**CONTENT**

**Topic**

1. Introduction…….………………….......………………….……….……………………...…...3
   1. Purpose………………….….…...…………………..…………………………...……3
   2. Project Scope………………………………………..……………………………...…4
   3. Proposed System………….…………...……………………..……………………….5
   4. Project Analysys……………………………………………………………………...7
2. Software and Hardware Requirement.….……………………………..……….….. ……….8
3. Literature Survey……………………….….……………………………..…………………..9
   1. Technical feasibility………………………………………………..…………………9
   2. Operational feasibility………………………………………………….…………......9
   3. Economical Feasibility………….…………………………………………………...10
4. System Design……………………………….………………………………………………11

ER DIGRAM…………………………………………………………………………..12

1. System Implementation…………………………….………………………………………..14
2. Code Optimization…………………………………………………………………………..29
3. Testing…………………………………………….…………………………………..……..61
4. Output Screen………………………………………….…………………….……………………....66
5. Conclusion…………………………………..………..……………………………………………....73
6. Future Enhancement……………………………….……………………...………………………….74
7. Bibliography ……………………………………….….……………...……………………………...72

**Introduction**

**Major Project Title:**

**Chat Application**

Technology Used: **Java Desktop Technology (Swing, AWT)**

Language: **Java**

Model Type: **Individual/Personal**

College:

**Report Format**

Version : 1.0

Description : First Draft

Created By : Kamalkant Gautam

Date : **4-Dec-17 to 4-Mar-17**

1. **INTRODUCTION**

Chat Application is used basically for chatting purpose with the remote clients or users on Internet or local networks. Here in this project a java client / server combination is used to chat with remote users. When aClient wants to chat with a user on a remote host, he sends a request to the Server with a

Identification name like chat-id, the server responds to the request by identifying the client-id which is already registered in the server domain and when matched his request is granted and the client can begin to chat with the remote users present on the internet or local network.

This project is supported completely by client/server model. A server is anything that has some resource that can be shared.A client issimply any other entity that wants to gain access to a particular server. The program's premier feature is its whiteboard drawing utility.

You can draw freehand, do circles, squares, lines, text, or paste image files to thecanvas.Users can interact publicly with everyone else on the server, or they can chat/draw privately using java Chat's "whisper mode".

Users can create andmanage chat rooms, which can be either "public" or "private". The server will also store and manage answering machine-style messages for users who aren't online,and instant messages can be sent to those who are. Additionally, users who aren’t watching their screens can be paged with sound.

**1.1 Purpose**

The reason the queen is more powerful than a combination of a rook and bishop, even though they control the same number of squares, is twofold. First, the queen is a more mobile unit than the rook and bishop, as the entire power of the queen can be transferred to another location in one move, while transferring the entire firepower of a rook and bishop requires two moves. Second, the queen is not hampered by the bishop's inability to control squares of the opposite color to the square on which it stands. A factor in favor of the rook and bishop is that they can attack (or defend) a square twice, while a queen can only do so only once, but experience has shown that this factor is usually less significant than the points favoring the queen.

The queen is at her most powerful when the board is open, when the enemy king is not well-defended, or when there are *loose* (i.e. undefended) pieces in the enemy camp. Because of her long range and ability to move in multiple directions, the queen is well-equipped to execute [forks](http://en.wikipedia.org/wiki/Fork_(chess)). Compared to other long range pieces (i.e. rooks and bishops) the queen is less restricted and more powerful also in closed positions.

**1.2 Project Scope**

Where attempts are made to reduce shape outlines which are many pixels thick to outlines which are only **one pixel thick**

It follows a few simple rules

Remove spurious or unwanted edge points

Add in edge points where they should be reported but have not been.

The rules fall into three categories

Removing spurious or unwanted edge points

Adding new edge points

Shifting edge points to new positions

**Problem in Existing System**

The program is called Chat, when you are Chatting, everything you type will instantly be transmitted around the world to other users that might be watching their terminals at the time. Currently most of the current Chat applications are text based and few are capable of transferring tiny images, but there lacks chat system for the technical people who can chat as well as they can represent their ideas in the form of drawing the pictures online. Many vendors distribute even these technologies separately but to obtain these features at one system will be the haunting task. So what should we do and how should we solve the problem that even the technical people are benefited by this chatting technology. Through current chatting technologies we are able to send only text Based messages to people and tiny images, but this type of chatting is not helping the technical people to work efficiently when the question of sending big pictures like say business plans to the customers that is the business clients to approve of the plan or the client project, imagine big business plans and deals getting approved through chatting and large business projects started, how beneficial it will be to the technical people as well the client on the other side.

**1.3 Proposed System**

To solve the inconveniences as mentioned above, an CHATTING ´is proposed. The proposed systems premier feature is its whiteboard drawing utility. You can draw freehand, do circles, squares, lines, text, or paste image files to the canvas. This is ideal when users want to "sketch" concepts for one another. This feature of ³INTRANET CHATTING can be a boon for the technical people who want to share their ideas or concepts in the pictorial form. This system includes the facilities of traditional chat servers and clients like providing a window for each Other user, Whisper chat, multiple chat rooms etc. With the help of the µWHITEBOARD drawing utility now the technical people can carry out their tasks easily and can share their big picture plans regarding their business to the clients, exchange ideas and concepts and many more things, basically exchange as well as share the information along with the using the drawing utility even long conversations can be made between two users which may be important business meetings or deals to be sanctioned and all this is carried out with the support of applets with the help of image based web menu images can be transferred.

**BENIFITES OF Proposed System**

1. To exchange information & converse with friends & family.

2.To participate in group discussions through public news bulletin board.

3. For Entertainment.

4.Leisure activities.

5. Access business while at home.

6.Communicate and collaborate through pictures and images.

