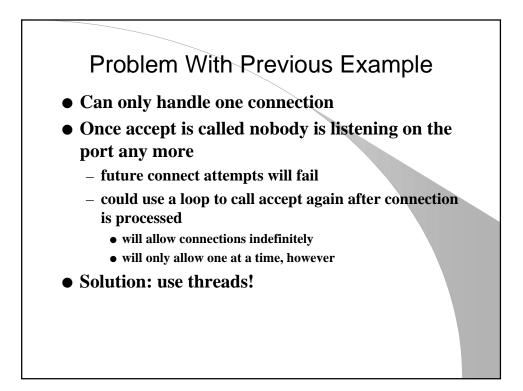


```
Echo server example
import java.io.*;
import java.net.*;
class EchoServer {
 public static void main( String [] args ) {
   Socket sock = null;
   try {
     ServerSocket ss = new ServerSocket( 3737 ); // Use port 3737
     sock = ss.accept( );
     InputStreamReader in = new InputStreamReader( sock.getInputStream( ) );
     BufferedReader is = new BufferedReader( in );
     PrintWriter os = new PrintWriter( sock.getOutputStream( ), true );
     os.println( "Welcome to the EchoServer!" );
     os.println( "Enter " + "***" + " to exit" );
     String str;
     while( ( str = is.readLine() ) != null && !str.trim().equals( "***" ) )
        os.println( str );
    }
    catch( IOException e ) { /* Could write to a log file */ }
    finally
                        { /* close stuff here */
                                                           }
 }
}
```

