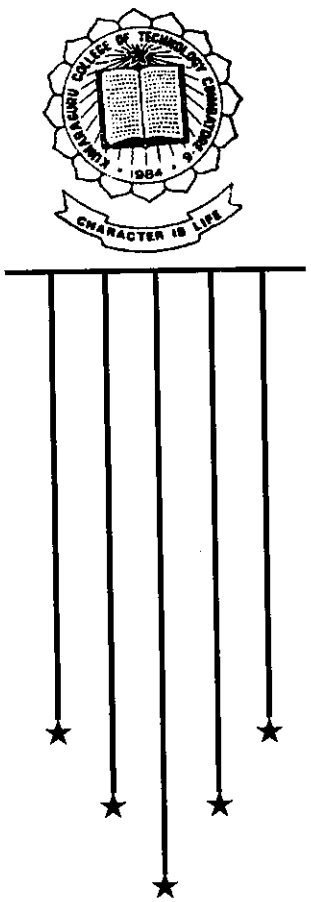
INDELIBLE

DOWNLOADING SOFTWARE

PROJECT REPORT



1999 -2000

SUBMITTED BY

AMARNATH. N

NIRMAL DEV. B

SATISH KUMAR. P

SUDHAKHAR. G

GUIDED BY

Mr. S. ANDREWS, M. Sc., PGDPM.

Department of Computer Science and Engineering

IN PARTIAL FULFILLMENT OF THE REQUIREMENTS

FOR THE AWARD OF THE DEGREE OF

BACHELOR OF SCIENCE IN

APPLIED SCIENCE - COMPUTER TECHNOLOGY

OF THE BHARATHIAR UNIVERSITY, COIMBATORE.

Department of Computer Science and Engineering

Kumaraguru College of Technology

Coimbatore - 641 006.

KUMARAGURU COLLEGE OF TECHNOLOGY

COIMBATORE – 641 006

Department of Computer Science and Engineering

Certificate

This is certify that the report entitled **"Indelible Downloading Software"** has been submitted by **Amarnath.N, Nirmal Dev.B, Satish Kumar.P, Sudhakhar.G** in partial fulfillment for the award of the degree of ***Bachelor of Science, in the Applied Science – Computer Technology*** branch, of Bharathiar University, Coimbatore – 641 046 during the academic year 1999-2000.

.....
Project Guide

.....
Head of the Department 27/2/2000

Certified that the Candidate with University Registration No.9727Q0003

Registration No 9727Q0024

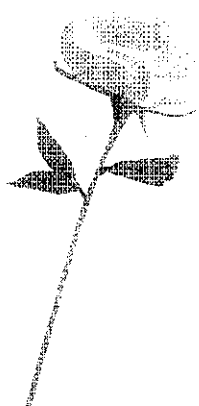
Registration No. 9727Q0034

Registration No. 9727Q0043

was examined in the project viva-voce examination held on .....

.....
Internal Examiner

.....
External Examiner



DEDICATION

We Opulently dedicate this project to
"The Lotus Feet of THE MOTHER"

To The Undisputed Legacy of
The BRAVE HEARTS

To Our Family for Their Incessant
Prayers for Our Pursuit of
Excellence

SRI AUROBINDO INTERNATIONAL INSTITUTE OF EDUCATIONAL RESEARCH

SAIIER is the research and development wing of Auroville. Auroville is actually a common wealth project well begun and has accomplished the status of an International colony in south India. Without any distinction even in the terms of age, all the Aurovillians have been working and are proceeding towards building a holy community of peace and goodwill.

Acknowleding the invitation of **The Mother** and **Shri Aurobindo**, the World Community has responded in one way or the other and we are proud that we have associated ourselves with **SAIIER**, towards a sustaining technical prospect.

PROF. R. MEENAKSHI, B.Sc., M.A.,,
ILAIGNARKAL EDUCATION CENTRE
(Sri Aurobindo International Institute of Educational Research)
Matrimandir Nursery Gardens,
Auroville - 605101.
☎ 0413-622141/622037
E-mail: tamil@auroville.org.in and *Meenakshi@auroville.org.in*

Auroville
10.03.2000

To Whomsoever It May Concern

The following students of **Kumaraguru College of Technology, Coimbatore,** studying final year B.Sc. (C.T) have completed their project for our organisation for the year 1999 – 2000.

Mr. N. Amarnath
Mr. G. Sudhakhar
Mr. P. Satishkumar
Mr. B. Nirmal Dev

Name of the project	: **"Indelible Downloading Software "**
Operating system	: Windows 3.x and above
Hardware configuration	: 24 MB RAM, Pentium class with min 1 MB HDD free space, Modem.
Technical utility of the project	: It enables to download web contents throw http and stores it permanently and allows automatic updating of the downloaded file every 20 seconds whenever online .
Duration of the project	: 4 months
Project implementation date	: 10.03.2000

They have put up many hours of hard work to achieve perfection in their project work.
We appreciate their efforts and wish them a bright future and success in all their endeavour.

R Meenakshi

(R.MEENAKSHI)

Acknowledgement

We express our sincere thanks to our Principal Dr.K.K Padmanabhan, Ph.D., for allowing us to undertake this project work.

We are grateful to Prof. Dr. S. Thangasamy Ph.D., Head of Department for his kindness in according permission to do our project.

We are forever indebted to our philosopher & guide, our class advisor *Mr. S.Andrews, M.Sc.,* for his continual and innovative guidance that led us to design this project and realize our true potential.

We will be failing in our duty, if we do not acknowledge our Course coordinator Mr. K.R. Baskaran B.E., MS. .We wish to place deep sense of gratitude for his valuable guidance.

We would always remain grateful to our friend & Teacher *Mr. Anand Ravi Deshpande MBA, M.Sc.,* for his down to earth suggestions and being in our own frequency to enable us to achieve the perfection in our efforts.

We sincerely convey our gratitude to Prof. B.S.Mishra M.Tech., (Amritha Institute of Technology & Science), for initiating and defining us the technical aspects during the proceedings & Completion of the project.

We offer our heart-bound love and perpetual respect to Prof. Mrs. R. Meenakshi, (SAIIER, Auroville) who imbibed in us the sense of confidence and sincerity.

Friends in need are friends indeed
The following souls remain an epitome of this great anecdote.
Ms. Srividhya.C.R
Mr.Bhuvanesh Kumar. R
Mr. Dinesh Sarvanan. K
Mr. Sundhararaman.L
And last but not the least
THE NERD MATRIX.

PRELUDE

The main objective of the project "The Indelible Downloading Software" is to download the necessary information from the net, especially the HTML and TXT files and store them permanently for future use.

This project involves simultaneous downloading of files, storing them permanently in a directory created for that purpose and updating the downloaded file once for every 20 seconds, checking the status of the URL entered the previous day, i.e., whether all the URLs are downloaded or not, if not downloading them immediately and our major focus of achievement in this project is to store the downloaded files permanently which is not possible in the existing system.

The scheme that is used to download is the hypertext Transfer Protocol – HTTP. The basic requirement is an HTTP server and the file that we require to be downloaded.

The method we use is the input of the HTTP URL. This URL is used for retrieving HTML documents on any network, for instance the World Wide Web.

For example, if a file index.htm has to be retrieved from the directory /home, on the HTTP server "www.msn.com" the corresponding URL would be:

http://www.msn.com/home/index.htm

The summery of the outputs obtained is the downloading of the specified file with the time and date of the downloaded period and with every 20 seconds elapsing we have a updated copy of the downloaded file.

Finally, suppose we have about 10 files to download, we would add all their addresses into the download list and them continue downloading them. We do not require being on a specific server address to have the updating done of a pertinent downloaded file.

Table of Contents

INTRODUCTION

Software and file downloaders have been important applications in Internet and the World Wide Web services. There are several software downloaders that download homepages and freeware from the net. But the existing system of downloading files from URLs does not permit permanent Storage of the downloaded files.

This project overcomes the later said barrier and avails permanent Storage of the files. The scope of this project starts from standalone Computers with local host address to web servers. We can download files from the local host and also from web sites knowing their URL addresses.

To understand more about this project we have to know about the URL, the HTTP and their original links and their method of working. How does HTTP work? This hypertext transfer protocol is a server program that listens to request through a port. This is a request response protocol that connects to some client program, typically a browser that in turn connects to a server through a server IP address. When we consider a file downloading, We consider the following paradigms

1. Platform Independent.
2. Simultaneous downloads of many files from different hosts.
3. Downloading to existing directories or directories created for specific purpose.
4. Changes updated to file already downloaded files.

5. Status of the downloaded files.

With the above paradigms into consideration we initiate the downloading by first specifying the URL address of the file that is to be downloaded. There are three modes by which the address of the files to be downloaded can be specified. We use the Universal Resource Locator URL, the www address or the Universal Document Identifier. In this method we use the URL which gets the client the access of the location of the resource.

When this process is undergone we encounter some principal results. The files from the servers through the scheme are downloaded with the most recent time and date of the download process. In the existing system, once the file is downloaded we lose contact of the URL. After some time the master copy of the downloaded file may undergo some changes or additions.

If we require accessing them we would need to download the whole page once again. To avoid this we implement the automatic updating through an appropriate interface.

To have this automatic updating we would consider being online with the respective server. This need not be the case. In this system of the automatic updating we just and only require to be online and need not necessarily be on that server.

Being online we would get into some other job while this software enables undercover access over the master copy of all previously downloaded files and updates all the files once in 20 seconds.

This updating is however valid only when the client is online. Every download and updating is permanently stored in some specified location in the user's hard disk. Technically emphasizing this updating process in the stipulated time is called **Scheduling**.

When we tend to download a lot of files we would come across in downloading the same file once again. To inform the client about the status of the files we give the status of the URL address as either Not attempted if it is a first timer, or Downloaded for downloaded files and also a status that says about its non availability of any information of the input URL address.

1) The main method-the first method to be executed-performs the setup by creating objects of the URL Connection for connection, the DataInputStream and the Data Output Stream class for file I/O and any other that are required.

2) The buttons can be Menu item load group/download can be triggered to perform the respective action.

3) The Main Module in the HTTP Downloading Software development implements a Frame that consists of the following components:

Buttons

Menu Items

Text Fields

List Box

The applet which depicts the actual processing window is named the HTTP_Down loader, which has the following, features:

1) Text Fields-Downloaded /Not Attempted/Found.

2) Interfaces-Mouse Listener/Window Listener/Action Listener.

3) Buttons-ADD URL, OPTIONS, DELETE, CLEAR.

4) List Box, Text Field, Menu and Text Area classes are defined in java.awt package.

5) All the interfaces are defined in a package called java.awtEvent.