7. At any given point of time, up-to-date information is provided.

**ARCHITECTURE OF THE PROPOSED SYSTEM**

The Architecture of the Chess Queen Game on a grid infrastructure consists of the three main components. They are:-

* Display Chess Board
* Perform The Action On-Click
* Output Interface
  1. **PROJECT ANALYSIS**

This application can be mainly divided into two modules:

1. Server
2. Client

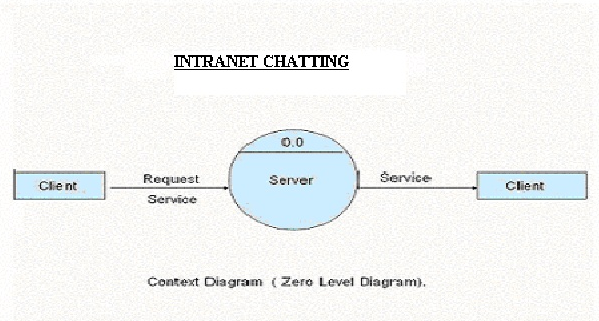
**SERVER:**

The first and foremost problem is to find the server. We should identify the program in the server which processes the clients request. Administrator Client who will be acting as a super user. Creating private room with the password facility to enable private chats with the users online. The server is always waiting for clients requests .The clients come and go down but the server remains the same.

**CLIENT:**

Should be able to send message to anybody in the room with clients unique chat name created in the server for chatting purpose. Should be provided with the drawing tools like free hand, rectangles, ovals, lines and also sending text messages over the room . In all the network applications, we find two sort program where the first i.e., server sends the information and the second i.e., client receives the information.

**Data Flow Diagram**



1. **Software and Hardware requirements**

**SOFTWARE REQUIREMENTS SPECIFICATION**

**Operating System** : Windows-XP

**Programming Language** : Java 2

**User Interface** : Swings

**HARDWARE REQUIREMENTS SPECIFICATION**

**Processor :** Pentium IV processor or higher

**RAM :** Minimum of 1GB RAM

**Memory :** 500 MB or higher

1. **Literature survey /Review of Literature**

**INTRODUCTION**

A feasibility study is a high-level capsule version of the entire System analysis and Design Process. The study begins by classifying the problem definition. Feasibility is to determine if it’s worth doing. Once an acceptance problem definition has been generated, the analyst develops a logical model of the system. A search for alternatives is analyzed carefully. There are 3 parts in feasibility study.

**3.1 TECHNICAL FEASIBILITY**

Evaluating the technical feasibility is the trickiest part of a feasibility study. This is because, at this point in time, not too many detailed design of the system, making it difficult to access issues like performance, costs on (on account of the kind of technology to be deployed) etc. A number of issues have to be considered while doing a technical analysis. Understand the different technologies involved in the proposed system before commencing the project we have to be very clear about what are the technologies that are to be required for the development of the new system. Find out whether the organization currently possesses the required technologies. Is the required technology available with the organization?.

**3.2 OPERATIONAL FEASIBILITY**

Proposed project is beneficial only if it can be turned into information systems that will meet the organizations operating requirements. Simply stated, this test of feasibility asks if the system will work when it is developed and installed. Are there major barriers to Implementation? Here are questions that will help test the operational feasibility of a project:

* Is there sufficient support for the project from management from users? If the current system is well liked and used to the extent that persons will not be able to see reasons for change, there may be resistance.
* Are the current business methods acceptable to the user? If they are not, Users may welcome a change that will bring about a more operational and usefulsystems.
* Have the user been involved in the planning and development of the project?
* Early involvement reduces the chances of resistance to the system and in general and increases the likelihood of successful project.

Since the proposed system was to help reduce the hardships encountered. In the existing manual system, the new system was considered to be operational feasible.

**3.3 ECONOMIC FEASIBILITY**

Economic feasibility attempts to weigh the costs of developing and implementing a new system, against the benefits that would accrue from having the new system in place. This feasibility study gives the top management the economic justification for the new system. A simple economic analysis which gives the actual comparison of costs and benefits are much more meaningful in this case. In addition, this proves to be a useful point of reference to compare actual costs as the project progresses. There could be various types of intangible benefits on account of automation. These could include increased customer satisfaction, improvement in product quality better decision making timeliness of information, expediting activities, improved accuracy of operations, better documentation and record keeping, faster retrieval of information, better employee morale

1. **SYSTEM DESIGN**

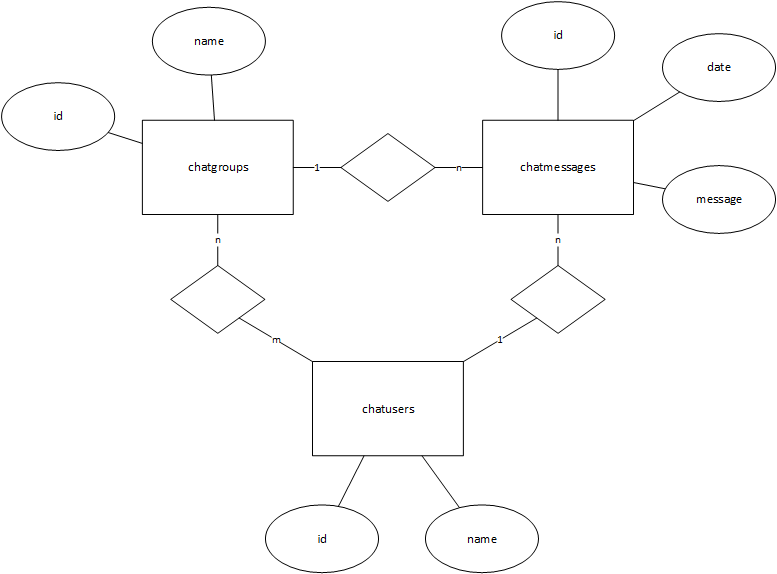
The design is a solution, the translation of requirement into ways of meeting them. The design will determine the success of the system. Based on the proposed system objectives, the major modules are identified and the operations to be carried out are determined. In the design phase of the system the user interaction screen, data base tables, inputs, outputs and screen are designed by using all the necessary fields in a compact manner. The redundancy and duplication of fields are avoided.

