Network Programming

Object Orientated Programming in Java

Benjamin Kenwright

Outline

- Essential Networking with Java
- Introduction to Java networking features
  - It is much easier to write networking programs in Java than in C++
  - But less efficient
- Today's Practical
- Review/Discussion

Today

- Networking basics
  - IP addresses, ports, protocols, client-server interaction
- Socket-level programming
  - Writing a client (Socket)
  - Writing a server (ServerSocket)
- Communicating with web servers
  - Retrieving information (URL, URLConnection)
  - Sending information

Why is Networking Important?

Networking Basics

Internet protocol (IP) addresses

- Every host on Internet has a unique IP address

  `143.89.40.46, 203.184.197.198`

  `203.184.197.196, 203.184.197.197, 127.0.0.1`

- More convenient to refer to using hostname string

  `cs.ust.hk, tom.com, localhost`

- One hostname can correspond to multiple internet addresses:
  - www.yahoo.com:

    66.218.70.49; 66.218.70.50; 66.218.71.80; 66.218.71.84; …

- Domain Naming Service (DNS) maps names to numbers

Networking Basics

`java.net.InetAddress` class converts between hostnames and internet addresses

```
InetAddress tm = InetAddress.getByName("www.yahoo.com");
InetAddress tm= InetAddress.getByName("localhost");
                                          //127.0.0.1
InetAddress tm = InetAddress.getLocalHost();
```

Can get array of addresses (if more than one)

```
InetAddress[] addrs;
addrs=InetAddress.getAllByName("www.yahoo.com");
for (int i = 0; i < addr.length; i++)
    System.out.println(addrs[i].getHostAddress());
```

Networking Basics

Ports

Many different services can be running on the host

A **port** identifies a service within a host

Many standard port numbers are pre-assigned

time of day 13, ftp 21, telnet 23, smtp 25, http 80

see `/etc/services` on workstation for list of all assigned ports

IP address + port number = "phone number" for service

Networking Basics

protocols : rules that facilitate communications between machines

Examples:

- HTTP: HyperText Transfer Protocol
- FTP: File Transfer Protocol
- SMTP: Simple Message Transfer Protocol
- TCP: Transmission Control Protocol
- UDP: User Datagram Protocol, good for, e.g., video delivery)

Protocols are standardized and documented

- So machines can reliably work with one another

Networking Basics

Client-Server interaction

Communication between hosts is two-way, but usually the two hosts take different roles

Server waits for client to make request

Server registered on a known port with the host ("public phone number")

Usually running in endless loop

Listens for incoming client connections

Networking Basics

Client "calls" server to start a conversation

Client making calls uses hostname/IP address and port number

Sends request and waits for response

Standard services always running

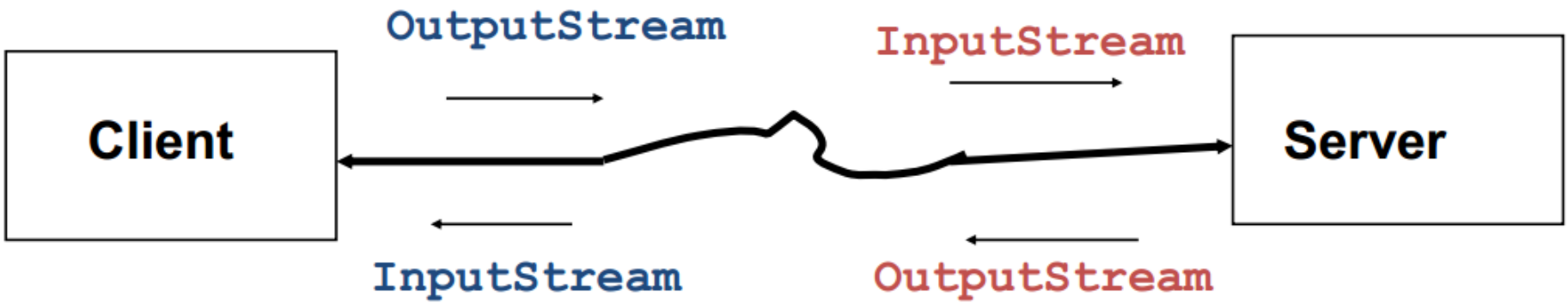
ftp, http, smtp, etc. server running on host using expected port