The above description emphasizes the very fundamental yet functional details of the project. The following sections explain the distinct features of all the components used in this project functioning.

THE HTTP ODYSSEY

The web is based upon the Hypertext Transfer Protocol. This is Precisely the Http that we type in our browser such as

http://www.msn.com/home

Http is what is known as a "stateless" protocol. That is Http cannot Remember the state that it was last in. What really happens is that when a Web browser connects to a site, it sends a request for a particular file on the server. The Web server returns the file and then totally forgets about it when the next file is needed because the viewer clicked a link or submitted to a form, the entire form is repeated.

2.1 Web servers and HTTP

A World Wide Web server is a program that answers requests for documents from clients over the Internet. Web servers perform a number of tasks such as...

Logging activities

Authenticating users

Recording the Internet address, the time and the details of requests made from each connection

Protecting files from non- authenticated users

Accepting and forwarding requests for data.

Web servers transfer hypertext documents using the hypertext transfer protocol (Http). This protocol is stateless, i.e., there is no continuous connection between the server and the client. The server simply receives a request from the browser, at which time a connection is established, and sends back the requested data, after which the connection is severed. If the Browser wants more data, a new connection is established. This is a **one transaction and out** delivery mechanism. Hence every time a browser requests data a new session is established between the browser and the Server.

Using HTTP, it is possible to exchange various kinds of data- HTML Documents graphics and so on. The files being sent by the server are Identified by the browser from the MIME (Multimedia Internet Mail Extension) header.

When you ask the browser to open a URL, the request is converted by the browser to an *HTTP GET* request. It means – get a document and send it to me using HTTP.

Let us have an example, considering the URL such as,

http://www.msn.com/adminst.htm is specified, the browser converts to the GET command.Htm command. This connects to the server running on the host www.msn.com.

Issues the command and waits for a response. The response could be the requested document or an error message. The reason for the error messages could be that the requested document is not accessible or the Server software is not running.

2.2 HTTP Transactions

All HTTP transactions, take place with the help of a TCP/IP (Transmission Control Protocol/Internet Protocol) connection. HTTP Transaction consists of 4 phases.

2.2.1 **Connection** – In this phase, the browser attempts to establish a connection with ht Web server. Some browsers give a message on the status line, indicating the connection is being established. After a connection has been established, it waits for a reply from the server. On getting a reply from the server, a two way communication between the browser and the server is established and the next phase begins.

2.2.2 **Request** – Once the connection is established, the browser sends a request to the server. This request specifies the protocol to be used, for example, HTTP , the document to be retrieved, and How the browser wants the server to respond to the request. The method by which the server responds is usually GET.

2.2.3 Response – On fulfilling the request, the server sends a response to the browser. The server first sends the MIME type of data type that is being sent, and then spends the data. The browser checks the MIME type against the configuration file to check which function can handle the in-coming document. If it does not find the required function, it passes the data to the auxiliary program configured to handle it. If there is no such data available, the browser gives an option of saving the document so that it can be viewed later using a suitable tool.

2.2.4 **Close** – After getting the response from the server, the connection between the browser and the server is terminated and the browser handles the data.

The above format is how the Http along with URL and the Appropriate web server is used up in the implementation of the project.

URL & Its Usage

One of the important aspects of the web is the creation of a scalable way to locate all the resources of the net and you can reliably name anything and everything, it becomes a very powerful paradigm. This is what the URL does.

URL specification involves four main components.

1) The first component is the scheme that is FTP, GOPHER or Http. any URL address starts with the scheme.

   For e.g. http://

2) The second component is the host name or IP address of the Host. This follows the // and ends with / or :

3) The third component, the port number is an optional parameter which starts from / or : and ends with a /.

4) And the final component is the access path.

Therefore the URL address is the HTTP is http://www.msn.com/index.html.

3.1 URL Connection:

The URL is a general purpose for accessing the attributes of a remote source. This connection inspects the properties of the remote object before transporting. The connection is basically estimated through the HTTP scheme. The URL file retrieval method varies, depending on the type of the document to be retrieved. A URL need not always point to a document but can also point to a document but can also point to a query, image or the result of a command.

General Format:

Scheme: //host: domain/path/dataname.

3.2 Working with URLs

What Is a URL?

If you've been surfing the Web, you have undoubtedly heard the term URL and have used URLs to access HTML pages from the Web.

It's often easiest, although not entirely accurate, to think of a URL as the name of a file on the World Wide Web because most URLs refer to a file on some machine on the network. However, remember that URLs also a point to other resources on the network, such as database queries and command output.

Definition: URL is an acronym for Uniform Resource Locator and is a reference (an address) to a resource on the Internet.

URL has two main components:

Protocol identifier

Resource name

Note that the protocol identifier and the resource name are separated by a colon and two forward slashes. The protocol identifier indicates the name of the protocol to be used to fetch the resource. The example uses the Hypertext Transfer Protocol (HTTP), which is typically used to serve up hypertext documents. HTTP is just one of many different protocols used to access different types of resources on the net. Other protocols include File Transfer Protocol (FTP), Gopher, File, and News.

The resource name is the complete address to the resource. The format of the resource name depends entirely on the protocol used, but for many protocols, including HTTP, the resource name contains one or more of the components listed in the following table:

host name

the name of the machine the resource lives on

filename

the pathname to the file on the machine

port number

the port number to connect to (this is typically optional)

reference

a reference to a named anchor within a resource; usually identifies a specific location within a file (this is typically optional)

For many protocols, the host name and the filename are required, while the port number and reference are optional. For example, the resource name for an HTTP URL must specify a server on the network (Host Name) and the path to the document on that machine (Filename); it also can specify a port number and a reference. In the URL for the Java Web site java.sun.com is the host name and the trailing slash is shorthand for the file named /index.html.

The Java Phenomenon

4.1 What Is Java?

Java is two things: a programming language and a platform.

The Java Programming Language:

Java is a high-level programming language that is all of the following:

Java is also unusual in that each Java program is both compiled and interpreted. With a compiler, you translate a Java program into an intermediate language called Java bytecodes--the platform-independent codes interpreted by the Java interpreter. With an interpreter, each Java bytecode instruction is parsed and run on the computer. Compilation happens just once; interpretation occurs each time the program is executed.

Java bytecodes serves as the machine code instructions for the Java Virtual Machine (Java VM). Every Java interpreter, whether it's a Java development tool or a Web browser that can run Java applets, is an implementation of the Java VM. The Java VM can also be implemented in hardware.

Java bytecodes help make "write once, run anywhere" possible. You can compile your Java program into bytecodes on any platform that has a Java compiler. The bytecodes can then be run on any implementation of the Java VM. For example, the same Java program can run on Windows NT, solaris, and Macintosh.

4.2 The Java Platform

A platform is the hardware or software environment in which a program runs. The Java platform differs from most other platforms in that it's a software-only platform that runs on top of other,hardware-based platforms. Most other platforms are described as a combination of hardware and operating system.

The Java platform has two components:

The Java Virtual Machine (Java VM)

The Java Application Programming Interface (Java API)

The Java API is a large collection of ready-made software components that provide many useful capabilities, such as graphical user interface (GUI) widgets. The Java API is grouped into libraries (packages) of related components.

As a platform-independent environment, Java can be a bit slower than native code. However, smart compilers, well-tuned interpreters, and just-in-time bytecode compilers can bring Java's

performance close to that of native code without threatening portability.

4.3 Design Goals of Java

Java is a high level programming language that has all of the following:

- Simple
- Object-Oriented
- Portable
- Distributed
- High Performance
- Interpreted
- Multithreaded
- Robust
- Dynamic
- Secure

4.3.1 Java – Architecture Neutral

Due to the bytecode compilation process, and interpretation by the browser, Java can work with a variety of hardware and operating systems. The only requirement is that the system should have Java enabled Internet Browser.

Java programs are compiled to an architectural neutral bytecode format. The primary advantage of this approach is that it allows the Java application to run on any system, as long as that system implements Java Virtual Machine.

4.4 Java Packages

Java Packages are collections of classes and interfaces that are related to each other in some useful way. Such classes need to access

each other's instance variables and methods directly.

Primary benefit of packages is the ability to organize many class definitions to a single unit. For example all the Java input output system code is collected in a single package called java.io. The secondary benefit from the programmers' viewpoint is that friendly instance variables and methods are available to all classes within the same package, but not to class outside the package. Frequently used packages are

- Java.lang
- Java.io
- Java.awt
- Java.awtEvent
- Java.util
- Java.net

4.4.1 Description of Packages:

Java.Lang

The Java language package provides the core classes that make up to the Java Programming environment, strings and objects as well as classes for handling compilation, the run time environment, security and threaded programming . The language package is included automatically.

Java.io

The Java IO package gives classes support for reading and writing data to and from different input and output devices, including files, strings and other data sources. The I/O package includes different classes

for inputting streams of data, outputting streams of data, working with files, and tokenizing streams of data.

Java.util

This package provides some of the most useful Java classes that you will come to rely on. It introduces ten classes, two exception classes and two interfaces.

Java.applet

The java.applet package contains that applet classes as well as three interfaces: applet context, applet stub and audioclip.

Java.awt

The Java.awt package contains classes that makeup the prebuild GUI (Graphical User Interface) components that are available to Java developers.Components can be colors, fonts, buttons and scrollbars. Java.awt package also contains two subpackages:

Java.awt.image & Java.awt.peer.

Enhancements

The architecture has been improved to make large-scale GUI development more feasible and add to basic functionality that was missing.

Architectural support has been added for event handling by non-components ("delegation"), data transfer (such as cut-copy-paste),

desktop color schemes (to improve consistency of appearance), printing, mouseless operation, component-specific cursors, and lightweight components.

Method names, arguments, and functionality have been made consistent. These changes make it possible for programs such as GUI builders to query components to determine the components' properties.

Java.awt.event

Java.awt.event uses a dedication event model, which is much more robust and flexible and allows for a greater of compiled time checking.

Java.net

This embeds the HTTP rules and status codes in them. The classes in this package is used for making network connections. Using the classes such as URL and Urlconnection we can retrieve, send, make and break connections.

Implementing Applets , Threads and Frames

5.1 The Anatomy of a Java Applet

Java applet is a program that adheres to a set of conventions that allows it to run within a Java-compatible browser.

Here again is the code for the "Hello World" applet.

```
import java.applet.Applet;
import java.awt.Graphics;
public class HelloWorld extends Applet {
   public void paint(Graphics g) {
      g.drawString("Hello world!", 50, 25); } }
```

5.1.1 *Importing Classes and Packages*:

The code above starts off with two import statements. By importing classes or packages, a class can more easily refer to classes in other packages. In the Java language, packages are used to group classes, similar to the way libraries are used to group C functions. Importing Classes and Packages gives you more information about packages and the import statement.

