**A**

**PROJECT SYNOPSIS**

**ON**

**LAN MULTIUSER CHAT IN JAVA**

**Submitted in Partial Fulfillment for the Award of**

**Bachelor in Computer Applications**

**Of**

**VSICS, Kanpur**

**(2015-2016)**

**Name of the Project:** LAN MULTIUSER CHAT on Java

**Abstract:**

A very simple cross--‐platform client--‐server chat application has been implemented in Java. Its design is described, limitations are discussed, and improvements are proposed and

A user manual is included.

**Introduction:**

Implementing a chat server application provides a good opportunity for a beginner to design and implement a network-­‐based system. The design is very simple. It is implemented in Java, since is easy to program in, it precludes the need to deal with low-­‐level memory management and includes powerful libraries for sockets and threads.

**Related Work/Background**

Implementing a chat server application is one of the most popular network programming projects with newbie programmers. Tons of free source code is available on the web. Excellent highly configurable applications are available both as open source well as proprietary software. Some examples of open source Chat applications implemented in Java are Llama Chat, FreeCS, Chipchat and OpenCHAT. With little experience in network programming as well as a short duration for the project, my intension was not to match or improve the existing implementations button implement a basic version on my own.

**Design**

TCP is used as the transport layer protocol, since it provides reliable delivery which is critical for the given application. TCP does not provide timing guarantee, which is not very important in the given scenario.

**Server**

The server is implemented as a singleton class. The main thread opens a server socket on the local inetaddress and port 2000, which has been arbitrarily chosen and hard coded. It then waits for clients to connect to it. When a client connects to the server, it creates a separate thread as well as a User object dedicated to it. The server maintains a hash map of User objects associated with the clients connected to it hashed against the user name. The thread dedicated to the client opens a buffered stream to read input messages from the client and a PrintWriter stream to send messages to the client.

The messages going from a client to the server are strings with atleast two words separated by a space. The first word is the header, which is the destination username. The messages going from the server to a client are strings with at least two words separated by a space. The first word is the header, which is the source username.

The thread dedicated to the client waits for incoming messages in the input buffered stream, tokenizes the messages into the header and the message string, and checks if the destination user is online by looking into the hash map. If the user is online, it attaches a new header displaying the source user name to the beginning of the message string and copies it to the print writer stream of the User object associated with the destination user. If the destination user isn’t online, it sends back an error message to the source client. Messages destined for the server (control messages) contain the string “server” as the header. Control messages are sent to know who is online and to exit. In case of the former, the thread dedicated to the user, gets all the keys from the hash map and returns a string containing the online user names to the user. When exit is requested, the thread closes the streams and the socket associated with the client and exits.

**Client**

The Client is implemented using two threads, one each for incoming and outgoing messages. The main thread opens the socket and connects to the server. It then opens input buffered stream and print writer for incoming and outgoing messages respectively. It also creates a buffered stream to read from the console. The main thread takes care of the outgoing message, while another thread deals with incoming messages. In order to send a message, the user types the destination user name following by the message string on the console, which is then read into the buffered stream reading from the console. The main thread creates the formatted message by adding the destination user name to the beginning of the message string and writes it to the print writer stream. Control messages have the string “server” as the header.

The reader thread, waits for incoming messages on the buffered stream. When a message arrives, it tokenizes it into the header and the message string and prints <source user> says: <message string> on the console. Responses from the server are displayed as “server says: <message string>”. When exit is requested, the client sends exit message to the server, waits for its response, and closes the streams and exits.

**Limitations**

There is no graphical user interface (GUI). So there are no chat windows and the user needs to write destination user name before message string. Also the user needs to enquire, who is online, rather than the server automatically updating the list of users logged in a window.

The absence of unique user identifications will lead to the nickname collision problem.

There is no encryption of message strings being sent across the network.