Server offers shared resource (information,database, files, printer, compute power) to clients

Socket-Level Programming

Socket is an abstraction of one type of bi-directional communication channel between hosts

Send and receive data using streams



Next:

- How to write a client
- How to write a server

Writing Clients

To write a client socket using **java.net.Socket**

- Create a new `Socket` with hostname and port number of the connection

```
Socket s = New Socket(String hostName,int
  portNumber);
```

- Call `s.getOutputStream()` and `s.getInputStream()` to get streams for sending and receiving infomation

- Need to learn protocol used to communicate
  - Know how to properly form requests to send to server
  - Know how to interpret the server's responses

Writing Clients

SocketTest:

Makes a socket connection to the atomic clock in Boulder, Colorado, and prints the time that the server sends.

```
try
{ Socket s = new Socket("time-A.timefreq.bldrdoc.gov", 13);

  BufferedReader in = new BufferedReader
        (new InputStreamReader( s.getInputStream() ));

  // read from in

}
catch (IOException e)
{  e.printStackTrace();
}
```

Writing Servers

To write a server using **java.net.ServerSocket**

- Create a new `ServerSocket` with a port number to listen on the port

```
ServerSocket s = New ServerSocket( portNumber);
```

- Use `accept()` to listen on the port.

- `accept()` returns a socket `incoming` when a client calls

```
Socket incoming = s.accept();
```

- Call `incoming.getOutputStream()` and `incoming.getInputStream()` to get streams for sending and receiving information

Writing Servers

Example: Echo server

```
ServerSocket s = new ServerSocket(8189);
 Socket incoming = s.accept( );
 BufferedReader in = new BufferedReader
  (new InputStreamReader(incoming.getInputStream()));
 PrintWriter out = new PrintWriter
  (incoming.getOutputStream(), true /* autoFlush */ );
 out.println( "Hello! Enter BYE to exit." );
```

…

Security Note

- Many machines now refuse socket connections due to security considerations

Writing Servers

Multithread server: starts a separate thread for each connection.

```
public class ThreadedEchoServer
{  public static void main(String[] args )
   {  int i = 1;
      try{ServerSocket s = new ServerSocket(8190);
      while (true)
      { Socket incoming = s.accept( );
        System.out.println("Spawning " + i);
        new ThreadedEchoHandler(incoming, i).start();
        i++;
      }
     } catch (Exception e) …. //ThreadedEchoServer.java
```

Writing Servers

```
class ThreadedEchoHandler extends Thread
{   public ThreadedEchoHandler(Socket i, int c)
    { incoming = i; counter = c; }
    public void run()
    {   try
        {   BufferedReader in = new BufferedReader
            (new InputStreamReader(incoming.getInputStream()));
            PrintWriter out = new PrintWriter
                (incoming.getOutputStream(), true /* autoFlush */);
            out.println( "Hello! Enter BYE to exit." );
        …
    private Socket incoming;
    private int counter; }
```

Communicating with **Web** Servers

Reason for communicating with web servers

- To retrieve/send information

Need to indicate location of resource

- URL stands for Uniform Resource Locator
  - Neat scheme for uniquely identifying all kinds of network resources

- Basic form **<protocol>:<sitename><pathname>**

  http://www.zjnu.cn/index.html

  ftp://ftp.zjnu/pub//test.java

  file:/MyDisk/Letters/file.txt

  Protocols include files, http, ftp, gopher, news, mailto, etc

Communicating with web servers

Class **`java.net.URL`** represents a Uniform Resource Locator

- Create an java object that represents an URL

```
URL url = new
  URL("http://www.cats.com/index.html");
```

  - getHost(), getPath(), getPort(), getProtocol()

java.net.URLConnection represents a communication link between the application and a URL.

- Constructor:
  - **`URLConnection cnn = new`** URLConnection(url)
- Obtainable also from URL:
  - **`URLConnection cnn = url.openConnection();`**

Communicating with web servers

Steps for working with **`java.net.URLConnection`**

- Set properties of connection:
  - **`setDoInPut(true)`** //default
  - **`setDoOutPut`**`(true)` for sending information to the server
  - …
- Make connection: **`cnn.connect();`**
- Query header information:
  - **`getContentType, getContentLength, getContentEncoding, getDate, getExpiration, getLastModified`**
- **`getInputStream`** for reading and **`getOutputStream`** for writing

API of the class has a more detailed description.