5.1.2 *Defining an Applet Subclass:*

Every applet must define a subclass of the Applet class. In the "Hello World" applet, this subclass is called HelloWorld. Applets inherit a great deal of functionality from the Applet class, ranging from communication with the browser to the ability to present a graphical user interface (GUI).

Implementing Applet Methods:

The HelloWorld applet implements just one method, the paint method. Every applet must implement at least one of the following methods: init,start, or paint. Unlike Java applications,applets do not need to implement a main method. Implementing Applet Methods talks about the paint method,how the "Hello World" applet implements it, and the other methods applet commonly implement.

5.1.3 *Running an Applet:*

Applets are meant to be included in HTML pages. Using the <APPLET> tag, you specify (at a minimum) the location of the Applet subclass and the dimensions of the applet's onscreen display area. When a Java-capable browser encounters an <APPLET> tag, it reserves onscreen space for the applet, loads the Applet subclass onto the computer the browser is executing on, and creates an instance of the Applet subclass. Running an Applet gives more details.

5.2 *The <APPLET> Tag*

When we build <APPLET> tags, keep in mind that words such as APPLET and CODEBASE can be typed in either as shown or in any mixture of uppercase and lowercase. Bold font indicates something you should type in exactly as shown (except that letters don't need to be uppercase). Italic font indicates that you must substitute a value for the

word in italics. Square brackets ([and]) indicate that the contents of the brackets are optional.

```
< APPLET
   [CODEBASE = codebaseURL]
   CODE = appletFile
   [ALT = alternateText]
   [NAME = appletInstanceName]
   WIDTH = pixels
   HEIGHT = pixels
   [ALIGN = alignment]
   [VSPACE = pixels]
   [HSPACE = pixels]
>
[< PARAM NAME = appletParameter1 VALUE = value >]
[< PARAM NAME = appletParameter2 VALUE = value >]
. . .
[alternateHTML]
</APPLET>
```

CODEBASE = codebaseURL

This optional attribute specifies the base URL of the applet -- the directory or folder that contains the applet's code. If this attribute is not specified, then the document's URL is used.

CODE = appletFile

This required attribute gives the name of the file that contains the applet's compiled Applet subclass. This file is relative to the base URL of the applet. It cannot be absolute.

ALT = alternateText

This optional attribute specifies any text that should be displayed if the browser understands the APPLET tag but can't run Java applets.

NAME = appletInstanceName

This optional attribute specifies a name for the applet instance, which makes it possible for applets on the same page to find (and communicate with) each other.

WIDTH = pixels
HEIGHT = pixels

These required attributes give the initial width and height (in pixels) of the applet display area, not counting any windows or dialogs that the applet brings up.

ALIGN = alignment

This required attribute specifies the alignment of the applet. The possible values of this attribute are the same (and have the same effects) as those for the IMG tag: left, right, top, texttop, middle, absmiddle, baseline, bottom, absbottom.

VSPACE = pixels
HSPACE = pixels

These optional attributes specify the number of pixels above and below the applet (VSPACE) and on each side of the applet (HSPACE). They're treated the same way as the IMG tag's VSPACE and HSPACE attributes.

< PARAM NAME = appletParameter1 VALUE = value >
<PARAM> tags are the only way to specify applet-specific parameters. Applets read user-specified values for parameters with the getParameter() method. See Defining and Using Applet Parameters for

information about the getParameter() method

5.2.1 *Alternate HTML*

If the HTML page containing this <APPLET> tag is viewed by a browser that doesn't understand the <APPLET> tag, then the browser will ignore the <APPLET> and <PARAM> tags, instead interpreting any other HTML code between the <APPLET> and </APPLET> tags. Java-compatible browsers ignore this extra HTML code.

5.3 Practical Considerations when Writing Applets

5.3.1 *Threads in Applets:*

Every applet can run in multiple threads. Applet drawing methods (paint and update) are always called from the AWT drawing and event handling thread. The threads that the major milestone methods -- init. start, stop, and destroy -- are called from depends on the application that's running the applet. But no application ever calls them from the AWT drawing and event handling thread.

Many browsers allocate a thread for each applet on a page, using that thread for all calls to the applet's major milestone methods. Some browsers allocate a thread group for each applet, so that it's easy to kill all the threads that belong to a particular applet. In any case, you're guaranteed that every thread that any of an applet's major milestone methods creates belongs to the same thread group.

PrintThread is a modified version of SimpleApplet that prints the thread and thread group that its init, start, stop, destroy, and update methods are called from.

5.3.2 *Why would an applet need to create and use its own threads?*

let us imagine an applet that performs some time-consuming initialization -- loading images, for example -- in its init method. The thread that invokes init cannot do anything else until init returns. In some

browsers, this might mean that the browser can't display the applet or anything after it until the applet has finished initializing itself. So if the applet is at the top of the page, for example, then nothing would appear on the page until the applet has finished initializing itself.

Even in browsers that create a separate thread for each applet, it makes sense to put any time-consuming tasks into an applet-created thread, so that the applet can perform other tasks while it waits for the time-consuming ones to be completed.

5.3.3 ***Rule of Thumb:***

If an applet performs a time-consuming task, it should create and use its own thread to perform that task. Applets typically perform two kinds of time-consuming tasks: tasks that they perform once, and tasks that they perform repeatedly.

5.4*Synchronizing Threads*

Usually thread contains all of the data and methods required for its execution and do require any outside resources or methods. In addition,the threads run at their own pace without concern over the state or activities of any other concurrently running threads.

However, there are many interesting situations where separate, concurrently running threads do share data and must consider the state and activities of other threads. One such set of programming situations

are known as producer/consumer scenarios where the producer generates a stream of data which then is consumed by a consumer.

For example, imagine a Java application where one thread (the producer) writes data to a file while a second thread (the consumer) reads data from the same file. Or, as you type characters on the keyboard, the producer thread places key events in an event queue and the consumer thread reads the events from the same queue. Both of these examples use concurrent threads that share a common resource: the first shares a file, the second shares an event queue. Because the threads share a common resource, they must be synchronized in some way.

5.5 How to Use Frames

The Frame class provides windows for applets and applications. Every application needs at least one Frame.

If an application has a window that should be dependent on another window -- disappearing when the other window is iconified, for example then you should use a Dialog instead of a Frame for the dependent window.

Unfortunately, applets currently can't use dialogs well, so they generally need to use frames instead.

The pack() method, which is called in the main() method above, is defined by the Window class.

Besides the no-argument Frame constructor implicitly used by the MenuWindow constructor shown above, the Frame class also provides a one-argument constructor. The argument is a String that specifies the title of the frame's window.

Other interesting methods provided by Frame are:

String getTitle() and void setTitle(String)-
Returns or sets (respectively) the title of the frame's window.

Image getIconImage() and void setIconImage(Image)-

Returns or sets (respectively) the image displayed when the window is iconified.

MenuBar getMenuBar() and void setMenuBar(MenuBar)-
Returns or sets (respectively) the menu bar for this Frame.

void remove(MenuComponent)-
Removes the specified menu bar from this Frame.

boolean isResizable() and void setResizable(boolean)-
Returns or sets whether the user can change the window's size.

int getCursorType() and void setCursor(int)-

Gets the current cursor image or sets the cursor image. The cursor must be specified as one of the types defined in the Frame class. The pre-defined types are

Frame.DEFAULT_CURSOR, Frame.CROSSHAIR_CURSOR,
Frame.HAND_CURSOR, Frame.MOVE_CURSOR,
Frame.TEXT_CURSOR, Frame.WAIT_CURSOR, or
Frame.X_RESIZE_CURSOR, where X is SW,
SE, NW, NE, N, S, W, or E.

Event model

An event model has two parts:

- Sources
- Listeners

6.1 Sources of Events

The sources used are

Buttons	Generates action events when the button is pressed.
Checkbox	Generates Item events when the checkbox is selected or deselected.
List	Generates action events when the item is double clicked. Generates Item events when the checkbox is selected or deselected.
Menu item	Generates action events when a menu item is selected; generates item events when a checkable menu item is selected or deselected.
Scrollbar	Generates adjustment events when the scrollbar is manipulated.
Text components	Generates text events when the user enters a character.
Window	Generates window events when a window is activated, closed, deactivated, iconified, deiconified, opened or quit.

6.2 *Event Listener Interfaces*

The event listener interfaces are

Action listener

Mouse listener

Window listener, that are dealt in detail as follows:

6.2.1 *Writing an Action Listener*

Action listeners are probably the easiest -- and most common -- event handlers to implement. You implement an action listener to respond to the user's indication that some implementation-dependent action should occur.

When the user clicks a button, double clicks a list item, chooses a menu item, or presses return in a text field, an action event occurs. The result is that an action Performed message is sent to all action listeners that are registered on the relevant component.

6.2.2 *Action Event Methods*

The Action Listener interface contains a single method, and thus has no corresponding adapter class. Here is the lone Action Listener method: void action Performed(ActionEvent), called by the AWT just after the user informs the listened-to component that an action should occur.

Examples of Handling Action Events

Here is the action event handling code from an applet named Beeper:

```
public class Beeper ... implements ActionListener {
  ...
  //where initialization occurs:
    button.addActionListener(this);
  ...
  public void actionPerformed(ActionEvent e) {
    ...//Make a beep sound...
  }
}
```

6.2.3 *The ActionEvent Class*

The actionPerformed method has a single parameter: an ActionEvent object. The ActionEvent class defines two useful methods:

String getActionCommand()

Returns the string associated with this action. Most objects that can generate actions support a method called setActionCommand that lets you set this string. If you don't set the action command explicitly, then it's generally the text displayed in the component. For objects with multiple items, and thus multiple possible actions, the action command is generally the name of the selected item.

int getModifiers( )

Returns an integer representing the modifier keys the user was pressing when the action event occurred. You can use the ActionEvent-defined constants SHIFT_MASK, CTRL_MASK, META_MASK, and

ALT_MASK to determine which keys were pressed. For example, if the user Shift-selects a menu item, then the following expression is nonzero:

actionEvent.getModifiers() & ActionEvent.SHIFT_MASK

Also useful is the getSource method, which returns the object (a component or menu component) that generated the action event. The getSource method is defined in one of ActionEvent's superclasses, EventObject.

6.3 *WRITING A MOUSE LISTENER*

Mouse events tell you when the user uses the mouse (or similar input device) to interact with a component. Mouse events occur when the cursor enters or exits a component's on-screen area and when the user presses or releases the mouse button. Because tracking the cursor's motion involves significantly more system overhead than tracking other mouse events, mouse motion events are separated into a separate listener type.

