8. Java Network Programming

Outline

- Overview of TCP/IP Protocol
- Uniform Resource Locator (URL)
- Network Clients and servers
- Networking Support Classes
- InetAddress Class and Examples
- URL class and Examples
- Networking with TCP/IP Sockets
- Socket Examples
- Applet and HTML Documents
- Applet and URL Examples
- Simple Web Browser

Introduction

Java language includes a set of API classes and methods in the java.net for supporting TCP/IP network access. However, the API's for such things as internet email, USENET news, and ftp were removed from the java.net.* hierarchy and placed in the sun.net.* hierarchy.

TCP/IP Protocol

The Internet Layer

Internet Protocol - (IP, RFC791)

- A protocol that provides the packet delivery service for TCP, UDP, and IMCP
- A connectionless protocol
- Unreliable service (no error detection and recovery)
- Has handshaking: verify the readiness; agreement parameters
- Routing

IP Datagrams

- An unreliable layer, connection less and unreliable delivery system
- Every IP datagram contains: source address, destination address
- Each IP datagram independent of all others, and can be routed independently

IP Address Structure

- The Internet Network Information Center (InterNIC)
- For information and Policy: http://rs.internic.net/rs-internic.html
- TCP/IP uses the administrator assigned 32-bit (48-bit) IP addresses
- A unique set of address that identifies each network on the inter-network
- Each device on the network has its unique address
- Symbolic address: (1) the network address, (2) the node address
- Dotted decimal notation (Most Left bit = bit 0, Most Right bit = bit 31)

- Network classes: A, B, C, D

Network Classes

- Class A:
  - bit 0= 0, 7-bit network ID (127 class A networks), 3 node bytes
  - Millions of hosts
  - Decimal Range 0 - 127
- Class B:
  - First two bits = 10, 2-byte network ID, 2 node bytes
  - Thousands of hosts
  - Decimal range 128-191
- Class C:
  - First three bits = 110, 3-byte network ID, 1 node byte
  - 254 host
  - Decimal range 192-223
- Class D:
  - First 4 bits = 1110
  - Multicast address
  - Identifying a group of computers that can run a distributed applications on the network

Subnet Masks

Due to:

- Network traffic
- Type of Medium Access Control: Token-ring, Ethernet, etc
- Type of Application: real-time multimedia and imaging (Fiber-optic DDI)
- Organizational mergers
- Geographical proximity

We can:

- Break a single network into smaller networks
- Each subnet accommodating certain number of users or
- Each subnet accommodating certain number of applications

Class B network

Without subnet mask

- 2-byte network ID, 2-byte node ID

With subnet mask

- 2-byte network ID, 1 byte subnet ID, 1 byte node ID

Default Subnet masks

Class A: 255.0.0.0

Class B: 255.255.0.0

Class C: 255.255.255.0

UDP (User Datagram Protocol)

- UDP Support Protocols
    - Internet Name Server Protocol (INSP)
    - Simple Network Management Protocol (SNMP)
    - Echo service (ping)
    - Trivial File Transfer Protocol (TFTP)
- Possible UDP Applications
    - Broadcast or multicast services
    - Real-time data (video, audio, industrial control, etc)
    - Short transaction time that assume implicit acknowledgement and tolerance on duplicate datagrams
- Service Ports
    - 16-bit port address, fixed binding
    - Both UDP and TCP use port addressing to deliver information
    - Well-known port (reserved): port number from 1024 and below
    - Port number between 1024 and 5000 are usually for custom servers

TCP (Transmission Control Protocol)

- A connection oriented protocol that provides a reliable, full-duplex byte stream for a user process.
- Since TCP uses IP, the entire Internet protocol suite is often called the TCP/IP protocol family.
- Peer-to-peer transport protocol

The Service File (UNIX)

