



Jaypee University of Information Technology
Solan (H.P.)
LEARNING RESOURCE CENTER

Acc. Num. *SP07016* Call Num:

General Guidelines:

- ◆ Library books should be used with great care.
- ◆ Tearing, folding, cutting of library books or making any marks on them is not permitted and shall lead to disciplinary action.
- ◆ Any defect noticed at the time of borrowing books must be brought to the library staff immediately. Otherwise the borrower may be required to replace the book by a new copy.
- ◆ The loss of LRC book(s) must be immediately brought to the notice of the Librarian in writing.

Learning Resource Centre-JUIT



SP07016

Fault Tolerant Voice Enabled Chat **Application**

SAARTHAK GUPTA (071240)

MONISH KAUL (071250)

UNDER THE SUPERVISION OF

MR. AMOL VASUDEVA



Submitted in partial fulfilment of the Degree of
Bachelor of Technology

DEPARTMENT OF CSE & IT

JAYPEE UNIVERSITY OF INFORMATION TECHNOLOGY,
WAKNAGHAT



TABLE OF CONTENTS

Chapter No.	Topics	Page No.
	Certificate from the Supervisor	II
	Acknowledgement	III
	Summary	IV
	List of Figures	V
	List of Symbols and acronyms	VI
Chapter 1	Introduction	1
Chapter 2	Types Of Client Server Architectures	6
Chapter 3	Introduction to Netbeans	15
Chapter 4	Sample Codes	30
Chapter 5	Testing	61
Chapter 6	Conclusion	65
References		

CERTIFICATE

This is to certify that the work entitled **"Fault Tolerant Voice Enabled Chat Application"** submitted by **"Saarthak Gupta and Monish Kaul"** in partial fulfilment for the award of degree of B.Tech of Jaypee University of Information Technology; Work has been carried out under my supervision. This work has not been submitted partially or wholly to any other University or Institute for the award of this or any other degree or diploma.

Signature of Supervisor 

Name of Supervisor 

Designation

Date

ACKNOWLEDGEMENT

No venture can be completed without the blessing of Almighty .We consider it our bounded duty to bow to Almighty whose kind blessings always inspire us on the right path of life This project is our combined effort and realizes the potential of team and gives us a chance to work in co-ordination.

Science has caused many frontiers so has human efforts towards human research. Our revered guide Mr Amol Vasudeva, Lecturer, Department of Computer Science and IT, JUIT, has indeed acted as a light house showing us the need of sustained effort in the field of Image Processing to learn more and more. So we take this opportunity to thank him, for lending us stimulating suggestions, innovative quality guidance and creative thinking. He provides us the kind of strategies required for the completion of a task.

Signature:

Name:

Date:

Signature:

Name:

Date:

SUMMARY

The project implements a full-featured chat application in the form of a fault tolerant and voice enabled system. The client and server services operate over a modularized WINDOWS OS network architecture. In addition, this application infers the occurrence of client communication with the connected peers and the server itself via Ethernet Lan and VOIP. It thus provides a network with an advanced functionality of communication which is both user friendly and hassle free. The rules implemented allow efficient communication and broaden the scope of a simple Ethernet connection by enabling transfer of voice data packets over it.

Signature:

Name:

Date:

Signature:

Name:

Date:

LIST OF FIGURES

Figure Number	Figure Name	Page NO.
Fig 1	Client Server Architecture	5
Fig 2	Two tier Architecture	6
Fig 3	Three tier Architecture	7
Fig 4	N Tier Architecture	9
Fig 5	Netbeans New Project	15
Fig 6	Netbeans New project II	16
Fig 7	Netbeans Main Project Window	17
Fig 8	Project Execution Window	18
Fig 9	Command Prompt Window	24
Fig 10	Apache Ant Home Screen	27

LIST OF ABBREVIATIONS AND SYMBOLS

TCP: Transfer Control Protocol

UDP: User Datagram Protocol

CHAPTER 1

1.1 INTRODUCTION TO NETWORKING :

A network is simply a collection of computers or other hardware devices that are connected together, either physically or logically, using special hardware and software, to allow them to exchange information and cooperate. Networking is the term that describes the processes involved in designing, implementing, upgrading, managing and otherwise working with networks and network technologies.

Networks are used for an incredible array of different purposes. In fact, the definitions above are so simple for the specific reason that networks can be used so broadly, and can allow such a wide variety of tasks to be accomplished.

The widespread networking of personal computers is a relatively new phenomenon. For the first decade or so of their existence, PCs were very much “islands unto themselves”, and were rarely connected together. In the early 1990s, PC networking began to grow in popularity as businesses realized the advantages that networking could provide. By the late 1990s, networking in homes with two or more PCs started to really take off as well.

This interconnection of small devices represents, in a way, a return to the “good old days” of mainframe computers. Before computers were small and personal, they were large and centralized machines that were shared by many users operating remote terminals. While having the entire computer power in one place had many disadvantages, one benefit was that all users were connected because they shared the central computer.

Individualized PCs took away that advantage, in favor of the benefits of independence. Networking attempts to move computing into the middle ground, providing PC users with the best of both worlds: the independence and flexibility of personal computers, and the connectivity and resource sharing of mainframes. In fact, networking is today considered so vital that it's hard to conceive of an organization with two or more computers that would not want to connect them together.

The advantages of networking include:

➤ **Connectivity and Communication:**

Networks connect computers and the users of those computers. Individuals within a building or work group can be connected into *local area networks (LANs)*; LANs in distant locations can be interconnected into larger *wide area networks (WANs)*. Once connected, it is possible for network users to communicate with each other using technologies such as electronic mail. This makes the transmission of business (or non-business) information easier, more efficient and less expensive than it would be without the network.

➤ **Data Sharing:**

One of the most important uses of networking is to allow the sharing of data. Before networking was common, an accounting employee who wanted to prepare a report for her manager would have to produce it on his PC, put it on a floppy disk, and then walk it over to the manager, who would transfer the data to her PC's hard disk.

True networking allows thousands of employees to share data much more easily and quickly than this. More so, it makes possible applications that rely on the ability of many people to access and share the same data, such as databases, group software development, and much more. Intranets and extranets can be used to distribute corporate information between sites and to business partners.

➤ **Hardware Sharing:**

Networks facilitate the sharing of hardware devices. For example, instead of giving each of 10 employees in a department an expensive color printer one printer can be placed on the network for everyone to share.

➤ **Internet Access:**

The Internet is itself an enormous network, so whenever you access the Internet, you are using a network. The significance of the Internet on modern society is hard to exaggerate, especially for those of us in technical fields.

➤ **Internet Access Sharing:**

Small computer networks allow multiple users to share a single Internet connection. Special hardware devices allow the bandwidth of the connection to be easily allocated to various individuals as they need it, and permit an organization to purchase one high-speed connection instead of many slower ones.

➤ **Data Security and Management:**

In a business environment, a network allows the administrators to much better manage the company's critical data. Instead of having this data spread over dozens or even hundreds of small computers in a haphazard fashion as their users create it, data can be centralized on shared servers. This makes it easy for everyone to find the data, makes it possible for the administrators to ensure that the data is regularly backed up, and also allows for the implementation of security measures to control that can read or change various pieces of critical information.

➤ **Performance Enhancement and Balancing:**

Under some circumstances, a network can be used to enhance the overall performance of some applications by distributing the computation tasks to various computers on the network.

1.2 INTRODUCTION TO THE CLIENT SERVER MODEL:

Client/server model is a concept for describing communications between computing processes that are classified as service consumers (clients) and service providers (servers). The basic features of a C/S model are:

- Clients and servers are functional modules with well defined interfaces (i.e., they hide internal information). The functions performed by a client and a server can be implemented by a set of software modules, hardware components, or a combination thereof. Clients and/or servers may run on dedicated machines, if needed.
- Each client/server relationship is established between two functional modules when one module (client) initiates a service request and the other (server) chooses to respond to the service request. Examples of service requests (SRs) are "retrieve customer name," "produce net income in last year," etc. For a given service request, clients and servers do not reverse roles (i.e., a client stays a client and a server stays a server). However, a server for SR R1 may become a client for SR R2 when it issues requests to another. For example, a client may issue an SR that may generate other SRs.
- Information exchange between clients and servers is strictly through messages (i.e., no information is exchanged through global variables). The service request and additional information is placed into a message that is sent to the server. The server's response is similarly another message that is sent back to the client. This is an extremely crucial feature of C/S model.
- Messages exchanged are typically interactive. In other words, C/S model does not support an off-line process. There are a few exceptions. For example, message queuing systems allow clients to store messages on a queue to be picked up asynchronously by the servers at a later stage.
- Clients and servers typically reside on separate machines connected through a network. Conceptually, clients and servers may run on the same machine or on separate machines. In this book, however, our primary interest is in *distributed client/server systems* where clients and servers reside on separate machines.

The implication of the last two features is that C/S service requests are real-time messages that are exchanged through network services. This feature increases the appeal of the C/S model (i.e., flexibility, scalability) but introduces several technical issues such as portability, interoperability, security, and performance.

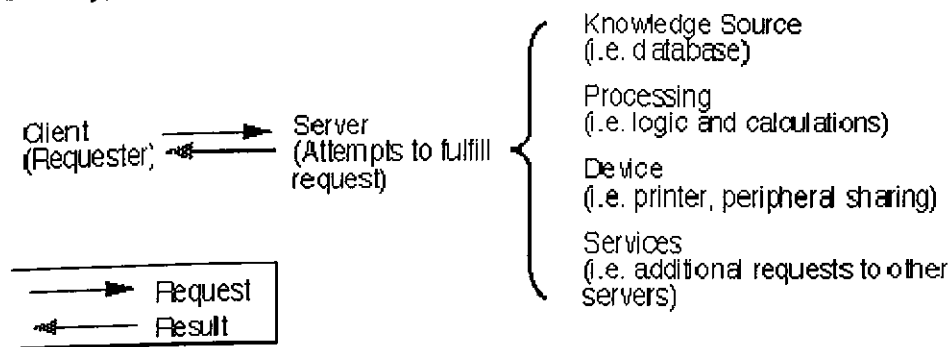


Fig 1: Client Server Communication

CHAPTER 2

2.1 Different Types of Client Server Architectures:

Firewalls fall into three broad categories: Two Tier Architecture, Three Tier Architecture and N Tier Architecture.

2.1.1 Two Tier Architecture:

In this implementation, the three components of an application (presentation, processing, and data) are divided among two software entities (tiers): client application code and database server.

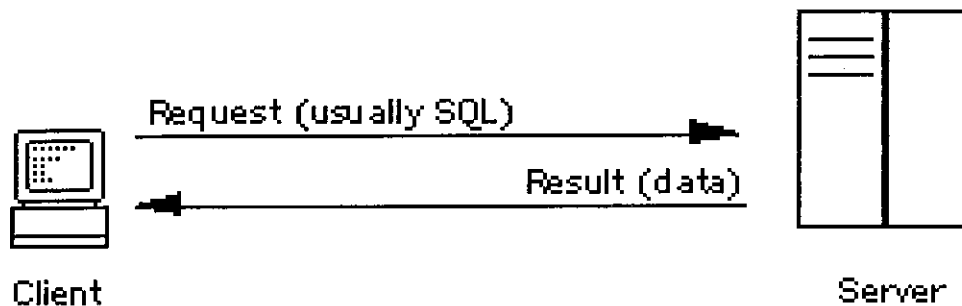


Figure 2. Two Tier Architecture

Presentation is handled exclusively by the client, processing is split between client and server, and data is stored on and accessed via the server. The PC client assumes the bulk of responsibility for application (functionality) logic with respect to the processing component, while the database engine - with its attendant integrity checks, query capabilities and central repository functions - handles data intensive tasks.

Two-tier architectures work well in relatively homogeneous environments with fairly static business rules. However this architecture is less suited for dispersed, heterogeneous environments with rapidly changing rules.

2.1.2 Three Tier Architecture:

The three tier architecture attempts to overcome some of the limitations of the two-tier scheme by separating presentation, processing, and data into separate, distinct software entities (tiers).

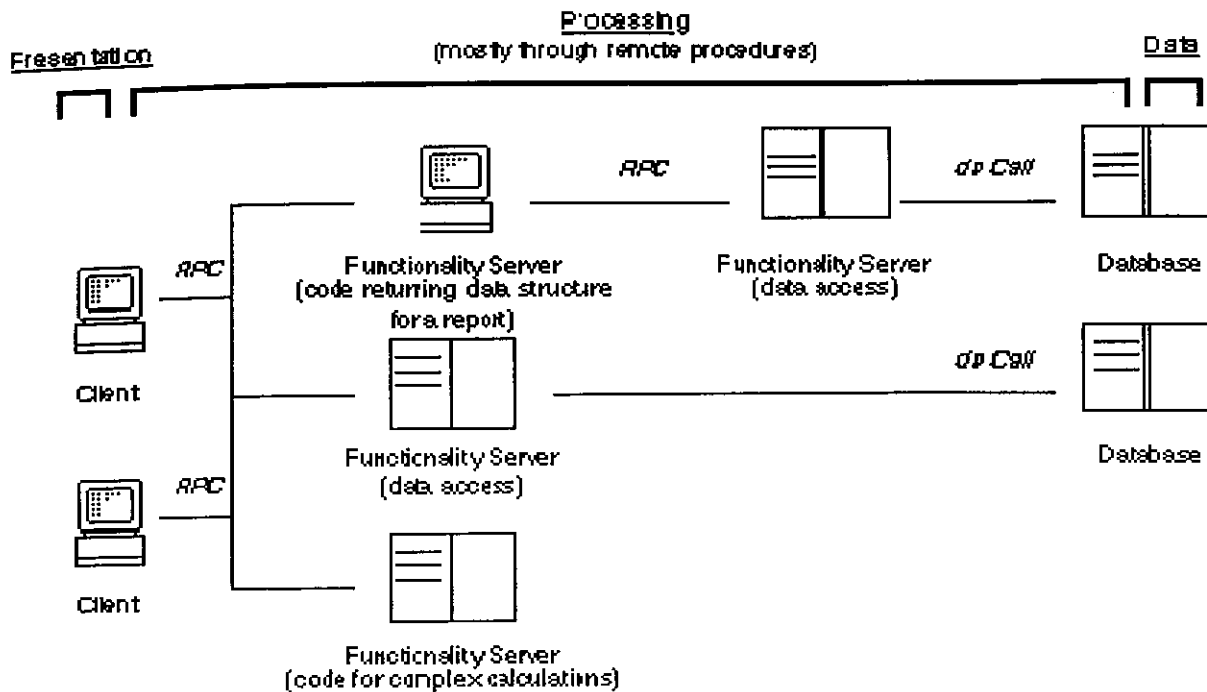


Figure 3. Three Tier Architecture

When calculations or data access is required by the presentation client, a call is made to a middle tier functionality server. This tier can perform calculations or can make requests as a client to additional servers. Middle-tier functionality servers may be multi-threaded and can be accessed by multiple clients, even those from separate applications.