Communicating with web servers

Can directly open a stream for reading in URL class:

- public final InputStream **openStream**() throws IOException

  url.opentStream()

  - Opens a connection to this URL and returns an InputStream for reading from that connection.
  - This method is a shorthand for: openConnection().getInputStream()

Retrieving Information

```
URL url = new URL(urlName);
URLConnection connection = url.openConnection();

connection.connect();

// print header fields
int n = 1;
String key;
while ((key = connection.getHeaderFieldKey(n)) != null)
{
   String value = connection.getHeaderField(n);
   System.out.println(key + ": " + value);
   n++;
}
```

Retrieving Information

```
// print convenience functions

  System.out.println("----------");
  System.out.println("getContentType: "
     + connection.getContentType() );

  System.out.println("getContentLength: "
     + connection.getContentLength() );

  System.out.println("getContentEncoding: "
     + connection.getContentEncoding() );
  ….
```

Retrieving Information

```
    // print first ten lines of contents
BufferedReader in = new BufferedReader(new
    InputStreamReader( connection.getInputStream() ));

String line;
n = 1;
while ((line = in.readLine()) != null && n <= 10)
{
        System.out.println(line);
        n++;
}
if (line != null) System.out.println(". . .");
```

Sending Information

Web servers receive information from clients using either GET or POST

- GET requests are requests made by browsers when the user
  - types in a URL on the address line,
  - follows a link from a Web page, or
  - makes an HTML form that does not specify a METHOD or specifically use the GET method.
- POST requests are generated when someone creates an HTML form that specifies METHOD="POST"
- Examples:
  - http://maps.yahoo.com/py/maps.py: python,

<form action="/py/maps.py?Pyt=Tmap&YY=28457" method=GET> … </form>

  - http://www.census.gov/ipc/www/idbprint.html:

<form method=post action="/cgi-bin/ipc/idbsprd">

Sending Information

Appropriate CGI (common gateway interface) script is called to process info received and produce an HTML page to send back to client

CGI scripts usually written in C, Perl, shell script. (Out of the scope of this course.)

Will discuss servlets, Java alternative to CGI scripts

Sending Information

Send information to CGI script using GET

- Attach parameters to the end of URL

  http://host/script?parameters

- Separate parameters using "&" and encode parameters as follows to avoid misinterpretation (URL encoding)
  - Replace space with "+"
  - Replace each non-alphanumeric character with "%" followed by the hexadecimal code of the character

    "Mastering C++" ➔ "Mastering+C%2b%2b"
- Disadvantage: long parameter string, might exceed limits of browsers.

Sending Information

Sending information to CGI script using POST:

Open URLConnection and send parameter using a stream

- Open a URLConnection:

```
URL url = new URL("http:/host/script");
URLConnection cnn = url.openConnection();
```

- Set up connection for output:

```
cnn.setDoOutput(true);
```

Sending Information

- Get a stream for sending data:

```
PrinterWriter out = new
    PrintWriter(cnn.getOutputStream());
```

- Send parameters

```
Out.print(name1 + "=" + URLEncoder.encode(value1, "UTF-8") + "&" );
Out.print(name2 + "=" + URLEncoder.encode(value2, "UTF-8") ) + "\n");
```

Note: URLEncoder: Utility class for HTML form encoding.

This class contains static methods for converting a String to the application/x-www-form-urlencoded MIME (*Multipurpose Internet Mail Extensions*) format.

The World Wide Web Consortium Recommendation states that the UTF-8 encoding scheme should be used.

