P-1204

BOARD ROOM

Project Report

Submitted in partial fulfillment of the
Requirement for the award of the degree of the

Bachelor of Computer Science and Engineering
Of
Bharathiar University, Coimbatore.

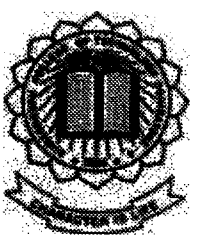
Submitted by

R.Arvind Srinivasan
0027K0161

B.Bharath
0027K0167

Under the guidance of

Mrs.D.Chandrakala M.E.
Lecturer



DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

KUMARAGURU COLLEGE OF TECHNOLOGY,
COIMBATORE – 641006.

MARCH 2004.

BOARD ROOM

Project Report

Submitted in partial fulfillment of the
requirement for the award of the degree of the

Bachelor of Computer Science and Engineering
Of
Bharathiar University, Coimbatore.

Submitted by

R.Arvind Srinivasan
0027K0161

B.Bharath
0027K0167

Under the guidance of

Mrs.D.Chandrakala M.E.
Lecturer

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

KUMARAGURU COLLEGE OF TECHNOLOGY,
COIMBATORE – 641006.

MARCH 2004.

CERTIFICATE

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

KUMARAGURU COLLEGE OF TECHNOLOGY
(Affiliated to Bharathiar University, Coimbatore)

CERTIFICATE
This is to certify that the project entitled

BOARD ROOM

is done by

R.Arvind Srinivasan
0027K0161

B.Bharath
0027K0167

and submitted in partial fulfillment of the
requirement for the award of the degree of the

Bachelor of Computer Science and Engineering
Of
Bharathiar University, Coimbatore.

Professor & Head of the department
(Dr.S.THANGASAMY)

16/3/'04

Guide
(Mrs.D.Chandrakala M.E.)

Certified that the candidates were examined by us in the project work
Viva voce examination held on 23.03.2004.

23/3/'04
Internal Examiner

23/3/04
External Examiner

DECLARATION

Declaration

We,

B.Bharath	**0027K0167**
R.Arvind Srinivasan	**0027K0161**

declare that the project entitled "Board Room", has been done by us and to the best of our knowledge, a similar work has not been submitted earlier to the Bharathiar University or any other institution, for fulfillment of the requirement of the course study.

This project report is submitted on partial fulfillment of the requirement for all awards of the degree of Bachelor Of Computer Science and Engineering of Bharathiar University.

Place: Coimbatore.

Date : 18/3/2k4

B.Bharath

B.Bharath

R. Arvind Srinivasan

R.Arvind Srinivasan

ACKNOWLEDGEMENT

ACKNOWLEDGEMENT

The exhilaration achieved upon the successful completion of any task should be definitely shared with the people behind the venture. This project is an amalgam of study and experience of many people without whose help this project would not have taken shape.

At the onset, we take this opportunity to thank the management of our college for having provided us excellent facilities to work with. We express our deep gratitude to our Principal Dr.K.K.Padmanabhan B.Sc(Engg),M.Tech., for ushering us in the path of triumph.

We are always thankful to our beloved Professor and the Head of the Department, Prof.S.Thangasamy B.E.(HONS)., whose consistent support and enthusiastic involvement helped us a great deal.

We are greatly indebted to our beloved guide and project coordinator Mrs.D.Chandrakala M.E., Lecturer, Department of Computer Science and Engineering for her excellent guidance and timely support during the course of this project. As a token of our esteem and gratitude, we honour her for her assistance towards this cause.

We also thank our beloved class advisor Ms.Hema Guptha B.E., for her invaluable assistance.

We also feel elated in manifesting our deep sense of gratitude to all the staff and lab technicians in the Department of Computer Science and Enineering.

We feel proud to pay our respectful thanks to our Parents for their enthusiasm and encouragement and also we thank our friends who have associated themselves to bring out this project successfully.

SYNOPSIS

SYNOPSIS

This project is an intranet application.The application simulates a board room session.The application can be used by board directors during meetings.It facilitates the directors to enter specific rooms appropriate to their topic of discussion.The users log into their respective rooms and start their session.At the basic level ,this application provides a chat facility for the workgroup.Also with it comes a white board which provides various other tools which enable them to share information more effectively. The tools include those used for drawing basic geometric shapes.The graphical tools help the directors to share information about the company's statistics amongst each other in more effective, interactive and understandable manner. User administration is possible , where unwarranted users can be banned. New rooms can be created by the administrator as well as the user. Rooms involving confidential discussions can be made private and a password can be associated with it, thereby restricting the users who take part in the discussion. On the other hand, general topics can be broadcasted to all the users. The product also comes with an instant messaging and paging option, to facilitate short messaging among the users. Messages for offline users can also be saved. An option for saving the chat and the canvas has also been provided to facilitate references to previous discussions

CONTENTS

CONTENTS

SNO	TOPIC	PAGE NO
1.	Introduction 1.1 Existing System 1.2 Proposed System	1
2.	System Requirement Analysis 2.1Product Definition 2.2Project Plan	2
3.	Software Requirement Specification 3.1 Introduction 3.2.Overall description 3.3.Specific requirements 3.4 Performance Requirements	4
4.	System Design 4.1Introduction 4.2Decomposition Descriptions 4.3Dependency Descriptions	7
5.	System Testing 5.1Introduction 5.2Definitions 5.3Unit testing Activities 5.4Functional Testing 5.5Integration Testing	16
6.	Future Enhancements	27
7.	Conclusion	28
8.	Bibilography	29
9.	Annexure 9.1 Sample Source Code 9.2 Sample Output	30

INTRODUCTION

1) INTRODUCTION

1.1) Existing system:

The problem in existing system is that, in any chat application we cannot communicate (on the spot) by graphics using drawing controls. The other facility available is net meeting, which works in windows alone. The other hi tech systems include video conferencing ,which are very costly and cannot be afforded by smaller organizations. When it comes to chat applications there is no facility of a board meeting.

1.2) Proposed System :

The aim of the proposed system is to give a good chatting application for conferencing purpose with the following advantages :-

- On the spot graphics communication using drawing controls
- It is platform independent. Unlike Net meeting , it can also work in linux and unix.
- Not only is this application useful for conferencing amongst the board members, it can also be used for the transfer of text and graphic files.

SYSTEM REQUIREMENT ANALYSIS

2)SYSTEM REQUIREMENT ANALYSIS

2.1) Product Definition:

Boardroom is very simple; The constraint of understandability requires that it is of fairly limited expanse. Hopefully, however, the techniques demonstrated will be of fairly broad use, and will allow you to easily modify the application to suit your own needs. Board room is a way of chatting and transfer of information in the form of graphic files and text.

2.2) Project Plan:

The project entitled 'Board Room' is to develop an online communication between the clients and the relay server. A client tool is provided to connect to the server and exchange information. Boardroom provides a media through which board members can share information. The information that is shared will be in the form of graphics, text and files. The session can take place in any of the three ways :-

i) one to one

ii) one to many

iii) many to many

This application provides a board which mirrors each other. The server also provides user administration. In this application, two important features included are:

- Chat room control
- Drawing control

Chat room control:

Using this the user can create chat rooms, enter, update and also view the details about the chat room such as creator, users present etc.,

Drawing control:

Using this the user can select any tool present in the control panel to help himself drag and draw upon the canvas. The other option included helps us to paste pictures and send it to others Drawing utility, commonly supplied as part of a collaboration framework is to allow distributed users to share a common drawing space

SOFTWARE REQUIREMENT SPECIFICATION

3) SOFTWARE REQUIREMENT SPECIFICATION

3.1) Introduction :

The following section gives the purpose and scope of the software requirement specification

3.1.1) Purpose of the document:

The purpose of this document is to provide the requirements for the whiteboard (client server communication). Sections 1 and 2 are primarily intended for customers of the application, but will also be of interest for network programmers. Section 3 is of use for network engineers, software people and customers of this application.

3.1.2) Scope:

The document is limited to the description of whiteboard, a communicator used for communication between client and server.

3.1.3) Definition, acronyms and abbreviation:

Board : It mirrors each member of the board for communication purposes.

Canvas: It provides a drawing surface for the board members.

Socket: It helps in establishing a connection between two computers.

Server: It accepts the connection and relays messages between clients.

Client: They connect to the server and communicate with each other.

3.2) Overall description:

The main function associated with this product is described in this section. The characteristics of a user of this product are indicated.

3.2.1) Product perspective:

This product indented to fulfill the needs for user who are good at socket programming or the user who have a greater influence over the content of the product with AWT programming. The design and documentation for this will make it convenient to expand and modify the tools. It is interactive to users.

3.2.2) Product function:

The product is in form of a chatting facility, which is used for transfer of text and graphic files. This consists of creating chartrooms, drawing utility (board room) which is very useful for the users. If more than one user is connected, the information is served by server. If one of the client wants to clarify, to interrupt the server, the particular client can send message to the server and it will be interrupted and the server will clarify the clients doubt.

3.2.3) User characteristics:

Those using this product are common users, who are able to understand every aspect. The user must know about the basic socket programming and AWT concepts. They need training to understand and use the system effectively.

3.3) Specific requirements:

In this section we will see about the specific requirements like interface, functional requirements, performance requirement of the product.

3.3.1) Interface requirements:

The user interface of this product requires a text area for information transfer and a drawing area to draw.

3.3.2) Functional requirements:

This will define the fundamental actions which take place in the product

3.3.2.1) Information flow:

The information flow is in the form of graphics, files and text .The client will first connect to the server and on connecting , the information is served by server to the client.

3.3.2.2) Process description:

The client in order to enter the communication, will first get a connection with the server and server will transfer information to client.If the client wants to communicate to the server for clarification, it can interrupt the server and clear its doubt. The process also consists of creating chartrooms , the details of which are maintained by the server.