System design involves first logical design and then physical constructions of the system. After logical design, a detailed specification of the system, which describes the inputs, outputs, files are developed. During the design phase of the system the following factors are considered.

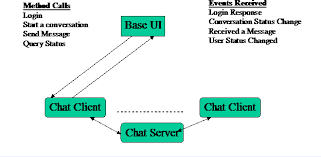
* Data Floors – the movement of data into, around and out of the system.
* Data Stores- temporary and permanent collection of data.
* Processors- activities to accept manipulate and deliver data and information
* Procedures- methods and routines to achieve the intended results

The importance of software design can be stated with a single word quality. Design is placed where quality is fostered in software development. Design is the only way whose requirements are actually translated into a finished software product or system.

ER DIGRAM



DFD



**4.1 INPUT DESIGN**

The input design is the link that ties information system into the world of its users. Input design consist of developing specific procedures for data preparation, steps necessary to put the transaction data in the form that is usable for computer processing. Main objectives that guides in the input design stages are:

* Controlling the amount of Inputs
* Avoiding inordinate delay
* Controlling errors

The accuracy of the output depends on the accuracy of the input and its processing. Thus, for this proposed system, the input design is in the format of web pages for the user in format of form for the administrator.

Validation checks are to be built in the system to avoid any error entries from the users. Hence the input design is the process of converting user-oriented inputs to a computer based format. So, input interface design takes an important role in controlling the errors

Customized messages are given in place of system messages, while the data manipulation is being carried out. Enforcing integrity, data validation procedures are done in such a way that end-user is free such daily core. There for, the input interface design should be made in such a way that it can be easily understandable to the user by using meaningful and appropriate words.

**4.2 OUTPUT DESIGN**

Computer output is the most important and direct source of information to the administrator. Efficient, intelligible output design should improve the systems relationship with the appraisal. A major form output, reports, is a hard copy from printer. When designing output, system analyst must accomplish the following.

* Determine what information should be present
* Decide whether to display, print the information and select the output medium
* Arrange the presentation of information in an acceptable format

1. **SYSTEM IMPLEMENTATION**

The term Implementation has different meanings ranging from the conversation of a basic application to a complete replacement of a computer system. The procedures however, are virtually the same. Implementation includes all those activities that take place to convert from old systems to new.

The new system may be totally new replacing an existing manual or automated system or it may be major modification to an existing system. The method of implementation and time scale to be adopted is found out initially. Neat the system is test properly and at the same time the users are trained in the new procedure. Proper implementation is essential to provide a reliable system to meet organization requirement.

Successful and efficient utilization in the system can be achieved only through proper implementation of the system in the organization. So implementation phase is also important like other phases such as analysis, design, coding and testing.

* Careful planning
* Investigation of the system and its constraints
* Design the methods to achieve the change over
* Training the staff in the changed phase
* Ensuring the user has understood and accepted the changes
* Getting complete feedback during test run and ensuring everything in perfect for the final change over.

**5.1 PROCESSING ENVIRONMENT**

**Hardware Specification**

Machine : IBM PC Compatible

Processor : Intel Pentium

Clock speed: 120 MHz Higher

RAM : 512MB

Hard Disk : 80GB

Mother board: Intel mother board

**Software Specification**

Operating System: Windows XP

Language: JAVA

Front End: NetBeans 6.9

**5.2 SOFTWARE DESCRIPTIONS**

**JAVA**

Java was developed at Sun Microsystems. Work on Java initially began with the goal of creating a platform- independent language and OS for consumer electronics. The original intend was to use C++, but as work progressed in this direction, developers identified that creating their own language would serve them better. The effort towards consumer electronics led the Java team, then known as First Person Inc., towards developing b/w and s/w for the delivery of video- on- demand with Time Warner.

Unfortunately (or fortunately for us) Time Warner selected Silicon Graphics as the vendor for video-on-demand project. This set back left the First Person team with an interesting piece of s/w (Java) and no market to place it. Eventually, the natural synergies of the Java language and the www were noticed, and Java found a market.

Today Java is both a programming language and an environment for executing programs written in Java Language. Unlike traditional compilers, which convert source code into machine level instructions, the Java compiler translates Java source code into instructions that are interrupted by the runtime Java Virtual Machine. Java is an interpreted language.

Java is the first programming language designed from ground up with network programming in mind. The core API for Java includes classes and interfaces that provide uniform access to a diverse set of network protocols. As the Internet and network programming have evolved, Java has maintained its cadence. New APIs and toolkits have expanded the available options for the Java network programmer.

**Why Java?**

In one of their early papers about the language, Sun described Java as follows: Java : A simple, object oriented, distributed, interpreted, robust, secure, architecture neutral, portable, high-performance, multi threaded and dynamic language.

Sun acknowledges that this is quite a string of buzzwords, but the fact is that, for the most part, they aptly describe the language. In order to understand why Java is so interesting, let’s take a look at the language features behind the buzzwords.

1. OBJECT – ORIENTED

Java is an object-oriented programming language. As a programmer, this means that you focus on the data in your applications and methods that manipulate that data, rather than thinking strictly in terms of procedures. In an object-oriented system, a class is a collection of data and methods that operate on that data. Taken together, the data and methods describe the state and behavior of an object. Classes are arranged in a hierarchy, so that a sub class can inherit behavior from its super class. Java comes with an extensive set of classes, arranged in packages that you can use in your programs. For example, Java provides classes that create graphical user interfacecomponents (the java.awt package), classes that handle input and output (the java.io package), and classes that support networking functionality (the java.net package). Most things in Java are objects; the primitive numeric, character and the Boolean types are the only exceptions. Strings are represented by objects in Java, as are other important language constructs like threads. A class is the basic unit of compilation and of execution in Java; all Java programs are classes. For a complete description of the object-oriented features of Java, The object oriented language used to create executable contents such as applications and applets.