The calling mechanism from client to server in such a system is most typically the remote procedure call or RPC. In an RPC, the requesting client passes the parameters needed for the request and specifies a data structure to accept returned values.

The three tier architecture provides for more flexible resource allocation. Middle-tier functionality servers are highly portable and can be dynamically allocated and shifted as the needs of the organization change. Network traffic can potentially be reduced by having functionality servers strip data to the precise structure required before distributing it to individual clients at the LAN level. Multiple server requests and complex data access can emanate from the middle tier instead of the client, further decreasing traffic.

2.1.3 N Tier Architecture :

Usually N-Tier Architecture begins as a 3-Tier model and is expanded. It provides finer granularity. Granularity is the ability of a system, in this case, an application, to be broken down into smaller components or granules. The finer the granularity, the greater the flexibility of a system. It can also be referred to as a system's modularity. Therefore, it refers to the pulling apart of an application into separate layers or finer grains. One of the best examples of N-Tier Architecture in web applications is the popular shopping-cart web application. The client tier interacts with the user through GUIs (Graphic User Interfaces) and with the application and the application server. In web applications, this client tier is a web browser. In addition to initiating the request, the web browser also receives and displays code in dynamic HTML (Hypertext Markup Language), the primary language of the World Wide Web. In a shopping cart web application, the presentation tier displays information related to such services as browsing merchandise, purchasing, and shopping cart contents. It communicates with other tiers by outputting results to the browser/client tier and all other tiers in the network. This layer calls custom tags throughout the network and to other networks. It also calls database stored procedures and web services, all in the goal of providing a more sophisticated response. This layer glues the whole application together and allows different nodes to communicate with each other and be displayed to the user through the browser. It is located in the application server. In N-Tier Architecture, the business logic tier is pulled out from the presentation tier and, as its own layer; it controls an application's functionality by performing detailed processing. For example, in our shopping cart example, this tier completes credit card authorization and calculates things like shipping costs and sales tax. The tools used to encapsulate an application's business logic into its own layer include web services, custom tags, and stored procedures. The business tier can also be considered the integration layer. Encapsulation allows the application to communicate with the data tier or the business logic tier in a way that is intelligible to all nodes. Encapsulation is one of the principles of object-oriented programming (OOP) and refers to an object's ability to conceal its data and methods. Encapsulated objects only publish the external interface so any user interacting with them only needs to understand the interface and can remain ignorant as to the internal specifications. This way a user can call all sorts of services into action by calling the custom tag without having to know the code details of what made communication or implementation possible. The services just have to be accessible in the custom tag library. Encapsulation in the integration tier also liberates the network from just one vendor. The integration tier allows N-Tier Architecture to be vendor independent. In the shopping cart example, the application may have a simple custom tag for searching inventory

and providing the most up-to-date, detailed information. The final application tier is the data tier. It usually consists of database servers. It keeps data neutral and independent from application servers or business logic. Giving data its own tier also improves scalability and performance. As it grows, it is easily moved to another, more powerful machine.

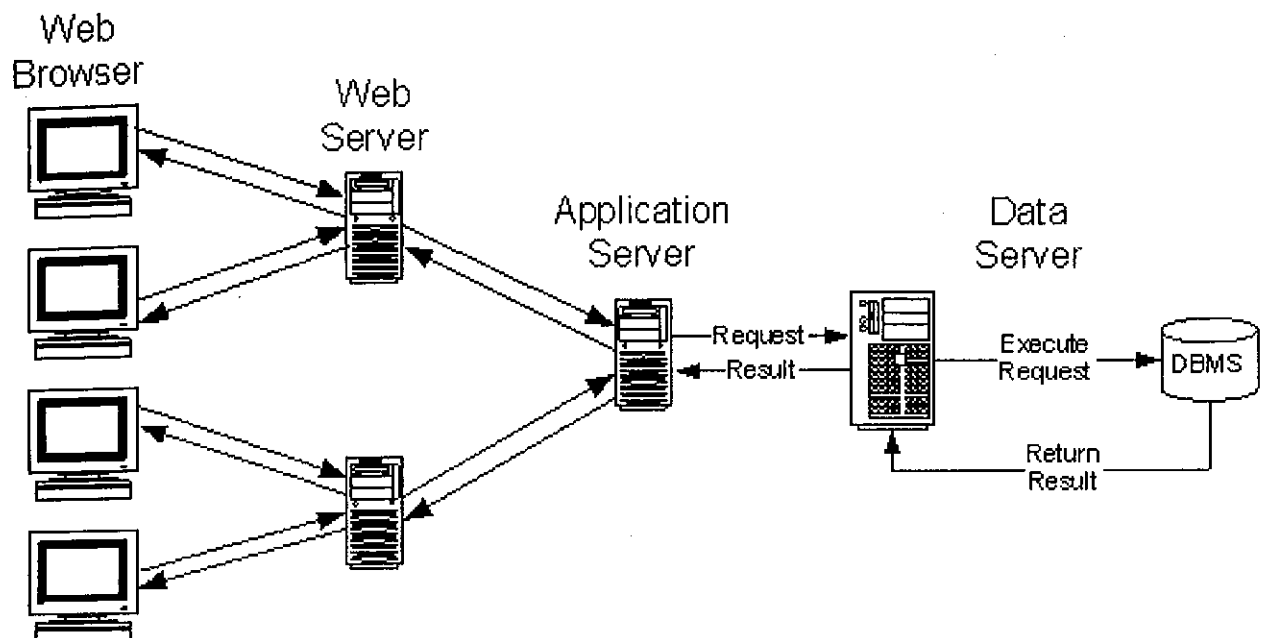


Fig 4: N Tier Architecture

2.1.3.1 Benefits of N Tier Architecture

There are many business benefits to N-Tier Architecture. For example, a small business can begin running all tiers on a single machine. As traffic and business increases, each tier can be expanded and moved to its own machine and then clustered. This is just one example of how N-Tier Architecture improves scalability and supports cost-efficient application building. N-Tier model also make applications more readable and reusable. It reduces the amount of spaghetti code. Custom tag libraries and EJBs are easier to port to readable applications in well-maintained templates. Reusability multiplies developer productivity and improves application maintainability. It is an important feature in web applications. N-Tier Architectures make application more robust because there is no single point of failure. Tiers function with relative independence. For example, if a business changes database vendors, they just have to replace the data tier and adjust the integration tier to any changes that affect it. The business logic tier and the presentation tier remain unchanged. Likewise, if the presentation layer changes, this

will not affect the integration or data layer. In 3-Tier Architecture all the layers exist in one and affect each other. A developer would have to pick through the entire application code to implement any changes. Again, well-designed modules allow for applications or pieces of applications to be customized and used across modules or even projects. Reusability is particularly important in web applications. As demonstrated N-Tier Architecture offers innovations in the standard Client-Server technology that spawned the Internet itself. It is but one of many web application frameworks. These are used to develop dynamic web sites, web applications or web services. They provide database access libraries, templates, and, as previously stated code re-use. Most web application frameworks follow the Model View Controller (MVC) which separate the user interface, the business rules and the data model. They provide authentication and authorization to provide security. This allows the web server to restrict user access based on pre-determined criteria. Web application frameworks also provide a unified API (Application programming Interface). This allows web application to work with various databases without requiring any code change. These frameworks also maintain a web template system. Finally, N-Tier Architecture helps developers build web applications because it allows developers to apply their specific skill to that part of the program that best suits their skill set. Graphic artists can focus on the presentation tier, while administrators can focus on the database tier.

2.1.4 Comparison with P2P model for Communication

Peer-to-peer networks are typically less secure than client-server networks because security is handled by the individual computers, not on the network as a whole. The resources of the computers in the network can become overburdened as they have to support not only the workstation user, but also the requests from network users. It is also difficult to provide system wide services because the desktop operating system typically used in this type of network is incapable of hosting the service. Where the client-server networks have a higher initial setup cost. It is possible to set up a server on a desktop computer, but it is recommended that businesses invest in enterprise-class hardware and software. They also require a greater level of expertise to configure and manage the server hardware and software.

Client Server Architecture:

1. A client-server network involves multiple clients connecting to a single, central server. The file server on a client-server network is a high capacity, high speed computer with a large hard disk capacity.
2. A client-server model works with any size or physical layout of LAN and doesn't tend to slow down with a heavy use.

Peer-to-Peer Model

1. Peer-to-peer networks involve two or more computers pooling individual resources such as disk drives, CD-ROMs and printers. These shared resources are available to every computer in the network. Each computer acts as both the client and the server which means all the computers on the network are equals, that is where the term peer-to-peer comes from.
2. In the peer to peer network, software applications can be installed on the single computer and shared by every computer in the network. They are also cheaper to set up because most desktop operating systems have the software required for the network installed by default.

2.2 Paradigms Used In Client Server Architectures

2.2.1 Remote Procedure Call (RPC).

In this paradigm, the client process invokes a remotely located procedure (a server process), the remote procedure executes and sends the response back to the client. The remote procedure can be simple (e.g., retrieve time of day) or complex (e.g., retrieve all customers from Chicago who have a good credit rating). Each request/response of an RPC is treated as a separate unit of work, thus each request must carry enough information needed by the server process. RPCs are supported widely at present.

2.2.2 Remote Data Access (RDA).

This paradigm allows client programs and/or end-user tools to issue ad hoc queries, usually SQL, against remotely located databases. The key technical difference between RDA and RPC is that in an RDA the size of the result is not known because the result of an SQL query could be one row or thousands of rows. RDA is heavily supported by database vendors.

2.2.3 Queued Message Processing (QMP)

In this paradigm, the client message is stored in a queue and the server works on it when free. The server stores ("puts") the response in another queue and the client actively retrieves ("gets") the responses from this queue. This model, used in many transaction processing systems, allows the clients to asynchronously send requests to the server. Once a request is queued, the request is processed even if the sender is disconnected (intentionally or due to a failure). QMP support is becoming commonly available.

Initial implementations of client/server architecture were based on the "two-tiered" architectures shown in Figure (a) through Figure (e) (these architectural configurations are known as the "Gartner Group" configurations). The first two architectures (Figure (a) and Figure (b) are used in many presentation intensive applications (e.g., XWindow, multimedia presentations) and to provide a "face lift" to legacy applications by building a GUI interface that invokes the older text-based user interfaces of legacy applications. Figure (c) represents the distributed application program architecture in which the application programs are split between the client and server machines, and they communicate with each other through the remote procedure call (RPC) or queued messaging middleware. Figure (d) represents the remote data architecture in which the remote data is typically stored in a "SQL server" and is accessed through ad hoc SQL statements sent over the network.

2.3 Protocols used in Client Server Architecture

2.3.1 UDP

The UDP protocol provides for communication that is not guaranteed between two applications on the network. UDP is not connection-based like TCP. Rather, it sends independent packets of data, called *datagram*, from one application to another. Sending datagram is much like sending a letter through the postal service: The order of delivery is not important and is not guaranteed, and each message is independent of any other.

For many applications, the guarantee of reliability is critical to the success of the transfer of information from one end of the connection to the other. However, other forms of communication don't require such strict standards. In fact, they may be slowed down by the extra overhead or the reliable connection may invalidate the service altogether.

Consider, for example, a clock server that sends the current time to its client when requested to do so. If the client misses a packet, it doesn't really make sense to resend it because the time will be incorrect when the client receives it on the second try. If the client makes two requests and receives packets from the server out of order, it doesn't really matter because the client can figure out that the packets are out of order and make another request. The reliability of TCP is unnecessary in this instance because it causes performance degradation and may hinder the usefulness of the service.

Another example of a service that doesn't need the guarantee of a reliable channel is the ping command. The purpose of the ping command is to test the communication between two programs over the network. In fact, ping needs to know about dropped or out-of-order packets to determine how good or bad the connection is. A reliable channel would invalidate this service altogether. The UDP protocol provides for communication that is not guaranteed between two applications on the network. UDP is not connection-based like TCP.

Rather, it sends independent packets of data from one application to another. Sending datagram is much like sending a letter through the mail service: The order of delivery is not important and is not guaranteed, and each message is independent of any others.

2.3.2 TCP

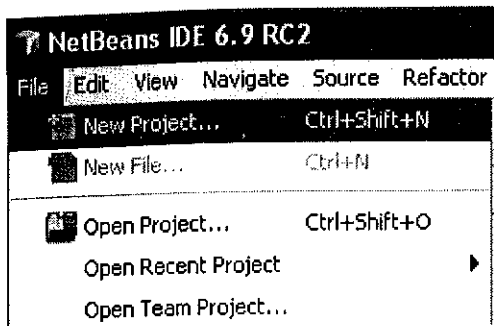
When two applications want to communicate to each other reliably, they establish a connection and send data back and forth over that connection. This is analogous to making a telephone call. If you want to speak to Aunt Beatrice in Kentucky, a connection is established when you dial her phone number and she answers. You send data back and forth over the connection by speaking to one another over the phone lines. Like the phone company, TCP guarantees that data sent from one end of the connection actually gets to the other end and in the same order it was sent. Otherwise, an error is reported. TCP provides a point-to-point channel for applications that require reliable communications. The Hypertext Transfer Protocol (HTTP), File Transfer Protocol (FTP), and Telnet are all examples of applications that require a reliable communication channel. The order in which the data is sent and received over the network is critical to the success of these applications. When HTTP is used to read from a URL, the data must be received in the order in which it was sent.

CHAPTER 3

3.1 Setting up the Project

To create an IDE project:

1. Start NetBeans IDE.
2. In the IDE, choose File > New Project (Ctrl-Shift-N), as shown in the figure below.



3. In the New Project wizard, expand the Java category and select Java Application as shown in the figure below. Then click Next.