3.4) Performance requirements:

The static as well as dynamic numerical requirements placed on the software or on human interaction with the product are described in this section. Examples of the performance requirements are:

Dynamic: All click and drop operation is accomplished within some milliseconds.

Static: Each control involved should be static.

SYSTEM DESIGN

4) SYSTEM DESIGN

4.1) Introduction:

This document will specify the logical structure and will also show the detailed description of the product.

4.1.1) Purpose:

The purpose of the software design document (SDD) is to provide an overall description of "board room" and to specify the system's detailed design.

4.1.2) Scope:

The scope of the software design document is constrained by the requirements stated in the "Board room" software requirement specification.

4.1.3) Definitions, Acronyms and Abbreviations:

Board: It mirrors each member of the board for communication purposes.

Canvas: It provides a drawing surface for the board members .

Socket: It helps in establishing a connection between two computers.

Server: It accepts the connection and relays messages between clients.

Client: They connect to the server and communicate with each other.

Conference: This will display the information which the server or client typed in the text area.

Text area: It is area for text based communication between client and server.

4.2) Decomposition Descriptions

This section describes the modules involved in the project and the descriptions of their functions.

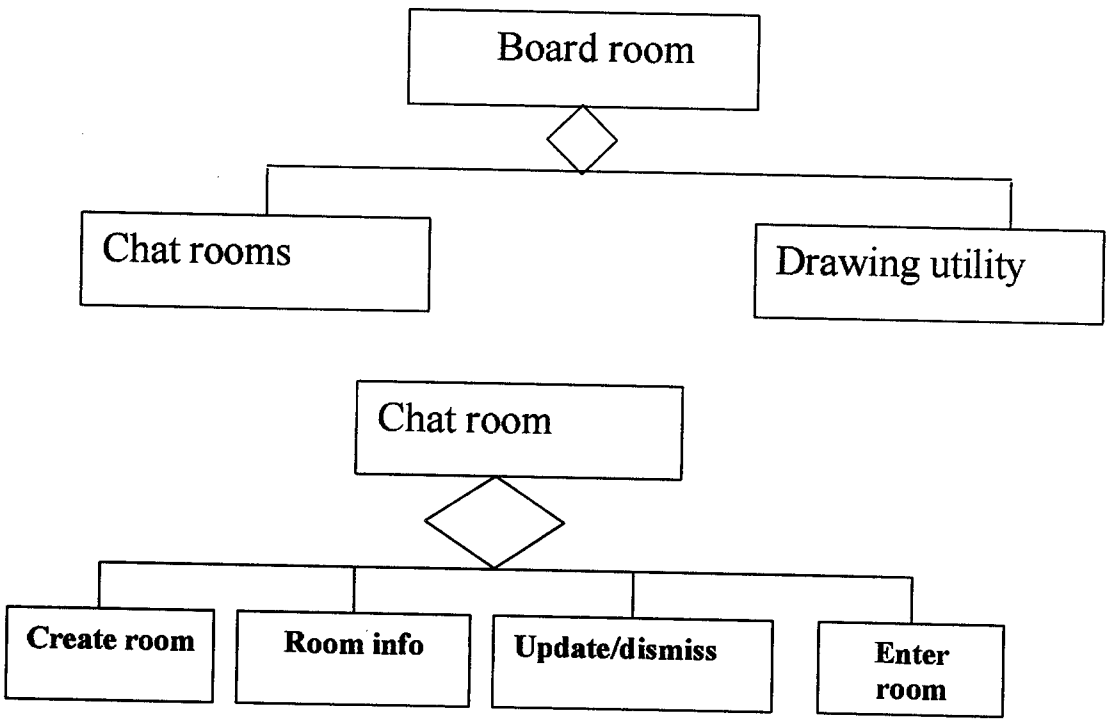
4.2.1) Module Decomposition

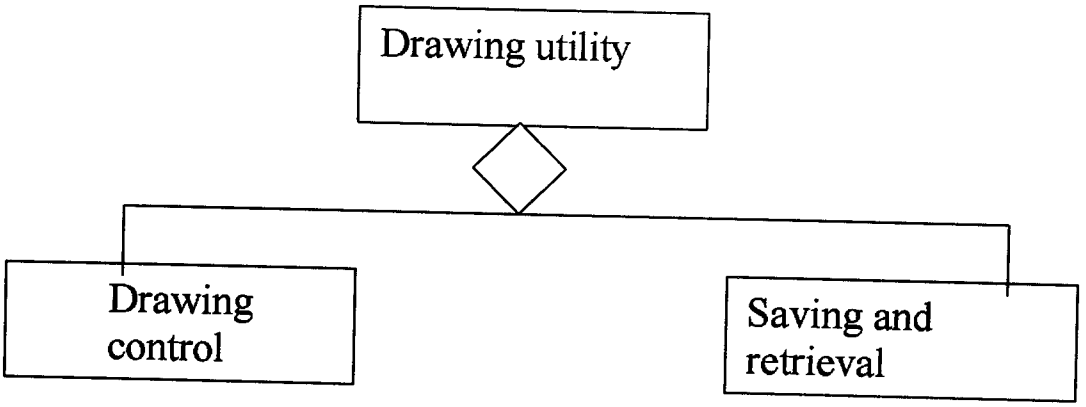
In this the modules are classified based on the type of communication. They are :

i) Text

ii) Drawing

Aggregation diagram:





4.2.2) Functional decomposition

Chat rooms:The chat rooms consist of creating, entering, updating and dismissing users.

- **Creating**: Both the server and client can create chat rooms. If the client or server want it private they can set password.
- **Entering into chat room:** If the user wants to enter into a particular room, the room list is displayed and he can select any one of them.
- **Room information**: The user can view the room information such as the room name, creator, status (private\public),number of users, names etc using this facility.
- **Update and dismiss**: The user can update the room information and dismissal of unwanted users is also possible.
- **Chat save** : The chat can be saved and retrieved

Drawing utility: This consists of a canvas and the drawing controls which it supports.

- **Canvas**: It is a simple drawing surface. It provides a drawing area to draw upon. Using mouse the user can select particular control and drag upon the canvas. In this canvas the user can paste pictures and the user has a option of saving the canvas and then retrieving it.

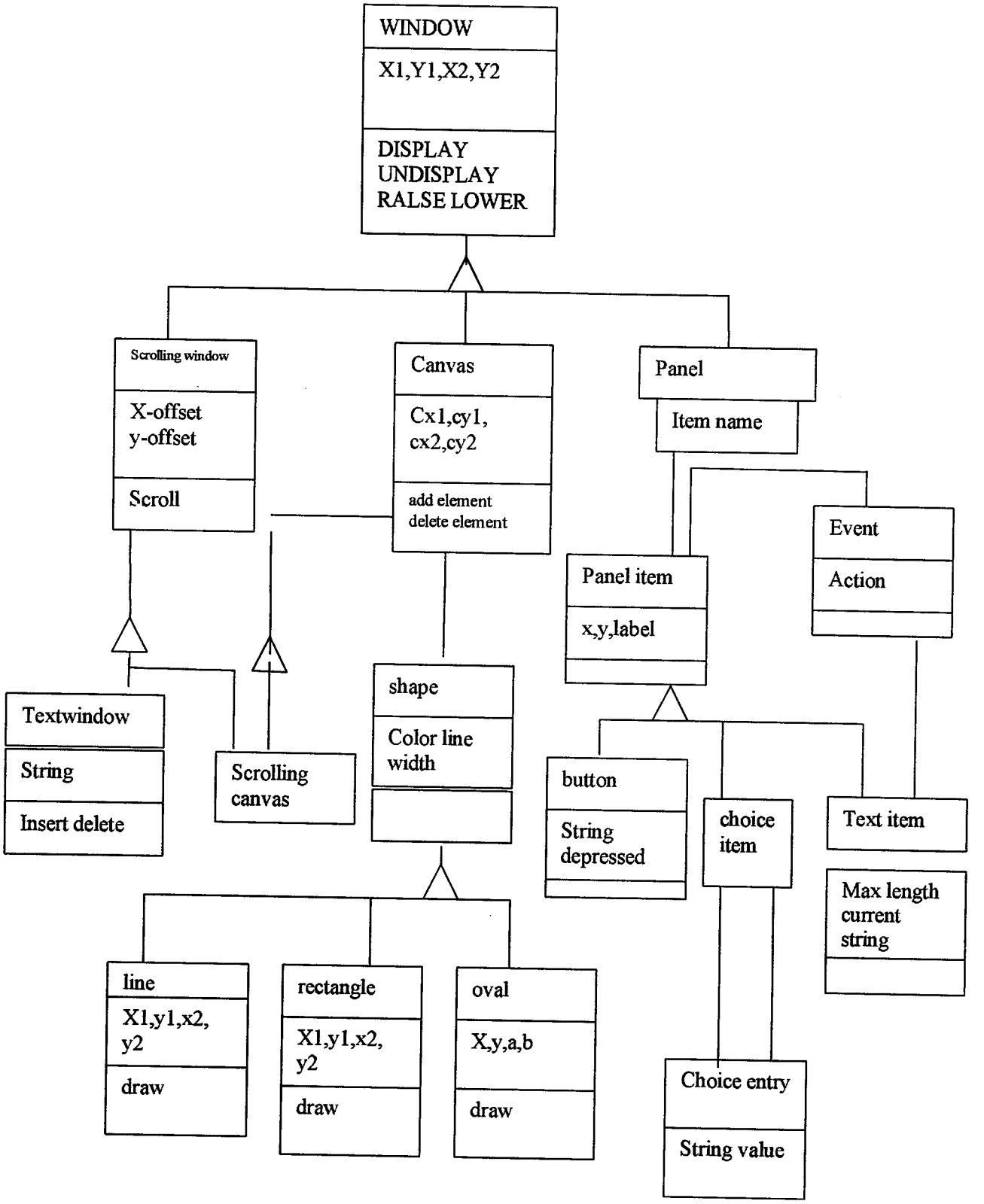
- **Drawing control**: This consists of freehand drawing, draw rectangle, line, oval and text. The user selects a particular control and draws upon the canvas

- **Rectangle\oval**: When user clicks the rectangle\oval he has a filled/ outlined option.