6.3.1 *Mouse Event Methods:*

The MouseListener interface and its corresponding adapter class, MouseAdapter, contain three methods:

void mouseClicked(MouseEvent)

Called by the AWT just after the user clicks the listened-to component.

void mouseEntered(MouseEvent)

Called by the AWT just after the cursor enters the bounds of the listened-to component.

void mouseExited(MouseEvent)

Called by the AWT just after the cursor leaves the bounds of the listened-to component.

void mousePressed(MouseEvent)

Called by the AWT just after the user presses a mouse button while the cursor is over the listened-to component.

void mouseReleased(MouseEvent)

Called by the AWT just after the user releases a mouse button after a mouse press over the listened-to component.

One complication affects mouse-entered, mouse-exited, and mouse-released events. When the user drags (presses and holds the mouse button and then moves the mouse), then the component that the cursor was over when the drag started is the one that receives all subsequent mouse and mouse-motion events up to and including the mouse button release. That means that no other component will receive a single mouse event -- not even a mouse-released event -- while the drag is occurring.

6.3.2*Examples of Handling Mouse Events:*

The following applet demonstrates mouse events. At the top of the applet is a blank area (implemented, strangely enough, by a class named BlankArea). A mouse listener listens for events both on the BlankArea and on its container, which is an instance of MouseEventDemo. Each time a mouse event occurs, a descriptive message is displayed under the blank area. By moving the cursor on top of the blank area and clicking mouse buttonsoccasionally, you can see when mouse events are generated.

6.3.3 *The MouseEvent Class*

Each mouse event method has a single parameter: a MouseEvent object. The MouseEvent class defines the following useful methods:

int getClickCount()

Returns the number of quick, consecutive clicks the user has made (including this event).

int getX()
int getY()
Point getPoint()

Return the (x,y) position at which the event occurred, relative to the component over which the event occurred.

boolean isPopupTrigger()

Returns true if the mouse event should cause a popup menu to appear. Because popup triggers are platform dependent, if your program uses popup menus, you should call isPopupTrigger for both mouse down and mouse up events.

The MouseEvent class extends InputEvent, which extends ComponentEvent.

ComponentEvent provides the handy getComponent method. InputEvent provides the following useful methods:

int getWhen()

Returns the timestamp of when this event occurred. The higher the timestamp, the more recently the event occurred.

boolean isAltDown()
boolean isControlDown()
boolean isMetaDown()
boolean isShiftDown()

Convenient methods giving you the state of the modifier keys when the event was generated.

int getModifiers()

Returns a flag indicating the state of all the modifiers when the event was generated. Besides the modifier keys, this flag indicates which mouse button was pressed.

For example, the following expression is true if the right button was pressed:

```
(mouseEvent.getModifiers() &
InputEvent.BUTTON3_MASK)
== InputEvent.BUTTON3_MASK
```

The SwingUtilities class contains convenience methods for determining whether a particular mouse button has been pressed:

```
static boolean isLeftMouseButton(MouseEvent)
static boolean isMiddleMouseButton(MouseEvent)
static boolean isRightMouseButton(MouseEvent)
```

6.4 *Window listener:*

Window events are generated by a Window just after the window is opened, closed, iconified, deiconified, activated,or deactivated. Opening a window means showing it for the first time; closing it means removing the window from the screen. Iconifying it means substituting a small icon on the desktop for the window; deiconifying means the opposite. A window is activated if it or a component it contains has the keyboard focus; deactivation occurs when the window and all of its contents lose the keyboard focus.

The most common use of window listeners is closing windows. If a program doesn't handle window-closing events,then nothing happens when the user attempts to close a window. An application that features a single window might react to window-closing events from that window by exiting.

Other programs usually react to window-closing events by disposing of the window or making it invisible.

Another common use of window listeners is to stop threads and release resources when a window is iconified, and to start up again when the window is deiconified. This way, you can avoid unnecessarily using the processor or other resources. For example, when a window that contains animation is iconified, it should stop its animation thread and free any large buffers. When the window is deiconified, it can start the thread again and recreate the buffers.

6.4.1 *Window Event Methods:*

The WindowListener interface and its corresponding adapter class, WindowAdapter, contain these methods:

void windowOpened(WindowEvent)

Called by the AWT just after the listened-to window has been shown for the first time.

void windowClosing(WindowEvent)

Called by the AWT in response to a user request that the listened-to window be closed. To actually close the window, the listener should invoke the window's dispose or setVisible(false) method.

void windowClosed(WindowEvent)

Called by the AWT just after the listened-to window has closed.

void windowIconified(WindowEvent)
void windowDeiconified(WindowEvent)

Called by the AWT just after the listened-to window is iconified or deiconified, respectively.

void windowActivated(WindowEvent)
void windowDeactivated(WindowEvent)

Called by the AWT just after the listened-to window is activated or deactivated, respectively.

6.4.2*Examples of Handling Window Events:*

The following applet demonstrates window events. By clicking the top button in the applet, you can bring up a small window. The controlling classlistens for window events from the window, printing a message whenever it detects a window event. You can find the applet's code in:

WindowEventDemo.java.

6.4.3 *The WindowEvent Class:*

Each window event method has a single parameter: a WindowEvent object. The WindowEvent class defines one useful method, getWindow, which returns the Window that generated the window event.

WEB SERVER- The Platform

In Java Web Server 1.1, we are using a new JavaServer Toolkit feature which simplifies and enhances (both!) the process of starting up the WebServer. The jserv program is a master program that handles whether or not to use an included JRE, classpath settings that should be kept (with or without JRE!) and other such items. To get a full rundown of the options that can be handed off to the jserv program, execute it with -help argument:

C:\JavaWebServer1.1\bin\jserv.exe -help

Usage: jserv [-jre|-nojre] [-javahome <javahome>] [-classpath <classpath>]
[-ssl] [-threads <green|native>] [-Dsystem_property=value]...
[-passfile] [-verbose] [-help]

7.1 ***Arguments:***

-jre Use the Java Runtime Environment (JRE) instead of JDK

-javahome Location of the Java executable, <javahome>/bin/java

-classpath Additional classpath settings

-threads Use portable Green threads or native threads

-ssl Enable use of Secure Sockets Layer (SSL)

-passfile Read SSL passphrase from file

<server_root>/keys.passphrase

-verbose Display internal settings such as CLASSPATH

-help Print this message

By default, and to acknowledge historical precedent, we ship two programs that sit in front of the jserv program -- httpd and hpttdnojre. The httpd program runs with the included JRE (in distributions that include a JRE), the httpdnojre program runs assuming that you already have a properly configured Java Runtime Environment (such as the JDK) installed and available.

So, if you want to run with the JRE, but specify on the command line the classpath to use, simply run:

C:\JavaWebServer1.1\bin\httpd -classpath C:\JDK1.1.4\classes

7.2 *The Java Runtime Environment*

CONTENTS

7.2.1 *OVERVIEW OF JAVA RUNTIME*

7.2.1.1 *INTRODUCTION*

This is version 1.1.4 of the Java Runtime Environment, also known as the Java Runtime, or JRE. The Java Runtime is the minimum standard Java Platform for running Java programs. It contains the Java Virtual Machine, Java Core Classes and supporting files.

The JRE can be invoked from the command line by using the jre (see the following section). On Windows platforms, the jre tool will ignore the CLASSPATH environment variable. For both Windows platforms, the -cp option is recommended to specify an application's class path.

The Win32 version of JRE 1.1.4 has an installer suitable for use by end-users. This gives software developers the option of not bundling the JRE with their applications. Instead, they can direct end-users to download and install the JRE themselves.

7.2.1.2 *THE JAVA RUNTIME INTREPRETER*

The jre tool invokes the Java Runtime interpreter for executing Java applications. The tool is available in the Win32, Solaris-Sparc, and Solaris x86 downloads of the JRE. The syntax for the jre command is:

jre [options] classname <rags>

The classname argument is the name of the class file to be executed. Any arguments to be passed to the class must be placed after the classname on the command line.

On Windows platforms, the jre tool will ignore the CLASSPATH environment variable. For both Windows and Solaris platforms, the –cp option is recommended to specify an application's class path.

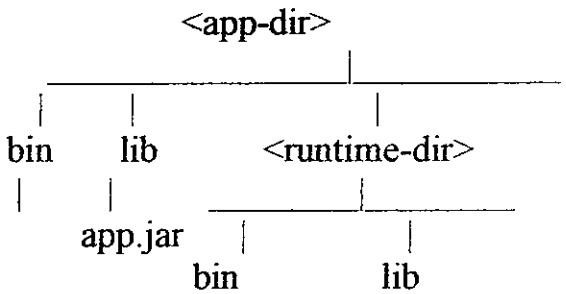
In general, the optional files provide localization support for languages.The JRE includes the bin and lib directories which both must reside in the same directory. We call this directory <runtime-dir>.

In the following lists, all paths are relative to the <runtime-dir> directory (which is originally "jre1.1.4 ").

7.2.2 *BUNDLING AND RUNNING THE JAVA RUNTIME*

BUNDLING THE JAVA RUNTIME

An example directory structure might look like the following:



7.2.2.1 *JAVA RUNTIME EXAMPLE*

The following web page has a Hello World example that you can download demonstrating how to create a simple Java application that runs on, and is bundled with, the Java Runtime Environment.

http://java.sun.com/products/jdk/1.1/jre/example/

This example shows how to make a simple, seamless transition from developing an application with the JDK, to deploying it with the more-lightweight JRE.

7.2.2.2 RUNTIME DOCUMENTATION

Runtime documentation is any documentation that an end-user Might need after they have installed a Java program that runs on the JRE. We supply the following runtime documentation:

- Each property file contains comments that describe what the file is useful for and how to modify it.
- awt.properties file - KeyEvent uses it to print out properties of key events, usually for debugging purposes. This might be used by a GUI debugger that needs to print out events.

THIS PROJECT IS ASSOCIATED WITH THE SIMULTANEOUS DOWNLOADING OF HTTP FILES AND AS WELL AS HAVING THE LATEST UPDATIONS OF THOSE DOWNLOADED FILES AS AND WELL WE ARE BROWSING THE INTERNET.

8.1 *SYSTEM STUDY*

1) The problem can be defined as downloading the significant information from a remote host to the user's hard disk.
2) The User-defined code indicates whether the files are completely downloaded, Not attempted or attempted.
3) Data on the net may change very frequently, so the changes are updated in the already existing downloaded files and this provision implemented in this project is termed as **"SCHEDULING".**

8.2 *SYSTEM JUSTIFICATION*

This system is efficient as it is accomplished using Java. Moreover the client software developed is an HTTP client since most of the information on the www is in this protocol.