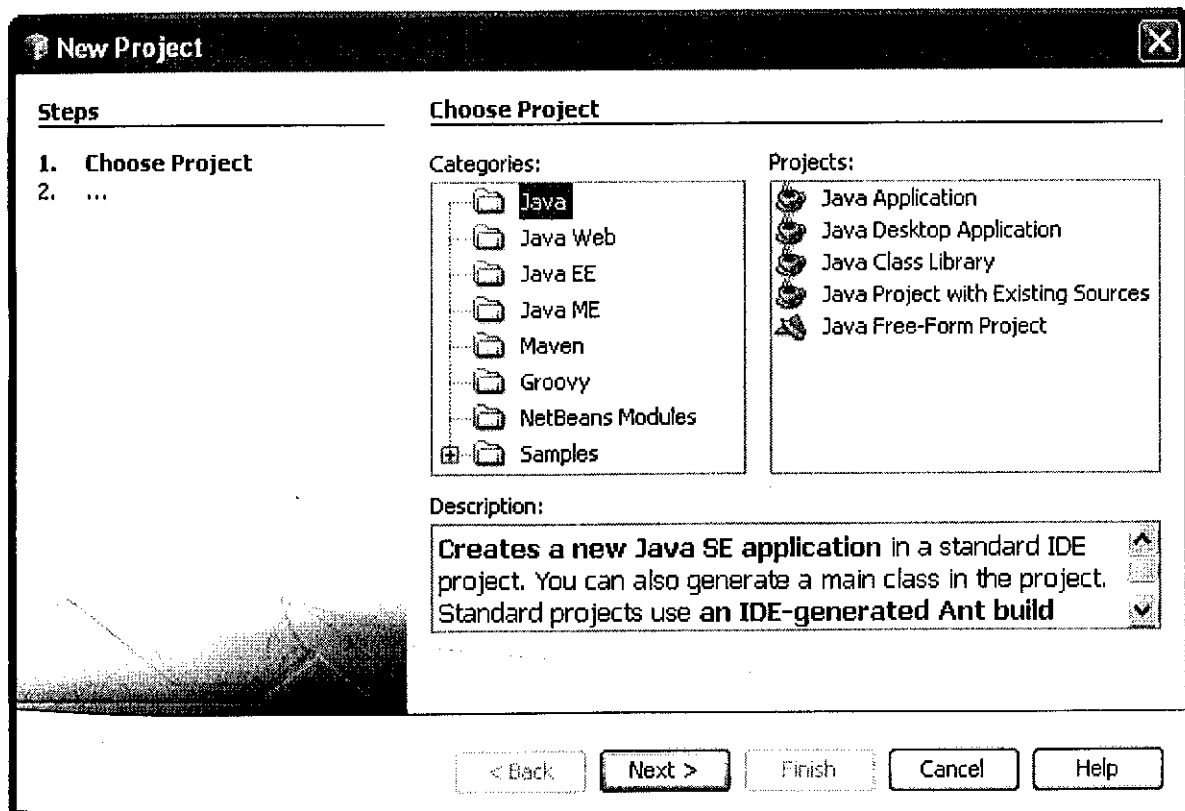


Fig 5: NetBeans New Project

4. In the Name and Location page of the wizard, do the following (as shown in the figure below):

- In the Project Name field, type project name.
- Leave the Use Dedicated Folder for Storing Libraries checkbox unselected.
- In the Create Main Class field, type the main class name.
- Leave the Set as Main Project checkbox selected.

New Java Application

Steps

1. Choose Project
2. Name and Location

Name and Location

Project Name: HelloWorldApp

Project Location: C:\work\tmp\nbproj Browse...

Project Folder: C:\work\tmp\nbproj\HelloWorldApp Browse...

☐ Use Dedicated Folder for Storing Libraries

Libraries Folder: Browse...

Different users and projects can share the same compilation libraries (see Help for details).

☒ Create Main Class helloworldapp.HelloWorldApp

☒ Set as Main Project

< Back Next > Finish Cancel Help

Fig 6: Netbeans New Project II

5. Click Finish.

The project is created and opened in the IDE. You should see the following components:

- The Projects window, which contains a tree view of the components of the project, including source files, libraries that your code depends on, and so on.
- The Source Editor window with a file called project name open.
- The Navigator window, which you can use to quickly navigate between elements within the selected class.
- The Tasks window, which lists compilation errors as well other tasks that are marked with keywords such as XXX and TODO.

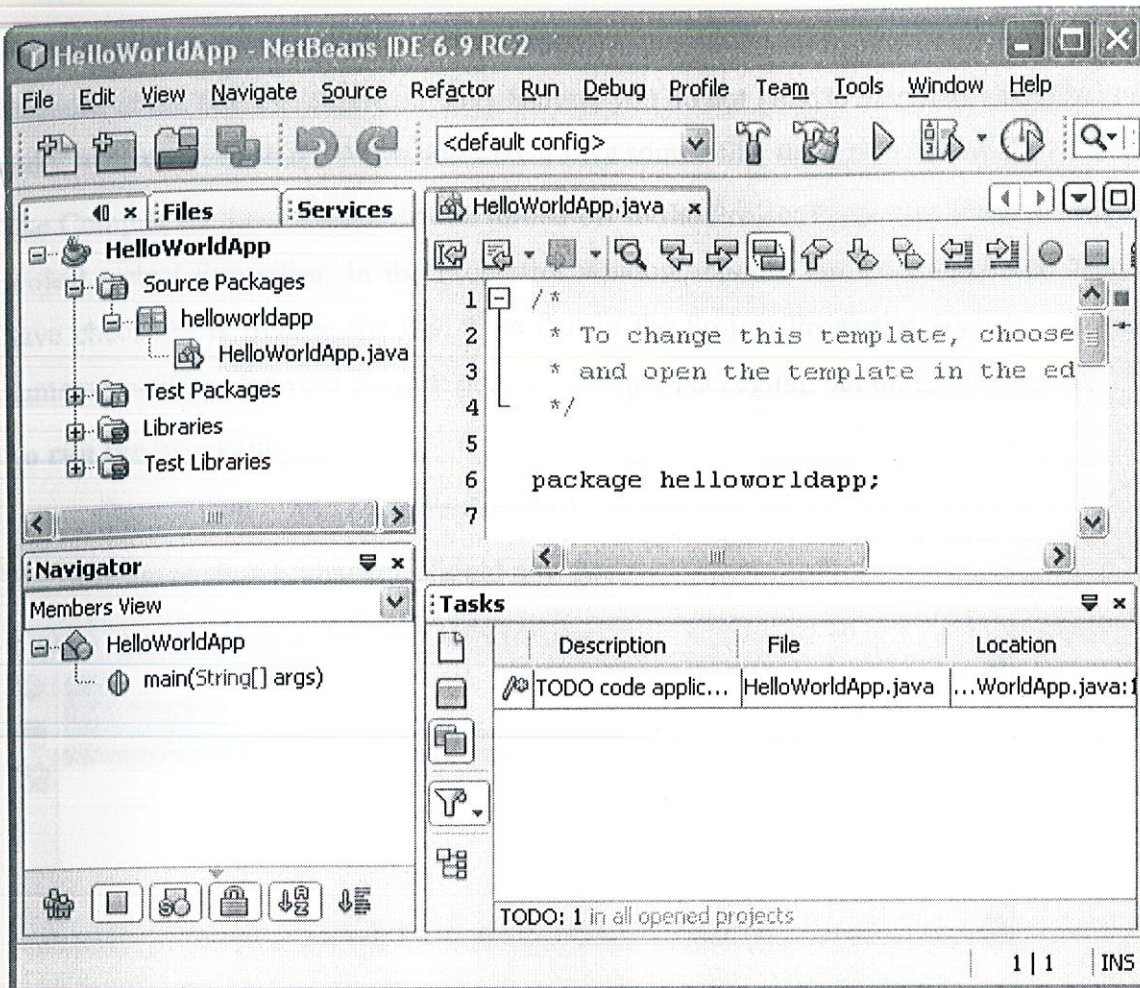


Fig 7: new project main window

3.2 Adding Code to the Generated Source File

Because you have left the Create Main Class checkbox selected in the New Project wizard, the IDE has created a skeleton main class for you. You can add the "Hello World!" message to the skeleton code by replacing the line:

```
// TODO code application logic here
```

with the line:

```
System.out.println("Hello World!");
```

Save the change by choosing File > Save.

3.3 Compiling and Running the Program

Because of the IDE's Compile on Save feature, you do not have to manually compile your project in order to run it in the IDE. When you save a Java source file, the IDE automatically compiles it.

The Compile on Save feature can be turned off in the Project Properties window. Right-click your project, select Properties. In the Properties window, choose the Compiling tab. The Compile on Save checkbox is right at the top. Note that in the Project Properties window you can configure numerous settings for your project: project libraries, packaging, building, running, etc.

To run the program:

- Choose Run > Run Main Project (F6).

The next figure shows what you should now see.

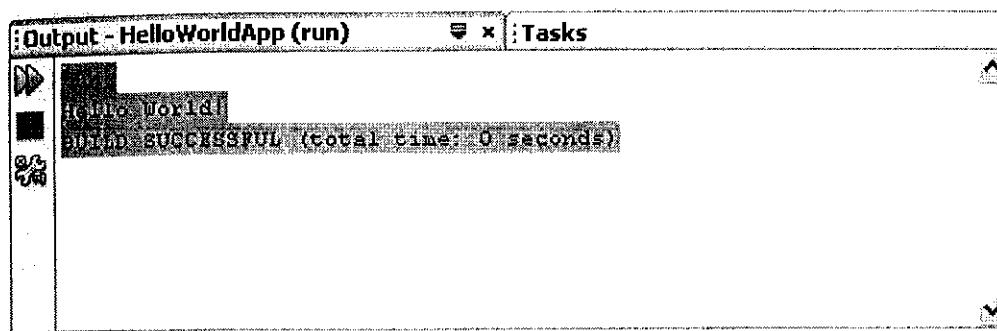


Fig 8: Program Execution window

If there are compilation errors, they are marked with red glyphs in the left and right margins of the Source Editor. The glyphs in the left margin indicate errors for the corresponding lines. The glyphs in the right margin show all of the areas of the file that have errors, including errors in lines that are not visible. You can mouse over an error mark to get a description of the error. You can click a glyph in the right margin to jump to the line with the error.

3.4 Building and Deploying the Application

Once you have written and test runs your application, you can use the Clean and Build command to build your application for deployment. When you use the Clean and Build command, the IDE runs a build script that performs the following tasks:

- Deletes any previously compiled files and other build outputs.
- Recompile the application and builds a JAR file containing the compiled files.

To build your application:

- Choose Run > Clean and Build Main Project (Shift-F11)

3.4 Project Setup

The application you create will contain two projects:

- A Java Class Library project in which you will create a utility class.
- A Java Application project with a main class that implements a method from the library project's utility class.

After you create the projects, you will add the library project to the classpath of the application project. Then you will code the application. The library project will contain a utility class with an acrostic method. The acrostic method takes an array of words as a parameter and then generates an acrostic based on those words. The MyApp project will contain a main class that calls the acrostic method and passes the words that are entered as arguments when the application is run.

3.4.1 Creating a Java Class Library Project

1. Choose File > New Project (Ctrl-Shift-N). Under Categories, select Java. Under Projects, select Java Class Library. Click Next.
2. Under Project Name, type MyLib. Change the Project Location to any directory on your computer. From now on, this tutorial refers to this directory as *NetBeans_projects*.
3. (Optional) Select the Use Dedicated Folder for Storing Libraries checkbox and specify the location for the libraries folder. See Sharing Project Libraries for more information on this option.
4. Click Finish. The MyLib project opens in both the Projects window and the Files window.

3.4.2 Creating a Java Application Project

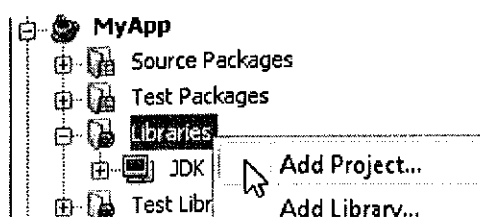
1. Choose File > New Project. Under Categories, select Java. Under Projects, select Java Application. Click Next.
2. Under Project Name, type MyApp. Make sure the Project Location is set to *NetBeans_projects*.
3. (Optional) Check the Use Dedicated Folder for Storing Libraries checkbox.
4. Enter acrostic.Main as the main class.
5. Ensure that the Set as Main Project and Create Main Class checkboxes are checked.
6. Click Finish. The MyApp project is displayed in the Project window and Main.java opens in the Source Editor.

3.4.3 Configuring the Compilation Classpath

Since MyApp is going to depend on a class in MyLib, you have to add MyLib to the classpath of MyApp. Doing so also ensures that classes in the MyApp project can refer to classes in the MyLib project without causing compilation errors. In addition, this enables you to use code completion in the MyApp project to fill in code based on the MyLib project. In the IDE, the classpath is visually represented by the Libraries node.

To add the library's utility classes to the project classpath:

1. In the Projects window, right-click the Libraries node for the MyApp project and choose Add Project as shown in the image below.



2. Browse to *NetBeans_projects/* and select the MyLib project folder. The Project JAR Files pane shows the JAR files that can be added to the project. Notice that a JAR file for MyLib is listed even though you have not actually built the JAR file yet. This JAR file will get built when you build and run the MyApp project.
3. Click Add Project JAR Files.
4. Expand the Libraries node. The MyLib project's JAR file is added to the MyApp project's classpath.

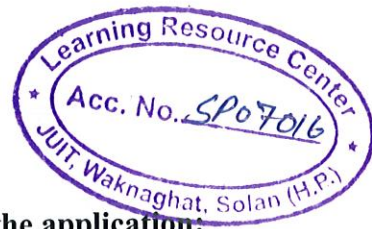
3.4.4 Creating and Editing Java Source Code

Now you need to create a Java package and add the method that you will use to construct the acrostic. After that you need to implement the acrostic method in the Main class.

3.4.4.1 Creating a Java Package and Class File

1. Right-click the MyLib project node and choose New > Java Class. Type LibClass as the name for the new class, typeorg.me.mylib in the Package field, and click Finish. LibClass.java opens in the Source Editor.
2. In LibClass.java, place the cursor on the line after the class declaration (public class LibClass {.
3. Type or paste in the method code.

4. If the code that you pasted in is not formatted correctly, press Alt-Shift-F to reformat the entire file.
5. Press Ctrl-S to save the file.



3.4.4.2 To add the arguments for the IDE to use when running the application:

1. Right-click the MyApp project node, choose Properties, and select the Run node in the dialog's left pane.

The main class should already be set to `acrostic.Main`.

2. Type `However we all feel zealous` in the Arguments field and click OK.

Running the Application

Now that you have created the application and provided runtime arguments for the application, you can test run the application in the IDE.

3.4.5 To run the application in the IDE:

- Choose Run > Run Main Project (F6).

In the Output window, you should see the output from the program, `Result = Hello` (the acrostic of the phrase that was passed to the program as an argument).

Testing and Debugging the Application

Now you will create and run a test for the project using JUnit and then run the application in the IDE's debugger to check for errors. In the JUnit test, you will test the `LibClass` by passing a phrase to the `acrostic` method and using an assertion to indicate what you think the result should be.