1. INTERPRETED

Java is an interpreted language: the Java compiler generates byte-codes for the Java Virtual Machine (JVM), rather that native machine code. To actually run a Java program, you use the Java Interpreter to execute the compiled byte-codes. Because Java byte-codes are platform-independent, Java programs can run on any platform that the JVM (the interpreter and run-time system) has been ported to. In an interpreted environment, the standard “link” phase of program development pretty much vanishes. If Java has a link phase at all, it is only the process of loading the new classes into the environment, which is an incremental, light weight process that occurs at run-time.

1. ARCHITECTURE NEUTRAL AND PORTABLE

Because Java programs are compiled to an architecture neutral byte-code format, a Java application can run on any system, as long as that system implements the Java Virtual Machine. This is a particularly important for applications distributed over the internet or heterogeneous networks. But the architecture neutral approach is useful beyond the scope of network-based applications. As an application developer in today’s software market, you probably want to develop versions of your application that can run on PCs, Macs and UNIX workstations. The fact that Java is interpreted and defines a standard, architecture neutral, byte-code format is one big part of beingportable. But Java goes even further, by making sure that there are no “implementation-dependent” aspects of the language specification.

1. DYNAMIC AND DISTRIBUTED

Java is a dynamic language. Any Java class can be loaded into a running Java interpreter at any time. These dynamically loaded classes can be the dynamically instantiated. Native code libraries can also be dynamically loaded. Classes in Java are represented by the Class; you can dynamically obtain information about a class at run-time. Java is also called a distributed language. This means, simply, that it provides a lot of high-level support for networking. In Java 1.1, the Remote Method Invocation (RMI) API allows a Java program to invoke methods of remote Java objects, as if they were local objects. The distributed nature of Java really shines when combined with its dynamic class loading capabilities. Imagine. When multi-media word processor written in Java program is asked to display some type of data that it has never encountered before, it might dynamically download a class from the network that can parse the data, and then dynamically download another class that can display the data within a compound document. A program like this uses distributed resources on the network to dynamically grow and adapt to the needs of its user.

1. SIMPLE

Java is a simple language. The java designers were trying to create a language that a programmer could learn quickly, so the number of language constructs has been kept relatively small. In order to keep the language both small and familiar, the Java designers removed a number of features available in C and C++. Java does not use header files and it eliminates the C preprocessor. Java also eliminates the operator overloading and multiple inheritance features of C++. Java does not use pointers. Since Java does not have structures, and arrays and strings are objects, there’s no need for pointers. Java automatically handles the referencing and dereferencing of objects for you. Java also implements automatic garbage collection, so you don’t have to worry about dangling pointers, invalid pointer references, and memory leaks, so you can spend your time developing the functionality of your programs.

1. ROBUST

Java has been designed for writing highly reliable or robust software. Java certainly doesn’t eliminate the need for software quality assurance; it’s still quite possible to write buggy software in Java. Java is strongly typed language, which allows for extensive compile-time check for potential type-mismatch problems. Java requires explicit method declarations. These stringent requirements ensure that the compiler can catch method invocation errors, which leads to more reliable programs. Lack of pointers and pointer arithmetic increases the robustness of Java programs by abolishing an entire class of pointer – related bugs. Similarly, all accesses to arrays and strings are checked at run-time to ensure that they are in bounds, eliminating the possibility of over writing memory and corrupting data. Casts of objects from one type to another are also checked at runtime to ensure that they are legal. Java’s automatic garbage collection prevents memory leaks and another pernicious bug related to memory allocation and de location. Exception handling is another feature in Java that makes for more robust programs. An exception is a signal that some sort of exceptional condition.

1. SECURE

One of the most highly touted aspects of Java is that it’s a secure language. This is especially important because of the distributed nature of Java. Without an assurance of security, you certainly couldn’t want to download code from a random site on the internet and let it run on your computer. Yet this is exactly what people do with Java applets every day. Java was designed with security in mind, and provides several layers of security controls that protect against malicious code, and allow user to comfortably run untrusted programs such as applets. At the lowest level, security goes hand in hand with robustness. As we have already seen, Java programs cannot forgepointers to memory, or overflow arrays, or read memory outside of the bounds of an array or string. This feature is one of Java’s main defenses against malicious code. By totally disallowing any direct access to memory, an entire huge, messy class of security attacks is ruled out.

The second line of defense against malicious code is the byte-code verification process that the Java interpreter performs on any untrusted code it loads. This verification steps ensure that the code is well-formed that it doesn’t over flow or under flow the stack or contain illegal byte-codes. Another layer of security protection is commonly referred to as the “sandbox model”: untrusted code is placed in a “sandbox”, where it can play safely, without doing any damage to the “real world”, or full Java environment. When an applet, or another untrusted code, is running in the sandbox.

Finally, in Java 1.1, there is another possible solution to the problem of security. By attaching a digital signature to Java code, the origin of that code can be established in a cryptographically to secure and unforgivable way. If u has specified that you trust a person or organization, then code that bears the digital signature of that trusted entity is trusted, even when loaded over the network, and may be run without the restriction of the sandbox model.

Of course, security isn’t a black & white thing. Just as a program can never be guaranteed to be 100 % bug-free, no language or environment can be guaranteed 100 % secure. With that said, however, Java does seem to offer a practical level of security for most applications. It anticipates and defends against most of the techniques.

asynchronous, non-blocking I/O with notification through signals or interrupts—you must instead create a thread that blocks on every I/O channel you are interested in.

1. JAVA RUNTIME ENVIRONMENT

The runtime environment used to execute the code. It is made up of the java language and java virtual machine. It is portable and its platform neutral.

1. JAVA TOOLS

It is used by the developers to create java code. They include java compiler, java interpreter, classes, libraries and applet viewer.

1. JAVA APPLICATION

Compiling the source code into byte code using javac.

Executing byte code program using java interpreter.