**Possible Improvements**

Limitations mentioned above can be addressed in improvements. A graphical user interface (GUI) will preclude the need to write the name of destination user when sending messages and make it more users friendly. Also users will not need to ask server who is online. A window displaying the logged in users, which is periodically updated will be a great enhancement.

In the current design, the Server hashes the users using usernames as keys. This creates problem with multiple clients using the same name. This can be addressed by hashing with IP and port number.

Current design does not encrypt the text strings. An enhanced design can use encryption of string using SSL for improved security.

A database of users containing username and password can be coupled with the existing design to maintain user accounts.

An improved version can include multiple servers, serving different geographical locations, while talking to each other. This will preclude messages between clients located close to each other being routed through a server located in a far off location, thus decreasing the delay.

**Objective**

The aim of this project is to develop desktop chat application incorporated with java multi threaded client-server program which would allow users to communicated private and public way and share resources while chatting and archive communicated messages.

This report is to use java, swing, multi threading and TCP-IP technology to design and create desktop based multi-user chat application design for chat and communicated over internet.

The methodology for this report is to understand how to carry out research into multi-user chat application, resource sharing while communicating through the Internet. Also finding out how users would feel. Interviews and research would be ongoing before starting the actual designing to identify the key elements of the project.

**ABOUT THE PROJECT : -**

The users from different places all over the world can communicate with each other through Internet by using this project. Now most of people uses internet mailing concept to send and messages between them. Because it is faster and efficient than existing system like sending letters and telegraphs. Best example for Internet mailing is Chat Server system.

This project is user-friendly, satisfies all needs of the user. User can send and receive messages quickly by using this project. It provides all security measures to secure the stored information from unauthorized users.

**Existing System: -**

The transfer of messages can be done only between two clients or users in the existing manual system. It is not having multi-user environment. When sending a picture message from a client to another client the format of message cannot be changed and it is not possible to add graphical emotional pictures to message. It is limited to only two clients. The existing Chat Server System is only meant for transfer of messages from one client to the other. But the messages cannot be formatted and can't add graphical emotional pictures to sending the picture messages, which makes look of messages more attractive. Also sending and receiving greetings between users on different occasions is not possible in existing system. User will not get facility of chatting with his added friends exclusively through online in existing system.

**Proposed System: -**

The performance of agents who are online in the proposed system is monitored by the PDS [Productivity Dialing System]

Some best examples are: Avaia, Maclenn, SRC & so on. Also it is used to improve the performance of the proposed system

Chat server application is the one which fulfills the all aspects of user and has all the facilities of a chat application. This system provides user id to each and every system user. The online users can communicate with each other at anytime, anywhere through internet in chat server. After a user logs into system using his user id, he can view a list of online users. User can start chatting with any online user and he can involve in group chat also.

**The font size and style of the texts in the message can be changed by user.**

**This application also provides facility of including graphical emotional pictures when sending message. The message will look more attractive by adding emotional pictures. User can browse in the application and at the same time he can chat with other users who are online.**

**Client Module:**

The login page is provided to the chat server by this module. The user can need to provide his user id and password in the login page for accessing the chat server application. The user can add image to the message when sending it to other users in this system. After login, the user will be directed to home page where he will get text area to enter the message, message size and style can be changed and online users list also he will get. The profile of the user can be updated in this application. This system provides facility of adding any online users to user’s friend list and he can start chatting with them.

**Server Module:**

For checking whether given user information is correct or wrong this module is used. The authorized user will get all the facilities provided by this application through. The list of users who are online will be displayed by this module. The functions like updating existing user profile, allowing new user to create account in this application will be performed by this module. This module can access user information through its connection to the database server.

**Advantages of Proposed System: -**

The major advantages of the proposed system i.e. Chat Server are given as following:

The Chat Server system is developed by using the latest version of Java Swing on the JAVA technology. The Java technology has in built support for the drivers of different databases so this proposed system also supports different types RDBMS packages in storing the information. In this system all individual system are integrated with each other to provide smooth communication among different technologies and solutions on the JAVA platform. The proposed system is completely platform independent and architecture independent.