3.5 Creating JUnit Tests

1. Right-click the `LibClass.java` node in the Projects window and choose Tools > Create JUnit Tests (Ctrl-Shift-U).

If this is the first time you have created JUnit tests in the IDE, you will be prompted with the Select JUnit Version dialog box. Press Enter to select JUnit 4.x and continue to the Create Tests dialog box.

2. In the Create Tests dialog box, click OK to run the command with the default options. The IDE creates `theorg.me.mylib` package and the `LibClassTest.java` file in a separate test folder. You can find this file by expanding the Test Packages node and the `org.me.mylib` subnode.
3. In `LibClassTest.java`, delete the body of the public void `testAcrostic()` method.

4. In place of the deleted lines, type or paste in the following:
5. `System.err.println("Running testAcrostic...");`
6. `String result = LibClass.acrostic(new String[]`
7. `{"fnord", "polly", "tropism"});`
`assertEquals("Correct value", "foo", result);`
8. Save the file by pressing Ctrl-S.

3.6 Running JUnit Tests

1. Select the MyLib project node and choose Run > Test Project (MyLib) or press Alt-F6. The MyLib (test) tab opens in the Output window. The JUnit test cases are compiled and run. The JUnit test result shows that the test passes.
2. You can also run a single test file rather than testing the entire project. Select the LibClass.java tab in the Source Editor and choose Run > Test File.

3.7 Debugging the Application

1. In the LibClass.java file, go to the acrostic method and place the insertion point anywhere inside `b.append(args[i].charAt(i));`. Then press Ctrl-F8 to set a breakpoint.
2. Choose Debug > Debug Main Project (Ctrl-F5). The IDE opens the Debugger windows and runs the project in the debugger until the breakpoint is reached.
3. Select the Local Variables window in the bottom of the IDE and expand the args node. The array of strings contains the phrase you entered as the command arguments.
4. Press F7 (or choose Debug > Step Into) to step through the program and watch the b variable change as the acrostic is constructed.

When the program reaches the end, the debugger windows closes.

3.8 Building, Running, and Distributing the Application

Once you are satisfied that your application works properly, you can prepare the application for deployment outside of the IDE. In this section you will build the application's JAR file and then run the JAR file from the command line.

3.8.1 Building the Application

The main build command in the IDE is the Clean and Build command. The Clean and Build command deletes previously compiled classes and other build artifacts and then rebuilds the entire project from scratch.

To build the application:

- Choose Run > Clean and Build Main Project (Shift-F11).

Output from the Ant build script appears in the Output window. If the Output window does not appear, you can open it manually by choosing Window > Output > Output.

When you clean and build your project, the following things occur:

- Output folders that have been generated by previous build actions are deleted ("cleaned"). (In most cases, these are thebuild and dist folders.)
- build and dist folders are added to your project folder (hereafter referred to as the *PROJECT_HOME* folder). You can view these folders in the Files window.
- All of the sources are compiled into .class files, which are placed into the *PROJECT_HOME/build* folder.
- A JAR file containing your project is created inside the *PROJECT_HOME/dist* folder.
- If you have specified any libraries for the project (in addition to the JDK), a lib folder is created in the dist folder. The libraries are copied into dist/lib.
- The manifest file in the JAR is updated to include entries that designate the main class and any libraries that are on the project's classpath.

3.8.2 Running the Application Outside of the IDE

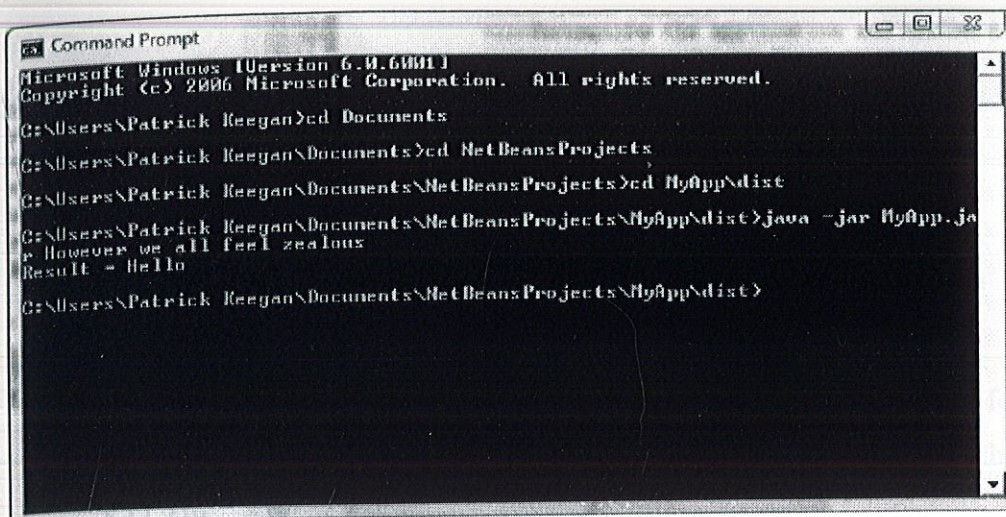
To run the application outside of the IDE:

1. On your system, open up a command prompt or terminal window.
2. In the command prompt, change directories to the MyApp/dist directory.
3. At the command line, type the following statement:

```
java -jar MyApp.jar However we all feel zealous
```

The application then executes and returns the following output as shown in the image below:

```
Result = Hello
```

```
Microsoft Windows [Version 6.0.6002.1.80402]
Copyright (c) 2006 Microsoft Corporation. All rights reserved.

C:\Users\Patrick Keegan>cd Documents
C:\Users\Patrick Keegan\Documents>cd NetBeansProjects
C:\Users\Patrick Keegan\Documents\NetBeansProjects>cd MyApp\dist
C:\Users\Patrick Keegan\Documents\NetBeansProjects\MyApp\dist>java -jar MyApp.jar
Hello
C:\Users\Patrick Keegan\Documents\NetBeansProjects\MyApp\dist>
```

Fig 9: Command Prompt Execution of Java Programs

3.8.3 Distributing the Application to Other Users

Now that you have verified that the application works outside of the IDE, you are ready to distribute the application.

To distribute the application:

1. On your system, create a zip file that contains the application JAR file (MyApp.jar) and the accompanying lib folder that contains MyLib.jar.
2. Send the file to the people who will use the application. Instruct them to unpack the zip file, making sure that the MyApp.jar file and the lib folder are in the same folder.
3. Instruct the users to follow the steps in the Running the Application Outside of the IDE section above.

3.9 Other Common Tasks

You have now completed the main part of the tutorial, but there are still some basic tasks that have not been covered. This section includes a few of those tasks.

3.9.1 Making the Javadoc Available in the IDE

To view the JavaSE API documentation in the NetBeans IDE, use the Source > Show Documentation command or choose Window > Other > Javadoc from the main menu to view API documentation in a separate window.

However, for some third-party libraries, API documentation is not available. In these cases, the Javadoc resources must be manually associated with the IDE.

To make the Javadoc API documentation available for the Show Javadoc command:

1. Download the Javadoc API documentation source.
2. Choose Tools > Libraries.
3. In the Libraries list, select the library that your project is using.
4. Click the Javadoc tab.
5. Click the Add ZIP/Folder button and navigate to the zip file or the folder that contains the Javadoc API documentation on your system. Select the zip file or the folder and click the Add ZIP/Folder button.
6. Click Close.

3.9.2 Generating Javadoc for a Project

You can generate compiled Javadoc documentation for your project based on Javadoc comments that you have added to your classes.

To generate Javadoc documentation for a project:

1. Select the MyLib project.
2. Choose Run > Generate Javadoc for "MyLib" from the IDE's main menu.

The generated Javadoc is added to the dist folder of the project. In addition, the IDE opens a web browser that displays the Javadoc.

3.10 Basic Project Concepts

This section provides an overview of some background information on the IDE's project system.

3.10.1 Projects

In the IDE, you always work inside of a project. In addition to source files, an IDE project contains metadata about what belongs on the classpath, how to build and run the project, and so on. The IDE stores project information in a project folder which includes an Ant build script and properties file that control the build and run settings, and a project.xml file that maps Ant targets to IDE commands.

3.10.2 Ant

Apache Ant is a Java-based build tool used to standardize and automate build and run environments for development. The IDE's project system is based directly on Ant. All of the project commands, like Clean and Build Project and Debug, call targets in the project's Ant script. You can therefore build and run your project outside the IDE exactly as it is built and run inside the IDE.

It is not necessary to know Ant to work with the IDE. You can set all the basic compilation and runtime options in the project's Project Properties dialog box and the IDE automatically updates your project's Ant script. If you are familiar with Ant, you can customize a standard project's Ant script or write your own Ant script for a project.

Creating a Project

To create a new project:

- Choose File > New Project (Ctrl-Shift-N).

When the New Project wizard appears, select the right template for your project and complete the remaining wizard steps. In the releases later than NetBeans IDE 6.7, the project template icon can be displayed in gray, which means that this project type has not been activated. Proceed with creating the project and this functionality will be activated in the IDE.

The IDE contains the following standard project templates for Java desktop and web applications:

- **Java Application.** Creates a skeleton Java SE project with a main class.
- **Java Desktop Application.** Creates an application based on the Swing Application Framework. Skeletons are offered for a basic desktop application and a database application that makes use of the Beans Binding and Java Persistence API libraries.
- **Java Class Library.** Creates a skeleton Java class library without a main class.
- **Java Project with Existing Sources.** Creates a Java SE project based on your own Java sources.
- **Web Application.** Creates a skeleton web application, including options to add various web frameworks
- **Web Application with Existing Sources.** Creates a web project based on your own web and Java sources.

In addition, the IDE also contains templates for EJB modules, enterprise applications, Java ME applications, and more.

The Java and Web project categories also have free-form project templates. The free-form templates enable you to use an existing Ant script for a project but require manual configuration.

When you finish creating a project, it opens in the IDE with its logical structure displayed in the Projects window and its file structure displayed in the Files window:

- The Projects window is the main entry point to your project sources. It shows a logical view of important project contents such as Java packages and Web pages. You can right-click any project node to access a popup menu of commands for building, running, and debugging the project, as well as opening the Project Properties dialog box. The Projects window can be opened by choosing Window > Projects (Ctrl-1).
- The Files window shows a directory-based view of your projects, including files and folders that are not displayed in the Projects window. From the Files window, you can open and edit your project configuration files, such as the project's build script and properties file. You can also view build output like compiled classes, JAR files, WAR files, and generated Javadoc documentation. The Files window can be opened by choosing Window > Files (Ctrl-2).

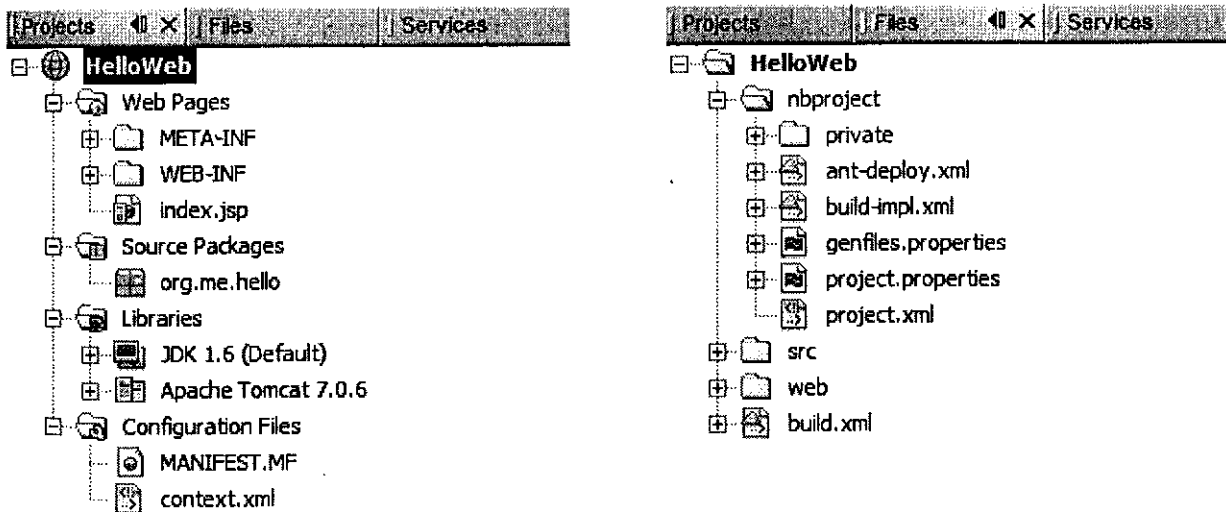


Fig 10: Apache Ant Home Screen

3.10.3 Importing a Project

This section shows you how to handle the initial importing of projects into the IDE.

3.10.3.1 Importing an Eclipse Workspace

For Eclipse projects, you can use the Import Eclipse Project wizard to help you create NetBeans projects from projects in an Eclipse workspace and import the project classpaths and other settings. When you use the Import Eclipse Project wizard, you do not need to use create and configure the NetBeans project manually. Open the wizard by choosing File > Import Project > Eclipse Project.

3.10.3.2 Setting Up a Java Project Based on Existing Sources

For other Java projects developed outside of NetBeans, you use an "Existing Sources" template in the New Project wizard to make a NetBeans project. In the wizard, you identify the location of the sources and specify a location for the NetBeans project metadata. You then use the Project Properties dialog box to configure the project.

To set up a NetBeans project for an existing Java application:

1. Choose File > New Project (Ctrl-Shift-N).
2. Choose Java > Java Project with Existing Sources. Click Next.
3. In the Name and Location page of the wizard, follow these steps:
 - Type a project name.
 - (Optional) Change the location of the project folder.
 - (Optional) Change the name of the build script used by the IDE. This might be desirable if there is already a build script called build.xml that is used to build the sources.
 - (Optional) Select the Use Dedicated Folder for Storing Libraries checkbox and specify the location for the libraries folder.
 - (Optional) Select the Set as Main Project checkbox. When you select this option, keyboard shortcuts for commands such as Clean and Build Main Project (Shift-F11) apply to this project.
4. Click Next to advance to the Existing Sources page of the wizard.

5. In the Source Packages Folder pane, click Add Folder. Then navigate to your sources and select the source roots, click Open.

When you add a folder containing source code, you must add the folder that contains the highest folder in your package tree. For example, for the `com.mycompany.myapp.ui` package, you add the folder that contains the `com` folder.

6. (Optional) In the Test Package Folders pane, click Add Folder to select the folder containing the JUnit package folders.
7. Click Next to advance to the Includes & Excludes page of the wizard.
8. (Optional) In the Includes & Excludes page of the wizard, enter file name patterns for any files that should be included or excluded from the project. By default, all files in your source roots are included.
9. Click Finish.