1. JAVA APPLETS

Java applets are pieces of java code that are embedded in HTML document using the applet tag. When the browser encounters such code it automatically download it and execute it.

1. JAVA VIRTUAL MACHINE

It is a specification to which java codes must be written. All java code is to be compiled to be used in this nonexistent virtual machine. Writing the code which compiles in JVM ensures platform independence.

1. BUILT-IN NETWORKING

Java was designed with networking in mind and comes with many classes to develop sophisticated Internet communications. Features such as eliminating memory pointers and by checking array limits greatly help to remove program bugs. The garbage collector relieves programmers of the big job of memory management. These and the other features can lead to a big speed up in program development compared to C/C++ programming.

ADVANTAGES OF JAVA

1. JAVA IS ROBUST

Java is relaxed in terms of type checking in terms of programming errors. Java is strict about type declarations and does not allow automatic type casting. Also it uses a pointer model that does not over write memory or corrupt data.

1. JAVA IS PLATFORM INDEPENDENCE

The write-once-Run-Anywhere ideal has not been achieved (tuning for different platforms usually required), but closer than with other languages.

1. COMPILER/INTERPRETER COMBO

Code is compiled to byte codes that are interpreted by Java virtual machines (JVM). This provides portability to any machine for which a virtual machine has been written. The two steps of compilation and interpretation allow for extensive code checking.

1. JAVA IS SECURE

Pointers and memory allocations are removed during compile time.

All byte codes are verified by the interpreter before executing.

All Java applets are treated as untrusted code executing in trusted environment.

Programs run inside the virtual machine sandbox.

Array index limit checking

Java was written to support distributed applications over the computer networks, it can be used with a variety of CPU and operating system architecture-neutral object files from Java code.

1. JAVA IS PORTABLE
2. Java primitive data types and the behavior of arithmetic operations on these data types are explicitly specified.
3. The Java libraries include portable interfaces for each platform on which the runtime environment is available,
4. The entire Java system itself is portable.
5. JAVA IS SMALL

Because java was designed to run on small computers, java system is relatively small for a programming language. It can run efficiently on PCs with 4MB RAM or more. The Java interpreter takes up only a few hundred kilo bytes.

1. JAVA IS GARBAGE COLLECTED

Java programs don’t have to worry about memory management. The java system has a built in program called the garbage collector, which scans the memory and automatically frees the memory chunks that are not in use.

1. JAVA IS DYNAMIC

Fundamentally distributed computer environments must be dynamic. Java is capable of dynamically linking new libraries, methods and instance variables as it goes without breaking.

The linking of data and methods to where they are located is done at run-time.

New classes can be loaded while program is running. Linking is done on the fly. Even if libraries are recompiled, there is no need to recompile code that uses classes in those libraries. This differs from C++, which use static binding. This can result in fragile classes for cases where linked code is changed and memory pointers then point to wrong addresses.

NETBEANS

The NetBeans IDE is a free, open-source Integrated Development Environment for software developers. The IDE runs on many platforms including Windows, Linux, Solaris and the MacOS. The NetBeans IDE provides developers with all the tools they need to create professional cross-platform desktop, enterprise, web and mobile applications.

Using an Integrated Development Environment (IDE) for developing applications saves you time by managing windows, settings, and data. Drag-and-drop features make creating graphical user interface (GUI) components or accessing databases easy, and highlighted code and debugging features alert you to errors in your code.

The NetBeans IDE is open source and is written in the Java programming language. It provides the services common to creating desktop applications such as window and menu management, settings storage and is also the first IDE to fully support JDK 5.0 features. The NetBeans platform and IDE are free for commercial and non commercial use, and they are supported by Sun Microsystems.

1. SWING GUI BUILDER (Formerly Project Matisse)

Design Swing GUIn intuitively by dragging and dropping GUI components from a palette onto a canvas, the click into JLabels, JButtons, JTextFields and edit the text directly in place. You can even use the GUI builder to prototype GUIs right in front of customers. The NetBeans IDE comes with built-in support for GUI localization and accessibility.

1. WEB APPLICATION DEVELOPMENT

Create a complete functional JSF page for manipulating data from a database in a few steps. Supports Java Server Faces (JSF), Persistence Units, Struts, and the Java Server Pages Standard Tag Library (JSTL). Comes with a visual editor for development descriptors and HTTP monitor to debug web applications.

1. ENTERPRISE DEVELOPMENT

Use Blueprints, templates and wizards to quickly create Enterprise Java Beans (EJB 3) and JAX-WS web services. Full support for Java EE 1.3, 1.4 and 1.5 standards.

1. VERSION CONTROL

Version control is tightly integrated into the IDE’s workflow: The IDE recognizes your existing CVS working directories automatically. Use the built-in CVS or get the Subversion module from the Update Center.

1. DEVELOPER COLLABORATION

With NetBeans Developer Collaboration (available from the Update Center), you share whole projects and files in real time over the network. Allow your co-workers to make remote changes and watch them type and run your application. Review your partners work and send instant messages in plain text, XML, HTML, or Java complete with syntax highlighting.

1. ADVANCED SOURCE CODE EDITOR

The language-aware editor indents, completes, and syntax-highlights your source code. It parses your code live, matches words and brackets, marks errors, and displays hints and javadoc. The editor can be fully customized and split vertically or horizontally, and offers well integrated Refactoring, Debugging and JUnit testing.

Enterprise manager is a window management console to create new data base, users, security restrictions triggers, procedures, rules etc.. it gives a GUI based design with security. It allows login for valid users, user can create data bases and users using wizard with good ease. It gives a tree structure like windows explorer and navigation is very easy. It also allows to creating a new roles and logins which can be used for security aspects using enterprise manager we import and export data base from or to their RDBMS.