- The Service File (Port Translation) is a flat file database
- Service Name: Port Number: Protocol
- The Internet Request for Comment RFC 1060 defines the file format
- A partial listing of a common service file:

```
apollo% cat /etc/services
#ident  "@(#)services   1.16    97/05/12 SMI"   /* SVr4.0 1.8   */
#
# Network services, Internet style
#
# <service name> <port number>/<protocol> [aliases ..] [#<comment>]
tcpmux          1/tcp
echo            7/tcp
echo            7/udp
discard         9/tcp           sink null
discard         9/udp           sink null
```

The Service File (continue)

```
systat          11/tcp          users
daytime         13/tcp
daytime         13/udp
netstat         15/tcp
chargen         19/tcp          ttytst source
chargen         19/udp          ttytst source
ftp-data        20/tcp
ftp             21/tcp
telnet          23/tcp
smtp            25/tcp          mail
time            37/tcp          timserver
time            37/udp          timserver
name            42/udp          nameserver
whois           43/tcp          nicname         # usually to sri-nic
domain          53/udp
domain          53/udp
domain          53/tcp
bootps          67/udp                          # BOOTP/DHCP server
bootpc          68/udp                          # BOOTP/DHCP client
hostnames       101/tcp         hostname        # usually to sri-nic
sunrpc          111/udp         rpcbind
sunrpc          111/tcp         rpcbind
#
```

Uniform Resource Locator (URL)

- A key concept in the operation of the World-Wide Web (WWW)
- A compact representation of the location and access method for a resource available via the internet
- Four address elements:
  - Protocol
  - Host
  - Port
  - Document file

RFC 1738 defines URL formats for the following access schemes

- file Host-specific file names
- ftp File Transfer Protocol
- gopher The Gopher Protocol
- http Hypertext Transfer Protocol
- mailto Electronic mail address
- news USENET news
- nntp USENET news using NNTP access
- prospero Prospero Directory Service
- telent Telnet Protocol for Interactive sessions
- wais Wide-Area Information Servers

URL Schemes

```
<scheme>:<scheme-specific-part>
Scheme      Default Port            Syntax
ftp             21
ftp://<user>:<password>@<host>:<port>/<cwd1>/<cwd2>/../<cwdN>/<name>;
type=<typecode>

http            80          http://<host>:<port>/<path>?<searchpart>
gopher          70          gogher://<host>:<port>/<selectror>
```

Network Clients and Servers

Network Client

- Actively forms a connection to a computer, printer, etc., on the network
- Have a network address (dynamic or static)

Network Server

- Connected to a network continuously and wait for a client to connect to it and provide services
- Has a static network address

Distributed Client/Server Applications

- Point-to-point connection
- Identified as a <host, socket> pair

Networking Support Classes

The **java.net** library contains the following classes for networking support

- Web HTTP support classes: `InetAddress` , `java.net.URL`, `java.net.URLConnection`, `HttpURLConnection`
- Client-side networking support: Socket, and InetAddress
- TCP-based client/server application support classes: ServerSocket, and InetAddresss
- Datagram networking using UDP protocol: `java.net.DatagramSocket`
- Extension classes: ContentHandler and URLStreamHandler

Java Sockets

- Provide an interface between a client and a server
- TCP connection oriented version:
    - Java client side socket: Socket class
    - Java server side socket: ServerSocket class
- UDP connectionless version: DatagramSocket class

InetAddress Class

- **java.net.InetAddress** represents an Internet address class
- Two fields: hostname (a name string), and address (IP address of integer type)
- Examples

```
import java.net.*;  // class library
InetAddress netaddr1 =
      InetAddress.getByName("149.164.36.20");

InetAddress netaddr2 =
      InetAddress.getByName("www.msdn.microsoft.com");

InetAddress netaddr3 =  InetAddress.getByName("www.microsoft.com ");

InetAddress[]netaddrs =
      InetAddress.getAllByName("www.microsoft.com ");

InetAddress loacladdr = InetAddress.getLocalHost();

If (netaddrs.equals(netaddr3){action()};

public int hashCode()  .. generate a key for hash table

public String toString()  .. for passing InetAddress objects to
System.out.println()
```