8.3 *SYSTEM GOALS*

1) The user is entitled to do some work or the other while Simultaneously downloading files and hence saves time.
2) To implement scheduling
3) The user defined status codes in turn enhances the system's Operations.

8.4 *SYSTEM REQUIREMENTS*

Operating system: MS-DOS (optional)

Memory configuration: 32 MB

Software: JDK1.1.5

Requirement: Internet connection

CPU: Pentium 100mhz

8.5 *SYSTEM DEVELOPMENTS*

1) The main method-the first method to be executed-performs the setup by creating objects of the URL Connection for connection, the DataInputStream and the Data Output Stream class for file I/O and any other that are required.
2) The buttons can be Menu item load group/download can be triggered to perform the respective action.
3) The Main Module in the HTTP Client Software development implements a Frame that consists of the following components:

Buttons
Menu Items
Text Fields
List Box

8.6 *DESIGN PHASE*

HTTP_Down loader will have the following features:

1) Text Fields-Downloaded /Not Attempted/Found.

2) Interfaces-Mouse Listener/Window Listener/Action Listener.

3) Buttons-ADD URL, OPTIONS, DELETE, CLEAR.

4) List Box, Text Field, Menu and Text Area classes are defined in java.awt package.

5) All the interfaces are defined in a package called java.awtEvent.

TESTING

We tested the software Indelible Downloading Software, by downloading the files from the network servers and also from the local host and we were able to download simultaneously many files from them. The other subsequent goals of this software were also tested and found that the results were working to the anticipated outputs of the system design.

System Limitations

The two important limitations that are encountered are

- Host Migration
- Change in configuration of Network

The change of IP address when a host moves from the network to another has to be taken into account.

//PROJECT JAVA//

```
import java.io.*;
import java.awt.event.*;
import java.awt.*;
import java.util.*;

class project extends Frame implements
WindowListener,DownLoadObserver,MouseListener
{
      String dloadDir,gru;
      String gotme,tofile;
      String bloadFile;
      collectUrl url;
      OptionSaver osr;
      Hashtable urlH;
      URLProperty   urlprop;
      TextField edit1 = new TextField();
      List list1;
      TextField edit3 =new TextField();
      TextField edit4 =new TextField();
      TextField edit5 =new TextField();
      Hashtable urlh=new Hashtable();
      Hashtable rtry=new Hashtable();
      Hashtable cloned =new Hashtable();
      MenuItem menuitem1,menuitem2,menuitem3;

     project()
      {
      super("Http-DownLoader");
      setLayout(null);
      MenuBar menubar1=new MenuBar();

      Menu menu1=new Menu("File");


      menuitem2=new MenuItem("Save Group");
      menuitem2.addActionListener(new mlis());
      menu1.add(menuitem2);

      menuitem3=new MenuItem("Load Group");
```

```
menuitem3.addActionListener(new mlis());
menu1.add(menuitem3);

menubar1.add(menu1);
setMenuBar(menubar1);

list1=new List(7,false);
Label label1 = new Label();
Label label2 =new Label();
url=new collectUrl();
osr=new OptionSaver();
Button button1 = new Button("Add URL");
Button button2 =new Button("DownLoad");
Button button3 =new Button("Options");
Button button4 =new Button("Delete");
Button button5 =new Button("Clear");

edit1.setLocation(80,45);
edit1.setSize(344, 20);
edit1.setText("");


list1.setLocation(80,104);
list1.setSize(344, 200);
list1.addMouseListener(this);

edit3.setLocation(80,368);
    edit3.setSize(344,20);
     edit3.setText("");

     label1.setLocation(5,45);
     label1.setSize(144, 24);
         label1.setText("Enter URL");

     label2.setLocation(440, 104);
     label2.setSize(88, 16);
         label2.setText("Status :");

     edit4.setLocation(440, 128);
     edit4.setSize(168, 20);
```

```
edit4.setText("");
edit5.setLocation(80,400);
edit5.setSize(344, 20);
edit5.setText("");

button1.setLocation(200,65);
button1.setSize(80, 32);

button2.setLocation(200,320);
button2.setSize(80,32);

button3.setLocation(440,160);
button3.setSize(80,32);

button4.setLocation(440,200);
button4.setSize(80,32);

button5.setLocation(440,240);
button5.setSize(80,32);

add(edit1);
add(label1);
add(list1);
add(edit3);
add(button1);
add(button2);
add(label2);
add(edit4);
add(edit5);
add(button3);
add(button4);
add(button5);
button1.addActionListener(new blis1());
button2.addActionListener(new blis2());
button3.addActionListener(new blis3());
button4.addActionListener(new blis4());
button5.addActionListener(new blis5());
this.addWindowListener(this);
osr=OptionSaver.retriveFromFile();
```

```
        for(int i=url.cu.size()-20;i<url.cu.size();i++)
          {
          try{
               String co=(String)url.cu.elementAt(i);
               list1.addItem(co);
               }catch(Exception app)
               { }
          }
}

     private class mlis implements ActionListener{
     public void actionPerformed(ActionEvent e){


               if(e.getSource()==menuitem2){
                    FileDialog fd=new FileDialog(project.this,"Save group
as",FileDialog.SAVE);
                    fd.show();
                    gru=fd.getFile();
                    System.out.println("gru"+gru);
                    System.out.println(gru);
                    url.saveInfile(gru);
               }

               if(e.getSource()==menuitem3)
               {
                    FileDialog fd=new FileDialog(project.this,"Save group
as",FileDialog.LOAD);
                    fd.show();
                    gru=fd.getFile();
                    url=collectUrl.retriveFromfile(gru);

                   for(int i=0;i<url.cu.size();i++)
                   {
                   try{
                        String co=(String)url.cu.elementAt(i);
                        list1.addItem(co);
                        }catch(Exception app)
                        { }
                   }}}}
```

```
    public void mouseClicked(MouseEvent e)
    {

    System.out.println("url.hash"+url.hash);

        if(url.hash.containsKey(list1.getSelectedItem()))
        {

System.out.println("true"+url.hash.containsKey(list1.getSelectedItem()));
                URLProperty
str=(URLProperty)url.hash.get(list1.getSelectedItem());
                int g=str.getstatus();
                System.out.println("g"+g);
                if(g==1)
                {
                edit4.setText("Downloaded");
                }
                else if(g==2)
                {
                edit4.setText("File not found");
                }
                else if(g==0)
                {
                edit4.setText("Not Attempted");
                }
        }
        else
        {
         edit4.setText("Status Not Avail");
         }
    }

    public void mouseEntered(MouseEvent e)
    {}
    public void mouseExited(MouseEvent e)
    {}
    public void mousePressed(MouseEvent e)
    {}
```

```
    public void mouseReleased(MouseEvent e)
    {}
    public void windowActivated(WindowEvent e)
    {}
    public void windowClosed(WindowEvent e)
    {}
    public void windowClosing(WindowEvent e)
    {
        try{
                setVisible(false);
                try{
               url.saveInfile(gru);
                }catch(Exception ty)
                {
                System.out.println("filesave "+ty);
                }

                System.exit(0);
          }catch(Exception er1){System.out.println("window error");  }
    }
    public void windowDeactivated(WindowEvent e)
    {}
    public void windowDeiconified(WindowEvent e)
    {}
    public void windowIconified(WindowEvent e)
    {}
    public void windowOpened(WindowEvent e)
    {}

private class blis1 implements ActionListener
{
     public void actionPerformed(ActionEvent e){
     String e1=edit1.getText().trim();
     if(e1.equals(""))
     {
     edit1.requestFocus();
     }
     else
     {
     list1.addItem(e1);
```

```
        url.addurl(e1);

        url.addStatus(edit1.getText(), new
URLProperty(URLProperty.NOT_ATTEMPTED));

        System.out.println("u"+url.hash);
        url.saveInfile(gru);
        }
          }
        }

private class blis3 implements ActionListener{
        public void actionPerformed(ActionEvent e){
        Options option=new Options();
        option.setLocation(100,100);
        option.setSize(400,200);
        option.show();




        }
        }
private class blis4 implements ActionListener{
        public void actionPerformed(ActionEvent e){
        try{
        gotme =list1.getSelectedItem();
        url.removeFromVector(gotme);
        url.removeStatus(gotme);
        url.saveInfile(gru);
        list1.delItem(list1.getSelectedIndex());
        }catch(NullPointerException n1)
        {}
      }
     }

private class blis5 implements ActionListener{
       public void actionPerformed(ActionEvent e){
       list1.removeAll();
       }}
```

```
private class blis2 implements ActionListener{
collectUrl urat;
OptionSaver osr1;
String tofile;
FileDialog fd;
      public void actionPerformed(ActionEvent e){
      try{

            urat=collectUrl.retriveFromfile(gru);
            osr1=OptionSaver.retriveFromFile();
            boolean sd=osr1.getStateDir();
            boolean su=osr1.getStateUrl();
            String dl=osr1.getdLoad();
            System.out.println("sd "+sd+"  su "+su);

            if(sd && su)
            {

                        for(int i=0;i<urat.cu.size();i++)
                         {

                          System.out.println("urat.hash"+urat.hash);
                         String st=(String)urat.cu.elementAt(i);
                         System.out.println("sd"+sd);
                         int len=st.indexOf("//");
                         String
sub=(dl+System.getProperty("file.separator")+st.substring(len+2)).replace('/',(Syst
em.getProperty("file.separator")).charAt(0));


                         System.out.println("dl"+dl);
                         System.out.println("File.separator"+File.separator);
                         System.out.println("st.substring(len+2)"+st.substring(len+2));




                         tofile=st.substring(st.lastIndexOf("/")+1);
```

```
                    System.out.println("sub"+sub);
                    System.out.println("tofile"+tofile);

                    saveThread au=new
saveThread((DownLoadObserver)project.this,st,sub,tofile,true);

                    au.start();


                    }

          }
           if(sd )
           {
                    for(int i=0;i<urat.cu.size();i++)
                    {
                    String st=(String)urat.cu.elementAt(i);
                    int len=st.indexOf("//");
                    tofile=st.substring(st.lastIndexOf("/")+1);

                    String sub=(dl);
                    System.out.println(sub);
                    System.out.println(tofile);

                    saveThread au=new
saveThread((DownLoadObserver)project.this,st,sub,tofile,true);
                    au.start();
                    }


          }
          else if(su)
          {
                    String cd=System.getProperty("user.dir");
                    for(int i=0;i<urat.cu.size();i++)
```

```
                    {
                    String st=(String)urat.cu.elementAt(i);
                    int len=st.indexOf("//");
                    String
sub=(cd+System.getProperty("file.separator")+st.substring(len+2)).replace('/',(Syst
em.getProperty("file.separator")).charAt(0));
                    System.out.println(sub);
                    tofile=st.substring(st.lastIndexOf("/")+1);

                    System.out.println(tofile);

                    saveThread au=new
saveThread((DownLoadObserver)project.this,st,sub,tofile,true);
                    au.start();
                    }
          }
          }catch(StringIndexOutOfBoundsException f)
          {
          System.out.println("blis2"+f);
          }
     }
     }

     public void Connecting(String s)
     {
          this.edit3.setText("Connecting to site "+s);
     }
     public void Connected(String s)
     {
          project.this.edit3.setText("Connected to site "+s);
     }
     public void downLoaded(String file)
     {
          project.this.edit3.setText(file +" DownLoaded");
     }
     public void downLoading(String file)
     {
          project.this.edit3.setText("DownLoading "+file);
     }
```

```
    public void Finish()
    {
        project.this.edit3.setText("All URLS has been DownLoaded");
    }
    public void ErrorMessage(String err)
    {
        project.this.edit3.setText(err);
    }
    public void saveHash(String s, URLProperty urlprop)
    {
    url.addStatus(s,urlprop);
    }
    public void addDate(String date)
    {
    project.this.edit5.setText("Time at which Downloaded is"+date);
    }
    public void addDate1(String date1)
    {
    project.this.edit5.setText("Time at which Connected is"+date1);
    }

    public void addurl(String s,URLProperty urp)
    {
    }
     public static project ret()
     {
     project p=new project();
     return p;
     }

   public static void main(String args[])
        {
        project au=new project();
        au.setSize(650,450);
        au.setLocation(0,0);
        au.show();
        }
}
```