3.10.3.3 Setting Up a Web Project Based on Existing Sources

For web projects developed outside of NetBeans, you use an "Existing Sources" template in the New Project wizard to make a NetBeans project. In the wizard, you identify the location of the sources and specify a location for the NetBeans project metadata. You then use the Project Properties dialog box to configure the project.

CHAPTER 4

Sample Code.

4.1 Recorder.java:

```
package org.multichat.client;

import org.multichat.CommonSoundClass;

import java.io.ByteArrayOutputStream;

import javax.sound.sampled.DataLine;
import javax.sound.sampled.TargetDataLine;
import javax.sound.sampled.AudioFormat;
import javax.sound.sampled.AudioSystem;
import javax.sound.sampled.AudioInputStream;
import javax.sound.sampled.LineUnavailableException;
import javax.sound.sampled.AudioFileFormat;

public class Recorder
    extends Thread {
    private TargetDataLine m_line;
    private AudioFileFormat.Type m_targetType;
    private AudioInputStream m_audioInputStream;
    private boolean m_bRecording;
    private boolean m_bQuitting;

    public byte bs[];
    public static CommonSoundClass cs;

    boolean onlyonce = false;

    public Recorder(CommonSoundClass csPtr) {
        this.cs = csPtr;

        boolean gotrecordingline = true;

        ByteArrayOutputStream outputFile = new ByteArrayOutputStream();
        AudioFormat audioFormat = null;
        // 8 kHz, 8 bit, mono
        audioFormat = new AudioFormat(AudioFormat.Encoding.PCM_SIGNED,
ClientShared.sampleRate, ClientShared.sampleSize, 1, 1,
ClientShared.frameRate, false);
        // 44.1 kHz, 16 bit, stereo
        // audioFormat = new AudioFormat( AudioFormat.Encoding.PCM_SIGNED,
44100.0f, 16, 1, 2, 44100.0f, false );
        // audioFormat = new AudioFormat(AudioFormat.Encoding.PCM_SIGNED,
44100.0F, 16, 2, 4, 44100.0F, false);

        // Get Line (microphone) Information
        DataLine.Info info = new DataLine.Info(TargetDataLine.class,
audioFormat);
        TargetDataLine targetDataLine = null;
```

```

try {
    // Connect to line
    targetDataLine = (TargetDataLine) AudioSystem.getLine(info);
    targetDataLine.open(audioFormat);
}
catch (LineUnavailableException e) {
    System.err.println("Error: Unable to get a recording line");
    gotrecordingline = false;
    //e.printStackTrace();
    //System.exit(1);
}

if (gotrecordingline) {
    AudioFileFormat.Type targetType = AudioFileFormat.Type.AU;
//    Recorder recorder = null;
    RecorderInit(targetDataLine, targetType);
    m_bRecording = true;
    m_bQuitting = false;
    this.start();
}

}

public void RecorderInit(TargetDataLine line, AudioFileFormat.Type
targetType) {
    m_line = line;
    m_audioInputStream = new AudioInputStream(line);
    m_targetType = targetType;
}

/**
 * Starts the recording.
 * To accomplish this, (i) the line is started and (ii) the
 * thread is started.
 */
public void start() {
    m_bRecording = true;
    m_line.start();
    super.start();
}

public void startRecording() {
    m_bRecording = true;
}

public void stopRecording() {
    m_bRecording = false;
    onlyonce = true;
}

synchronized public void run() {
    while ( !m_bQuitting ) {
        byte bs[] = new byte[ClientShared.audioReadBytes];
        m_line.read(bs, 0, ClientShared.audioReadBytes);
        if (m_bRecording) {

```

```

        cs.writebyte(bs);
    } else if (onlyonce) {
        cs.writebyte(("NT|").getBytes());
        onlyonce = false;
    }
}

m_line.stop();
m_line.close();
}

public void onExit() {
    m_bQuitting = true;
}
}

```

4.2 CommonSoundClass.java:

```

// $Id: Queue.java,v 1.1 2001/05/04 21:22:05 mito Exp $
package org.multichat;

import java.util.*;

public class CommonSoundClass {
    public Vector vec = new Vector();
    boolean lock = true;
    private byte b[];

    public CommonSoundClass() {
    }

    synchronized public Object readbyte() {
        try {
            while (vec.isEmpty()) {
                wait();
            }
        } catch (InterruptedException ie) {
            System.err.println("Error: CommonSoundClass readbyte
interrupted");
        }

        if (! vec.isEmpty()) {
            b = (byte[]) vec.remove(0);
            return b;
        } else {
            byte[] b = new byte[5];
            return b;
        }
    }

    synchronized public void writebyte(Object e) {
        vec.addElement(e);
    }
}

```

```

        lock = false;
        notifyAll();
    }
}

```

4.3 Playback.java:

```

package org.multichat;

// $Id: org.multichat.Playback.java,v 1.5 2001/05/04 21:13:09 mito Exp $

import org.multichat.client.ClientShared;

import javax.sound.sampled.*;

public class Playback implements Runnable {
    // Data written to this is played by the soundcard
    private SourceDataLine sdl;

    // Write this many bytes per inner loop execution
    static private final int innerLoopWriteSize = 2048;

    // new sounds to be played are placed on this queue
    private Queue incoming = new Queue();

    // lock to wait on when waiting for a sound to play
    private Object soundLock = new Object();

    // assume a standard sampling rate
    static final public int sampRate = 8000;

    // the latency through the low-level sound system
    // this must be tuned for each system
    static private final double sysLatencyTime = 0.695;

    // the sound latency expressed in samples
    static private final int sysLatency =
        (int)(sysLatencyTime*sampRate);

    // a standard audio format for playing audio
    static public AudioFormat stdFormat =
        new AudioFormat(AudioFormat.Encoding.PCM_SIGNED,
ClientShared.sampleRate, ClientShared.sampleSize, 1, 1,
ClientShared.frameRate, false);

    // The construtor opens a playback audio line
    // and creates a background thread for streaming data
    public Playback() {
        // Open the playback line
        sdl = getOutputLine();

        // Create a background thread for streaming audio
        Thread t = new Thread( this, "playback" );
        t.start();
    }
}

```

```

public Playback( byte b[] ){
    sdl = getOutputLine();
    // Create a background thread for streaming audio
    Thread t = new Thread( this, "playback" );
    t.start();
    this.setSound( b );
}

// Tell the playback to play the next sound
// set to null to turn sound off
public void setSound( byte raw[] ) {
    synchronized(soundLock) {
        // place the new sound on the queue
        incoming.put( raw );

        // tell the thread that there's a new sound
        soundLock.notifyAll();
    }
}

// Background streaming thread
public void run() {
    // The currently playing sound
    byte currentRaw[] = null;

    // The position within the currently playing sound
    int cursor = 0;

    // open the output line for playing
    try {
        // open it with our standard audio format
        sdl.open( stdFormat );

        // start it playing
        sdl.start();
    } catch( LineUnavailableException lue ) {
        throw new RuntimeException( lue.toString() );
    }

    while (true) {
        synchronized (soundLock) {
            while( true ){
                if(incoming.numWaiting()>0){
                    currentRaw = (byte[])incoming.get();
                    break;
                }else{
                    try {
                        soundLock.wait(50);
                    } catch( InterruptedException ie ) {}
                }
            }
            cursor = 0;
            int bytesLeft = currentRaw.length - cursor;

```

```

        do{
            if (sdl.available()>0){
                int r = Math.min(sdl.available(),
currentRaw.length-cursor);
                sdl.write( currentRaw, cursor, r );
                if (r==-1)
                    throw new RuntimeException( "Can't write to
line!" );
                cursor+=r;
            } else {
                try {
                    soundLock.wait(10);
                } catch( InterruptedException ie ) {}
            }
        } while(currentRaw.length-cursor > 0);
    }
}

// Utility class -- open an output line with our standard
// audio format
public SourceDataLine getOutputLine() {
    try {
        DataLine.Info info =
            new DataLine.Info( SourceDataLine.class, stdFormat );
        SourceDataLine sdl = (SourceDataLine)AudioSystem.getLine(
info );
        return sdl;
    } catch( LineUnavailableException lue ) {
        throw new RuntimeException( "Can't get output line" );
    }
}
}

```

4.4 Playback.java:

```

package org.multichat;

// $Id: Queue.java,v 1.1 2001/05/04 21:22:05 mito Exp $

import java.util.*;

public class Queue {
    // Internal storage for the queue'd objects
    private Vector vec = new Vector();
    boolean prebuffer = true;
    // Lower this rate for quicker delivery of audio
    int queueWaitSize = 12; /* 16 * 5 + 1 */

    synchronized public int numWaiting() {
        // makes the client wait for more packets before starting to play
        them
    }
}

```

```

// this takes care of the buffer :)
// This makes sound clearer but also causes delays in receiving
audio.
if( prebuffer && vec.size() < queueWaitSize){
    return 0;
} else {
    prebuffer = false;
    return vec.size();
}
}

synchronized public void put( Object o ) {
    // Add the element
    vec.addElement( o );
    // There might be threads waiting for the new object --
    // give them a chance to get it
    notifyAll();
}

synchronized public Object get() {
    while (true) {
        if ( numWaiting() > 0 /*vec.size()>0*/ ) {

            // remove the bytes if its more then 1.5 seconds delay
            while( vec.size() > (24 /*16 * 10*/) ){
                vec.removeElementAt(0);
            }

            // There's an available object!
            Object o = vec.elementAt( 0 );

            // Remove it from our internal list, so someone else
            // doesn't get it.

            /*int vectorsize = vec.size();

            if( vectorsize > 1 ){
                vectorsize = 1;
            }

            int j = 0;
            int i = 0;
            byte[] bigbyte = new byte[vectorsize * (1024*8)];
            for(; i < vectorsize * 512; i++){
                if( j == 1024*8 ){
                    j = 0;
                    vec.removeElementAt(0);
                }
                bigbyte[i] = ((byte[]) (vec.elementAt( 0 )))[j];
                j++;
            }*/

            vec.removeElementAt( 0 );

            //System.out.println("buffer is " + vec.size() );
            if( vec.size() == 0 ){

```



```

        prebuffer = true; // we have reached the last element in
the stack we shuld buffer more data before playing
    }

    // Return the object
    return o;
    //return (Object)bigbyte;
} else {
    // There aren't any objects available. Do a wait(),
    // and when we wake up, check again to see if there
    // are any.
    try { wait(); } catch( InterruptedException ie ) {}
}
}
}
}

```

4.5 Sock.java:

```

/*
 * To change this template, choose Tools | Templates
 * and open the template in the editor.
 */

package chat_application;

import java.io.BufferedReader;
import java.io.IOException;
import java.io.InputStream;
import java.io.InputStreamReader;
import java.io.OutputStream;
import java.io.OutputStreamWriter;
import java.io.PrintWriter;
import java.net.ConnectException;
import java.net.ServerSocket;
import java.net.Socket;
import java.net.SocketException;

public class sock {
    String name;
    long time_entry;
    long timespend;
    String ip;
    Socket s,temp;
    client_entry chaman;
    BufferedReader br,br2;
    String str;
    PrintWriter pr;

    sock()
    {
        chaman =new client_entry();
        java.awt.EventQueue.invokeLater(new Runnable() {
            public void run() {
                chaman.setVisible(true);
            }
        })
    }
}

```

```

    });

}

Socket getsock()
{
    return s;
}
String getname()
{
    return name;
}
void setsock()
{
    try
    {
        name=chaman.str1;                // bhokaal error ....
        time_entry=System.currentTimeMillis()/1000;
        ip=chaman.getstr4();
        s=new Socket(ip,4437);
    }
    catch(SocketException ll)
    {
        System.out.println("check point 1");
    }
    catch(IOException e)
    {
        System.out.println("io exception ....");
    }
}

}

public static void main(String args[])
{
    sock cl=new sock();
    try
    {
        Thread.sleep(13000);
    }
    catch(Exception e)
    {}
    cl.setsock();
    Clinfol c =new Clinfol(cl.getsock());
    Thread t=new Thread(c);
    Clinfo2 c2 =new Clinfo2(cl.getsock(),cl.getname());
    Thread t2=new Thread(c2);
    t.start();
    t2.start();

}
}

```

4.6 Pop_up.java:

```
public class pop_up extends javax.swing.JFrame {

    /** Creates new form pop_up */
    public pop_up() {
        initComponents();
    }

    /** This method is called from within the constructor to
     * initialize the form.
     * WARNING: Do NOT modify this code. The content of this method is
     * always regenerated by the Form Editor.
     */
    @SuppressWarnings("unchecked")
    // <editor-fold defaultstate="collapsed" desc="Generated Code"> //GEN-
BEGIN: initComponents
    private void initComponents() {

        jLabel1 = new javax.swing.JLabel();
        jButton1 = new javax.swing.JButton();

        setDefaultCloseOperation(javax.swing.WindowConstants.EXIT_ON_CLOSE);

        jLabel1.setFont(new java.awt.Font("Tahoma", 1, 12)); // NOI18N
        jLabel1.setText("Name & Email id can't be left blank .");

        jButton1.setText("OK");
        jButton1.addActionListener(new java.awt.event.ActionListener() {
            public void actionPerformed(java.awt.event.ActionEvent evt) {
                jButton1ActionPerformed(evt);
            }
        });

        javax.swing.GroupLayout layout = new
javax.swing.GroupLayout(getContentPane());
        getContentPane().setLayout(layout);
        layout.setHorizontalGroup(
layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
            .addGroup(layout.createSequentialGroup()
                .addContainerGap()
                .addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
                    .addComponent(jLabel1,
javax.swing.GroupLayout.PREFERRED_SIZE, 233,
javax.swing.GroupLayout.PREFERRED_SIZE)
                    .addGroup(layout.createSequentialGroup()
                        .addGap(119, 119, 119)
                        .addComponent(jButton1)))
                .addContainerGap())
        );
    }
}
```

```

        .addContainerGap(20, Short.MAX_VALUE))
    );
    layout.setVerticalGroup(

layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
        .addGroup(layout.createSequentialGroup()
            .addComponent(jLabel1,
                javax.swing.GroupLayout.PREFERRED_SIZE, 35,
                javax.swing.GroupLayout.PREFERRED_SIZE)

            .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED)
                .addComponent(jButton1)
                .addContainerGap(javax.swing.GroupLayout.DEFAULT_SIZE,
Short.MAX_VALUE))
        );

    pack();
} // </editor-fold> // GEN-END: initComponents

private void jButton1ActionPerformed(java.awt.event.ActionEvent evt)
{ // GEN-FIRST: event_jButton1ActionPerformed
    this.setVisible(false);

    // GEN-LAST: event_jButton1ActionPerformed

    /**
     * @param args the command line arguments
     */

    // Variables declaration - do not modify // GEN-BEGIN: variables
    private javax.swing.JButton jButton1;
    private javax.swing.JLabel jLabel1;
    // End of variables declaration // GEN-END: variables
}