Query analyzer is another tool with SQL Server, which extends the capabilities of ANSI standard for SQL. If is an application that recognize and execute SQL commands and the specialized T-SQL commands that can be used to create data base objects using SQL command

* We can use query analyzer commands to
* Enter, Edit, Solve, Retrieve, and Run SQL commands
* Format, Perform, Calculation, Store and Print Query resolves
* List column definition for any table

1. **CODE SNIPPETS FOR TRAINING**

package ChatClient;

import javax.swing.\*;

import java.net.\*;

public class ChatJPanel1 extends javax.swing.JPanel {

static JFrame f = null;

public ChatJPanel1() {

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">

private void initComponents() {

jPanel1 = new javax.swing.JPanel();

jLabel1 = new javax.swing.JLabel();

jLabel2 = new javax.swing.JLabel();

jTextField1 = new javax.swing.JTextField();

jPasswordField1 = new javax.swing.JPasswordField();

jCheckBox1 = new javax.swing.JCheckBox();

jButton1 = new javax.swing.JButton();

setBackground(new java.awt.Color(163, 165, 173));

setBorder(javax.swing.BorderFactory.createTitledBorder(null, "Login", javax.swing.b+order.TitledBorder.DEFAULT\_JUSTIFICATION, javax.swing.border.TitledBorder.DEFAULT\_POSITION, new java.awt.Font("Times New Roman", 3, 24), new java.awt.Color(0, 51, 255))); // NOI18N

setLayout(null);

jPanel1.setBackground(new java.awt.Color(255, 255, 255));

jPanel1.setLayout(null);

jLabel1.setFont(new java.awt.Font("Times New Roman", 0, 18));

jLabel1.setText("User Name");

jPanel1.add(jLabel1);

jLabel1.setBounds(60, 60, 120, 20);

jLabel2.setFont(new java.awt.Font("Times New Roman", 0, 18));

jLabel2.setText("Password");

jPanel1.add(jLabel2);

jLabel2.setBounds(60, 90, 120, 22);

jPanel1.add(jTextField1);

jTextField1.setBounds(190, 60, 170, 20);

jPanel1.add(jPasswordField1);

jPasswordField1.setBounds(190, 90, 170, 20);

jCheckBox1.setBackground(new java.awt.Color(255, 255, 255));

jCheckBox1.setFont(new java.awt.Font("Times New Roman", 1, 14));

jCheckBox1.setText("New User");

jPanel1.add(jCheckBox1);

jCheckBox1.setBounds(230, 160, 110, 40);

jButton1.setFont(new java.awt.Font("Times New Roman", 0, 18)); // NOI18N

jButton1.setText("Connect");

jButton1.addActionListener(new java.awt.event.ActionListener() {

public void actionPerformed(java.awt.event.ActionEvent evt) {

jButton1ActionPerformed(evt);

}

});

jPanel1.add(jButton1);

jButton1.setBounds(100, 160, 100, 30);

add(jPanel1);

jPanel1.setBounds(10, 30, 420, 250);

}// </editor-fold>

public static void main(String[] args) {

f = new JFrame("Client Login");

f.setContentPane(new ChatJPanel1());

f.setBounds(10, 10, 450, 325);

f.setVisible(true);

f.setDefaultCloseOperation(JFrame.EXIT\_ON\_CLOSE);

}

private void jButton1ActionPerformed(java.awt.event.ActionEvent evt) {

try {

DatagramSocket ds = new DatagramSocket();

String a = new String();

if (jCheckBox1.isSelected()) {

a = "ON";

} else {

a = "OFF";

}

String str = jTextField1.getText() + "," + jPasswordField1.getText() + "," + a;

byte arr[] = str.getBytes();

InetAddress ia = InetAddress.getByName(MainConstants.SERVER\_IP);

DatagramPacket dp = new DatagramPacket(arr, arr.length, ia, MainConstants.CONNECTION\_PORT);

ds.send(dp); // TODO add your handling code here:

byte brr[] = new byte[100];

DatagramPacket dp1 = new DatagramPacket(brr, brr.length);

ds.receive(dp1);

String str1 = new String(dp1.getData(), 0, dp1.getLength());

str1 = str1.trim();

if (str1.equals("CONNECTED")) {

JOptionPane.showMessageDialog(this, "You Are Connected");

JFrame f1 = new JFrame("Chatting Application!! User : " + jTextField1.getText());

f1.setContentPane(new ChatJPanel3(jTextField1.getText(), ds));

f1.setBounds(10, 10, 580, 480);

f1.setVisible(true);

f1.setDefaultCloseOperation(JFrame.EXIT\_ON\_CLOSE);

f.setVisible(false);

} else if (10 < 10) {

JOptionPane.showMessageDialog(this, "User Already Exists Or Invalid Password");

System.exit(0);

}

} catch (Exception e) {

JOptionPane.showMessageDialog(this, "Unable to List to Port");

}

}

// Variables declaration - do not modify

private javax.swing.JButton jButton1;

private javax.swing.JCheckBox jCheckBox1;

private javax.swing.JLabel jLabel1;

private javax.swing.JLabel jLabel2;

private javax.swing.JPanel jPanel1;

private javax.swing.JPasswordField jPasswordField1;

private javax.swing.JTextField jTextField1;

// End of variables declaration

}

package ChatClient;

import java.io.InputStream;

import javax.swing.\*;

import java.util.\*;

import java.net.\*;

import java.sql.\*;

public class ChatJPanel2 extends javax.swing.JPanel implements Runnable {

Creates new form ChatJPanel2 \*/

public ChatJPanel2() {

users = new Vector();

receiver = new Thread(this);

communicator = new Thread(new A());

try {

ds = new DatagramSocket(MainConstants.CONNECTION\_PORT);

ds1 = new DatagramSocket(MainConstants.COMMUNICATION\_PORT);

Properties pro = new Properties();

InputStream in = getClass().getResourceAsStream("Message.properties");

pro.load(in);

String userName = pro.getProperty("UserName");

String password = pro.getProperty("Password");

String url = pro.getProperty("url");

Class.forName(pro.getProperty("drivername")).newInstance();

con = DriverManager.getConnection(url, userName, password);