- **Text**: When the user selects text, a dialog box will be displayed which show the text type, style and size as attributes to be selected.

- **Clear canvas**: This option helps the user to clear the canvas.

ACTIVITY DIAGRAM



4.3) Dependency descriptions

4. 3.1)Intermodule Dependencies:

The use case is a view of the system from a specific User's view. This considers actions initiated by the user on the system, and consequently, the actions initiated by the system on the user.

The use case diagrams illustrate the effects on the client with the typical uses of the product. These diagrams provide a lower level implementation, such as the Order, Direction and Information of use case.

4.3.1.1) High Level Analysis Models

The Analysis Model has been generated by a process of refinements starting from the Use Case Models. Each use case role has been devised into the stages of the analysis model. This model is a very general and high level collaboration diagram. It takes the role from the use case diagram and expands each of the use cases to show how they will collaborate within the components of the design. The detailed use case descriptions are described using stereotypes. These are as follows:

Boundary Class

Models interaction between the system and its user (users and external systems).

Control Class

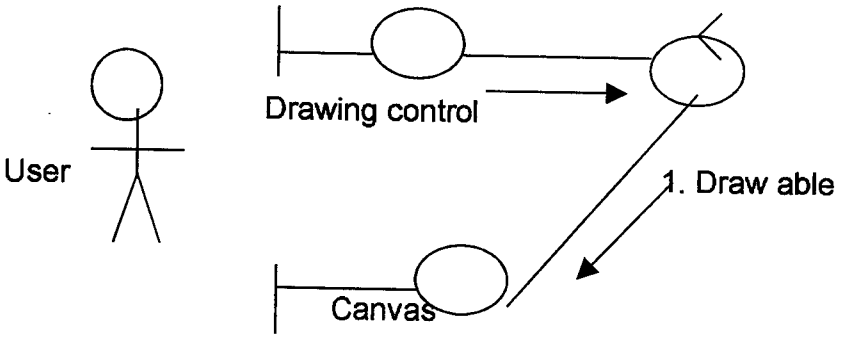
Represents coordination, sequencing, transition and control of other objects.

Entity Class

Modules information long-lived (persistent) data.

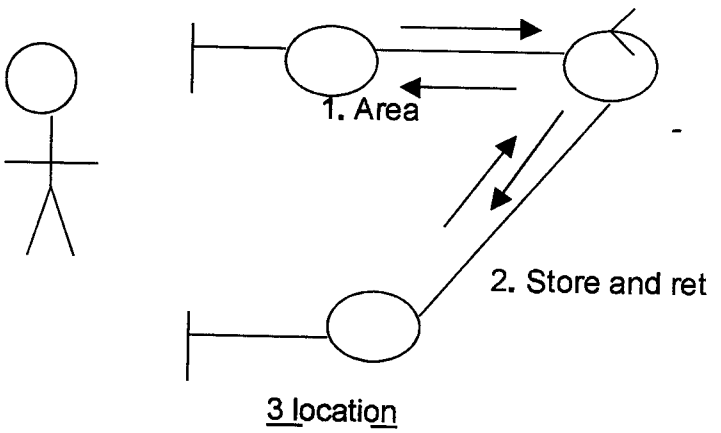
4.3.1.2) Stereotypes

Drawing:



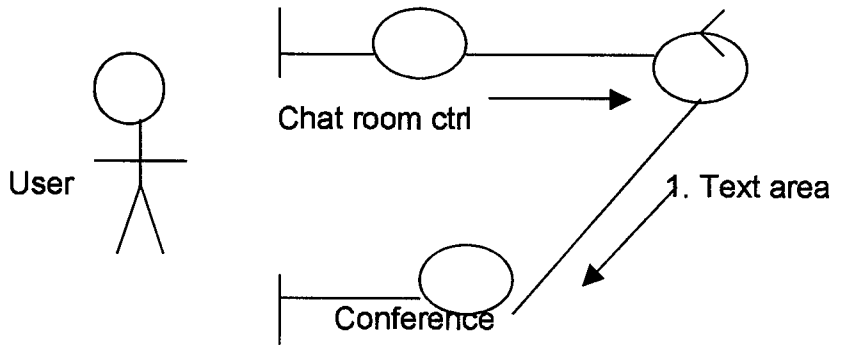
The above diagram shows that when the user wants to draw any shape (present) he clicks the appropriate control in the window. Then it calls the particular function to draw it, so that it will be displayed on the canvas...

Storage and Management (canvas\chat):



The storage of the chat or the canvas is carried out by selecting the storage and retrieval function to perform the saving operation, after which it gets stored in the particular location.

Chat Room Control:

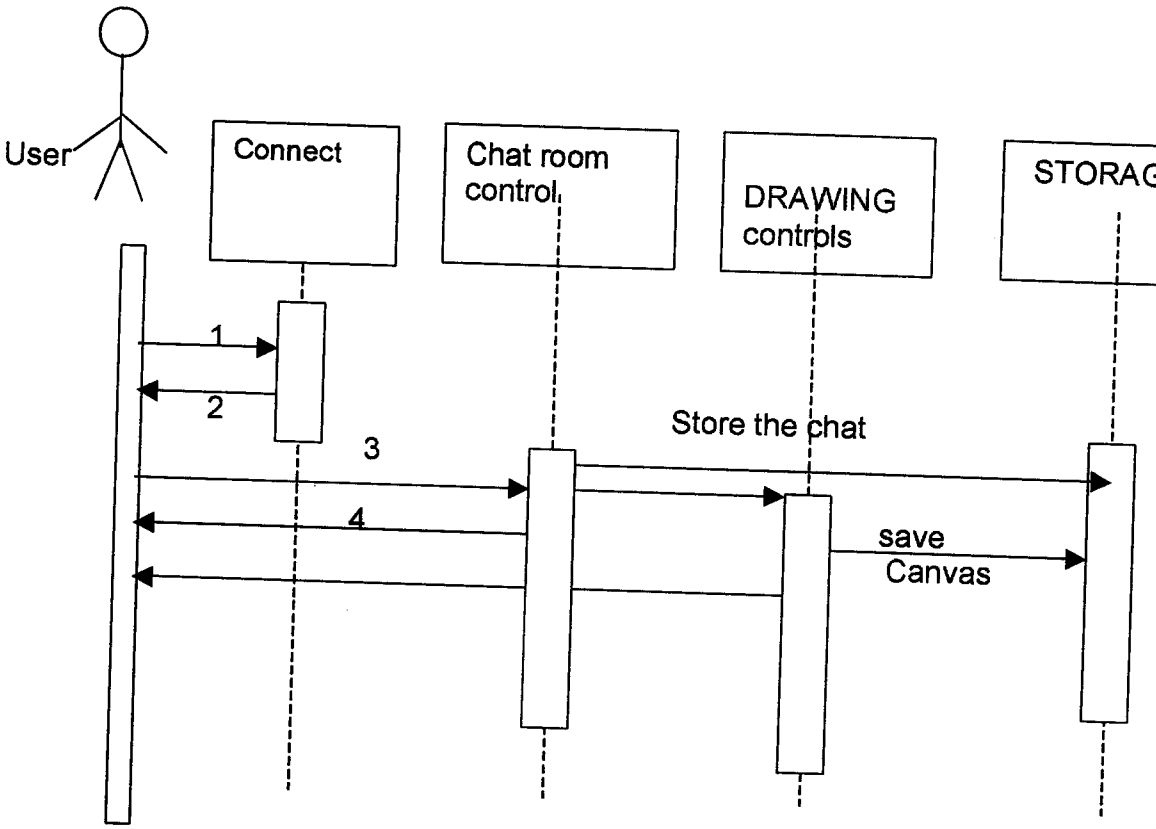


The above diagram shows the case when the user wants to chat. The user must enter the room (present) and he can view the users present in the room after which chatting is initiated. When the user types any information, it is displayed on the conference area of other users, and that information can be viewed by all users present in that room.

Chat room control is a management process in which the user can create his own chat rooms, enters any room he wants , updates room information and views the rooms information like the creator, number of users etc..,

4.3.2) Interprocess Dependencies

Sequence Diagrams



1.Connect.

2.accept:

3. select chat room

4. option dialog box:

The above diagram illustrates that the user has to first request connection. If connected , he can chat and draw using the drawing controls .He can save the chat and the canvas whenever he wishes.

SYSTEM TESTING

5) SYSTEM TESTING

5.1) Intoduction

5.1.1) Inside the scope:

The scope of this document is to give a test case for testing the product's functionality & performance. The test case should be designed in such a way that it has a high probability of finding as-yet undiscovered error.

5.2) Definitions

Board: It mirrors the members for communication

Canvas : It is a simple surface which provides an area to draw upon.

GUI Environment It is a graphical user interface, which displays the commands as menu and icons, which can be easily manipulated using the mouse.

Mouse: A pointing device used in GUI interfaces

Test units – a set of one or more computer program modules together with associated control data, usage procedure .

Unit- refers to the test unit

Refine – adding some improvements to the utility

5.3) Unit Testing Activities

5.3.1) Plan the general approach, resources & schedule:

The general approach for testing the product is to find the requirements, design document details and then plan the test case. We should also consider the environment & design the test case for specialized environments. In this case the software is a Graphical User Interface (GUI) & so the test case should be designed in such a way that it tests all the properties & functions of a GUI

Required Resources for the testing process:

System Requirement Specification (SRS) document.

System Design Description (SDD) documents.

5.3.2) Determine Features to be tested:

This is a test for a specialized environment like the GUIs and so we should take care in testing the qualities of GUI interfaces & other properties like interfacing of the system with the environment.