InetAddress Programming Examples

8-1. Examine local host IP address.

```
// seeLocalIP.java
//
// RESULTS:
// lin/149.164.36.104


import java.net.*;

class seeLocalIP{

  public static void main (String args[]) {

    try {
      InetAddress myIP = InetAddress.getLocalHost();
      System.out.println(myIP);
    }
    catch (UnknownHostException e) {
      System.out.println("No IP address available.");
    }

  }

}
```

8-2. Examine remote server IP address

```
// DomainToIp.java
// Examine IP addresses:
// www.amc.org/206.247.88.6
// www.triplecrown.com/205.207.122.203

import java.net.*;
class DomainToIp{
  public static void main (String args[]) {
    try {
      InetAddress[] addrs =
InetAddress.getAllByName("www.triplecrown.com");
      for (int n = 0; n < addrs.length; n++) {
        System.out.println(addrs[n]);
      }
    }
    catch (UnknownHostException e) {
      System.out.println("Could not find IP");
    }
  }

}
```

8-3. Examine all IP addresses assigned to a domain name

```
// seeAllMicrosoftIps.java
//
// Examine all IP addresses:
//www.microsoft.com/207.46.197.102
//www.microsoft.com/207.46.230.229
//www.microsoft.com/207.46.230.218
//www.microsoft.com/207.46.197.100

import java.net.*;

class seeAllMicrosoftIps {

  public static void main (String args[]) {

    try {
      InetAddress[] addrs =
                  InetAddress.getAllByName("www.microsoft.com");
      for (int n = 0; n < addrs.length; n++) {
        System.out.println(addrs[n]);
      }
    }
    catch (UnknownHostException e) {
      System.out.println("Could not find any www.microsoft.com");
    }

  }

}
```

8-4. Examine any IP address of a domain name passed through the command line.

```
// seeAnyIPs.java
// Examine all IP addresses of a domain name passed
// through the command line arguments.
// To run this program:
// java seeAnyIPs www.ipfw.edu
//
// args[0] -- point to the argment: www.ipfw.edu
//
import java.net.*;
class seeAnyIPs {
  public static void main (String args[]) {
    try {
      InetAddress[] addrs = InetAddress.getAllByName(args[0]);
      for (int n = 0; n < addrs.length; n++) {
        System.out.println(addrs[n]);
      }
    }
    catch (UnknownHostException e) {
      System.out.println("Could not find any IP address");
    }
  }
```

```
}
```

Output:

```
E:\LinJava\LinJavaExs>javac seeanyips.java

E:\LinJava\LinJavaExs>java seeAnyIPs www.ipfw.edu
www.ipfw.edu/149.164.187.23

E:\LinJava\LinJavaExs>java seeAnyIPs www.purdue.edu
www.purdue.edu/128.210.11.200
```

HTTP Support Classes

- HTTP servers provide a standard means for serving and accessing data objects
- On the client side, we need only to use the URL and/or URLConnection objects to access documents
- Require some extra communication bandwidth than sockets

URL Class

- **java.net.URL** is the class for downloading the contents of a remote resource named by the uniform resource locator (URL)
- Examples of creating URL classes

```
URL webURL, ftpURL;
URL u = new URL("http:www.microsoft.com");
webURL  = new URL("149.164.36.20");
URL u = new URL("http:www.microsoft.com");
```

- URL includes six fields
  - the protocol
  - the host
  - the port
  - the document file
  - the URLStreamHandler
  - the named anchor or ref
- Methods
  - String getFile() .. return the file part of the URL
  - String getHost() .. return the host name of the URL
  - int getPort() .. return the port number part of the URL

- String getProtocol() .. return the protocol part of the URL
- String getRef() ..return the reference part of the URL
- getContent()
- openConnection()
- openSteam()