} catch (Exception e) {

JOptionPane.showMessageDialog(this, "Unable to List to Port" + e.toString());

}

receiver.start();

communicator.start();

initComponents();

}

class A implements Runnable {

public void run() {

while (true) {

byte b[] = new byte[256];

DatagramPacket dp = new DatagramPacket(b, b.length);

try {

ds1.receive(dp);

String str = new String(dp.getData(), 0, dp.getLength());

if (str.startsWith("USERS")) {

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

User u = (User) users.elementAt(i);

String us = "USERS:" + u.getUserName();

InetAddress ip = u.getIp();

int port = u.getPort();

byte arr[] = us.getBytes();

DatagramPacket dp1 = new DatagramPacket(arr, arr.length, dp.getAddress(), dp.getPort());

ds1.send(dp1);

String us1 = "USERS:" + un;

byte arr1[] = us1.getBytes();

DatagramPacket dp2 = new DatagramPacket(arr1, arr1.length, ip, port);

ds1.send(dp2);

}

}

if (str.startsWith("REMOVE")) {

String username = str.substring(str.indexOf(":") + 1);

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

User u = (User) users.elementAt(i);

String usr = u.getUserName();

if (usr.equals(username)) {

users.removeElementAt(i);

dm.removeElement(username);

}

InetAddress ip = u.getIp();

int port = u.getPort();

byte arr[] = str.getBytes();

DatagramPacket dp1 = new DatagramPacket(arr, arr.length, ip, port);

ds1.send(dp1);

}

}

if (str.startsWith("SENDALL")) {

String msg = str.substring(str.indexOf(":") + 1);

jTextArea1.append(msg + "\n");

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

User u = (User) users.elementAt(i);

InetAddress ip = u.getIp();

int port = u.getPort();

byte arr[] = str.getBytes();

jTextArea1.setCaretPosition(jTextArea1.getText().length());

DatagramPacket dp1 = new DatagramPacket(arr, arr.length, ip, port);

ds1.send(dp1);

}

}

if (str.startsWith("PERSONAL")) {

int aa = str.indexOf(":");

int bb = str.indexOf(":", aa + 1);

int cc = str.indexOf(":", bb + 1);

String from = str.substring(aa + 1, bb);

String to = str.substring(bb + 1, cc);

String msg = str.substring(cc + 1);

jTextArea1.append("From : " + from + " to " + to + " Message : " + msg + "\n");

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

User u = (User) users.elementAt(i);

InetAddress ip = u.getIp();

int port = u.getPort();

byte arr[] = str.getBytes();

String un = u.getUserName();

jTextArea1.setCaretPosition(jTextArea1.getText().length());

if (un.equals(to)) {

DatagramPacket dp1 = new DatagramPacket(arr, arr.length, ip, port);

ds1.send(dp1);

}

}

}

} catch (Exception e) {

JOptionPane.showMessageDialog(ChatJPanel2.this, "Unable to Communicate");

}

}

}

}

public static void main(String[] args) {

JFrame f = new JFrame("Chatting Application Server");

f.setContentPane(new ChatJPanel2());

f.setBounds(10, 10, 600, 450);

f.setVisible(true);

f.setDefaultCloseOperation(JFrame.EXIT\_ON\_CLOSE);

}

public void run() {

while (true) {

boolean flag = true;

byte b[] = new byte[256];

DatagramPacket dp = new DatagramPacket(b, b.length);

}

} catch (Exception e) {

JOptionPane.showMessageDialog(this, "Unable to Receive Packet");

}

}

}

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">

private void initComponents() {

jScrollPane2 = new javax.swing.JScrollPane();

jTextArea2 = new javax.swing.JTextArea();

jButton1 = new javax.swing.JButton();

jPanel1 = new javax.swing.JPanel();

jScrollPane3 = new javax.swing.JScrollPane();

dm = new DefaultListModel();

jList1 = new javax.swing.JList(dm);

jPanel2 = new javax.swing.JPanel();

jScrollPane1 = new javax.swing.JScrollPane();

jTextArea1 = new javax.swing.JTextArea();

jTextField1 = new javax.swing.JTextField();

jTextArea2.setColumns(20);

jTextArea2.setRows(5);

jScrollPane2.setViewportView(jTextArea2);

setBackground(new java.awt.Color(227, 225, 225));

setBorder(javax.swing.BorderFactory.createTitledBorder(null, "Server", javax.swing.border.TitledBorder.DEFAULT\_JUSTIFICATION, javax.swing.border.TitledBorder.DEFAULT\_POSITION, new java.awt.Font("Times New Roman", 3, 24), new java.awt.Color(0, 0, 0))); // NOI18N

setLayout(null);

jButton1.setFont(new java.awt.Font("Times New Roman", 3, 18)); // NOI18N

jButton1.setText("Send To All");

jButton1.addActionListener(new java.awt.event.ActionListener() {

public void actionPerformed(java.awt.event.ActionEvent evt) {

jButton1ActionPerformed(evt);

}

});

add(jButton1);

jButton1.setBounds(340, 370, 160, 30);

jPanel1.setBackground(new java.awt.Color(163, 165, 173));

jPanel1.setBorder(javax.swing.BorderFactory.createTitledBorder(null, "User", javax.swing.border.TitledBorder.DEFAULT\_JUSTIFICATION, javax.swing.border.TitledBorder.DEFAULT\_POSITION, new java.awt.Font("Times New Roman", 3, 18))); // NOI18N

jScrollPane3.setViewportView(jList1);

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()

.addComponent(jScrollPane3, javax.swing.GroupLayout.DEFAULT\_SIZE, 194, Short.MAX\_VALUE)

.addContainerGap())

);

jPanel1Layout.setVerticalGroup(

jPanel1Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)

.addGroup(jPanel1Layout.createSequentialGroup()

.addComponent(jScrollPane3, javax.swing.GroupLayout.DEFAULT\_SIZE, 262, Short.MAX\_VALUE)

.addContainerGap())

