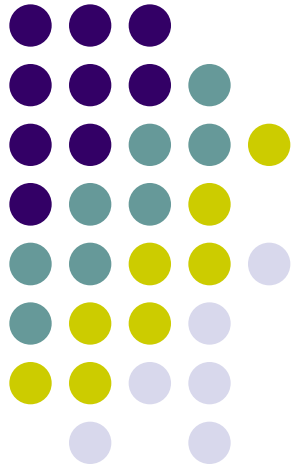
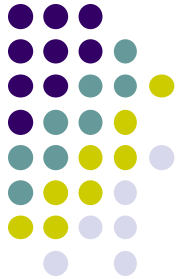
Networking Basics

10 Marks

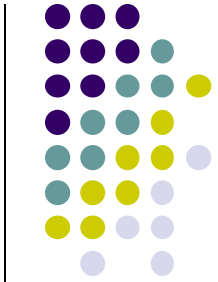


Socket



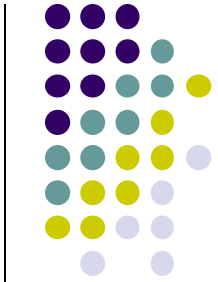
- A *network socket* is a lot like an electrical socket.
- Anything that understands the standard protocol can “plug in” to the socket and communicate.
- ***Internet Protocol (IP)*** is a low-level routing protocol that breaks data into small packets and sends them to an address across a network.

Socket (cont..)



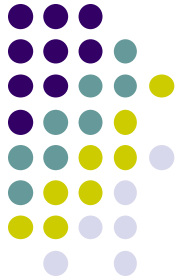
- ***Transmission Control Protocol (TCP)*** is a higher-level protocol that manages to robustly string together these packets, sorting and retransmitting them as necessary to reliably transmit your data.
- ***User Datagram Protocol (UDP)***, can be used directly to support fast, connectionless, unreliable transport of packets.

Client/Server



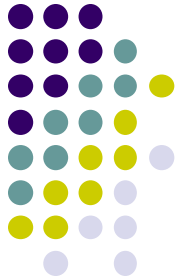
- A ***server*** is anything that has some resource that can be shared. There are *compute servers,* which provide computing power; *print servers,* which manage a collection of printers; *disk servers,* which provide networked disk space; and *web servers,* which store web pages.
- A ***client*** is simply any other entity that wants to gain access to a particular server.

Client/Server (cont..)



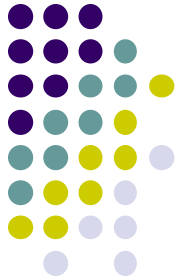
- A server process is said to “listen” to a port until a client connects to it.
- A server is allowed to accept multiple clients connected to the same port number, although each session is unique.
- To manage multiple client connections, a server process must be multithreaded or have some other means of multiplexing the
- simultaneous I/O.

Reserved Sockets



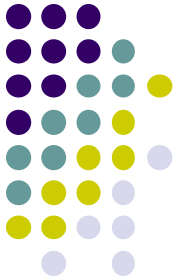
- TCP/IP reserves the lower 1,024 ports for specific protocols.
- Port number 21 is for FTP, 23 is for Telnet, 25 is for e-mail, 79 is for finger, 80 is for HTTP, 119 is for netnews—and the list goes on.
- It is up to each protocol to determine how a client should interact with the port.

Example of a client requesting a single file, /index.html, and the server replying that it has successfully found the file and is sending it to the client:



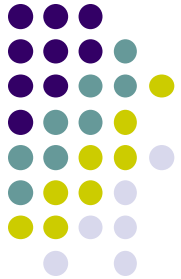
Server	Client
Listens to port 80.	Connects to port 80.
Accepts the connection.	Writes "GET /index.html HTTP/1.0\n\n".
Reads up until the second end-of-line (\n).	
Sees that GET is a known command and that HTTP/1.0 is a valid protocol version.	
Reads a local file called /index.html.	
Writes "HTTP/1.0 200 OK\n\n".	"200" means "here comes the file."
Copies the contents of the file into the socket.	Reads the contents of the file and displays it.
Hangs up.	Hangs up.

Proxy Servers



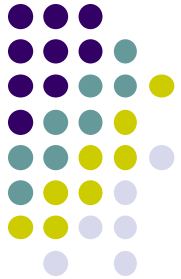
- A *proxy server* speaks the client side of a protocol to another server.
- A proxy server has the additional ability to filter certain requests or cache the results of those requests for future use.
- A caching proxy HTTP server can help reduce the bandwidth demands on a local network's connection to the Internet.

Internet Addressing



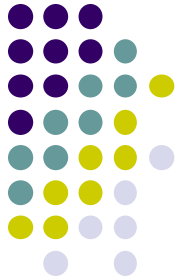
- Every computer on the Internet has an *address.*
- An Internet address is a number that uniquely identifies each computer on the Net.
- There are 32 bits in an IPv4 IP address, and we often refer to them as a sequence of four numbers between 0 and 255 separated by dots (.).