URL Programming Examples

8-5. Examine the HTTP and FTP protocols using URL objects.

```
// ieeeURL.java
//
// This program examine the URL of the IEEE.
//
// RESULT:
// http://www.ieee.org/index.html
// ftp://www.ieee.org/
import java.net.*;
public class ieeeURL {
  public static void main (String args[]) {
    URL http_URL, ftp_URL;
    try {
        http_URL = new URL("http://www.ieee.org/index.html");
        System.out.println(http_URL);
        ftp_URL = new URL("ftp://www.ieee.org");
        System.out.println(ftp_URL);
    }
    catch (MalformedURLException e)  {
      System.err.println(e);
    }
  }
}
```

Output:

```
E:\LinJava\DemoProgs\URL_Web>java ieeeURL
http://www.ieee.org/index.html
ftp://www.ieee.org
```

8-6. Download a HTML Web Page

```
//getWebpage.java
//
// This program is executed from command line. It requires a web
// page address as the argument. We can display the received page
// on the screen or save it on the disk using redirection.
//
//


import java.net.*;
import java.io.*;

public class getWebpage {

  public static void main (String args[]) {

    URL aURL;
    String inBuffer;

    if  (args.length > 0) {
      try {
        aURL = new URL(args[0]);

        try {
          DataInputStream HTMLpage = new
                              DataInputStream(aURL.openStream());

          try {
            while ((inBuffer = HTMLpage.readLine()) != null) {
              System.out.println(inBuffer);
            }
          }
          catch (Exception ex) {
            System.err.println(ex);
          }
        }
        catch (Exception ex) {
          System.err.println(ex);
        }
      }
      catch (MalformedURLException ex) {
        System.err.println(args[0] + " not a URL");
        System.err.println(ex);
      }
    }
  }
}
```

Output:

```
E:\LinJava\DemoProgs\URL_Web>java getWebpage http://149.164.36.204 >
ecetLabServ
E:\LinJava\DemoProgs\URL_Web>java getWebpage http://www.ipfw.edu
```

8-7. This program allows a user to display the text format or download a HTML page from the selected URL.

```
// GetURLdata.java
// This program is executed from command line. It requires a web
// page address as the argument.
// For examples, we can display the received page
// on the screen in text format:
// E:\Networking>java GetURLdata http://www.ipfw.edu
// OR save it on the disk:
// E:\Networking>java GetURLdata http://www.ipfw.edu ipfwpage.html
//
import java.io.*;
import java.net.*;

public class GetURLdata {
  public static void main(String[] args) {
    InputStream readIn = null;
    OutputStream writeOut = null;
    try {
      // Check the arguments
      if ((args.length != 1) && (args.length != 2))
        throw new IllegalArgumentException("Wrong number of
          arguments");

      // Set up the input and output streams
      URL newUrl = new URL(args[0]);
      readIn = newUrl.openStream();
      if (args.length == 2)
        writeOut = new FileOutputStream(args[1]);
      else writeOut = System.out;

      // Copy stream from the newUrl to the output stream
      byte[] buffer = new byte[4096];
      int bytes_read;
      while((bytes_read = readIn.read(buffer)) != -1)
        writeOut.write(buffer, 0, bytes_read);
    }
    // On exceptions, print error message and usage message.
    catch (Exception e) {
      System.err.println(e);
      }
    finally {
      try {
         readIn.close();
           writeOut.close();
          }
      catch (Exception e) {}
    }
  }
}
```

Output:

```
E:\java GetURLdata http://www.ipfw.edu ipfwpage.html
E:\java GetURLdata http://www.ipfw.edu
```

URLConnection Class

- Communicate directly with a server
- Access everything send by the server in the raw form (HTNL, plaint text, binary image data, protocol headers in use)

- URLConnection methods in the initialization phase
    - connect
    - setAllowUserInteraction
    - setContentHandleFactory
    - setDefaultUserCaches
    - setDefaultAllowUserInteraction
    - setDoInput
    - setDoOutput
    - setIfModifiedSince
    - setRequestProperty
    - setUseCaches