//COLLECT URL//

```
import java.io.*;
import java.util.*;
      public class collectUrl implements Serializable{
      Vector cu;
    Hashtable hash;

      collectUrl()
      {
      cu=new Vector();
      hash=new Hashtable();
      }

      public void addurl(String s)
      {
      cu.addElement(s);
      }

      public void addStatus(String s,URLProperty urlprop)
      {
      hash.put(s,urlprop);
      System.out.println("added to hashtable");
      System.out.println(hash+"at adding place");

      }

     public void removeStatus(String s)
     {
     hash.remove(s);
     }
     public void removeFromVector(String s)
     {
     cu.removeElement(s);
     }
```

```
public void saveInfile(String file)
{
try
{
      FileOutputStream f=new FileOutputStream(file);
      ObjectOutputStream o=new ObjectOutputStream(f);
      o.writeObject(this);
      o.writeObject(cu);
      o.writeObject(hash);
      System.out.println(this.cu+"  iam at saving to file");
      System.out.println(this.hash+"  iam at saving hash  to file");

      o.flush();
      o.close();
}catch(Exception a)
      {
      System.out.println(a);
      }
}

public static collectUrl retriveFromfile(String file)
{
collectUrl cU=null;
      try{
            FileInputStream fo=new FileInputStream(file);
            ObjectInputStream os=new ObjectInputStream(fo);
            cU=(collectUrl)os.readObject();
        //    cu=(Vector)os.readObject();
        //    hash=(Hashtable)os.readObject();
      }catch(Exception b)
      {
      System.out.println("Error while retriving :"+ b);
      cU=new collectUrl();
      }
return cU;
}
/*
```

```
public static  void main(String args[])
{
collectUrl cU=new collectUrl();
URLProperty urlprop=new  URLProperty( 1);
//String s,file;
cU.addurl("ramani");
cU.saveInfile("ram");
cU.retriveFromfile("ram");
cU.addStatus("ramani", urlprop);
cU.removeStatus("ramani");
cU.removeFromVector("ramani");
}*/
}
```

//DOWNLOAD OBSERVER//

```
interface DownLoadObserver{
      public void Connecting(String s);
      public void Connected(String s);
      public void downLoading(String file);
      public void downLoaded(String file);
      public void Finish();
      public void ErrorMessage(String err);
      public void saveHash(String s,URLProperty urlprop);
      public void addDate(String date);
      public void addDate1(String date1);
      }
```

//LIST//

```
import java.awt.*;
import java.awt.event.*;
import java.io.*;

public class List1 extends Dialog implements WindowListener,ActionListener
{
        Label l1;
        Label l2;
        List lis;
        TextField t1;
      String copied;
      Button b1,b2;
      File f3;

        public List1()
        {
           super(new Options(),true);
           setLayout(null);
           l1 = new Label();
                l2 = new Label("Save in : ");
                lis = new List(7);
                t1 = new TextField(25);
                b1 = new Button("Cancel");
          b2 = new Button("OK");
          f3 = new File(System.getProperty("user.dir"));

          l1.setLocation(56,24);
          l1.setSize(300,16);
              add(l1);
              l1.setText(f3.getAbsolutePath());

              l2.setLocation(8,60);
              l2.setSize(40,16);
              add(l2);

             t1.setLocation(56,56);
             t1.setSize(184,20);
```

```
        add(t1);
        lis.setLocation(56,88);
        lis.setSize(184,134);
        add(lis);
        lis.addActionListener(this);

        b1.setLocation(168,232);
        b1.setSize(56,24);
        add(b1);
     b1.addActionListener(new ActionListener() {
        public void actionPerformed(ActionEvent a) {
           setVisible(false);
        } });

     b2.setLocation(100,232);
     b2.setSize(56,24);
     add(b2);
     b2.addActionListener(new ActionListener() {
        public void actionPerformed(ActionEvent a) {
           setVisible(false);
        } });

        addWindowListener(this);

    addtolist(System.getProperty("user.dir"));

    }

void addtolist(String str)
{
        String[] str1 = new String[20];
        File f1 = new File(str);
        lis.removeAll();

        lis.add("..");
        if(f1.exists() && f1.isDirectory())
        {
     str1 = f1.list();
     for(int i=0;i<str1.length;i++)
        lis.add(str1[i]);            }}
```

```
public void actionPerformed(ActionEvent a)
{
        if(a.getActionCommand().equals(".."))
        {
                String str = l1.getText();
         if ((str.indexOf(File.separator))==-1)
         {
         l1.setText(str);
                addtolist(l1.getText());
         }
         else
         {
         l1.setText(str.substring(0,str.lastIndexOf(File.separator)));
         if(l1.getText().equals("C:")) {
            l1.setText("C:"+File.separator);
            addtolist(l1.getText());
         }
         else
            addtolist(l1.getText());
        }
       }
       else
       {
              l1.setText(l1.getText()+File.separator+a.getActionCommand());
              addtolist(l1.getText());
       }
}

public void windowClosing(WindowEvent w)
{
      this.dispose();
}

public void windowClosed(WindowEvent w)
{
}

public void windowOpened(WindowEvent w)
{
}
```

```
edit1.setLocation(176, 48);
edit1.setSize(208, 20);
add(checkBox1);
add(button1);
add(edit1);
add(checkBox2);
add(button2);
add(button3);
addWindowListener(new WindowAdapter()
{
public void windowOpened(WindowEvent e)
{
        OptionSaver osr1=OptionSaver.retriveFromFile();
        boolean sd=osr1.getStateDir();
        System.out.println(sd);
         boolean su=osr1.getStateUrl();
         String dl=osr1.getdLoad();
        edit1.setText(dl);
        checkBox1.setState(sd);
         checkBox2.setState(su);

}

 public void windowClosing(WindowEvent e){
Options.this.hide();
 }
 public void windowDeactivated(WindowEvent e)
 {}
 public void windowDeiconified(WindowEvent e)
 {}
 public void windowIconified(WindowEvent e)
 {}

 });
 }
 private class blis implements ActionListener{
 public void actionPerformed(ActionEvent e){
 osr =new OptionSaver();
 String action =e.getActionCommand();
```

```
        if(action.equals("Browse"))
        {
        List1 dlist=new List1();
        dlist.setSize(300,300);
        dlist.setLocation(100,100);
        dlist.show();
        edit1.setText(dlist.l1.getText());

        }
        else if(action.equals("CANCEL"))
        {
       Options.this.hide();
        }
        else if(action.equals("OK"))
        {
        osr.setStateDir(checkBox1.getState());
        osr.setStateUrl(checkBox2.getState());
        if(!((edit1.getText()).equals("")))
        {
        osr.setdLoad(edit1.getText());
        }
        else
        {
        edit1.requestFocus();
        osr.setdLoad("c:\\windows");
        }
        osr.saveInFile();
        Options.this.hide();
        System.out.println("dir name is : "+edit1.getText());
        System.out.println("check1 is : "+checkBox1.getState());
        System.out.println("check2  is : "+checkBox2.getState());
       loadDir=edit1.getText();

       }
       else
       {
//      Options.this.hide();
      }
   }
   }
```

```
    /*  private class blis implements ActionListener{
     public void actionPerformed(ActionEvent e){
     }
     }
     private class blis implements ActionListener{
     public void actionPerformed(ActionEvent e){
     }
     }
     private class blis implements ActionListener{
     public void actionPerformed(ActionEvent e){
     }
     } */


     public String getDirload()
     {
     return loadDir;
     }

/*     public static void main(String args[])
     {
     Options option=new Options();
     option.setSize(468,165);
     option.setLocation(100,100);
     option.show();
     } */

  }
```

//OPTIONS SAVER//

```
import java.io.*;
import java.util.*;
class OptionSaver implements Serializable{
boolean check1,check2;
String dLoad;

        OptionSaver()
        {
          String dLoad="c:\\";
        }

        public void saveInFile()
        {
        try{
          FileOutputStream fos=new FileOutputStream("opSaver");
          ObjectOutputStream oos =new ObjectOutputStream(fos);
          System.out.println("iam at optionsaver"+" "+getStateDir()+" "+getStateUrl()+" "
+getdLoad());

          oos.writeObject(this);
          oos.flush();
          oos.close();
          }catch(IOException e1)
          {
          System.out.println("Error at OptionSaver: error while saving to file >"+e1);

          }
        }


       public static OptionSaver retriveFromFile(){
          OptionSaver opsave;
          try{
          FileInputStream fis =new FileInputStream("opSaver");
          ObjectInputStream ois=new ObjectInputStream(fis);
          opsave=(OptionSaver)ois.readObject();
          }
```