The test includes:

The test for application window properties

i) The test for pull down menus

ii) The test for the mouse operations

5.3.3) Design the set of tests:

The language used for the development of the product is java, so that we can go through and test every action and hope that we get the expected response. We will create a template for every test. So the first three test sets are GUI based tests & the next two are action based & function based tests respectively.

Test Set 1: For Windows

1. Will the window open properly based on related typed or menu-based commands?
2. Can the window be resized, moved, and scrolled?
3. Is all data content contained within the window properly addressable with a mouse, function keys, directional keys, and keyboard?
4. Does the window properly regenerate when it is overwritten and then re-called?
5. Are all functions that relate to the window available when needed?
6. Are all functions that relate to the window operational?
7. Are all relevant pull-down menus, controls, scroll bars, dialog boxes, and buttons, available and properly displayed for the window?

Test Set 2: For pull-down menus and mouse operations:

1. Do pull-down operations work properly?
2. Are all functions and pull-down sub functions properly listed?
3. Are all functions properly addressable by the mouse?
4. Are text typeface, size, and format correct?
5. Does each function perform as advertised?
6. Do multiple or incorrect mouse picks within the window caused unexpected side effects?

Test Set 3: Function based testing

1. Line Draw
2. Check for the valid number of inputs, if valid then what does it do or what.
3. Does it do incase the arguments are invalid?
4. Does it use the proper draw color?
5. Rectangle Draw function
6. Oval Draw function
7. Free hand drawing function

5.3.4) Execute the test procedures:

The above set of tests is executed in sequence & if any errors or negative results occurred for the test, then it is corrected immediately and the test is preceded further.

5.3.5) Check for termination:

The results for the above test set are required to be acceptable to the requirements and other factors for termination of the test process.

5.3.6) Evaluate the test effort:

The following are the test results with the errors that have occurred & corrected:

Test Set 1:

1. Yes, the window opens properly according to the context.
2. Yes, all can be done.
3. Yes, previously some commands were not given proper keyboard shortcuts and that has been corrected.
4. Yes, the window re-draws work properly.
5. Yes, all functions are available when needed.
6. Yes, all functions relate to that window.
7. Yes, all controls & objects are displayed at their appropriate places.

Test Set 2:

1. Yes, the pull-down menus work properly.
2. Yes, all are listed properly.
3. Yes, all are addressable by the mouse.
4. Yes, the font properties are correct.
5. Yes, perform as advertised
6. No, incorrect mouse picks within the window do not generate unexpected side effects.

The above results are evaluated and accepted to be satisfactory & hence the test process is terminated.

5.4) Functional testing:

The testing refers to the operation of some important function, which is in the product.

The drawing operation is a process present in this application

In drawing a rectangle:

With the help of the mouse the user can click the object and drop the selected object upon the canvas

Clear the canvas:

When the user wants to clear the canvas, click the canvas button.

Save the chat and canvas:

When the user wants to save the chat and canvas, select save chat and save canvas button

Retrieve the chat and canvas:

When the user wants the retrieve back the chat and canvas, he will invoke the location where the chat and canvas saved and their he can select and retrieve.

Functional test result:

1. Yes, an object is selected and dropped upon the canvas
2. Yes, an object is selected and that object is cleared from the canvas by clicking the clear option
3. Yes, chat and canvas are saved using save chat and save canvas button
4. Yes, chat and canvas got retrieved from the location where the chat and canvas were saved

5.5) Integration testing:

It is a systematic technique for constructing the program structure while at the same time conducting tests to uncover errors associated with interfacing.

5.5.1) Top down integration:

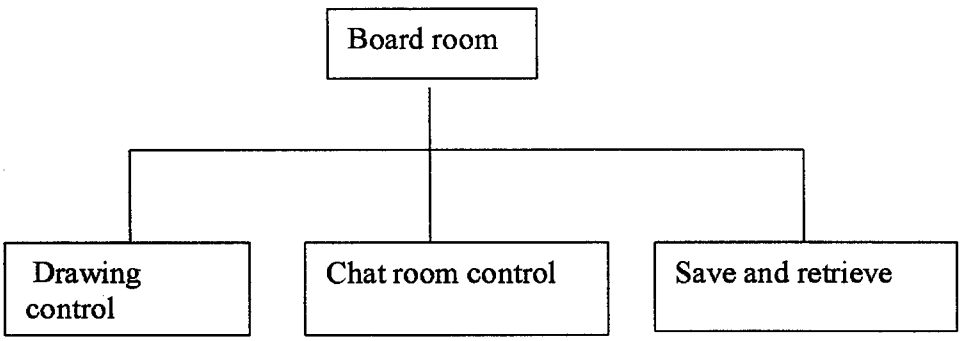
It is an incremental approach to the construction of program structure]

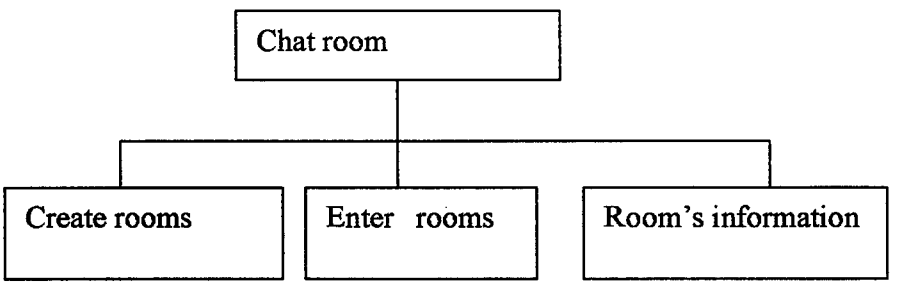
The modules are:

i) Chat room control module

ii) Drawing control module

iii) Save and retrieve module

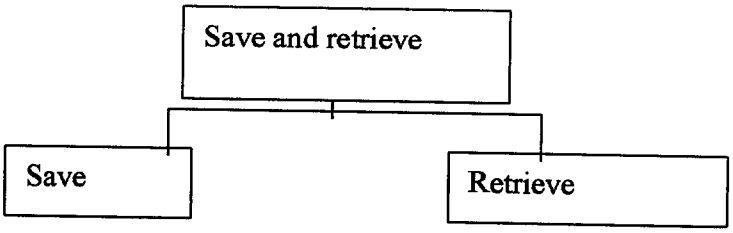




Create rooms: The user can create the rooms.

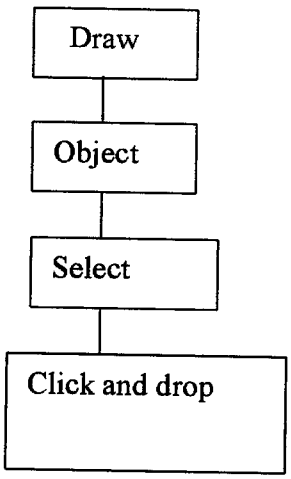
Enter room: The user can enter any room which the user has selected from the list.

Room information: The user can view the information about the room.



Save: Save the canvas\chat by selecting the save option.

Retrieve: Get back the canvas\chat from the location where the canvas is saved.



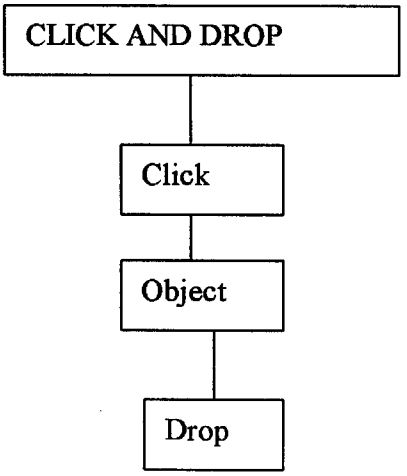
Definition:

Object: Drawing tools in the toolbar.

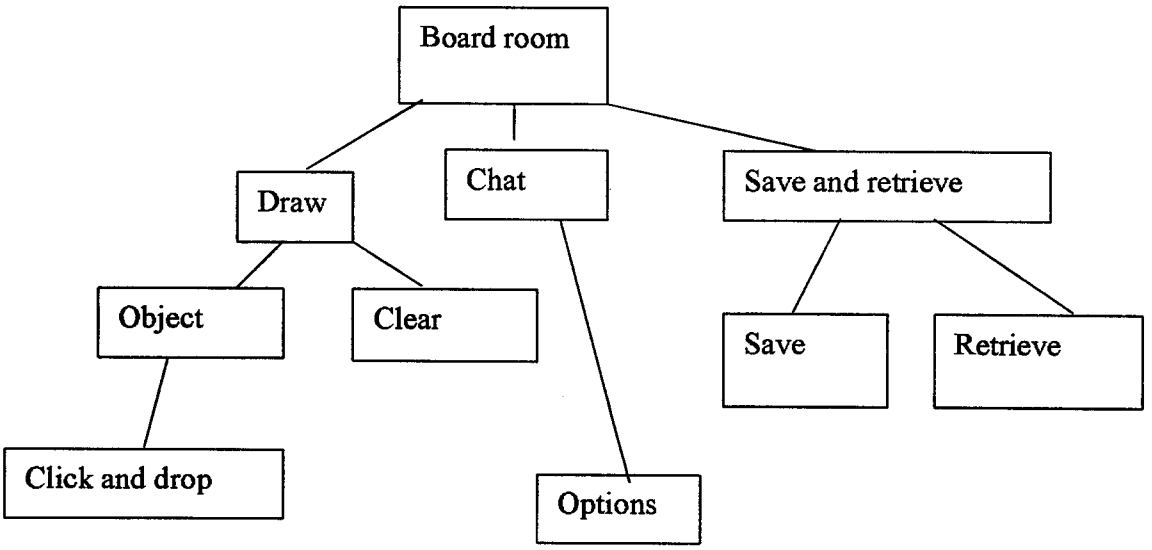
Draw select: Select an object from the toolbar.

Click: Mouse click in order select the object in the toolbar.

Drop: Drop the selected object into the drawing space.