- URLConnection methods in the connected state
    - getAllowUserInteraction
    - getDefaultRequestFactory
    - getInputStream
    - getOutputStream
    - getUseCaches
    - getDefaultUseCaches
    - getDefaultAllowUserInteraction
    - getIfModifiedSince
    - getDoOutput
    - getIfModifiedSince

- URLConnection methods in the idle state
    - Applicable methods used in the connected state

- URLConnection parses HTTP header through the following methods:
  getDate()
  getExpiration()
  getContentLength()
  getContentType()
  getLastModified()
  getContentEncoding()
  getHeaderField()

HttpURLConnection Class

- A subclass of URLConnection (new for JDK 1.1 and up)
- Provide additional HTTP/1.1 specific methods: GET, POST, PUT, HEAD, TRACE, DELETE, and OPTIONS
- HttpURLConnection Methods:
  disconnet()
  boolean getFollowRedirects()
  String getRequesteMethod()
  int getResponseCode()
  String getResponseMessage()
  getFollowRedirects(boolean)
  setRequestMethod(String)
  Boolean usingProxy()

Networking with TCP/IP Sockets

A TCP/IP socket performs the following basic operations:

- Open a full-duplex connection to a remote machine (ready for sending and receiving information)
- Send data stream
- Receive data stream
- Close a connection
- Bind to a port
- Listen for incoming data
- Accept connection for remote machine on a predetermine port

Networking with Java Sockets

- Provide a stream interface to the TCP protocol
- It will open a Java version of a TCP connection to the remote host with either a domain name or an IP address
- We can use obtain the information about the connection through the methods as listed above

java.net.Socket (for both clients and servers)

// Constructors

protected Socket()

    // Creating an unconnected socket, with default setup

Socket(InetAddress address, int port)

Socket(InetAddress address, int port, InetAddress localAddr, int localPort)

Socket(String host, int port)

Socket(String host, int port, InetAddress localAddr, int localPort)

// Methods

void close()

    // Close this socket and disconnect from the remote server

InetAddress getInetAddress()

```
        // Return the address through which the socket is connected
        // Get the Internet address for the connected Socket
InputStream getInputSream()
        // Get an input stream for the connection
boolean getKeepAlive()
        //
InetAddress getLocalAddress()
        // Get the local port to which this socket is bound
OutputStream getOutputStream()
        // Get an output stream for the connection
int getLocalPort()
        // Get the local port number for the Socket in use
int getPort()
        // Get the remote port number for the Socket in use
int getReceiveBufferSize()
int getSendBufferSize()
int getSoLinger()
        // Returns setting for SO_LINGER
int getSoTimeOut()
        // Returns setting for SO_TIMEOUT
boolean getTcpNoDelay()
        //Tests if TCP_NODELAY is enabled
void setKeepAlive(boolean on)
        // Enable/disable SO_KEEPALIVE
void  setReceiveBufferSize(int size)
void setSendBufferSize(int size)
static void setSocketImplFactory(SocketImpFactory fac)
        // Set the client socket implementation factory for the application
void setSoLinger(boolean on, int linger)
        // Enable/disable SO_LINGER with the specified linger time
        // in seconds
```

```
void setSoTimeOut(int timeout)
        //Setting for SO_TIMEOUT in milliseconds
boolean getTcpNoDelay()
        //Enable/disable TCP_NODELAY
void shutdownInput()
        // Places the input stream for this socket at "end of stream"
void shutdownOutput()
        // Disables the output stream for this socket
String toString()
        // Converts this socket to a String
```

Socket Client

1. Create a Socket object for the connection to a remote server
2. Create an output stream for sending data to the Socket
3. Create an input stream for receiving data
4. Do Input/Output
5. Close the Socket connection