**Different types of media used to create communication between users are telephonic, telegrams, fax etc. The existing system like concept of sending letters and telegraphs is taking more time to send/receive messages between two persons. To give a solution to problems in existing system proposed** chat server system **is developed. In this system, the online users who are from different locations all over the world can communicate with each other through Internet.**

**The information stored in the proposed system is completely protected by the high level security features of the chat server system. Chat Server is more secure than existing system. The Chat server system delivers messages to recipients quickly and efficiently. It is user-friendly and designed to provide complete user satisfaction.**

**EXISTING CHATSERVER SYSTEM AND ITS LIMITATIONS**  
  
Existing System:

The client from one side can send messages to another client through telegraphs, letters in this system. But biggest disadvantage of existing system is letters, telegraphs will be delivered in two or three days. It is difficult to send quick message to client in the existing system. It does not support Multi user environment. User will not get the facility of adding new friends and chatting with added friend. The graphical emotional pictures cannot be added to send the picture messages and formatting of texts like changing font size and font style is not possible. Also user cannot send and receive greetings on different occasions in the existing system.

**PROPOSED SYSTEM**

Chat Server application is user friendly and fulfills all the requirements of the user. It provides all facilities of a chat application, after successful registration the user will get a user id by the system. User needs to login to the system using his user id, then he will get list of online users and he can communicate with any online user. The delivery of message is quick in the proposed system. In this system, the users from different locations can communicate with each other at any time. The font style and size of the text in the message can be changed to make look of message more attractive. Also he will get facility of adding graphicalemotional pictures to the sending message. User can browse web and can chat with other users simultaneously.

**Client Module:**

The login page for system user will be created by using this module. The user can login to the system by providing the user id and password in this page. After successful login, he will be directed to his home page. In home page all options will be displayed like, area to enter message text, formatting text options, list of online users, updating his profile. The user can include emotion pictures, images in the sending message. He also get facility of adding any online friends to his friend list.

**Server Module:**

The entered user information during registration will be get validated by

the server module. This module restricts the unauthorized users to use facilities of application. The functions like the user profile updating, creating a new user account etc will be performed by this module. By using its connection to database server it can access information in the database.

**Users of the System:**

* 1. User
  2. Admin

**Non Functional Requirements:**

**Performance Requirements**

Some Performance requirements identified is listed below:

* The database shall be able to accommodate a minimum of 10,000 records of students.
* The software shall support use of multiple users at a time.
* There are no other specific performance requirements that will affect development.

#### Safety Requirements

The database may get crashed at any certain time due to virus or operating system failure. Therefore, it is required to take the database backup.

#### Security Requirements

Some of the factors that are identified to protect the software from accidental or malicious access, use, modification, destruction, or disclosure are described below. Keep specific log or history data sets

* Assign certain functions to different modules
* Restrict communications between some areas of the program
* Check data integrity for critical variables

**Technologies to be used**

1. **Java Platform**

Java is a set of several computer software products and specifications from Sun Microsystems (which has since merged with Oracle Corporation), that together provide a system for developing application software and deploying it in a cross-platform computing environment. Java is used in a wide variety of computing platforms from embedded devices and mobile phones on the low end, to enterprise servers and supercomputers on the high end. While less common on desktop computers, Java applets are sometimes used to provide improved and secure functions while browsing the World Wide Web.

## History

[James Gosling](http://en.wikipedia.org/wiki/James_Gosling), [Mike Sheridan](http://en.wikipedia.org/wiki/Mike_Sheridan), and [Patrick Naughton](http://en.wikipedia.org/wiki/Patrick_Naughton) initiated the Java language project in June 1991. Java was originally designed for interactive television, but it was too advanced. The language was initially called [*Oak*](http://en.wikipedia.org/wiki/Oak_%28programming_language%29) after an [oak tree](http://en.wikipedia.org/wiki/Oak) that stood outside Gosling's office; it went by the name *Green* later, and was later renamed *Java*, from a list of random words. Gosling aimed to implement a [virtual machine](http://en.wikipedia.org/wiki/Virtual_machine) and a language that had a familiar [C](http://en.wikipedia.org/wiki/C_%28programming_language%29)/[C++](http://en.wikipedia.org/wiki/C%2B%2B) style of notation.