Main control:



Draw : In this top down integration testing, first the product starts testing the main control of drawing.

i) Is the object selected by mouse click?
ii) Is the object dropped in the drawing space?

Chat : In this, the chat room control is tested
i) Can the user create chat rooms?

ii) Is the user able to enter the chat room that he selected?

iii) Can the user view the information in the chat room?

Clear :
Is the canvas cleared using the clear button option?

Save and retrieve :
i) Does the chat\canvas get saved using save option?

ii) Is the chat\canvas retrieved?

5.5.2) Bottom up integration:

Bottom up integration strategy may be implemented with the following steps:

i) Low-level modules are combined into clusters that perform a specific software sub function

ii) Cluster- sometimes called builds.

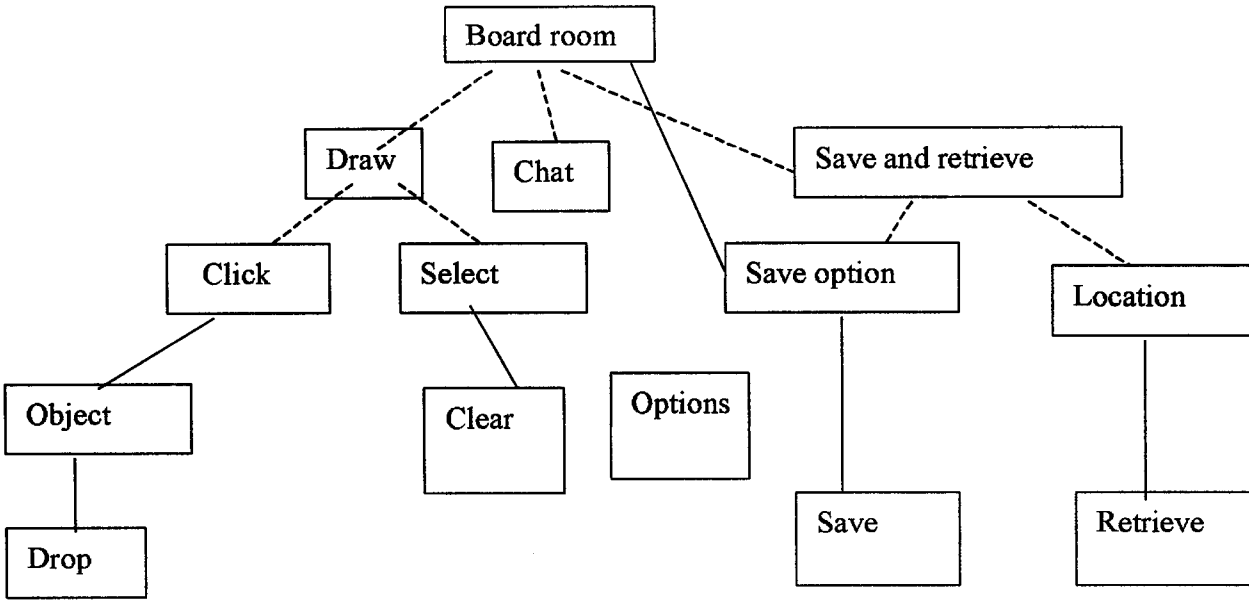
iii) A driver is written for coordinates test case input and output

iv) Driver- a control program for testing

v) The clusters are tested

vi) Drivers are removed and clusters are combined moving upwards in the program structure.

vii) Main controls in this bottom up testing:



Cluster

Main control

Click; select, save option, location - drivers

Cluster:

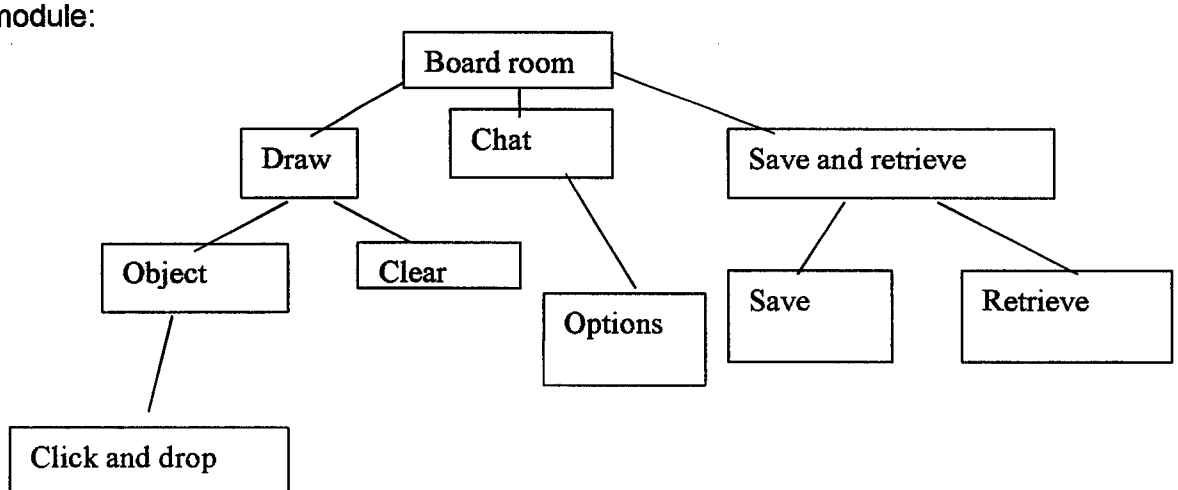
1. Is an object selected on mouse click and can it be dropped on the drawing space?

2. Is canvas cleared using clear button option?

3. Is chat\canvas saved using the save option?

4. Is the saved chat\canvas retrieved from the location?

Then the drivers are removed and the clusters are interfaced directly to the module:



First the driver's click and select are removed and the object cluster is interfaced with the drawing control module.

Next the save option and retrieve option drivers are removed and the save and retrieve clusters are interfaced with the save retrieve (location) module.

Test result:
Top down approach:
Draw
Yes, the object is selected by mouse click.

Yes, the object is dropped in the drawing space by using mouse drop.

Chat
Yes, the user is able to create chat rooms

Yes, the user can enter the room ,that he had selected

Yes, the user can view all information about that room

Clear
Yes, the canvas is cleared when the user selects the clear button option.

Save and retrieval:
Yes, the chat\canvas is saved using save option.

Yes, the object is retrieved by selecting the location. A window appears and the chat\canvas is displayed.

Bottom up integration:
Cluster:
Yes, an is selected and dropped in the drawing space.

Yes, the canvas is cleared when the user selects the clear button option

Yes, the chat\canvas is saved using save option.

Yes, the object is retrieved by selecting the location. A window appears and the chat\canvas is displayed.

FUTURE ENHANCEMENTS

6)FUTURE ENHANCEMENTS

i) We are planning to implement voice chat, so the users can communicate using voice also.

ii) We are planning to improve product towards video conferencing.

iii) We are also planning to implement search engine for reference by users while chatting i.e.. The user can search for any e-books, while related chatting is going on

CONCLUSION

7) CONCLUSION

Board room is a way of chatting and transfer of information in the form of text, files and graphics which help the users to get a control over the chat room by creating, entering, and also by viewing information about that room .The user can click on any drawing control and drop them in the canvas. The user can save chat \canvas and can as well retrieve from that location.

BIBLIOGRAPHY

8) BIBLIOGRAPHY

IEEE Standard for Software Requirement Specification 830 - 1993

IEEE Design document 1016-1998

IEEE standard for Software Unit testing – std.1008-1987.

Patrick naughton & Herbert schildt , "The Complete Reference Java 2 "5th Edition 2002