Networking Servers

- Properties of a server
    - Connected to a network continuously
    - Wait for a client to connect to it and provide services
    - Has a static network address
- Types of servers:
    - Sysnchronous server
    - Asynchronous server (background processing)
    - Multiclient web server
- States of a server
    - Instantiated or constructed
    - Accepting clients
    - Connected

java.net.ServerSocket methods

- ServerSocket()
- ServerSocket.accept() – Listen for incoming connections
- ServerSocket.close() -- No longer listen for new connections
- ServerSocket.getLocalPort() – Get the local port number for the Socket in use
- ServerSocket.getInetAddress() – Get the Internet address for the connected Socket

States of a ServerSocket

- Instantiated a new Server Object
- Accepting client connections
- Connected to the remote client
- Close all remote client connections

Socket Examples

Establishing a Simple Server (localhost: 127.0.0.1)

GUI supporting objects:

- `JTextField enter;`
- `JTextArea display`
- `JScrollPane(display);`

I/O Streams Objects:

- `ObjectOutputStream output;`
  - `output.writeObject(message);`
  - `output.flush()`
  - `output.close()`
- `ObjectInputStream input;`
  - `input.readObject`
  - `input.close()`

1. Create a ServerSocket object
2. Listen to client connections
3. Communicate with client
4. Processing Connection
5. Close the server socket

```
output.close()
input.close()
connection.close()
```

Client Java

Applets

- Properties
    - Embedded in a HTML page
    - It sends by a server for perform client-side application only
    - Cannot be used to build a server from an applet
    - Have limited access to files
    - Some security restrictions
    - Can only make connections back to the host from which they sent
    - Cannot utilized platform-native libraries
- Referring to host directories
    - codebase – the directory the applet come from
    - document base – the directory that contains the HTML page in which the applet is embedded
- Other Class Related Methods
    - public URL getDocumentBase()
    - public URL getCodeBase()
- Applications
    - Using applet for chat application
    - Using applet for downloading data such as images, sounds, etc

Applets Class

- The Applet class interface and runtime context of the browser/applet viewer are designed to isolate the code in the Applet from the machine on which the applet is running

java.applet.Applet (Java 2 Std. Ed. v1.3)

```
public class Applet extends Panel
//Applet Methods
void destroy()
        // Called by the browser or applet
AccessibleContext getAccessibleContext()
        // Get the accessible context associated with this applet
AppletContext getAppletContext()
        // Get this applet context that allows the applet to query and affect its
        // environment.
String getAppletInfo()
        // Return information about
AudioClip getAudioClip(URL url)
AudioClip getAudioClip(URL url, String name)
URL getCodeBase()
        //Gets the base URL
URL getDocumentBase()
        // Returns an absolute URL naming the directory of the document in which
        // the applet is embedded
Image getImage(URL url)
Image getImage(URL url, String name)
Local getLocal()
String getParameter(String name)
        // Return the value prepared of the named parameter in the HTML tags:
        // <PARAM NAME= "name" VALUE ="url">
String[ ][ ] getParameterInfo()
```

```
        // Returns information about the parameters than are undertood by this
        // applet
void init()
        //Informs the browser that this applet has been loaded into the system
boolean isActive()
        // Determine if this applet is active
static AudioClip newAudioClip(URL url)
        // Get an audio clip from the given URL
void play (URL url)
        // Plays the audio clip at the specified absolute URL
void play (URL url, String name)
        // Plays the audio clip at the specified absolute URL
void resize(Dimension d)
        // Request the applet to be resized
void resize(int width, int height)
        // Request the applet to be resized
void setStub(AppletStub stub)
        // Replace the Web document window with this
        // document referenced by URL
void showStatus(String msg)
        // Display msg in the status window
void start()
        // Called by the browser or applet to start its execution
void stop()
        // Called by the browser or applet to stop its execution
```

javax.swing.JApplet

public class JApplet extends Applet implements Accessible,
RootPaneConatianer

Applet Life Cycle Control

- The Java system provides methods for controlling the execution of an applet. The methods as listed below are contained in the class Applet:
    - init() -- initialize the applet each time that it is loaded or unloaded
    - start() -- when the applet is loaded, and begins executing its current task
    - stop() -- when the user leaves the page or exits the browser, the system will call this method to suspend any outstanding tasks and threads
    - destroy() -- clean up applet before it is unloaded from the browser. All tasks and threads are halted

A Complete Interactive Applet should be able to

- Read user input
- Make decisions
- Uses arrays
- Perform output
- Reacts to the applet life cycle

How To Create An Applet?