Sun Microsystems released the first public implementation as Java 1.0 in 1995. It promised "[Write Once, Run anywhere](http://en.wikipedia.org/wiki/Write_once,_run_anywhere)" (WORA), providing no-cost run-times on popular [platforms](http://en.wikipedia.org/wiki/Computing_platform). Fairly secure and featuring configurable security, it allowed network- and file-access restrictions. Major web browsers soon incorporated the ability to run Java [*applets*](http://en.wikipedia.org/wiki/Applet) within web pages, and Java quickly became popular. With the advent of *Java 2* (released initially as J2SE 1.2 in December 1998–1999), new versions had multiple configurations built for different types of platforms. For example, *J2EE* targeted enterprise applications and the greatly stripped-down version *J2ME* for mobile applications (Mobile Java). *J2SE* designated the Standard Edition. In 2006, for marketing purposes, Sun renamed new *J2* versions as [*Java EE*](http://en.wikipedia.org/wiki/Java_Platform,_Enterprise_Edition), [*Java ME*](http://en.wikipedia.org/wiki/Java_Platform,_Micro_Edition), and [*Java SE*](http://en.wikipedia.org/wiki/Java_Platform,_Standard_Edition), respectively.

Sun make most of its Java implementations available without charge, despite their [proprietary software](http://en.wikipedia.org/wiki/Proprietary_software) status. Sun generated revenue from Java through the selling of licenses for specialized products such as the Java Enterprise System. Sun distinguishes between its [Software Development Kit (SDK)](http://en.wikipedia.org/wiki/Software_development_kit) and [Runtime Environment (JRE)](http://en.wikipedia.org/wiki/HotSpot) (a subset of the SDK); the primary distinction involves the JRE's lack of the compiler, utility programs, and header files.

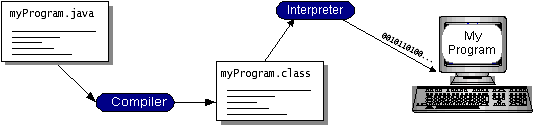
On November 13, 2006, Sun released much of Java as [open source software](http://en.wikipedia.org/wiki/Open_source_software) under the terms of the [GNU General Public License](http://en.wikipedia.org/wiki/GNU_General_Public_License) (GPL). On May 8, 2007, Sun finished the process, making all of Java's core code available under [free software](http://en.wikipedia.org/wiki/Free_software)/open-source distribution terms, aside from a small portion of code to which Sun did not hold the copyright.

Following [Oracle Corporation](http://en.wikipedia.org/wiki/Oracle_Corporation)'s acquisition of Sun Microsystems in 2009-2010, Oracle has described itself as the "steward of Java technology with a relentless commitment to fostering a community of participation and transparency".

**Characteristics:-**

* Simple
* Object oriented
* Distributed
* Interpreted
* Robust
* Secure
* Architecture neutral
* Portable
* High performance
* Multithreaded

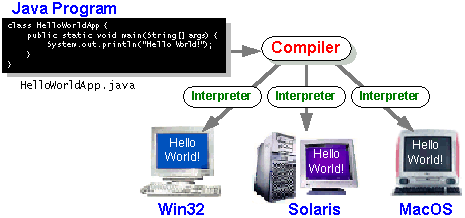
With most programming languages, you either compile or interpret a program so that you can run it on your computer. The Java programming language is unusual in that a program is both compiled and interpreted. With the compiler, first you translate a program into an intermediate language called Java byte codes —the platform-independent codes interpreted by the interpreter on the Java platform. The interpreter parses and runs each Java byte code instruction on the computer. Compilation happens just once; interpretation occurs each time the program is executed. The following figure illustrates how this works.