Richard E.Fairley,"Software Engineering Cncepts",Tata McGraw-Hill Edition 1997

Websites Visited

www.dur.ac.uk/`dcs8s08/design/design.html

http://cs.anu.sdu.au/student/comp3100/delivs/type2/pr-ut.html

ANNEXURE

9) ANNEXURE

9.1) Sample source code:

```
// mezochat.java
//

import java.awt.*;
import java.io.*;
import java.util.*;
import java.net.*;


public class mezochat
   extends Object
   implements Runnable
{
   public static final String VERSION = "2.1_BETA";

   public static String usernameParam     = "-username";
   public static String passwordParam     = "-password";
   public static String servernameParam   = "-servername";
   public static String portnumberParam   = "-portnumber";
   public static String chatroomParam     = "-chatroom";
   public static String widthParam        = "-xsize";
   public static String heightParam       = "-ysize";
   public static String nopasswordsParam  = "-nopasswords";
   public static String locksettingsParam = "-locksettings";
   public static String autoconnectParam  = "-autoconnect";
   public static String hidecanvasParam   = "-hidecanvas";

   private mezochatWindow window;
   private URL myURL = null;
   private Class thisClass = mezochat.class;
   private String name = "";
   private String password = "";
   private String host = "";
   private String port = "";
   private String room = "";
   private int windowWidth = 0;
   private int windowHeight = 0;
   private boolean requirePasswords = true;
   private boolean lockSettings = false;
   private boolean autoConnect = false;
```

```
  private boolean showCanvas = true;

  // For managing strings according to the locale
  protected mezochatStringManager strings = null;


  public mezochat(String[] args)
  {
       // Get a URL to describe the invocation directory
       try {
          myURL = new URL("file", "localhost", "./");
       }
       catch (Exception E) {
          System.out.println(E);
          System.exit(1);
       }

       // For managing strings according to locale
       strings = new mezochatStringManager(myURL,
Locale.getDefault().getLanguage());

       // Parse our args.  Only continue if successful
       if (!parseArgs(args))
          System.exit(1);

       // If "username" is blank, that's OK.  However, if the server
and/or
       // port are blank, we'll supply some default ones here
       if ((host == null) || host.equals(""))
          host = "visopsys.org";
       if ((port == null) || port.equals(""))
          port = "12468";

       // Open the window
       window = new mezochatWindow(new mezochatPanel(name,
password, host,
                                                     port, showCanvas,
                                                     myURL));

       // Set the window width and height, if applicable
       Dimension tmpSize = window.getSize();
       if (windowWidth > 0)
          tmpSize.width = windowWidth;
       if (windowHeight > 0)
          tmpSize.height = windowHeight;
```

```
        window.setSize(tmpSize);

        // Make the pretty icon
        window.setIcon(myURL);

        // Should the window prompt users for passwords
automatically?
        window.contentPanel.requirePassword = requirePasswords;

        // Should the user name, server name, and port name be
locked
        // against user changes?
        window.contentPanel.lockSettings = lockSettings;

        // Show the window
        window.show();

        // Are we supposed to attempt an automatic connection?
        if (autoConnect)
           window.contentPanel.connect();
        else
           window.contentPanel.offline();

        // Is the user supposed to be placed in an initial chat room?
        if (!room.equals(""))
           if (window.contentPanel.client != null)
              try {
                 window.contentPanel.client.sendEnterRoom(room,
false,
                                                          "");
              }
              catch (IOException e) {
                 window.contentPanel.client.lostConnection();
                 return;
              }

        // Done
        return;
    }

    public void run()
    {
        // Nothing to do here.
        return;
    }
```

```
private void usage()
{
    System.out.println("\n" + strings.get(thisClass, "usage"));
    System.out.println("java mezochat [" +
                       usernameParam + " name] [" +
                       passwordParam + " password] [" +
                       servernameParam + " host] [" +
                       portnumberParam + " port] [" +
                       chatroomParam + " room] [" +
                       widthParam + " number] [" +
                       heightParam + " number] [" +
                       nopasswordsParam + "] [" +
                       locksettingsParam + "] [" +
                       autoconnectParam + "] [" +
                       hidecanvasParam + "]");
    return;
}

private boolean parseArgs(String[] args)
{
    // Loop through any command line arguments
    for (int count = 0; count < args.length; count ++)
      {
        if (args[count].equals(usernameParam))
          {
            if (++count < args.length)
              name = args[count];
          }

        else if (args[count].equals(passwordParam))
          {
            if (++count < args.length)
              password = args[count];
          }

        else if (args[count].equals(servernameParam))
          {
            if (++count < args.length)
              host = args[count];
          }

        else if (args[count].equals(portnumberParam))
          {
            if (++count < args.length)
              port = args[count];
          }
```

```
                else if (args[count].equals(chatroomParam))
                  {
                      if (++count < args.length)
                        room = args[count];
                  }

                else if (args[count].equals(widthParam))
                  {
                      if (++count < args.length)
                        windowWidth =
Integer.parseInt(args[count]);
                  }

                else if (args[count].equals(heightParam))
                  {
                      if (++count < args.length)
                        windowHeight =
Integer.parseInt(args[count]);
                  }

                else if (args[count].equals(nopasswordsParam))
                  requirePasswords = false;

                else if (args[count].equals(locksettingsParam))
                  lockSettings = true;

                else if (args[count].equals(autoconnectParam))
                  autoConnect = true;

                else if (args[count].equals(hidecanvasParam))
                  showCanvas = false;

                else if (args[count].equals("-help"))
                  {
                      usage();
                      return (false);
                  }

                else
                  {
                      System.out.println("\n" + strings.get(thisClass,
                                                          "unknownarg")
+
                                              " " + args[count]);
```

```
                                System.out.println(strings.get(thisClass,
            "forusage"));

                                return (false);
                            }
                        }

                    return (true);
                }

                public static void main(String[] args)
                {
                    mezochat firstInstance = new mezochat(args);
                    firstInstance.run();
                    return;
                }
            }



    //



//
//  mezochatserver.java
//

import java.net.*;
import java.util.*;
import java.text.*;
import java.io.*;


class mezochatServerShutdown
   extends Thread
{
   // This gets called when the VM is shutting down.  If the server has
   // already terminated properly (for example, from the administrator
   // pushing the 'shut down' button), then there's nothing to do.  But,
   // if the administrator sends a KILL signal, or presses CTRL-C or
   // something like that, then this thread will shut down the server
   // properly

   private mezochatServer server;
```

```
  public mezochatServerShutdown(mezochatServer s)
  {
   server = s;
  }

  public void run()
  {
   if (server.stop)
      // The server has already shut down by itself (i.e. not
      // from an external signal)
      return;

   server.externalShutdown = true;
   server.shutdown();
  }
}

public class mezochatServer
  extends Thread
{
  protected static String userPasswordFileName = "User.passwords";
  protected static String serverLogName = "Server.log";
  protected static String messageFileName = "Messages.saved";
  protected static String welcomeMessageName = "WELCOME.TXT";

  protected static String portnumberParam = "-portnumber";
  protected static String usepasswordsParam = "-usepasswords";
  protected static String newusersParam = "-newusers";
  protected static String nographicsParam = "-nographics";
  protected static String chatlogsParam = "-chatlogs";
  protected static String bugreportsParam = "-bugreports";

  protected static int DEFAULTPORT = 12468;

  protected URL myURL;
  protected int port = DEFAULTPORT;
  protected mezochatServerWindow window;
  protected boolean graphics = true;
  protected boolean requirePasswords = false;
  protected boolean allowNewUsers = true;
  protected boolean logChats = false;
  protected boolean autoBugReports = false;
  private Integer userIdCounter = new Integer(1);

  protected ServerSocket myServerSocket;
  protected Socket mySocket;
```

```
protected ThreadGroup myThreadGroup;

protected Vector connections;
public Vector messages;
public mezochatChatRoom mainRoom;
public Vector chatRooms;

private FileOutputStream log;
protected mezochatPasswordEncryptor passwordEncryptor;

protected mezochatClientSocket administratorClient;

protected int currentConnections = 0;
protected int peakConnections = 0;
protected int totalConnections = 0;

protected boolean stop = false;
protected boolean extemalShutdown = false;

// For managing strings according to locale
protected mezochatStringManager strings = null;
private Class thisClass = mezochatServer.class;

protected SimpleDateFormat dateFormatter =
 new SimpleDateFormat("MMM dd, yyyy hh:mm a");


public mezochatServer(String[] args)
{
 super("mezochat Chat server");

 // Get a URL to describe the invocation directory
 try {
    myURL = new URL("file", "localhost", "./");
 }
 catch (Exception E) {
    System.out.println(E);
    System.exit(1);
 }

 strings = new mezochatStringManager(myURL,
                                     Locale.getDefault().getLanguage());

 // Parse our args.  Only continue if successful
 if (!parseArgs(args))
    System.exit(1);
```

```
// Start up the log file
try {
   File logFile = new File(serverLogName);
   log = new FileOutputStream(logFile);
}
catch (IOException e) {
   // Oops, no log file
   serverOutput(strings.get(thisClass, "noopen") + " " +
                serverLogName + "\n");
}

// if user wants graphics, set up simple window

if (graphics)
   {
        window =
           new mezochatServerWindow(this,
                                    strings.get(thisClass,
                                                "mezochatversion") +
                                    mezochat.VERSION);
        window.setSize(400, 400);
        window.setVisible(true);
   }

else
   {
        System.out.println("\n" + strings.get(thisClass,
                                               "serverstatus"));
        System.out.println(strings.get(thisClass, "listenport") +
                           " " + port);
        System.out.println(strings.get(thisClass, "connections"));
   }

try {
   myServerSocket = new ServerSocket(port);
}
catch (IOException e) {
   System.out.println("\n" + strings.get(thisClass,
                                         "noserversocket"));
   System.exit(1);
}

connections = new Vector();
messages = new Vector();
chatRooms = new Vector();
```

```
myThreadGroup = new ThreadGroup("Clients");

// Set up the object for encrypting passwords
passwordEncryptor = new mezochatPasswordEncryptor();

if (requirePasswords)
   // Try to make sure the password file exists
   try {
        new FileOutputStream(userPasswordFileName, true).close();
   }
   catch (IOException f) {}

// Create the initial 'main' chat room
mainRoom = new mezochatChatRoom("Main", "Administrator", false, null);
try {
   mainRoom.setLogging(logChats);
}
catch (IOException e) {
   serverOutput(strings.get(thisClass, "noroomlog") + " " +
                mainRoom.name + "\n");
}

serverOutput(strings.get(thisClass, "readingmessages") + "\n");
readMessages();

serverOutput(strings.get(thisClass, "waiting") + "\n");

// To catch shutdown events, such as CTRL-C.  Note that this Java
// feature was only introduced as of 1.3, so if you are getting
// compilation errors, you should check your Javac version or comment
// out this bit.
double javaVersion = 0.0;
try {
   javaVersion = new Double(System.getProperty("java.version")
                            .substring(0, 3)).doubleValue();
}
catch (NumberFormatException e) {}

// If you get compile errors here, you can upgrade to JDK 1.3
// or delete the following 2 lines
if (javaVersion >= 1.3)
   Runtime.getRuntime()
        .addShutdownHook(new mezochatServerShutdown(this));

start();
}
```

```
public boolean checkPassword(String fileName, String userName,
                             String password)
 throws Exception
{
 // Return true if the supplied user name/password combo match

 DataInputStream passwordStream = null;

 // Try to find the user name/password combo in the password file
 try {

    passwordStream =
        new DataInputStream(new FileInputStream(fileName));

    // Read entry by entry.
    while(true)
        {
           String tempUserName = "";
           String tempPassword = "";
           try {
                tempUserName = passwordStream.readUTF();
                tempPassword = passwordStream.readUTF();
           }
           catch (EOFException e) {
                // Reached the end of the file, no match
                throw new Exception(strings.get(thisClass,
                                                "nosuchuser"));
           }

           // Do the user name and password match?
           if (tempUserName.equals(userName))
                {
                  if (tempPassword.equals(password))
                       return (true);
                  else
                       // The user name exists, but the password
                       // doesn't match
                       return (false);
                }
        }
 }
 catch (IOException e) {
    serverOutput(strings.get(thisClass, "errorpasswordfile") +
                 " " + e.toString() + "\n");
    return (false);
```

```
 }
}

protected mezochatChatRoom findChatRoom(String roomName)
{
 // This will return the chat room object with the corresponding
 // name

 mezochatChatRoom tempRoom = null;
 mezochatChatRoom returnRoom = null;

 // Is it the main chat room?
 if (mainRoom.name.equals(roomName))
    returnRoom = mainRoom;

 else
    {
        for (int count = 0; count < chatRooms.size(); count ++)
           {
               tempRoom = (mezochatChatRoom)
                  chatRooms.elementAt(count);

               if (tempRoom.name.equals(roomName))
                  {
                      returnRoom = tempRoom;
                      break;
                  }
           }
    }

 return (returnRoom);
}

public boolean isUserAllowed(mezochatChatRoom room,
                    mezochatClientSocket client, String password)
{
 // Return true if a user is allowed to enter a chat room.  False
 // otherwise.

 // Private room?
 if (room.priv)
    {
        boolean invited = false;

        // Make sure that this user was either invited, or supplied
        // the correct password.  Check the password first
```

```
        if (!room.password.equals(password))
            {
                // No correct password supplied.  Check the list
                // of invitees

                for (int count2 = 0;
                    count2 < room.invitedUsers.size(); count2 ++)
                    {
                        Integer Id = (Integer) room.invitedUsers
                            .elementAt(count2);

                        if (Id.intValue() == client.user.id)
                            {
                                invited = true;
                                break;
                            }
                    }

                if (!invited)
                    {
                        try {
                            client.sendServerMessage(
                             client.strings.get(thisClass,
                                                "notallowed") +
                             " " + room.name);
                        }
                        catch (IOException e) {
                            disconnect(client, false);
                            return (false);
                        }
                        return (false);
                    }
            }
    }

// Make sure the user has not been banned from the room
for (int count1 = 0; count1 < room.bannedUserNames.size(); count1 ++)
    {
        if (room.bannedUserNames.elementAt(count1)
            .equals(client.user.name))
            {
                // This user has been banned from this room.  Send
                // them a message and quit
                try {
                    client.sendBanUser(client.user.id, room.name);
```

```
                }
                catch (IOException e) {
                   disconnect(client, false);
                   return (false);
                }
                return (false);
             }
    }

 // The user is allowed
 return (true);
}

public synchronized void disconnect(mezochatClientSocket who,
                                    boolean notify)
{
 int count;
 mezochatChatRoom chatRoom;

 if (notify)
    {
         try {
            // Try to let the user know they're being disconnected
            who.sendDisconnect(who.user.id, strings
                               .get(thisClass, "disconnected"));
         }
         catch (IOException e) {}
    }

 // Shut down the client socket
 who.shutdown();

 // Remove the user from their chat room.
 try {
    who.leaveChatRoom();
 }
 catch (IOException e) {}

 // Remove the user from our list of connections
 synchronized (connections)
    {
         connections.removeElement(who);
         connections.trimToSize();
         currentConnections = connections.size();

         // Tell all the other clients to ditch this user
```

```
        for (count = 0; count < currentConnections; count ++)
          {
              mezochatClientSocket other = (mezochatClientSocket)
                connections.elementAt(count);
              try {
                other.sendDisconnect(who.user.id, "");
              }
              catch (IOException e) {}
          }
    }

serverOutput(strings.get(thisClass, "user") + " " + who.user.name +
          " " + strings.get(thisClass, "disconnectedat") + " " +
          dateFormatter.format(new Date()) + "\n");

serverOutput(strings.get(thisClass, "thereare") + " " +
          currentConnections + " " +
          strings.get(thisClass, "usersconnected") + "\n");

try {
   sleep(250);
}
catch (InterruptedException l) {}

if (graphics)
   {
        synchronized (window.userList) {
          // Remove the user name from the list widget.  We do this
          // loop to make sure it hasn't already been removed,
          // since disconnect() can get called multiple times for
          // one disconnection.
          for (count = 0; count < window.userList.getItemCount();
                count ++)
              {
                if (window.userList.getItem(
                              count).equals(who.user.name))
                    {
                      window.userList.remove(who.user.name);
                      break;
                    }
              }
        }

        window.updateStats();

        window.disconnectAll.setEnabled(currentConnections > 0);
```

```
        synchronized (window.userList)
          {
              if ((currentConnections <= 0) ||
                 window.userList.getSelectedItem() == null)
                 window.disconnect.setEnabled(false);
          }
    }
 return;
}


public synchronized void disconnectAll(boolean notify)
{
 int count;

 synchronized (connections)
    {
        // Loop backwards through all of the current connections
        for(count = (currentConnections - 1); count >= 0; count --)
          {
              mezochatClientSocket temp = (mezochatClientSocket)
                 connections.elementAt(count);

              if (temp == null)
                 continue;

              disconnect(temp, notify);
          }
    }

 return;
}


 public void run()
 {
  while (!stop)
     {
         try {
            mySocket = myServerSocket.accept();
         }
         catch (IOException e) {
            serverOutput(strings.get(thisClass, "socketerror") +
                        "\n");
            try {
                myServerSocket.close();
```

```
    }
    catch (IOException f) {
        serverOutput(strings.get(thisClass,
                                 "closesocketerror") + "\n");
    }
    System.exit(1);
}

if (mySocket == null)
    {
        serverOutput(strings.get(thisClass, "nullsocket") +
                     "\n");
        try {
           myServerSocket.close();
        }
        catch (IOException g) {
           serverOutput(strings.get(thisClass,
                                    "closesocketerror") +
                        "\n");
        }
        System.exit(1);
    }

// Now, this might be an automatic reconnection attempt
// by a client who was accidentally disconnected.  Check
// to see whether we already have a client from this
// InetAddress
boolean reconnection = false;
String tmpAddr = mySocket.getInetAddress().getHostAddress();
synchronized (connections)
    {
        for (int count = 0; count < connections.size();
             count ++)
            {
                mezochatClientSocket tmpClient =
                   (mezochatClientSocket)
                   connections.elementAt(count);

                if (!tmpClient.online &&
                   tmpAddr.equals(tmpClient.mySocket
                                  .getInetAddress()
                                  .getHostAddress()))
                   {
                       // The client already exists.  Just
                       // give it the new socket and
                       // wake it up
```

```
                                acceptReconnect(tmpClient);
                                reconnection = true;
                                break;
                            }
                        }
                    }

                if (!reconnection)
                    // New connection
                    new mezochatClientSocket(this, mySocket, myThreadGroup);
            }
        }

        protected void acceptReconnect(mezochatClientSocket client)
        {
         try {
            client.mySocket.close();
         }
         catch (IOException e) {}

         client.mySocket = mySocket;

         // Get new input and output streams
         try {
            client.istream =
                new DataInputStream(client.mySocket.getInputStream());
            client.ostream =
                new DataOutputStream(client.mySocket.getOutputStream());
            client.interrupt();
         }
         catch (IOException a) {}
        }

        protected int getUserId()
        {
         // Returns a number to be used as a user Id

         int tmp;

         synchronized (userIdCounter)
            {
                tmp = userIdCounter.intValue();
                userIdCounter = new Integer(tmp + 1);
            }

         return (tmp);
```

```
    }

    protected void createNewUser(String userName, String
encryptedPassword)
     throws Exception
    {
     // Create a new user acount.
     new mezochatUserTool().createUser(userName, encryptedPassword);
    }

    protected void serverOutput(String message)
    {
     if (graphics)
        window.logWindow.append(message);
     else
        System.out.print(message);

     // Write it to the log file
     if (log != null)
        {
            try {
               byte[] messagebytes = message.getBytes();
               log.write(messagebytes);
            }
            catch (IOException F) {
               if (graphics)
                    window.logWindow
                       .append(strings.get(thisClass, "writelogerror") +
                              "\n");
               else
                    System.out.print(strings.get(thisClass,
                                          "writelogerror") + "\n");
            }
        }

     return;
    }

    protected void readMessages()
    {
     String tempFor = "";
     String tempFrom = "";
     String tempMessage = "";

     DataInputStream messageStream = null;
```

```
    try {
       messageStream =
            new DataInputStream(new FileInputStream(messageFileName));

       // Read entry by entry.
       while(true)
            {
               try {
                    tempFor = messageStream.readUTF();
                    tempFrom = messageStream.readUTF();
                    tempMessage = messageStream.readUTF();
               }
               catch (EOFException e) {
                    // Reached the end of the file
                    break;
               }

               messages.addElement(new mezochatMessage(tempFor,
tempFrom,
                                                       tempMessage));
            }

         messageStream.close();
      }
      catch (IOException E) {}

      return;
    }

    protected void saveMessages()
    {
      DataOutputStream messageStream = null;

      try {
         messageStream =
              new DataOutputStream(new
 FileOutputStream(messageFileName));

         for (int count = 0; count < messages.size(); count ++)
              {
                mezochatMessage tempMessage =
                     (mezochatMessage) messages.elementAt(count);

                messageStream.writeUTF(tempMessage.messageFor);
                messageStream.writeUTF(tempMessage.messageFrom);
                messageStream.writeUTF(tempMessage.text);
```