```

4.7 Client_entry.java:

```

package chat_application;
public class client_entry extends javax.swing.JFrame {
    private pop_up p;
    String str1, str2, str3, str4;

    public client_entry()
    {
        initComponents();
    }

    String getstr1()
    {
        return str1;
    }

    String getstr4()
    {

```

```

return str4;
}

```

```

@SuppressWarnings("unchecked")
// <editor-fold defaultstate="collapsed" desc="Generated Code"> //GEN-
BEGIN: initComponents
private void initComponents() {

```

```

    jLabel1 = new javax.swing.JLabel();
    jSeparator1 = new javax.swing.JSeparator();
    jLabel2 = new javax.swing.JLabel();
    jtf1 = new javax.swing.JTextField();
    jLabel3 = new javax.swing.JLabel();
    jtf2 = new javax.swing.JTextField();
    jLabel4 = new javax.swing.JLabel();
    jLabel5 = new javax.swing.JLabel();
    jLabel6 = new javax.swing.JLabel();
    jtf3 = new javax.swing.JTextField();
    jLabel7 = new javax.swing.JLabel();
    jButton1 = new javax.swing.JButton();
    jLabel8 = new javax.swing.JLabel();
    jLabel9 = new javax.swing.JLabel();
    jtf4 = new javax.swing.JTextField();
    jLabel10 = new javax.swing.JLabel();

```

```

setDefaultCloseOperation(javax.swing.WindowConstants.EXIT_ON_CLOSE);
setTitle("cLiEnT sUpPoRt WiNdOw");
setCursor(new java.awt.Cursor(java.awt.Cursor.DEFAULT_CURSOR));
setResizable(false);

```

```

    jLabel1.setFont(new java.awt.Font("Tahoma", 3, 36));
    jLabel1.setText("EnTeR DeTaIlS");

```

```

    jLabel2.setFont(new java.awt.Font("Tahoma", 1, 14));
    jLabel2.setText("Enter NAME : ");

```

```

    jtf1.addActionListener(new java.awt.event.ActionListener() {
        public void actionPerformed(java.awt.event.ActionEvent evt) {
            jtf1ActionPerformed(evt);
        }
    });

```

```

    jLabel3.setFont(new java.awt.Font("Tahoma", 1, 14));
    jLabel3.setText("Enter CUSTOM TAG : ");

```

```

    jLabel4.setText("*");

```

```

    jLabel5.setText("* Fields marked with ' * ' are compulsory to
enter .");
    jLabel5.setBorder(new javax.swing.border.MatteBorder(null));

```

```

    jLabel6.setFont(new java.awt.Font("Tahoma", 1, 14));
    jLabel6.setText("Email Address :");

```

```

    jLabel7.setText("*");

```



```

.addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.UNRELATED)
    .addComponent(jLabel4,
javax.swing.GroupLayout.DEFAULT_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE, Short.MAX_VALUE))
    .addGroup(layout.createSequentialGroup()
        .addComponent(jLabel6,
javax.swing.GroupLayout.PREFERRED_SIZE, 119,
javax.swing.GroupLayout.PREFERRED_SIZE)

.addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED)
    .addComponent(jtf3,
javax.swing.GroupLayout.PREFERRED_SIZE, 225,
javax.swing.GroupLayout.PREFERRED_SIZE)

.addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.UNRELATED)
    .addComponent(jLabel7))
    .addComponent(jLabel5,
javax.swing.GroupLayout.PREFERRED_SIZE, 298,
javax.swing.GroupLayout.PREFERRED_SIZE)))
    .addGroup(layout.createSequentialGroup())

.addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Alignment.TRAILING, false)

.addGroup(javax.swing.GroupLayout.Alignment.LEADING,
layout.createSequentialGroup()
    .addContainerGap()
    .addComponent(jLabel9,
javax.swing.GroupLayout.PREFERRED_SIZE, 122,
javax.swing.GroupLayout.PREFERRED_SIZE)

.addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED)
    .addComponent(jtf4))

.addGroup(javax.swing.GroupLayout.Alignment.LEADING,
layout.createSequentialGroup()
    .addGap(51, 51, 51)
    .addComponent(jLabel8,
javax.swing.GroupLayout.PREFERRED_SIZE, 271,
javax.swing.GroupLayout.PREFERRED_SIZE)))

.addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.UNRELATED)
    .addComponent(jLabel10))
    .addGap(38, 38, 38)
    .addComponent(jButton1,
javax.swing.GroupLayout.DEFAULT_SIZE, 44, Short.MAX_VALUE))
    .addGroup(javax.swing.GroupLayout.Alignment.TRAILING,
layout.createSequentialGroup()
    .addContainerGap(90, Short.MAX_VALUE)
    .addComponent(jLabel11,
javax.swing.GroupLayout.PREFERRED_SIZE, 290,
javax.swing.GroupLayout.PREFERRED_SIZE)
    .addGap(76, 76, 76))
);
layout.setVerticalGroup(

```

```

layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
    .addGroup(layout.createSequentialGroup()
        .addComponent(jLabel1,
            javax.swing.GroupLayout.PREFERRED_SIZE, 68,
            javax.swing.GroupLayout.PREFERRED_SIZE)
        .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED)
        .addComponent(jSeparator1,
            javax.swing.GroupLayout.PREFERRED_SIZE, 10,
            javax.swing.GroupLayout.PREFERRED_SIZE)
        .addGap(18, 18, 18)
    )
    .addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Alignment.TRAILING, false)
        .addGroup(layout.createSequentialGroup()
            .addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Alignment.BASELINE)
                .addComponent(jLabel2)
                .addComponent(jtf1,
                    javax.swing.GroupLayout.PREFERRED_SIZE,
                    javax.swing.GroupLayout.DEFAULT_SIZE,
                    javax.swing.GroupLayout.PREFERRED_SIZE)
                .addComponent(jLabel4))
            .addGap(29, 29, 29)
        )
        .addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING, false)
            .addComponent(jLabel3)
            .addComponent(jtf2,
                javax.swing.GroupLayout.DEFAULT_SIZE, 61, Short.MAX_VALUE))
            .addGap(17, 17, 17)
    )
    .addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Alignment.BASELINE)
        .addComponent(jLabel6)
        .addComponent(jtf3,
            javax.swing.GroupLayout.PREFERRED_SIZE,
            javax.swing.GroupLayout.DEFAULT_SIZE,
            javax.swing.GroupLayout.PREFERRED_SIZE)
        .addComponent(jLabel7))
        .addGap(27, 27, 27)
        .addComponent(jLabel8)
        .addGap(4, 4, 4)
    )
    .addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Alignment.BASELINE)
        .addComponent(jLabel9)
        .addComponent(jtf4,
            javax.swing.GroupLayout.PREFERRED_SIZE,
            javax.swing.GroupLayout.DEFAULT_SIZE,
            javax.swing.GroupLayout.PREFERRED_SIZE)
        .addComponent(jLabel10))
    .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED, 41,
        Short.MAX_VALUE)

```



```

        .addComponent(jLabel5))
        .addComponent(jButton1,
javax.swing.GroupLayout.Alignment.LEADING,
javax.swing.GroupLayout.PREFERRED_SIZE, 255,
javax.swing.GroupLayout.PREFERRED_SIZE))
        .addContainerGap())
    );

    pack();
} // </editor-fold> // GEN-END: initComponents

private void jtflActionPerformed(java.awt.event.ActionEvent evt)
{ // GEN-FIRST: event_jtflActionPerformed
    // TODO add your handling code here:
} // GEN-LAST: event_jtflActionPerformed

private void jButton1ActionPerformed(java.awt.event.ActionEvent evt)
{ // GEN-FIRST: event_jButton1ActionPerformed

    str1=jtfl1.getText();
    str2=jtf2.getText();
    str3=jtf3.getText();
    str4=jtf4.getText();
    if(str1.equals("") || str3.equals("") || str4.equals(""))
    {
        p=new pop_up();
        java.awt.EventQueue.invokeLater(new Runnable() {

            public void run() {
                p.setVisible(true);
            }

        });
    }
    else
    {
        this.setVisible(false);
        java.awt.EventQueue.invokeLater(new Runnable() {
            public void run() {
                new client(str1+" : "+str2,str3).setVisible(true);
            }
        });
    }
} // GEN-LAST: event_jButton1ActionPerformed

/**
 * @param args the command line arguments
 */

// Variables declaration - do not modify // GEN-BEGIN: variables
private javax.swing.JButton jButton1;
private javax.swing.JLabel jLabel1;
private javax.swing.JLabel jLabel10;
private javax.swing.JLabel jLabel2;
private javax.swing.JLabel jLabel3;
private javax.swing.JLabel jLabel4;

```

```

private javax.swing.JLabel jLabel5;
private javax.swing.JLabel jLabel6;
private javax.swing.JLabel jLabel7;
private javax.swing.JLabel jLabel8;
private javax.swing.JLabel jLabel9;
private javax.swing.JSeparator jSeparator1;
private javax.swing.JTextField jtf1;
private javax.swing.JTextField jtf2;
private javax.swing.JTextField jtf3;
private javax.swing.JTextField jtf4;
// End of variables declaration//GEN-END:variables
}

```

4.8 Cl_infol.java:

```

package chat_application;

import java.io.BufferedReader;
import java.io.DataInputStream;
import java.io.IOException;
import java.io.InputStream;
import java.io.InputStreamReader;
import java.io.OutputStream;
import java.io.OutputStreamWriter;
import java.io.PrintWriter;
import java.net.ConnectException;
import java.net.Socket;
import java.net.SocketException;
import exception.*;
import java.util.logging.Level;
import java.util.logging.Logger;
public class Clinfol extends Thread
{
    Socket s;
    BufferedReader br2;
    String str;
    PrintWriter pr;
    static boolean mode;
    public Clinfol()
    {

    }
    private String line;

    Clinfol(Socket s)
    {
        this.s=s;
    }

    @Override

```

```

public void run()
{
    try
    {
        br2=new BufferedReader(new InputStreamReader(s.getInputStream()));
        while(true)
        {
            line = br2.readLine();
            if(!line.equals(""))
                System.out.println("server : "+line);
        }
    }
    catch(ConnectException e)
    {
        System.err.println("Unable to connect !!! Raising AUTO SERVER
.....");

        Serinfo instance=new Serinfo();
        try {
            mode = instance.getMode();
        } catch (IOException ex) {

Logger.getLogger(Clinfol.class.getName()).log(Level.SEVERE, null, ex);
        }

        instance.startserver(null);

        Thread ts=new Thread(instance);
        ts.start();
    }
    catch(IOException e)
    {
        this.interrupt();
        System.err.println("IOException !!! Raising AUTO SERVER
.....");

        Serinfo instance=new Serinfo();
        try {
            mode = instance.getMode();    // change of mode error
            .....
        } catch (IOException ex) {

Logger.getLogger(Clinfol.class.getName()).log(Level.SEVERE, null, ex);
        }

        instance.startserver(null);

        Thread ts=new Thread(instance);
        ts.start();
    }
}

```

4.9 Client.java:

```
package chat_application;
import java.awt.Graphics;
public class client extends javax.swing.JFrame {

    /** Creates new form client */
    public client()
    {
        initComponents();
    }

    client(String str1, String str3)
    {
        this();
        jid.setText(str1);
        tag.setText(str3);
    }
    @SuppressWarnings("unchecked")
    // <editor-fold defaultstate="collapsed" desc="Generated Code"> //GEN-BEGIN:
    private void initComponents() {

        jLayeredPanel1 = new javax.swing.JLayeredPane();
        jPanel1 = new javax.swing.JPanel();
        jid = new javax.swing.JLabel();
        tag = new javax.swing.JLabel();
        jLabel8 = new javax.swing.JLabel();
        jLabel9 = new javax.swing.JLabel();
        jSeparator1 = new javax.swing.JSeparator();
        jScrollPane1 = new javax.swing.JScrollPane();
        list = new javax.swing.JList();
        jbl1 = new javax.swing.JButton();
        jTabbedPane1 = new javax.swing.JTabbedPane();
        jInternalFrame1 = new javax.swing.JInternalFrame();
        jInternalFrame2 = new javax.swing.JInternalFrame();
        jTextField1 = new javax.swing.JTextField();
        jButton2 = new javax.swing.JButton();

        setDefaultCloseOperation(javax.swing.WindowConstants.EXIT_ON_CLOSE);

        jid.setBackground(new java.awt.Color(255, 153, 0));
        jid.setForeground(new java.awt.Color(102, 0, 0));
        jid.setText("Id - custom .");

        tag.setForeground(new java.awt.Color(102, 0, 0));
        tag.setText("EMAIL ID");

        jLabel8.setText("Inbox");

        jLabel9.setText("Settings");
    }
}
```

```

list.setModel(new javax.swing.AbstractListModel() {
    String[] strings = { "Item 1", "Item 2", "Item 3", "Item 4",
"Item 5" };
    public int getSize() { return strings.length; }
    public Object getElementAt(int i) { return strings[i]; }
});
list.setCursor(new java.awt.Cursor(java.awt.Cursor.HAND_CURSOR));
list.addMouseListener(new java.awt.event.MouseAdapter() {
    public void mouseClicked(java.awt.event.MouseEvent evt) {
        listMouseClicked(evt);
    }
});
jScrollPane.setViewportView(list);

jbl.setBackground(new java.awt.Color(255, 255, 255));
jbl.setFont(new java.awt.Font("Tahoma", 3, 12)); // NOI18N
jbl.setForeground(new java.awt.Color(102, 0, 0));
jbl.setText("<html> <br>\nR <br>\nE <br>\nF <br>\nR <br>\nE  
<br>\nS <br>\nH <br>\n<html>");
jbl.setToolTipText("refreshes your friend list");
jbl.addActionListener(new java.awt.event.ActionListener() {
    public void actionPerformed(java.awt.event.ActionEvent evt) {
        jblActionPerformed(evt);
    }
});

jInternalFrame1.setTitle("Block Friend");

javax.swing.GroupLayout jInternalFrame1Layout = new
javax.swing.GroupLayout(jInternalFrame1.getContentPane());
jInternalFrame1.getContentPane().setLayout(jInternalFrame1Layout);
jInternalFrame1Layout.setHorizontalGroup(
jInternalFrame1Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
    .addGap(0, 257, Short.MAX_VALUE)
);
jInternalFrame1Layout.setVerticalGroup(
jInternalFrame1Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
    .addGap(0, 87, Short.MAX_VALUE)
);

jTabbedPane1.addTab("Security", jInternalFrame1);

jInternalFrame2.setTitle("Send Friend Request");

jTextField1.setText("enter id");
jTextField1.addActionListener(new java.awt.event.ActionListener()
{
    public void actionPerformed(java.awt.event.ActionEvent evt) {
        jTextField1ActionPerformed(evt);
    }
});