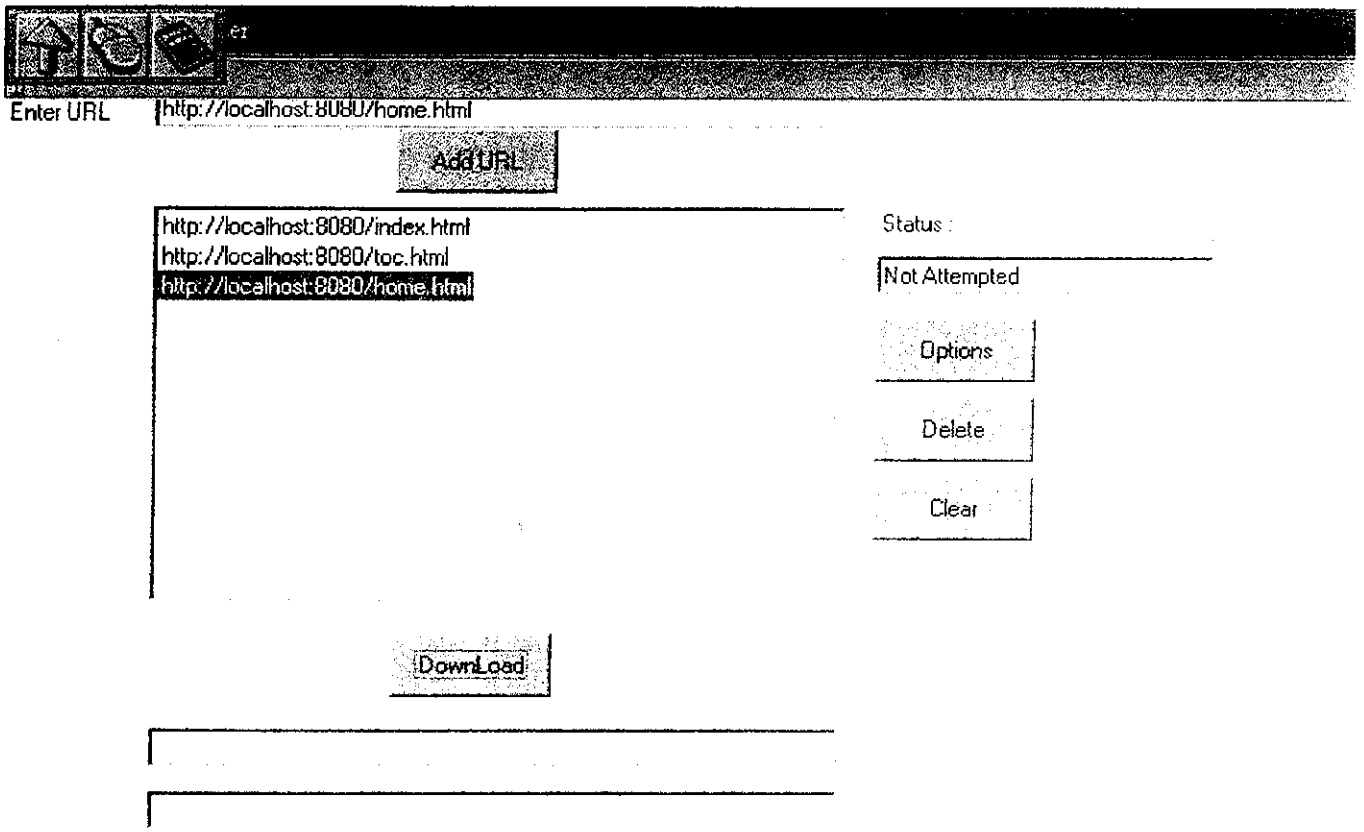
```
catch(Exception e)
 {
 System.out.println("Error at OptionSaver: error while retriving from file >"+e);
 opsave=new OptionSaver();
 }
 return opsave;
 }

 public void setStateDir(boolean b)
 {
 check1=b;
 System.out.println("iam at setstatedir "+check1);

 }

 public void setStateUrl(boolean b)
 {
 check2=b;
 System.out.println("iam at setstateurl "+check2);

 }
 public boolean getStateDir()
 {
 return check1;
 }
 public boolean getStateUrl()
 {
 return check2;
 }
 public String getdLoad()
 {
 return dLoad;
 }

 public void setdLoad(String s)
 {
 dLoad=s;
 System.out.println("iam at setdload "+dLoad);
 }
}
```

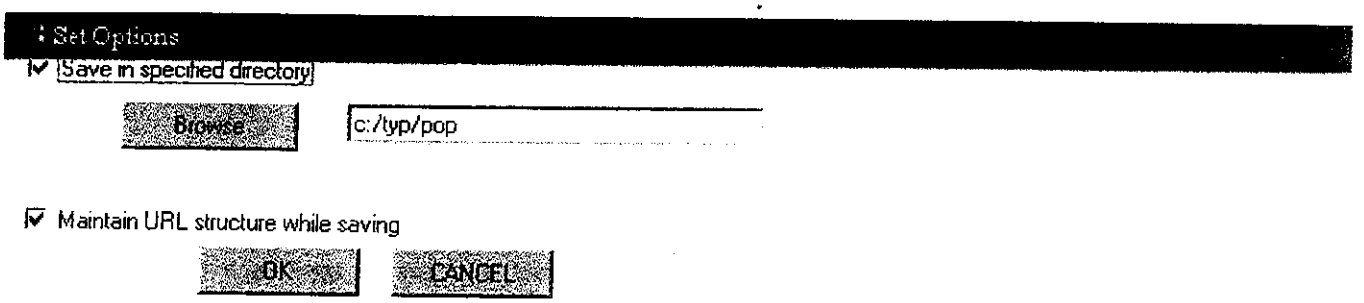
```
public void run()
{
check();
    try{
            if(this.create)
            {

            File p=new File(dirs);
                if(!p.exists())
                {
                created=p.mkdirs();
                }
            fp=new File(p,files);
            }
            else
            {
            fp=new File(dirs,files);
            }
            w=w+1;
          Integer t=new Integer(w);

        System.out.println("I AM AT SAVEdirs"+fp);

        System.out.println("t "+t.toString());


        FileOutputStream fo=new FileOutputStream(fp+t.toString());
        DataOutputStream out=new DataOutputStream(fo);

        InputStream is=hpcon.getInputStream();

        DataInputStream in=new DataInputStream(is);
        byte buf[];
         System.out.println("test "+test1);
        dObserver.downLoading(files);
        Date date=new Date();
        int data=date.getSeconds();
```

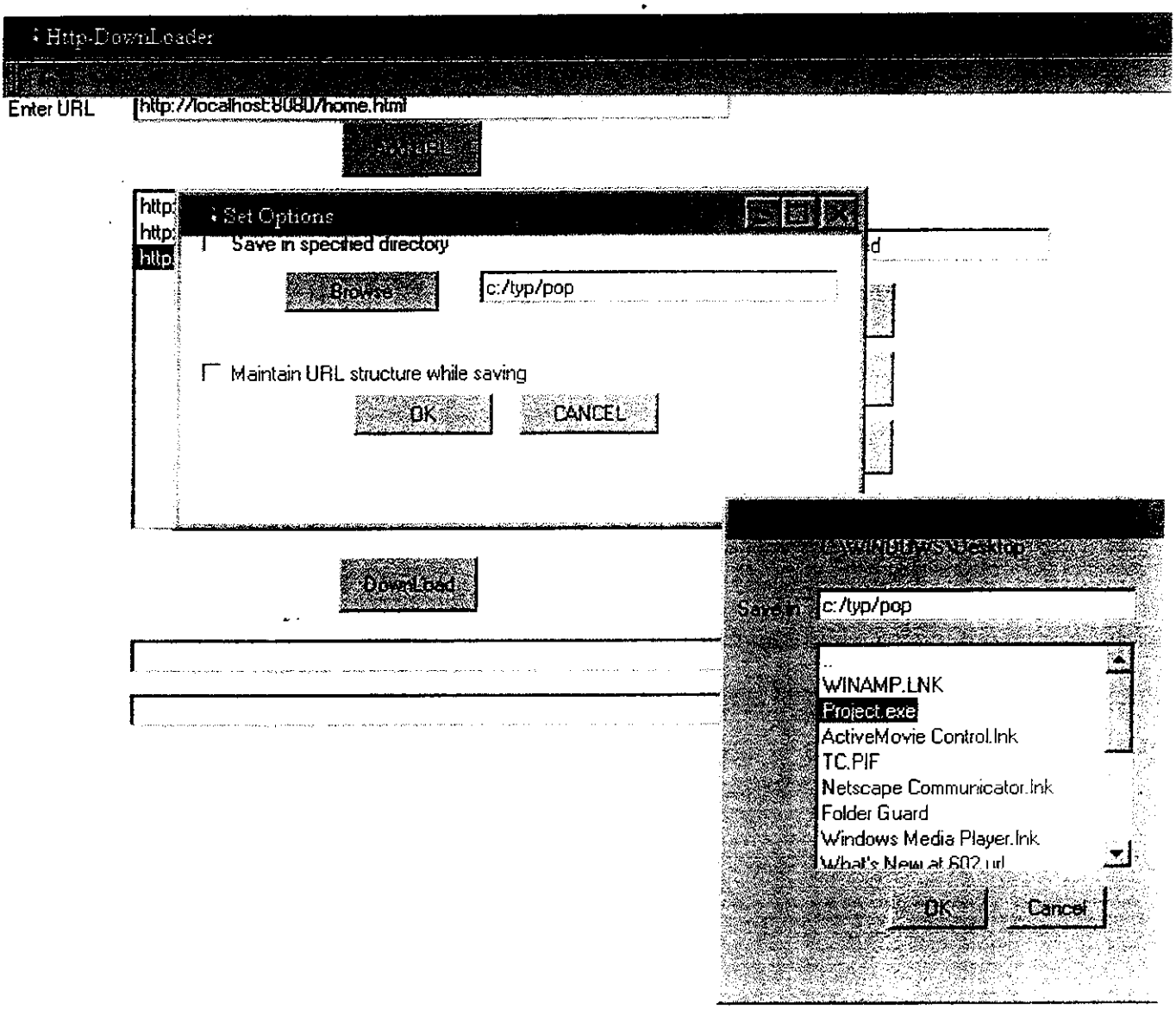
```
                    dObserver.saveHash(urlstrin,new
URLProperty(URLProperty.DOWN_LOADED));


                dObserver.downLoaded(fp+t.toString());
                  Date date1=new Date();
                  dObserver.addDate(date1.toString());


                  while(true)
                  {
                       buf = new byte[2056];
                       int count=in.read(buf);
                       if(buf!=null)
                       {
                            try{
                            out.write(buf,0,count);
                            out.flush();
                            }catch(Exception ui)
                            {
                            System.out.println("out "+ui);
                            break;

                            }
                       }
                       else
                       {
                            out.close();
                            break;
                       }
                AddDate dat=new AddDate(
dObserver,urlstrin,dirs,files,true,test1,date1);
                dat.start();
                this.sleep(20000*w);

                  }


          }catch(Exception t)
          {
          }
```