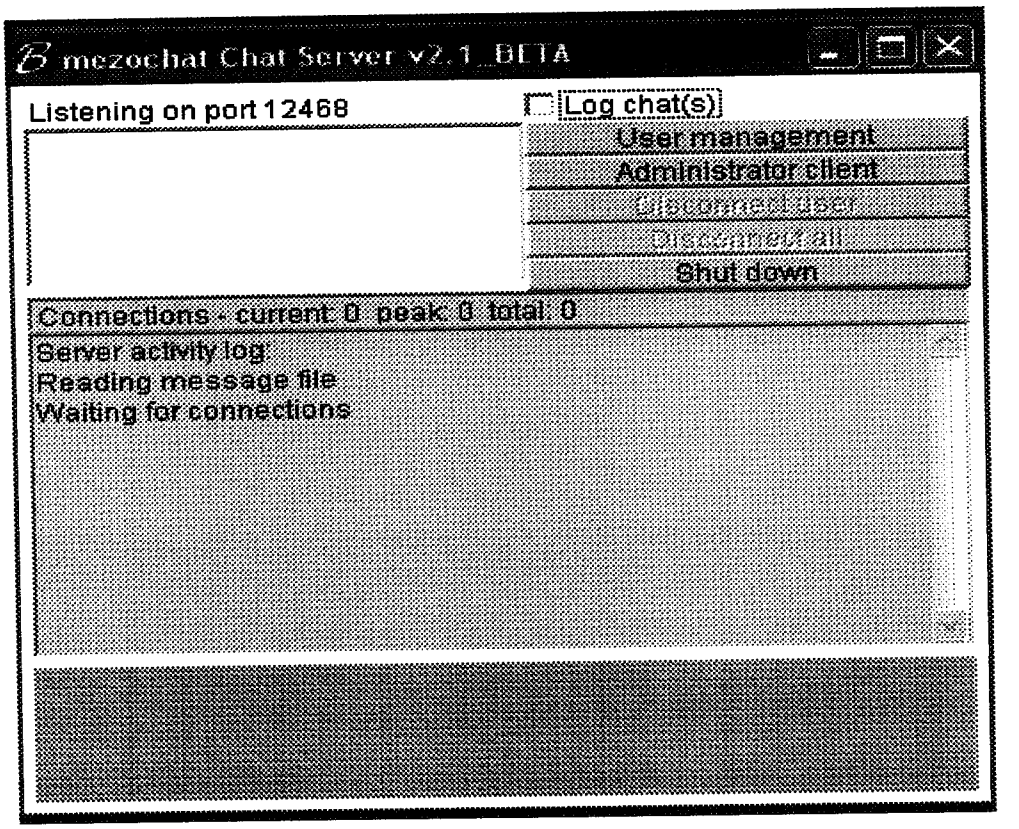
```
        }

    messageStream.close();
 }
 catch (IOException E) {
    serverOutput(strings.get(thisClass, "writemsgerror") + "\n");
 }

 return;
}

protected void shutdown()
{
 serverOutput(strings.get(thisClass, "shutting") + "\n");

 if (currentConnections > 0)
    {
        serverOutput(strings.get(thisClass, "disconnecting") + "\n");

        synchronized (connections)
           {
               // Loop through all of the users, sending them the
               // message that the server is shutting down
               for (int count = 0; count < currentConnections;
                   count ++)
                  {
                      mezochatClientSocket who =
                         (mezochatClientSocket)
                         connections.elementAt(count);

                      try {
                         who.sendDisconnect(who.user.id,
                                  who.strings.get(thisClass,
                                          "shuttinggoodbye"));
                      }
                      catch (IOException e) {}
                  }
           }

        // Make sure
        disconnectAll(true);
    }
  else
     serverOutput(strings.get(thisClass, "nousers") + "\n");

  // Print some stats
```