Internet Addressing (cont..)



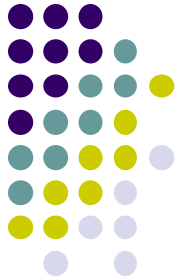
- The first few bits define which class of network, lettered A, B, C, D, or E, the address represents.
- Most Internet users are on a class C network, since there are over two million networks in class C.
- The first byte of a class C network is between 192 and 224, with the last byte actually identifying an individual computer among the 256 allowed on a single class C network.

Domain Naming Service (DNS)



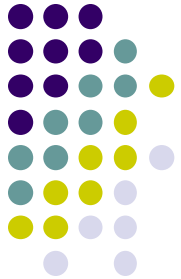
- It is difficult to imagine seeing “http://192.9.9.1/” at the bottom of an advertisement.
- A parallel hierarchy of names to go with all these numbers. It is called the *Domain Naming Service (DNS).*
- Just as the four numbers of an IP address describe a network hierarchy from left to right, the name of an Internet address, called its *domain name.*

InetAddress

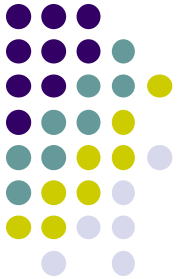


- The **InetAddress** class is used to encapsulate both the numerical IP address we discussed earlier and the domain name for that address.
- You interact with this class by using the name of an IP host, which is more convenient and understandable than its IP address.
- The **InetAddress** class hides the number inside.
- **InetAddress** can handle both IPv4 and IPv6 addresses.

Factory Methods



- The **InetAddress** class has no visible constructors.
- To create an **InetAddress** object, you have to use one of the available factory methods.
- *Factory methods* are merely a convention whereby static methods in a class return an instance of that class.



InetAddress factory methods are shown here.

static InetAddress getLocalHost( )
throws UnknownHostException

static InetAddress getByName(String *hostName*)
throws UnknownHostException

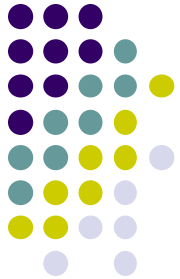
static InetAddress[ ] getAllByName(String *hostName*)
throws UnknownHostException

```
import java.net.*;

class InetAddressTest
{
  public static void main(String args[]) throws UnknownHostException {
    InetAddress Address = InetAddress.getLocalHost();
    System.out.println(Address);
    Address = InetAddress.getByName("osborne.com");
    System.out.println(Address);
    InetAddress SW[] = InetAddress.getAllByName("www.nba.com");
    for (int i=0; i<SW.length; i++)
      System.out.println(SW[i]);
  }
}
```

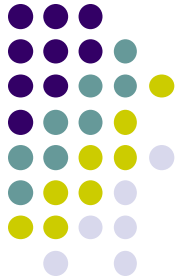
```
default/206.148.209.138
osborne.com/198.45.24.162
www.nba.com/64.241.238.153
www.nba.com/64.241.238.142
```

Instance Methods



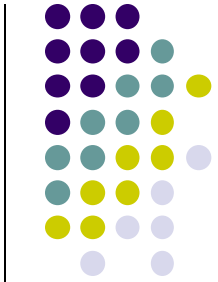
boolean equals(Object *other*)	Returns true if this object has the same Internet address as *other*.
byte[ ] getAddress( )	Returns a byte array that represents the object's Internet address in network byte order.
String getHostAddress( )	Returns a string that represents the host address associated with the **InetAddress** object.
String getHostName( )	Returns a string that represents the host name associated with the **InetAddress** object.
boolean isMulticastAddress( )	Returns true if this Internet address is a multicast address. Otherwise, it returns false.
String toString( )	Returns a string that lists the host name and the IP address for convenience.

TCP/IP Client Sockets



- TCP/IP sockets are used to implement reliable, bidirectional, persistent, point-to-point, stream-based connections between hosts on the Internet.
- A socket can be used to connect Java's I/O system to other programs that may reside either on the local machine or on any other machine on the Internet.

TCP/IP Client Sockets (cont..)

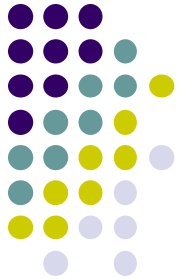


- There are two kinds of TCP sockets in Java.
  - The **ServerSocket** class is designed to be a "listener," which waits for clients to connect before doing anything.
  - The **Socket** class is designed to connect to server sockets and initiate protocol exchanges.
- The creation of a **Socket** object implicitly establishes a connection between the client and server.