**Figure 1: Java programming steps**

You can think of Java byte codes as the machine code instructions for the Java Virtual Machine (Java VM). Every Java interpreter, whether it's a development tool or a Web browser that can run applets, is an implementation of the Java VM.

Java byte codes help make "write once, run anywhere" possible. You can compile your program into byte codes on any platform that has a Java compiler. The byte codes can then be run on any implementation of the Java VM. That means that as long as a computer has a Java VM, the same program written in the Java programming language can run on Windows 2000, a Solaris workstation, or on an iMac.



**Figure 2: write once run everywhere**

**The Java Platform**

A platform is the hardware or software environment in which a program runs. We've already mentioned some of the most popular platforms like Windows 2000, Linux, Solaris, and Mac OS. Most platforms can be described as a combination of the operating system and hardware. The Java platform differs from most other platforms in that it's a software-only platform that runs on top of other hardware-based platforms.

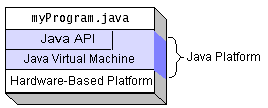
The Java platform has two components:

* The Java Virtual Machine (Java VM)
* The Java Application Programming Interface (Java API)

Java Virtual Machine is standardized hypothetical computer, which is emulated inside our computer by a program. It is base of Java platform and is ported onto various hardware-based platforms.

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

The following figure depicts a program that's running on the Java platform. As the figure shows, the Java API and the virtual machine insulate the program from the hardware.



**Figure 3 Java API**

**What Can Java Technology Do…?**

The most common types of programs written in the Java programming language are applets and applications. If you've accessed the Web, you're probably already familiar with applets. An applet is a program that adheres to certain conventions that allow it to run within a Java-enabled browser.

However, the Java programming language is not just for writing cute, entertaining applets for the Web. The general-purpose, high-level Java programming language is also a powerful software platform. Using the generous API, you can write many types of programs.

An application is a standalone program that runs directly on the Java platform. A special kind of application known as a server serves and supports clients on a network. Examples of servers are Web servers, proxy servers, mail servers, and print servers. Another specialized program is a servlet. A servlet can almost be thought of as an applet that runs on the server side. Java Servlets are a popular choice for building interactive web applications, replacing the use of CGI scripts. Servlets are similar to applets in that they are runtime extensions of applications. Instead of working in browsers, though, servlets run within Java Web servers, configuring or tailoring the server.

**JFC and Swing**

JFC and Swing is a part of java extension. JFC is short for Java Foundation Classes, which encompass a group of features to help people build graphical user interfaces (GUIs). The JFC was first announced at the 1997 Java One developer conference and is defined as containing the following features:

* **The Swing Components**

Include everything from buttons to split panes to tables.

* **Pluggable Look and Feel Support**

It gives any program that uses Swing components a choice of looks and feels. For example, the same program can use either the Java look and feel or the Windows look and feel. We expect many more look-and-feel packages -- including some that use sound instead of a visual "look" -- to become available from various sources.

* **Accessibility API**

It enables assistive technologies such as screen readers and Braille displays to get information from the user interface.

* **Java 2DTM API (Java 2 Platform only)**

It enables developers to easily incorporate high-quality 2D graphics, text, and images in applications and in applets.

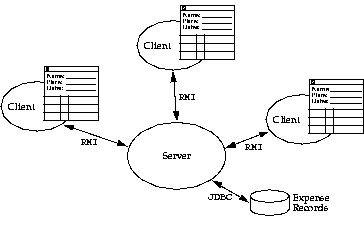
* **Drag and Drop Support (Java 2 Platform only)**

Provides the ability to drag and drop between a Java application and a native application.