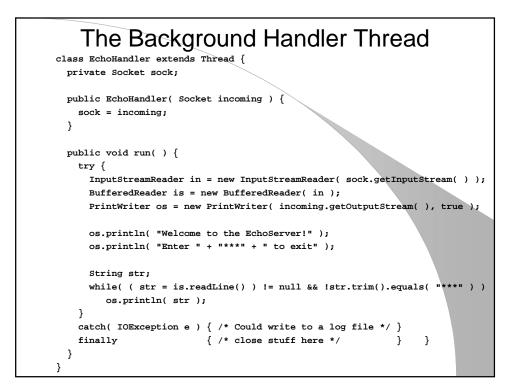


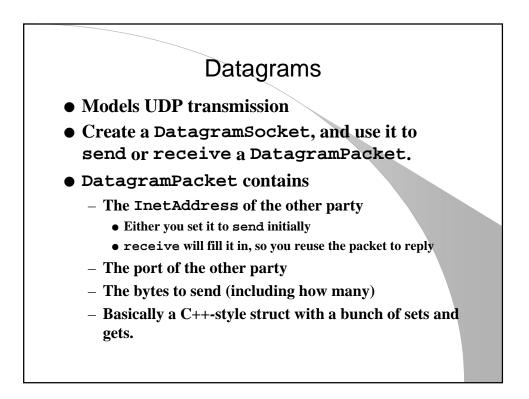
Threads And Networking

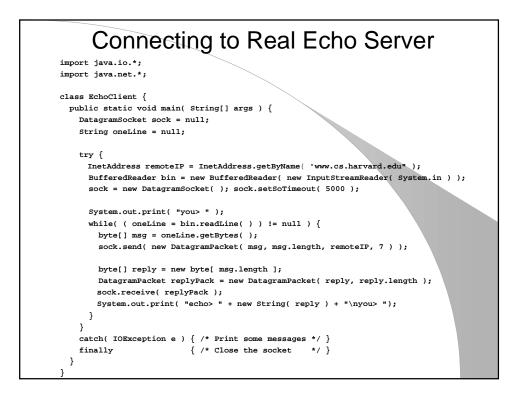
• main thread

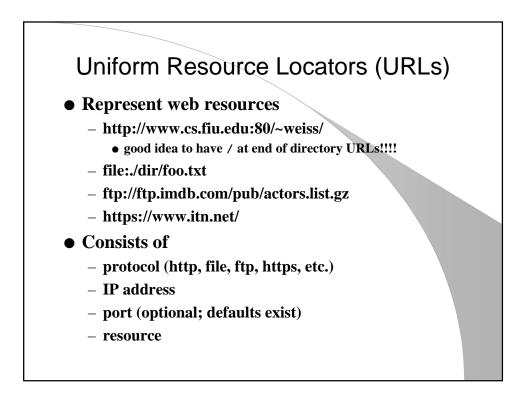
- creates the ServerSocket
- has a tight loop listening for a connection
- when a connection comes in, main thread spawns a background thread
 - passes the socket to the background thread
 - background thread processes the connection
 - main thread resumes listening for a connection
 - multiple simultaneous connections possible, subject to system limits
 - main thread runs indefinitely
 - main thread could keep a shared list of all background threads it has spawned: chatroom possibilities!

A Better Echo Server: Main
import java.io.*;
<pre>import java.net.*;</pre>
<pre>public class BetterEchoServer {</pre>
<pre>public static void main(String [] args) {</pre>
ServerSocket ss = null;
try {
<pre>ss = new ServerSocket(3737);</pre>
<pre>while(true) { Carbot area = cr accept(); </pre>
Socket sock = ss.accept(); Thread t = new EchoHandler(sock);
t.start();
}
}
<pre>catch(IOException e) { /* Could write to a log file */ }</pre>
finally { /* close stuff here */ }
}
}





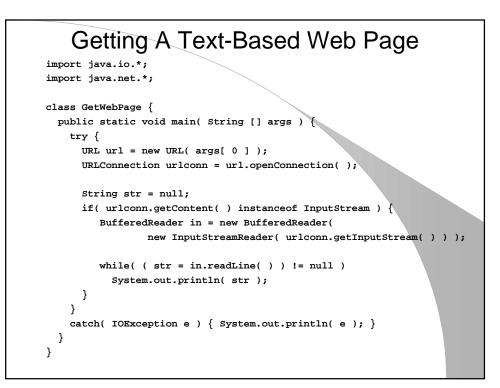


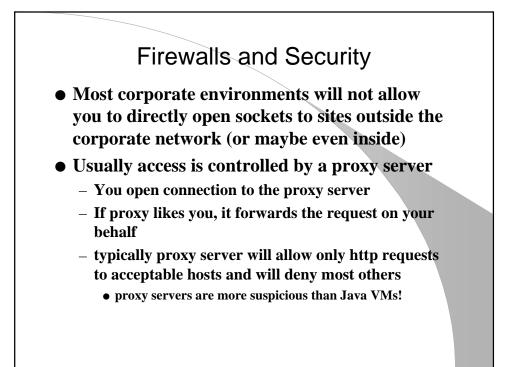


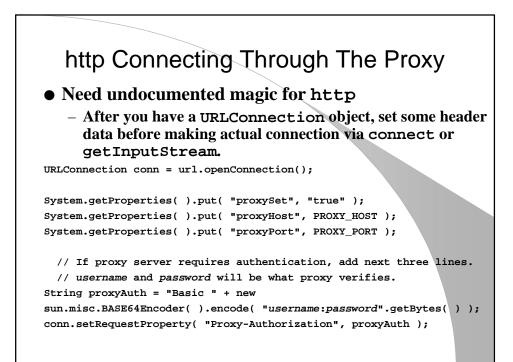
Java Classes

• URL

- abstracts the notion of a URL
- supports http, ftp, file
- https ok in browser if you download JSSE
- URLConnection
 - abstract class abstracts the notion of a connection
 - can optionally set request headers
 - then make connection
 - then optionally get returned header info
 - then access resource with both input stream and outputstream (for instance, to post forms)
 - can define your own protocols







Password-Protected Web Pages

• Can access password protected pages by including an additional property in the header

// name and pwd will be what web page verifies.
String webAuth = "Basic " + new
sun.misc.BASE64Encoder().encode("name:pwd".getBytes());
conn.setRequestProperty("Authorization", webAuth);