Socket(String *hostName*, int *port*)	Creates a socket connecting the local host to the named host and port; can throw an UnknownHostException or an IOException.
Socket(InetAddress *ipAddress*, int *port*)	Creates a socket using a preexisting InetAddress object and a port; can throw an IOException.

A socket can be examined at any time for the address and port information associated with it, by use of the following methods:

InetAddress getInetAddress( )	Returns the InetAddress associated with the Socket object.
int getPort( )	Returns the remote port to which this Socket object is connected.
int getLocalPort( )	Returns the local port to which this Socket object is connected.

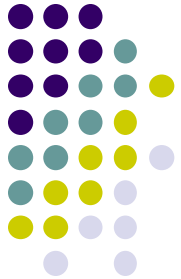


InputStream getInputStream( )	Returns the **InputStream** associated with the invoking socket.
OutputStream getOutputStream( )	Returns the **OutputStream** associated with the invoking socket.

```
//Demonstrate Sockets.
import java.net.*;
import java.io.*;

class Whois {
  public static void main(String args[]) throws Exception {
    int c;
    Socket s = new Socket("internic.net", 43);
    InputStream in = s.getInputStream();
    OutputStream out = s.getOutputStream();
    String str = (args.length == 0 ? "osborne.com" : args[0]) + "\n";
    byte buf[] = str.getBytes();
    out.write(buf);
    while ((c = in.read()) != -1) {
      System.out.print((char) c);
    }
    s.close();
  }
}
```

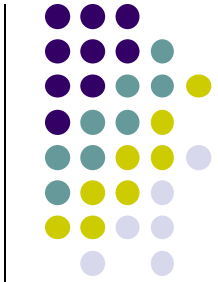
URL



- The *URL* provides a reasonably intelligible form to uniquely identify or address information on the Internet.
- Every browser uses them to identify information on the Web.
- **Format**

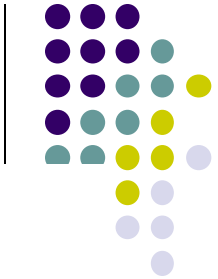
  http://www.osborne.com:80/index.htm

URL (cont..)



- URL(String *urlSpecifier*)
- URL(String *protocolName*, String *hostName*, int *port*, String *path*)
- URL(String *protocolName*, String *hostName*, String *path*)

URL (cont..)



```
// Demonstrate URL.
import java.net.*;
class URLDemo {
  public static void main(String args[]) throws MalformedURLException {
    URL hp = new URL("http://www.osborne.com/downloads");

    System.out.println("Protocol: " + hp.getProtocol());
    System.out.println("Port: " + hp.getPort());
    System.out.println("Host: " + hp.getHost());
    System.out.println("File: " + hp.getFile());
    System.out.println("Ext:" + hp.toExternalForm());
  }
}
```

```
Protocol: http
Port: -1
Host: www.osborne.com
File: /downloads
Ext:http://www.osborne.com/downloads
```

URL Connection

```
// Demonstrate URLConnection.
import java.net.*;
import java.io.*;
import java.util.Date;

class UCDemo
{
  public static void main(String args[]) throws Exception {
    int c;
    URL hp = new URL("http://www.internic.net");
    URLConnection hpCon = hp.openConnection();

    // get date
    long d = hpCon.getDate();
    if(d==0)
      System.out.println("No date information.");
    else
      System.out.println("Date: " + new Date(d));

    // get content type
    System.out.println("Content-Type: " + hpCon.getContentType());

    // get expiration date
    d = hpCon.getExpiration();
    if(d==0)
      System.out.println("No expiration information.");
    else
      System.out.println("Expires: " + new Date(d));

    // get last-modified date
    d = hpCon.getLastModified();
    if(d==0)
      System.out.println("No last-modified information.");
    else
      System.out.println("Last-Modified: " + new Date(d));

    // get content length
    int len = hpCon.getContentLength();
    if(len == -1)
      System.out.println("Content length unavailable.");
    else
      System.out.println("Content-Length: " + len);

    if(len != 0) {
      System.out.println("=== Content ===");
      InputStream input = hpCon.getInputStream();
      int i = len;
      while (((c = input.read()) != -1)) { // && (--i > 0)) {
        System.out.print((char) c);
      }
      input.close();

    } else {
      System.out.println("No content available.");
    }

  }
}
```

Output

Date: Sat Apr 27 12:17:32 CDT 2002
Content-Type: text/html
No expiration information.
Last-Modified: Tue Mar 19 17:52:42 CST 2002
Content-Length: 5299
=== Content ===

```
<html>

<head>
<title>InterNIC | The Internet's Network Information Center</title>
<meta name="keywords" content="internic,network information, domain registration">
<style type="text/css">
<!--
p, li, td, ul {  font-family: Arial, Helvetica, sans-serif}
-->
</style>
</head>
```