The Swing API is available in two forms:

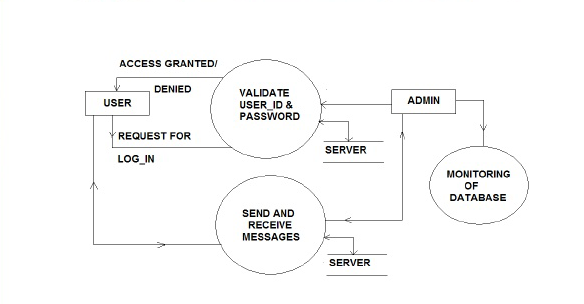
* As a core part of the Java 2 Platform, Standard Edition (including versions 1.2, 1.3, and 1.4)
* JFC 1.1 (for use with JDK 1.1)

Sun community recommends that to use the latest version of the Java 2 Platform. Not only will you be getting the latest bug fixes, but you'll get more features. (As we used JSDK1.5.0\_01 version, which is the latest release of java development kit up to the time of written of this report.)

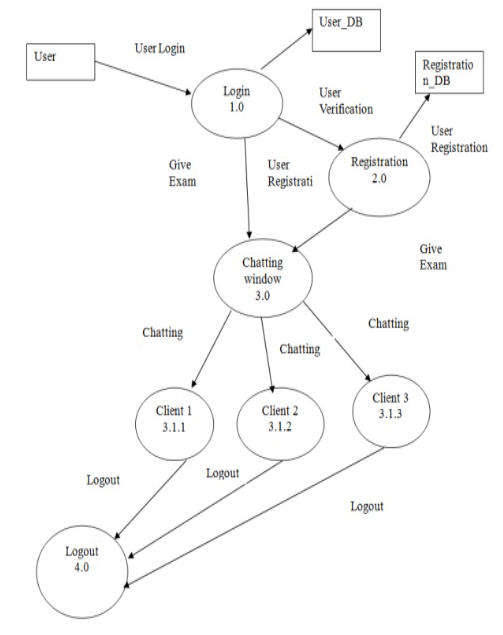
**DFD**

**0-Level**

**1-Level**

****

**2-Level**

****

**Tools to be Used**

* Netbeans / RAD
* MySQL Database
* Windows will be preferred OS
* CSV files for activity logs

**Final Deliverable must include**

* Online or offline help to above said users, Application deployment executive and developer
* Application archive ( .jar/.exe / .war ) with source code
* Complete Source code

**HARDWARE REQUIREMENT SPECIFICATION**

**Hardware Specification: -**

**Server side:**

|  |  |  |
| --- | --- | --- |
| Description | Minimum | Recommended |
| System type | IBM compatible PC with 1GHz | IBM compatible PC with 2GHz |
| RAM | 256 MB | 512 MB |
| Cache | 256 KB | 512 KB |
| Storage | 40 GB | 120 GB  (Convenient for Backup) |
| Disk Drive | Floppy or CD-RW  (Back up purpose) | Floppy or CD-RW or DAT  (Back up purpose) |
| Display | 15’’ VGA | 17’’ SVGA (LCD) |
| User Interface | Key Board and mouse | Compatible keyboard and mouse |
| Output media | DMP Printer | Laser Printer (For reports) |

**Client Side:**

|  |  |  |
| --- | --- | --- |
| Description | Minimum | Recommended |
| System type | IBM compatible PC with 1GHz | IBM compatible PC with 1GHz |
| RAM | 128 MB | 256 MB |
| Cache | 256 KB | 512 KB |
| Storage | 20 GB | 40 GB  (Convenient for Backup) |
| Disk Drive | Floppy or CD-RW  (Back up purpose) | Floppy or CD-RW  (Back up purpose) |
| Display (LCD) | 14’’ VGA | 15’’ SVGA |
| User Interface | Key Board | Compatible keyboard and mouse |
| Output media | DMP Printer | Laser Printer (For reports) |

**Software Requirements:**

* The software requirements are as follows:
* System Software
* Windows 2000/Windows XP
* Database Oracle, SQL, JDK 1.6, Java IDE like eclipse