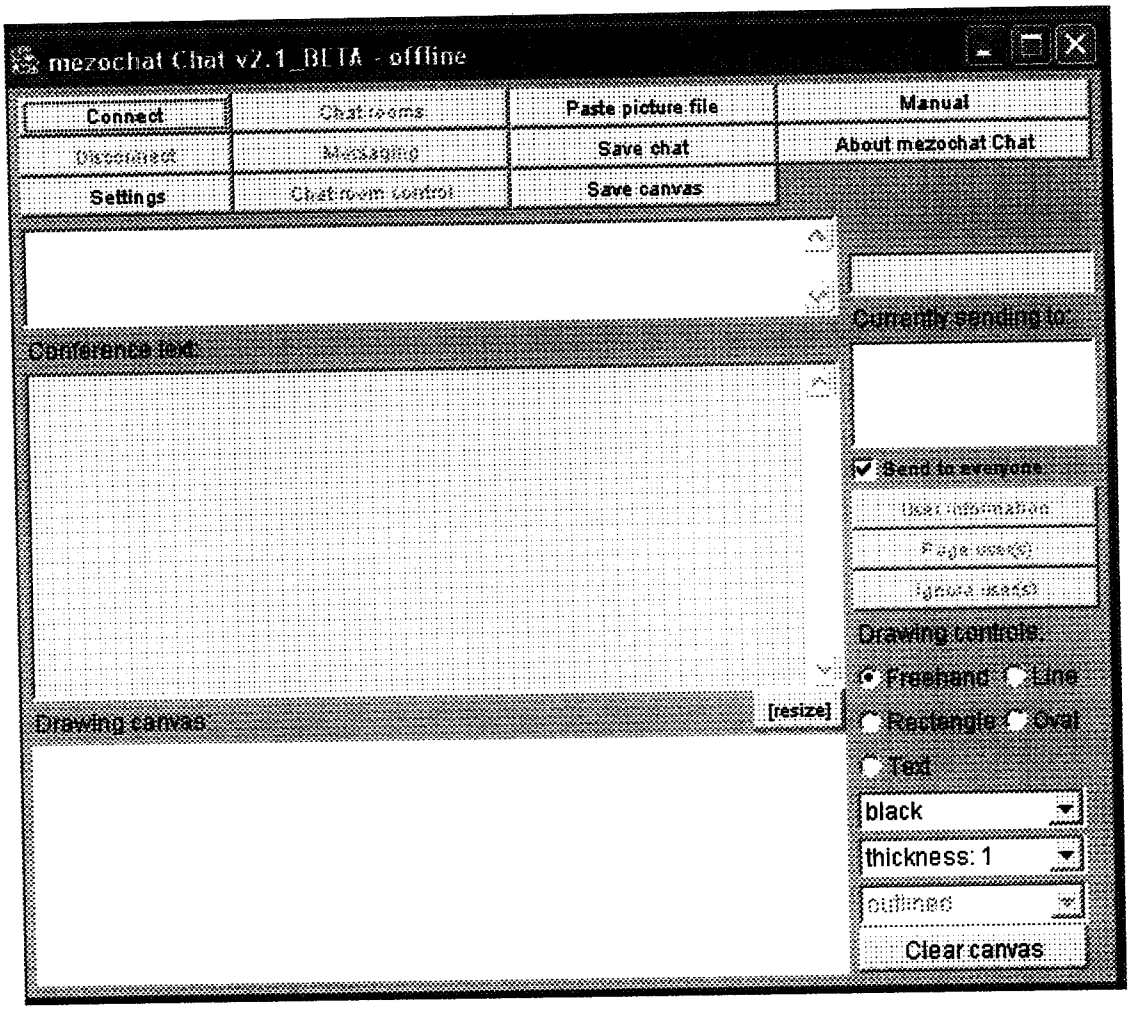
```
serverOutput(strings.get(thisClass, "peakconn") + " " +
            peakConnections + "\n");
serverOutput(strings.get(thisClass, "totalconn") + " " +
            totalConnections + "\n");

serverOutput(strings.get(thisClass, "savingmsg") + "\n");
saveMessages();

serverOutput(strings.get(thisClass, "closinglog") + "\n");
try {
   log.close();
}
catch (IOException F) {
   serverOutput(strings.get(thisClass, "closelogerror") + "\n");
}

if (graphics)
   window.dispose();

stop = true;

// If we aren't using the GUI window, we should provide some
// visual feedback that the server has terminated.
if (!graphics)
   {
        System.out.println("");
        System.out.println(strings.get(thisClass,
                                       "shutdowncomplete"));
   }

// This function can be called by the mezochatServerShutdown thread
// when the server gets killed by an external signal.  If so, it
// sets the externalShutdown flag, and we shouldn't call the
// System.exit() function
if (!externalShutdown)
    System.exit(0);
}

private void usage()
{
 System.out.println("\n" + strings.get(thisClass, "usage"));
 System.out.println("java mezochatServer [" +
                    portnumberParam + " number] [" +
                    usepasswordsParam + "] [" +
                    newusersParam + "] [" +
                    nographicsParam + "] [" +
```

```
                    chatlogsParam + "] [" +
                    bugreportsParam + "]");
 return;
}

private boolean parseArgs(String[] args)
{
 // Parse the arguments
 for (int count = 0; count < args.length; count ++)
    {
        if (args[count].equals(portnumberParam))
          {
              if (++count < args.length)
                {
                    try {
                       port = Integer.parseInt(args[count]);
                    }
                    catch (Exception E) {
                       System.out.println("\n" +
                         strings.get(thisClass, "illegalport") +
                         " " + args[count]);
                       System.out.println(strings
                                    .get(thisClass,
                                         "forusage"));
                       return (false);
                    }
                }
          }

        else if (args[count].equals(usepasswordsParam))
          requirePasswords = true;

        else if (args[count].equals(newusersParam))
          allowNewUsers = true;

        else if (args[count].equals(nographicsParam))
          graphics = false;

        else if (args[count].equals(chatlogsParam))
          logChats = true;

        else if (args[count].equals(bugreportsParam))
          autoBugReports = true;

        else if (args[count].equals("-help"))
          {
```

```
                usage();
                return (false);
            }

        else
            {
                System.out.println("\n" +
                        strings.get(thisClass, "unknownarg") +
                        " " + args[count]);
                System.out.println(strings.get(thisClass,
                                        "forusage"));
                return (false);
            }
        }

    return (true);
    }

    public static void main(String[] args)
    {
     // Start the server
     new mezochatServer(args);
     return;
    }
}
```

9.2) Sample Output:




mezochat Chat v2.1_BETA - offline
Connect
Chatrooms
Paste picture file
Manual
Disconnect
Messaging
Save chat
About mezochat Chat
Settings
Chatroom control
Save canvas
Conference text:
Currently sending to:
Send to everyone
[resize]
Drawing canvas
Drawing controls:
Freehand
Line
Rectangle
Oval
Text
black
thickness: 1
Clear canvas