the file name of this URL.
- **`String getHost()`** - returns the host name of this **`URL`**, if applicable.
- **`int getPort()`** - returns the port number of this URL.
- **`InputStream openStream()`** - opens a connection to this **`URL`** and returns an **`InputStream`** for reading from that connection.
- **`URLConnection openConnection()`** - returns a **`URLConnection`** object that represents a connection to the remote object referred to by the **`URL`**.

Reading contents of a URL

- Use **`openStream()`** method of **`URL`** class.

```
URL example = new
URL("http://www.cs.nott.ac.uk/~nza");
BufferedReader in = new
BufferedReader(new InputStreamReader(
                  example.openStream()));
String inputLine;
while((inputLine = in.readLine())!= null)
   System.out.println(inputLine);
in.close();
```

Creating a URL connection

- Alternative: create a Connection object and read and write using its methods.

```
URL example = new
URL("http://www.cs.nott.ac.uk/~nza");
URLConnection ex =
  example.openConnection();
BufferedReader in = new
BufferedReader(new InputStreamReader(
                   ex.getInputStream()));
```

Writing to a URL

- You can also write to a URL connection (filling in forms, for example). More on

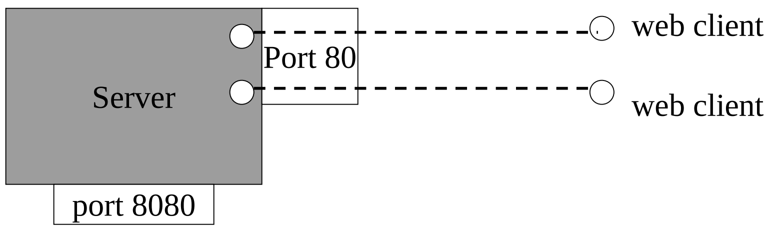
http://java.sun.com/docs/books/tutorial/networking/urls/
readingWriting.html

Ports and Sockets

- URLs are a high-level mechanism for accessing resources on the Internet. Client-server applications require lower-level network communication.
- A ***port*** is an abstraction of a physical place through which communication can proceed between a server and a client.
- A ***socket*** is an abstraction of a network software which enables communication in an out of the program.

Ports and Sockets continued

- Several sockets (for connecting clients) can be created on a single port (server).
- To be more precise, a server accepts a client on the server port and creates a socket for them on a different port. However, Java hides those details from you.



ServerSocket class

One of the constructors:

- **`ServerSocket(int port)`** - creates a server socket on a specified port.

Some methods:

- **`Socket accept()`** - listens for a connection to be made to this socket and accepts it.
- **`void close()`** - closes this socket.
- **`InetAddress getInetAddress()`** - returns the local address of this server socket.
- **`int getLocalPort()`** - returns the port on which this socket is listening.

Socket class

Some of the constructors:

- **`Socket(String host, int port)`** - - creates a stream socket and connects it to the specified port number on the named host.
- **`Socket(InetAddress address, int port)`** - creates a stream socket and connects it to the specified port number at the specified IP address.

Socket class

Some methods:

- **`void close()`** - closes this socket.
- **`InetAddress getInetAddress()`** - returns the address to which the socket is connected.
- **`int getLocalPort()`** - returns the local port to which this socket is bound.
- **`OutputStream getOutputStream()`** - returns an output stream for this socket.

Writing a client server application

- Write a server class (what does the server do; at least should open a ServerSocket)
- Write a client class
- Write a protocol for communication between client and server
- When the server is running, it creates a thread to deal with each new client.

Case study: chat server

- The program (slightly modified) from Judy Bishop’s ***Java Gently.***
- A chat server program is running on some machine listening on a specified port. When it gets a request for connection, it creates a thread which adds the new client to a list of clients and reads on a stream from that client.
- When a client types something, this message is read by the server and broadcast to all other clients.
- We don’t have to implement a protocol since we use a ready made one: telnet. Clients are also just people telnetting in and chatting, not programs.

ChatServer

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

public class ChatServer {

 private static LinkedList clientList =
  new LinkedList();
 private static int id = 0;
```

ChatServer contd.

```
static synchronized void broadcast(String
message, String name) throws IOException{
 Socket s;
 PrintWriter p;
 for (int i = 0; i < clientList.size();
  i++) {
  s = (Socket)clientList.get(i);
  p = new PrintWriter (
              s.getOutputStream(), true);
  p.println(name+": "+message); }}
```

ChatServer contd.

```
static synchronized void remove(Socket s)
  {
   clientList.remove(s);
   id--;
  }
```

ChatServer contd.

```
public static void main(String[] args)
  throws IOException {
//Get the port and create a socket there.
int port = 8190; // default
if (args.length > 0)
  port = Integer.parseInt(args[0]);
ServerSocket listener = new
  ServerSocket(port);
System.out.println("The Chat Server is
  running on port "+port);
```

ChatServer contd.

```
// Listen for clients.
// Start a new handler for each.
// Add each client to the list.
while (true) {
 Socket client = listener.accept();
 new ChatHandler(client).start();
 System.out.println("New client no."+id+
  " on client's port "+client.getPort());
 clientList.add(client);
 id++; } } // end while and end main()
```

ChatHandler

```
class ChatHandler extends Thread {
    private BufferedReader in;
    private PrintWriter out;
    private Socket toClient;
    private String name;

    ChatHandler(Socket s) {
        toClient = s;
    }
```

ChatHandler continued

```
public void run() {
 try {
 in = new BufferedReader(
      new InputStreamReader(
      toClient.getInputStream()));
 out = new PrintWriter(
       toClient.getOutputStream(), true);
 out.println("*** Welcome to the Chatter
  ***");
 out.println("Type BYE to end");
```

ChatHandler continued

```
out.print("What is your name? ");
out.flush();
String name = in.readLine();
ChatServer.broadcast(name+" has joined
  the discussion.", "Chatter");
```

ChatHandler continued

```
 while (true) {
   String s = in.readLine();
   if (s.startsWith("BYE")) {
    ChatServer.broadcast(name+" has left
    the discussion.", "Chatter");
    break;
   }
   ChatServer.broadcast(s, name);
} // end while
```

ChatHandler continued

```
 ChatServer.remove(toClient);
  toClient.close();
} catch (Exception e) {
  System.out.println("Chatter error: 
  "+e);
}}} // end catch, run(), class definition
```

Summary

- Java is highly suitable for networking and communication over the Internet.
- java.net provides classes for URLs and sockets.
- A detailed case study of client/server application (cash dispensers) can be found in Java Gently Chapter 14. More examples can be found on http://java.sun.com/docs/books/tutorial/networking/index.html.