Summary

- Essential Java Networking
  - Ports/IP/Sending/Receiving Data
  - Java Implementation Examples
- Hands-On/Practical
- Today is about becoming basic Java Networking

This Week

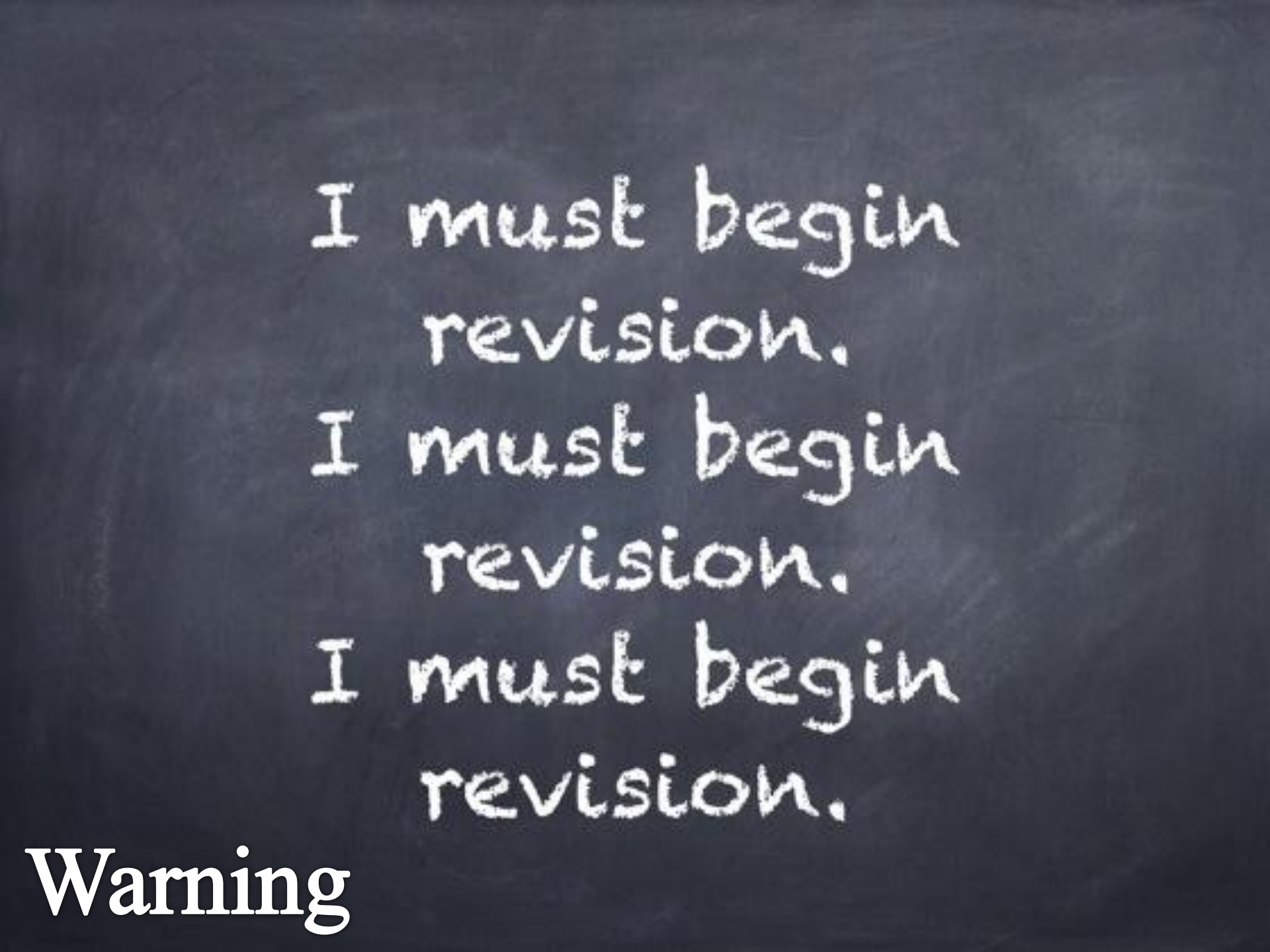
- Read Associated Chapters
- Review Slides
- Java Exercises
- Online Quizzes
  - ▷25th December (Last Date)
- Getting Ready Exam

Today's Exercises

- **Chapter 30 – Exercise 30.1**

Opportunity to review/revisit previous chapters

Solid understanding each of the concepts


I must begin
revision.
I must begin
revision.
I must begin
revision.

Warning

Questions/Discussion