```



```

jButton2.setText("Send Invite For Approval");

javax.swing.GroupLayout jInternalFrame2Layout = new
javax.swing.GroupLayout(jInternalFrame2.getContentPane());

jInternalFrame2.getContentPane().setLayout(jInternalFrame2Layout);
jInternalFrame2Layout.setHorizontalGroup(

jInternalFrame2Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
    .addGroup(jInternalFrame2Layout.createSequentialGroup()

.addGroup(jInternalFrame2Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)

.addGroup(jInternalFrame2Layout.createSequentialGroup()
    .addGap(46, 46, 46)
    .addComponent(jTextField1,
javax.swing.GroupLayout.PREFERRED_SIZE, 110,
javax.swing.GroupLayout.PREFERRED_SIZE))

.addGroup(jInternalFrame2Layout.createSequentialGroup()
    .addGap(46, 46, 46)
    .addComponent(jButton2,
javax.swing.GroupLayout.PREFERRED_SIZE, 110,
javax.swing.GroupLayout.PREFERRED_SIZE))
    .addContainerGap())
);
jInternalFrame2Layout.setVerticalGroup(

jInternalFrame2Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
    .addGroup(jInternalFrame2Layout.createSequentialGroup()
        .addGap(26, 26, 26)
        .addComponent(jTextField1,
javax.swing.GroupLayout.PREFERRED_SIZE, 110,
javax.swing.GroupLayout.PREFERRED_SIZE)
        .addGap(26, 26, 26)
        .addComponent(jButton2,
javax.swing.GroupLayout.PREFERRED_SIZE, 34,
javax.swing.GroupLayout.PREFERRED_SIZE)
        .addContainerGap())
);

jTabbedPane.addTab("Invite", jInternalFrame2);

javax.swing.GroupLayout jPanel1Layout = new
javax.swing.GroupLayout(jPanel1);
jPanel1.setLayout(jPanel1Layout);
jPanel1Layout.setHorizontalGroup(

jPanel1Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
    .addGroup(jPanel1Layout.createSequentialGroup()
        .addContainerGap()
        .addGroup(jPanel1Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.TRAILING)
            .addComponent(jTextField1,
javax.swing.GroupLayout.PREFERRED_SIZE, 110,
javax.swing.GroupLayout.PREFERRED_SIZE)
            .addComponent(jButton2,
javax.swing.GroupLayout.PREFERRED_SIZE, 34,
javax.swing.GroupLayout.PREFERRED_SIZE)
        )
        .addContainerGap())
);

```

```

.addGroup(jPanellLayout.createParallelGroup(javax.swing.GroupLayout.Align
ment.TRAILING)
    .addGroup(jPanellLayout.createSequentialGroup()
        .addContainerGap(198, Short.MAX_VALUE)
        .addComponent(jLabel8)
        .addGap(27, 27, 27)
        .addComponent(jLabel9))
    .addGroup(javax.swing.GroupLayout.Alignment.LEADING,
jPanellLayout.createSequentialGroup()
    .addGap(25, 25, 25)

.addGroup(jPanellLayout.createParallelGroup(javax.swing.GroupLayout.Align
ment.LEADING)
    .addComponent(tag)
    .addComponent(jid))

.addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED, 206,
Short.MAX_VALUE)))
    .addGap(27, 27, 27))
    .addComponent(jSeparator1,
javax.swing.GroupLayout.DEFAULT_SIZE, 319, Short.MAX_VALUE)
    .addGroup(javax.swing.GroupLayout.Alignment.TRAILING,
jPanellLayout.createSequentialGroup()
    .addContainerGap()
    .addComponent(jScrollPane1,
javax.swing.GroupLayout.PREFERRED_SIZE, 153,
javax.swing.GroupLayout.PREFERRED_SIZE)

.addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED, 115,
Short.MAX_VALUE)
    .addComponent(jb1,
javax.swing.GroupLayout.PREFERRED_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE,
javax.swing.GroupLayout.PREFERRED_SIZE))
    .addGroup(jPanellLayout.createSequentialGroup()
        .addContainerGap()
        .addComponent(jTabbedPane1,
javax.swing.GroupLayout.DEFAULT_SIZE, 272, Short.MAX_VALUE)
        .addGap(37, 37, 37))
    );
jPanellLayout.setVerticalGroup(
jPanellLayout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
NG)
    .addGroup(jPanellLayout.createSequentialGroup()

.addGroup(jPanellLayout.createParallelGroup(javax.swing.GroupLayout.Align
ment.BASELINE)
    .addComponent(jLabel8)
    .addComponent(jLabel9))
    .addGap(15, 15, 15)
    .addComponent(jid)

.addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED)
    .addComponent(tag)
    .addGap(23, 23, 23)

```



```

        .addComponent(jSeparator1,
javax.swing.GroupLayout.PREFERRED_SIZE, 10,
javax.swing.GroupLayout.PREFERRED_SIZE)

.addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED)

.addGroup(jPanellLayout.createParallelGroup(javax.swing.GroupLayout.Align
ment.LEADING)
        .addComponent(jb1,
javax.swing.GroupLayout.PREFERRED_SIZE, 134,
javax.swing.GroupLayout.PREFERRED_SIZE)
        .addComponent(jScrollPane1,
javax.swing.GroupLayout.PREFERRED_SIZE, 175,
javax.swing.GroupLayout.PREFERRED_SIZE))
        .addGap(17, 17, 17)
        .addComponent(jTabbedPane1,
javax.swing.GroupLayout.PREFERRED_SIZE, 150,
javax.swing.GroupLayout.PREFERRED_SIZE)
    );

    javax.swing.GroupLayout layout = new
javax.swing.GroupLayout(getContentPane());
    getContentPane().setLayout(layout);
    layout.setHorizontalGroup(

layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
        .addComponent(jPanell, javax.swing.GroupLayout.DEFAULT_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE, Short.MAX_VALUE)
    );
    layout.setVerticalGroup(

layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
        .addGroup(layout.createSequentialGroup()
            .addComponent(jPanell,
javax.swing.GroupLayout.PREFERRED_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE,
javax.swing.GroupLayout.PREFERRED_SIZE)
            .addContainerGap(43, Short.MAX_VALUE))
        );

    pack();
} // </editor-fold> // GEN-END: initComponents

private void jb1ActionPerformed(java.awt.event.ActionEvent evt)
{ // GEN-FIRST:event_jb1ActionPerformed

} // GEN-LAST:event_jb1ActionPerformed

private void jTextField1ActionPerformed(java.awt.event.ActionEvent
evt) { // GEN-FIRST:event_jTextField1ActionPerformed
    // TODO add your handling code here:
} // GEN-LAST:event_jTextField1ActionPerformed

private void listMouseClicked(java.awt.event.MouseEvent evt) { // GEN-
FIRST:event_listMouseClicked

} // GEN-LAST:event_listMouseClicked

```

```

// Variables declaration - do not modify//GEN-BEGIN:variables
private javax.swing.JButton jButton2;
private javax.swing.JInternalFrame jInternalFrame1;
private javax.swing.JInternalFrame jInternalFrame2;
private javax.swing.JLabel jLabel8;
private javax.swing.JLabel jLabel9;
private javax.swing.JLayeredPane jLayeredPanel1;
private javax.swing.JPanel jPanel1;
private javax.swing.JScrollPane jScrollPane1;
private javax.swing.JSeparator jSeparator1;
private javax.swing.JTabbedPane jTabbedPane1;
private javax.swing.JTextField jTextField1;
private javax.swing.JButton jbl1;
private javax.swing.JLabel jid;
private javax.swing.JList list;
private javax.swing.JLabel tag;
// End of variables declaration//GEN-END:variables

```

4.10 Request.java:

```

import java.io.BufferedReader;
import java.io.DataInputStream;
import java.io.IOException;
import java.io.InputStreamReader;
import java.io.OutputStreamWriter;
import java.io.PrintWriter;
import java.net.ConnectException;
import java.net.Socket;
import java.net.SocketException;
import java.util.ArrayList;
public final class Request extends Thread
{
    static ArrayList handlers = new ArrayList(20);
    private Socket socket;
    private BufferedReader in;
    private PrintWriter out;
    private String str;

    public ArrayList gethandler()
    {
        return handlers;
    }
    public Request(Socket socket) throws IOException {
        this.socket = socket;
        out=new PrintWriter(new
OutputStreamWriter(this.socket.getOutputStream()));
        in = new BufferedReader(
            new InputStreamReader(this.socket.getInputStream()));
    }

```

```

    }
    @Override
    public void run() {
        String line;
        synchronized(handlers) {
            handlers.add(this);
        }
        try {

while(true)
        {

            line =in.readLine();
            if(Serinfo.mode==true)
{
                for(int i=0;i<handlers.size();i++)
                {
                    synchronized(handlers) {
                        Request h =
                            (Request)handlers.get(i);
                        h.out.println(line + "\r");
                        h.out.flush();
                    }
                }

                System.out.println(line);
            }

        }
        catch(NullPointerException e)
        {
            System.out.println("client disconnected ....");
        }

        catch(SocketException t)
        {
            System.out.println("client disconnected ....");
        }
        catch(IOException e)
        {
            e.printStackTrace();
        }
        finally {
            try {
                in.close();

                socket.close();
            }
            catch(NullPointerException e)
            {
                System.out.println("client disconnected ....");
            }
            catch(IOException e)
            {
                e.printStackTrace();
            }
        }
    }
}

```

```

        finally {
            synchronized(handlers)
            {
                handlers.remove(this);
            }
        }
    }
}
}

```

4.11 Servergui.java:

```

public class servergui extends javax.swing.JFrame {

    public servergui() {
        initComponents();
    }
    @SuppressWarnings("unchecked")
    // <editor-fold defaultstate="collapsed" desc="Generated Code"> //GEN-BEGIN:
    private void initComponents() {

        jLabel1 = new javax.swing.JLabel();
        jSeparator1 = new javax.swing.JSeparator();
        jTabbedPane1 = new javax.swing.JTabbedPane();
        jInternalFrame1 = new javax.swing.JInternalFrame();
        jInternalFrame2 = new javax.swing.JInternalFrame();
        jInternalFrame3 = new javax.swing.JInternalFrame();
        jInternalFrame4 = new javax.swing.JInternalFrame();
        jLabel2 = new javax.swing.JLabel();
        jScrollPane1 = new javax.swing.JScrollPane();
        jList1 = new javax.swing.JList();
        jScrollPane2 = new javax.swing.JScrollPane();
        jList2 = new javax.swing.JList();
        jScrollPane3 = new javax.swing.JScrollPane();
        jList3 = new javax.swing.JList();
        jButton1 = new javax.swing.JButton();
        jButton2 = new javax.swing.JButton();
        jInternalFrame5 = new javax.swing.JInternalFrame();
        jInternalFrame6 = new javax.swing.JInternalFrame();

        setDefaultCloseOperation(javax.swing.WindowConstants.EXIT_ON_CLOSE);

        jLabel1.setFont(new java.awt.Font("Tahoma", 1, 36));
        jLabel1.setText("SeRvEr CoNtRoLs");

        jInternalFrame1.setVisible(true);

        javax.swing.GroupLayout jInternalFrame1Layout = new
        javax.swing.GroupLayout(jInternalFrame1.getContentPane());
        jInternalFrame1Layout.setHorizontalGroup(
            jInternalFrame1Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
            .add(jLabel1)
            .add(jList1)
            .add(jList2)
            .add(jList3)
            .add(jButton1)
            .add(jButton2)
            .add(jInternalFrame5)
            .add(jInternalFrame6)
        );
        jInternalFrame1Layout.setVerticalGroup(
            jInternalFrame1Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
            .add(jLabel1)
            .add(jList1)
            .add(jList2)
            .add(jList3)
            .add(jButton1)
            .add(jButton2)
            .add(jInternalFrame5)
            .add(jInternalFrame6)
        );
    }
}

```

```

        jInternalFrame1Layout.setHorizontalGroup(

jInternalFrame1Layout.createParallelGroup(javax.swing.GroupLayout.Alignme
nt.LEADING)
        .addGap(0, 504, Short.MAX_VALUE)
        );
        jInternalFrame1Layout.setVerticalGroup(

jInternalFrame1Layout.createParallelGroup(javax.swing.GroupLayout.Alignme
nt.LEADING)
        .addGap(0, 238, Short.MAX_VALUE)
        );

        jTabbedPane1.addTab(" Main ", jInternalFrame1);

        javax.swing.GroupLayout jInternalFrame2Layout = new
javax.swing.GroupLayout(jInternalFrame2.getContentPane());

jInternalFrame2.getContentPane().setLayout(jInternalFrame2Layout);
        jInternalFrame2Layout.setHorizontalGroup(

jInternalFrame2Layout.createParallelGroup(javax.swing.GroupLayout.Alignme
nt.LEADING)
        .addGap(0, 504, Short.MAX_VALUE)
        );
        jInternalFrame2Layout.setVerticalGroup(

jInternalFrame2Layout.createParallelGroup(javax.swing.GroupLayout.Alignme
nt.LEADING)
        .addGap(0, 238, Short.MAX_VALUE)
        );

        jTabbedPane1.addTab(" Configure ", jInternalFrame2);

        javax.swing.GroupLayout jInternalFrame3Layout = new
javax.swing.GroupLayout(jInternalFrame3.getContentPane());

jInternalFrame3.getContentPane().setLayout(jInternalFrame3Layout);
        jInternalFrame3Layout.setHorizontalGroup(

jInternalFrame3Layout.createParallelGroup(javax.swing.GroupLayout.Alignme
nt.LEADING)
        .addGap(0, 504, Short.MAX_VALUE)
        );
        jInternalFrame3Layout.setVerticalGroup(

jInternalFrame3Layout.createParallelGroup(javax.swing.GroupLayout.Alignme
nt.LEADING)
        .addGap(0, 238, Short.MAX_VALUE)
        );

        jTabbedPane1.addTab(" Statistics ", jInternalFrame3);

        jLabel2.setText("Client Name
| Time Conected | Actions");

        jList1.setModel(new javax.swing.AbstractListModel() {