1. Create a project and write Java code in files with `.java` as an extension
2. Compile the .java file into byte code with .class extension
3. Design an HTML document that includes a statement to call a compiled Java class
4. Open the HTML document using a Web browser such as Netscape Navigator, Microsoft Internet Explorer, or an AppletViewer

java.applet.AppletContext

Applet getApplet(String name)

    // Find and return the applet

```
Enumeration getApplets()
        // Find all applets in the document
AudioClip getAudioClip(URL url)
        // Create an audio clip
Image getImage(URL url)
        // Return an image object
void showDocument(URL url)
        // Replace the Web document window with this
        // document referenced by URL
void showDocument(URL url, String target)
        // document referenced by URL
void showStatus(String status)
        // Display information at status window
```

java.util.Hashtable

public class Hashtable extends Dictionary implements Map, Cloneable, Serializable

A hash table stores Key/Value pairs.

```
//Cosntructors
Hashtable()
Hashtable(int initialCapacity)
Hashtable(int initialCapacity, float loadFactor)
// Methods
void clear()
        // Clear this hash table and remove all keys
Object clone()
        // Creates a shallow copy
boolean contains(Object value)
        // Test if some key maps into this value
boolean containsKey(Object key)
        // Test if this is a key
```

```
Enumeration elements()
        // Returns an enumeration of the values in this hash table
Set entrySet()
        // Returns a Set view of the entries
boolean equals(Object o)
        // Compare the specified Object with this Map for equality
Object get(Object key)
        // Return the value associated with the key
int hashCode()
boolean isEmpty()
        // Tests if this hash table maps no keys to values
Enumeration keys()
        // Returns an enumeration of the keys in this hashtable
Set keySet()
        // Returns a Set view of the keys contained in this Hashtable
Object put(Object key, Object value)
        // Maps the specified key to the specified value
void putAll(Map t)
        // Copies all of the mappings from the specified Map to this Hash table
protected void rehash()
        // Increase the capacity
Object remove(Object key)
        // Remove the key and its corresponding value from the
        // Hash table
int size()
        // Returns the number of keys in this Hash table
String toString()
        // Returns a string representation of this Hash table
Collection values()
        // Returns a collection view of the values contained in this table
```

Applet and URL Examples

Reading a File on a Web Server

The class ReadServerFile and other system classes used in this application:

Class JTextField

- JTextField enter = new JTextField("Enter file URL here");
- enter.actionListener()
- actionPerformed(Action Event e)
- e.getActionCommand()
- enter.setText(location)

JFrame

setCursor(Cursor.getPredefinedCursor(Cursor.WAIT_CURSOR)):

setCursor(Cursor.getPredefinedCursor(Cursor.DEFAULT_CURSOR)):

javax.swing.JEditorPane

public class JEditorPane extends JTextComponent

Class JEditorPane

String getContentType()

    // text/plain, text/html, text/rtf

// Methods to load content into the JEditorPane componenets

void setText(String t)

void read(InputStream in, Object desc)

void setPage(String url)

Class JEditorPane

- For rendering both plain text and HTML formatted text

  JEditorPane contents = new JEditorPane()

  contents.setEditable(false);

```
contents.addHyperlinkListener(
        new HyperlinkListener(){
        public void hyperlinkUpdate(HyperlinkEvent e)
          {
            if (e.getEventType() ==
                HyperlinkEvent.EventType.ACTIVATED)
                getThepage(e.getURL.toString());
          }
        }
      }
```

- content.setPage(location)
  // Down load the document specified by location