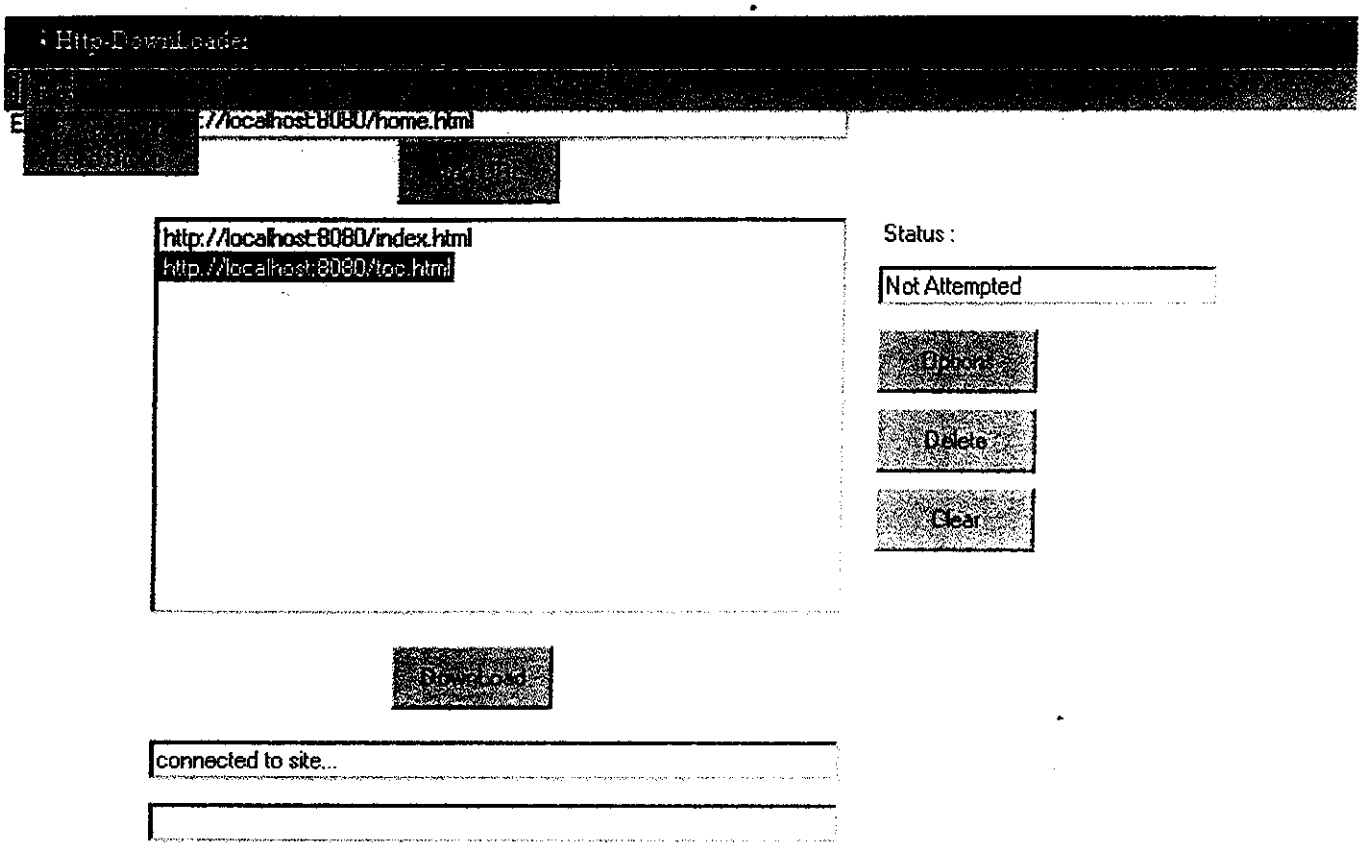
//THREAD//

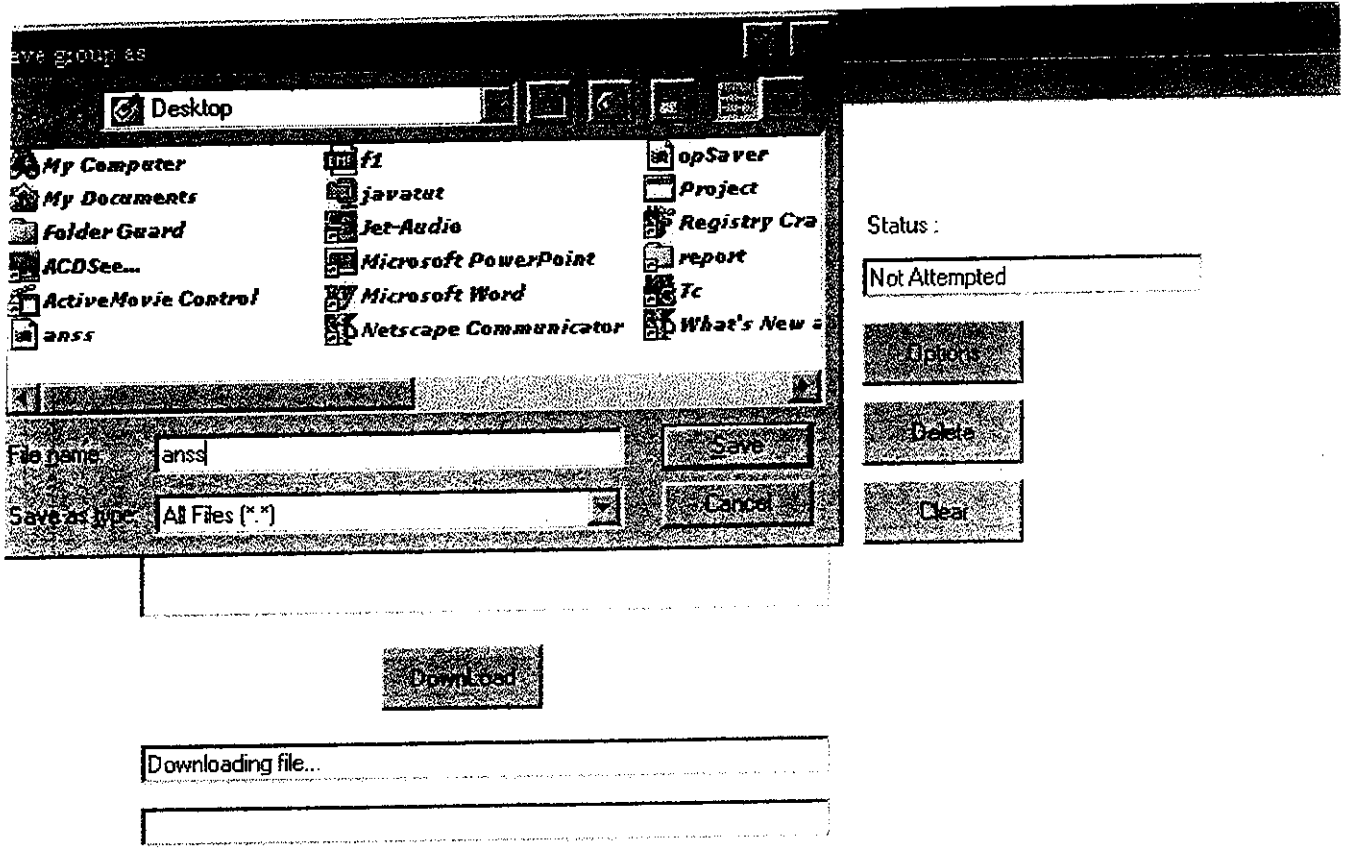
```
import java.util.*;
import java.io.*;
public class URLProperty implements Serializable
{
        int Status;
public static final int NOT_ATTENDED=0;
public static final DOWN_LOADED=1;
public static final int NOT_FOUND=2;
        public URLProperty (int int12)
        {
                Status=int12;
        }
        public int getstatus()
        {
        return Status;
        }
public static URLProperty setstatus(int stat)
{
URLProperty urlprop=new URLProperty(stat)
Return urlprop;
}
        public String getStatus()
        {
                String str=null;
                If(Status==1)
                {
                        str=("Downloaded");
                }
                else if(Status==2)
                {
                str=("File not found");
                }
                else if(Status=0)
                {
                str=("Not Attempted");
                }
                return str;
}
}
```


Enter URL
http://localhost:8080/home.html
Add URL
http://localhost:8080/index.html
http://localhost:8080/toc.html
http://localhost:8080/home.html
Status :
Not Attempted
Options
Delete
Clear
DownLoad


Set Options
Save in specified directory
Browse
c:/typ/pop
Maintain URL structure while saving
OK
CANCEL


Http-DownLoader
Enter URL
http://localhost:8080/home.html
Set Options
Save in specified directory
Browse
c:/typ/pop
Maintain URL structure while saving
OK
CANCEL
DownLoad
Save in:
c:/typ/pop
WINAMP.LNK
Project.exe
ActiveMovie Control.lnk
TC.PIF
Netscape Communicator.lnk
Folder Guard
Windows Media Player.lnk
OK
Cancel


Http-DownLoader
://localhost:8080/home.html
http://localhost:8080/index.html
http://localhost:8080/toc.html
Status :
Not Attempted
Upload
Delete
Clear
Download
connected to site...


Desktop
My Computer
My Documents
Folder Guard
ACDSee...
ActiveMovie Control
anss
f1
javatut
Jet-Audio
Microsoft PowerPoint
Microsoft Word
Netscape Communicator
opSaver
Project
Registry Cra
report
Tc
What's New
File name:
anss
Save
Save as type:
All Files (*.*)
Cancel
Status :
Not Attempted
Options
Delete
Clear
Download
Downloading file...

CONCLUSION

After careful verification and validation procedures, this project Indelible Downloading Software was confirmed to satisfy the organizational requirements in both performance and downloading aspects. The completion of this module helped the organization to simultaneously download many required files from different hosts. This module provided us with an handsome opportunity to learn Java, networking and also a saturated exposure to the internet.

BIBLIOGRAPHY

E. Balagurusamy, "Programming with Java – A Primer" @ TMH Publishing Company, New Delhi. 2nd Reprint Ed., 1998.

Patrick Naughton and Herbert Schildt, " The Complete Reference Java 2 @ TMH Publishing Company Limited, New Delhi. 2nd Ed., 1998.

Robert.A.Day, "How to Write & Publish a Scientific Paper" @ The Press Syndicate of the University of Cambridge, London. 3rd Ed., 1993.