```

```

        String[] strings = { "Client 1", "Client 2", "Client 3",
"Client 4", "Client 5" };
        public int getSize() { return strings.length; }
        public Object getElementAt(int i) { return strings[i]; }
    });
    jScrollPane1.setViewportViewView(jList1);

    jList2.setModel(new javax.swing.AbstractListModel() {
        String[] strings = { "Item 1", "Item 2", "Item 3", "Item 4",
"Item 5" };
        public int getSize() { return strings.length; }
        public Object getElementAt(int i) { return strings[i]; }
    });
    jScrollPane2.setViewportViewView(jList2);

    jList3.setModel(new javax.swing.AbstractListModel() {
        String[] strings = { "Item 1", "Item 2", "Item 3", "Item 4",
"Item 5" };
        public int getSize() { return strings.length; }
        public Object getElementAt(int i) { return strings[i]; }
    });
    jScrollPane3.setViewportViewView(jList3);

    jButton1.setText("Kick");

    jButton2.setText("Ban");

    javax.swing.GroupLayout jInternalFrame4Layout = new
javax.swing.GroupLayout(jInternalFrame4.getContentPane());
    jInternalFrame4.getContentPane().setLayout(jInternalFrame4Layout);
    jInternalFrame4Layout.setHorizontalGroup(
        jInternalFrame4Layout.createParallelGroup(javax.swing.GroupLayout.Alignment
nt.LEADING)
            .addGroup(jInternalFrame4Layout.createSequentialGroup()
                .addGroup(jInternalFrame4Layout.createParallelGroup()
                    .addContainerGap()

                .addGroup(jInternalFrame4Layout.createParallelGroup(javax.swing.GroupLayout
ut.Alignment.LEADING, false)
                    .addComponent(jLabel2,
javax.swing.GroupLayout.PREFERRED_SIZE, 424,
javax.swing.GroupLayout.PREFERRED_SIZE)

                .addGroup(jInternalFrame4Layout.createParallelGroup()
                    .addComponent(jScrollPane1,
javax.swing.GroupLayout.PREFERRED_SIZE, 175,
javax.swing.GroupLayout.PREFERRED_SIZE)
                    .addGroup(33, 33, 33)
                    .addComponent(jScrollPane2,
javax.swing.GroupLayout.PREFERRED_SIZE, 65,
javax.swing.GroupLayout.PREFERRED_SIZE)

                .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED,
javax.swing.GroupLayout.DEFAULT_SIZE, Short.MAX_VALUE)

```

```

        .addComponent(jScrollPane3,
javax.swing.GroupLayout.PREFERRED_SIZE, 106,
javax.swing.GroupLayout.PREFERRED_SIZE))

.addGroup(jInternalFrame4Layout.createSequentialGroup()
        .addComponent(jButton1,
javax.swing.GroupLayout.PREFERRED_SIZE, 87,
javax.swing.GroupLayout.PREFERRED_SIZE)

.addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED)
        .addComponent(jButton2,
javax.swing.GroupLayout.PREFERRED_SIZE, 90,
javax.swing.GroupLayout.PREFERRED_SIZE)))
        .addContainerGap(70, Short.MAX_VALUE))
);
jInternalFrame4Layout.setVerticalGroup(
jInternalFrame4Layout.createParallelGroup(javax.swing.GroupLayout.Alignme
nt.LEADING)
        .addGroup(jInternalFrame4Layout.createSequentialGroup()
                .addContainerGap()
                .addComponent(jLabel2)

.addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.UNRELATED)

.addGroup(jInternalFrame4Layout.createParallelGroup(javax.swing.GroupLayo
ut.Alignment.TRAILING)
        .addComponent(jScrollPane1,
javax.swing.GroupLayout.PREFERRED_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE,
javax.swing.GroupLayout.PREFERRED_SIZE)
        .addComponent(jScrollPane2,
javax.swing.GroupLayout.PREFERRED_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE,
javax.swing.GroupLayout.PREFERRED_SIZE)
        .addComponent(jScrollPane3,
javax.swing.GroupLayout.PREFERRED_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE,
javax.swing.GroupLayout.PREFERRED_SIZE))

.addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED, 38,
Short.MAX_VALUE)

.addGroup(jInternalFrame4Layout.createParallelGroup(javax.swing.GroupLayo
ut.Alignment.BASELINE)
        .addComponent(jButton1)
        .addComponent(jButton2))
        .addContainerGap())
);

jTabbedPanel.addTab("  Clients  ", jInternalFrame4);

javax.swing.GroupLayout jInternalFrame5Layout = new
javax.swing.GroupLayout(jInternalFrame5.getContentPane());

jInternalFrame5.getContentPane().setLayout(jInternalFrame5Layout);
jInternalFrame5Layout.setHorizontalGroup(

```

```

jInternalFrame5Layout.createParallelGroup(javax.swing.GroupLayout.Alignme
nt.LEADING)
    .addGap(0, 504, Short.MAX_VALUE)
);
jInternalFrame5Layout.setVerticalGroup(

jInternalFrame5Layout.createParallelGroup(javax.swing.GroupLayout.Alignme
nt.LEADING)
    .addGap(0, 238, Short.MAX_VALUE)
);
jTabbedPane1.addTab("  Bans    ", jInternalFrame5);

javax.swing.GroupLayout jInternalFrame6Layout = new
javax.swing.GroupLayout(jInternalFrame6.getContentPane());
jInternalFrame6.getContentPane().setLayout(jInternalFrame6Layout);
jInternalFrame6Layout.setHorizontalGroup(

jInternalFrame6Layout.createParallelGroup(javax.swing.GroupLayout.Alignme
nt.LEADING)
    .addGap(0, 504, Short.MAX_VALUE)
);
jInternalFrame6Layout.setVerticalGroup(

jInternalFrame6Layout.createParallelGroup(javax.swing.GroupLayout.Alignme
nt.LEADING)
    .addGap(0, 238, Short.MAX_VALUE)
);
jTabbedPane1.addTab("  Console  ", jInternalFrame6);

javax.swing.GroupLayout layout = new
javax.swing.GroupLayout(getContentPane());
getContentPane().setLayout(layout);
layout.setHorizontalGroup(

layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
    .addComponent(jLabel1,
javax.swing.GroupLayout.Alignment.TRAILING,
javax.swing.GroupLayout.DEFAULT_SIZE, 539, Short.MAX_VALUE)
    .addComponent(jSeparator1,
javax.swing.GroupLayout.DEFAULT_SIZE, 539, Short.MAX_VALUE)
    .addGroup(layout.createSequentialGroup())
        .addContainerGap()
        .addComponent(jTabbedPane1,
javax.swing.GroupLayout.DEFAULT_SIZE, 519, Short.MAX_VALUE)
        .addContainerGap())
);
layout.setVerticalGroup(

layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
    .addGroup(layout.createSequentialGroup())
        .addComponent(jLabel1,
javax.swing.GroupLayout.PREFERRED_SIZE, 64,
javax.swing.GroupLayout.PREFERRED_SIZE)

```



```

.addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED)
    .addComponent(jSeparator1,
javax.swing.GroupLayout.PREFERRED_SIZE, 10,
javax.swing.GroupLayout.PREFERRED_SIZE)

.addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.UNRELATED)
    .addComponent(jTabbedPane1,
javax.swing.GroupLayout.DEFAULT_SIZE, 301, Short.MAX_VALUE)
    .addContainerGap()

);

pack();
} // </editor-fold> // GEN-END: initComponents

```

```

// Variables declaration - do not modify // GEN-BEGIN: variables
private javax.swing.JButton jButton1;
private javax.swing.JButton jButton2;
private javax.swing.JInternalFrame jInternalFrame1;
private javax.swing.JInternalFrame jInternalFrame2;
private javax.swing.JInternalFrame jInternalFrame3;
private javax.swing.JInternalFrame jInternalFrame4;
private javax.swing.JInternalFrame jInternalFrame5;
private javax.swing.JInternalFrame jInternalFrame6;
private javax.swing.JLabel jLabel1;
private javax.swing.JLabel jLabel2;
private javax.swing.JList jList1;
private javax.swing.JList jList2;
private javax.swing.JList jList3;
private javax.swing.JScrollPane jScrollPane1;
private javax.swing.JScrollPane jScrollPane2;
private javax.swing.JScrollPane jScrollPane3;
private javax.swing.JSeparator jSeparator1;
private javax.swing.JTabbedPane jTabbedPane1;
// End of variables declaration // GEN-END: variables

```

```

}

```

Chapter 5

5.1 Testing

Testing is the process of executing the program(s) with the intention of finding out errors. During testing, the program to be tested is executed with a set of test cases and the output of the programs for the test case is evaluated to determine if the program is performing as it is expected to be. The success of testing in revealing errors in programs depends critically on the test cases

5.1.1 LEVELS OF TESTING

UNIT TESTING: The first level of testing is called unit testing. In this different modules are tested against the specifications produced during design of the modules. Unit testing is essentially for verification of the code produced during coding phase, and hence the goal is to test the internal logic of the modules. The programmer of the module typically does it.

INTEGRATION TESTING: The next level of testing is often called integration testing. In this, many unit-tested modules are combined into subsystems, which are then tested. The goal is here to see if the modules can be integrated properly. Hence, the emphasis is on testing interfaces between modules. The testing activity can be considered testing the design. The integration plan specifies the steps and order in which modules are combined to realize the full system. After each integration step, the partially integrated system is tested. An important factor that guides the integration is the module dependency graph.

SYSTEM TESTING: System tests are designed to validate a fully developed system to assure that it meets its requirements. There are essentially three main kinds of system testing:

ALPHA TESTING: Alpha refers to the system testing carried out by the test team within the developing organization.

BETA TESTING: Beta testing is the system testing performed by a select group of friendly customers.

ACCEPTANCE TESTING: Acceptance testing is the system testing performed by the customer to determine whether to accept or reject the delivery of the system.

5.1.2 TYPES OF TESTING:

BLACK BOX TESTING: This testing is also known as functional testing. The basis for deciding test cases in functional testing is the requirements or specifications of the system or modules. For the entire system test cases are designed from the requirement specification document from the system. There are number of techniques that can be used to select test cases that have been found to be very successful in detecting errors. Some of them are:

Equivalence class partitioning: In this we divide the domain of all the inputs into a set of equivalence classes. That is we want to identify classes of test cases such that the success of one test case in a class implies the success of others. It is often useful to consider equivalence classes in the output. For an output equivalence class, the goal is to generate test cases such that the output of that test case lies in the output equivalence class.

Boundary value analysis: In boundary value analysis, we choose an input for a test case from an equivalence class, such that the input lies at the edge of the equivalence class. Boundary value test cases are also called “extreme cases”.

Cause-effect graphing: It is a technique that aids in selecting combinations of input conditions in a systematic way. A cause is a distinct input condition, and an effect is a distinct output condition. Each condition forms a node in the cause-effect graph. Beyond generating high-yield test cases, it also aids the understanding of the functionality of the system, because the tester must identify the distinct causes and effects.

Special cases: It depends on the data structures and the function of the module. There are no rules to determine special cases, and the tester has to use his intuition and experience to identify such test cases. Consequently, determining special cases is also called “error guessing.”

WHITE BOX TESTING: There are several white box-testing strategies. Each testing strategy is based on some heuristic. One white box testing strategy is said to be stronger than another strategy, if all types of errors detected by the first testing strategy (say B) are also detected by the second testing strategy (say A), and the second strategy additionally detects some more types of errors. When two testing strategies detect errors that are different at least with respect to some types of errors, they are then called complementary.

Statement coverage: It aims to design test cases so that every statement in the program is executed at least once. The principal idea is that unless we execute a statement, we have no way of determining if error exists in that statement.

Branch coverage: In this strategy, test cases are designed to make each branch condition assume true and false value in turn. It is also known as “edge-testing” as in this testing scheme, each edge of a program’s control flow graph is traversed at least once.

Condition coverage: In this test cases are designed to make each component of a composite conditional expression assume both true and false values. Thus, condition testing is a stronger testing strategy than branch testing. For conditional coverage, the number of test cases increases exponentially with the number of component conditions.

Path coverage: It requires us to design test cases such that all linearly independent paths in the program are executed at least once. These paths are defined in terms of control-flow graph of a program.

As testing forms the first step towards determining the errors in a program it should be properly carried out.

UNIT TESTING:

Unit testing was carried out for each module against the specifications produced during the design of the module.

Unit testing of each program and module was done with the following perception.

USER INTERFACE:

User interface was tested which gave rise to more user understandable errors and help messages.

INTERNAL LOGIC:

While testing a module, the internal logic was tested.

INTEGRATED TESTING:

Tint the integrated testing, the unit-tested modules were combined into subsystems and then tested.

The strategies for integrated tested comprised of :

- Performance time testing.
- Logical cycle of data.
- Test data.
- Live data obtained from users.

5.1.3.3 Environments

The environment on which we have performed the firewall configuration and testing was a linux workstation that has virtual network of virtual computers that are managed by the MLN tool. The network schema is described in the lab document. The tests are done using external and internal UML instances.

CHAPTER 6

Conclusion:

The rules in this application can implement the following situations:

- ✓ Text chat (peer to peer)
- ✓ Text chat (peer to all) | Broadcast mode
- ✓ Voice chat (peer to peer)
- ✓ Voice chat (Broadcast mode)
- ✓ Server failure support
- ✓ Centralised server

Future Recommendations:

- ✓ The same can be implemented on a cloud based approach.
- ✓ Distributed server can be implemented instead of centralised server.
- ✓ Video chat can be implemented.
- ✓ Implementing functionality to allow user input of the search keyword.
- ✓ File transfer can be done using the same server and client architecture

REFERENCES

- Behrouz A. Forouzan, 2003, TCP/IP Protocol Suite, Mc Graw Hill.
- The "Networking" code in Linux, Teunis J. Ott and Rahul Jain July 29, 2004
- D0 Code – comprehensively cross – referenced and searchable code <http://www-d0.fnal.gov/D0Code/source/>
- <http://gnumonks.org/ftp/pub/doc/packet-journey-2.4.html>
- <http://gnumonks.org/papers/netfilter lk2000/presentation.html> Overview of Routing and Packet Filter
- Foley, M. (2002). Instant Messaging Reference in an Academic Library: A Case Study. *College & Research Libraries*, 63(1), 36-45. Retrieved 5 March 2004 from <http://www.vrd.org/conferences/VRD2001/proceedings/foley.shtml>.
- Francoeur, S. (2001). An Analytical Survey of Chat Reference Services. *Reference Services Review*, 29(3), 189-203.