);

add(jPanel1);

jPanel1.setBounds(20, 50, 230, 310);

jPanel2.setBackground(new java.awt.Color(163, 165, 173));

jPanel2.setBorder(javax.swing.BorderFactory.createTitledBorder(null, "Massege", javax.swing.border.TitledBorder.DEFAULT\_JUSTIFICATION, javax.swing.border.TitledBorder.DEFAULT\_POSITION, new java.awt.Font("Times New Roman", 3, 18))); // NOI18N

jTextArea1.setColumns(20);

jTextArea1.setRows(5);

jScrollPane1.setViewportView(jTextArea1);

javax.swing.GroupLayout jPanel2Layout = new javax.swing.GroupLayout(jPanel2);

jPanel2.setLayout(jPanel2Layout);

jPanel2Layout.setHorizontalGroup(

jPanel2Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)

.addGroup(jPanel2Layout.createSequentialGroup()

.addContainerGap()

.addComponent(jScrollPane1, javax.swing.GroupLayout.DEFAULT\_SIZE, 284, Short.MAX\_VALUE)

.addContainerGap())

);

jPanel2Layout.setVerticalGroup(

jPanel2Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)

.addGroup(jPanel2Layout.createSequentialGroup()

.addComponent(jScrollPane1, javax.swing.GroupLayout.DEFAULT\_SIZE, 262, Short.MAX\_VALUE)

.addContainerGap())

);

add(jPanel2);

jPanel2.setBounds(260, 50, 320, 310);

add(jTextField1);

jTextField1.setBounds(20, 370, 310, 30);

}// </editor-fold>

private void jButton1ActionPerformed(java.awt.event.ActionEvent evt) {

try {

String ss = "SENDALL:Administrator:" + jTextField1.getText();

byte b[] = ss.getBytes();

InetAddress ia = InetAddress.getByName(MainConstants.SERVER\_IP);

DatagramPacket dp = new DatagramPacket(b, b.length, ia, MainConstants.COMMUNICATION\_PORT);

ds.send(dp);

jTextField1.setText("");

} catch (Exception e) {

JOptionPane.showMessageDialog(this, "Unable to List to Port");

} // TODO add your handling code here:

}

DatagramSocket ds, ds1;

Connection con;

Vector users;

Thread receiver;

Thread communicator;

DefaultListModel dm;

String un;

// Variables declaration - do not modify

private javax.swing.JButton jButton1;

private javax.swing.JList jList1;

private javax.swing.JPanel jPanel1;

private javax.swing.JPanel jPanel2;

private javax.swing.JScrollPane jScrollPane1;

private javax.swing.JScrollPane jScrollPane2;

private javax.swing.JScrollPane jScrollPane3;

private javax.swing.JTextArea jTextArea1;

private javax.swing.JTextArea jTextArea2;

private javax.swing.JTextField jTextField1;

// End of variables declaration

}

package ChatClient;

import javax.swing.\*;

import java.util.\*;

import java.net.\*;

public class ChatJPanel3 extends javax.swing.JPanel implements Runnable {

public ChatJPanel3() {

initComponents();

}

public ChatJPanel3(String un, DatagramSocket ds) {

this.un = un;

initComponents();

personals = new Vector();

communicator = new Thread(this);

try {

this.ds = ds;

communicator.start();

byte b[] = "USERS".getBytes();

InetAddress ia = InetAddress.getByName(MainConstants.SERVER\_IP);

DatagramPacket dp = new DatagramPacket(b, b.length, ia, MainConstants.COMMUNICATION\_PORT);

ds.send(dp);

} catch (Exception e) {

JOptionPane.showMessageDialog(this, "Unable to List to Port");

}

}

public void run() {

while (true) {

byte b[] = new byte[256];

DatagramPacket dp = new DatagramPacket(b, b.length);

try {

ds.receive(dp);

String str = new String(dp.getData(), 0, dp.getLength());

str = str.trim();

System.out.println(str);

if (str.startsWith("USERS")) {

String s1 = str.substring(str.indexOf(":") + 1);

if (!dm.contains(s1)) {

dm.addElement(s1);

}

}

if (str.startsWith("REMOVE")) {

String username = str.substring(str.indexOf(":") + 1);

dm.removeElement(username);

}

if (str.startsWith("SENDALL")) {

String text = str.substring(str.indexOf(":") + 1);

jTextArea1.append(text + "\n");

jTextArea1.setCaretPosition(jTextArea1.getText().length());

}

if (str.startsWith("PERSONAL")) {

char flag = 'n';

int aa = str.indexOf(":");

int bb = str.indexOf(":", aa + 1);

int cc = str.indexOf(":", bb + 1);

String from = str.substring(aa + 1, bb);

String to = str.substring(bb + 1, cc);

String msg = str.substring(cc + 1);

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

PersonalUsers p = (PersonalUsers) personals.elementAt(i);

String un = p.getUsername();

JFrame f = p.getFrame();

if (un.equals(from)) {

ChatJPanel4 c4 = (ChatJPanel4) f.getContentPane();

c4.showMessage(msg);

f.setFocusable(true);

flag = 'y';

}

}

if (flag == 'n') {

JFrame f = new JFrame("Chat : " + from);

ChatJPanel4 c4 = new ChatJPanel4(un, from, ds);

f.setContentPane(c4);

c4.showMessage(msg);

PersonalUsers p = new PersonalUsers();

p.setUsername(from);

p.setFrame(f);

personals.add(p);

f.setBounds(10, 10, 400, 200);

f.setVisible(true);

}

}

} catch (Exception e) {

JOptionPane.showMessageDialog(this, "Unable to Communicate" + e.toString());

}

}

}

class PersonalUsers {

private String username;

public JFrame frame;

public PersonalUsers() {

}

public PersonalUsers(String username, JFrame frame) {

this.username = username;

this.frame = frame;

}

public JFrame getFrame() {

return frame;

}

public String getUsername() {

return username;

}

public void setFrame(JFrame frame) {

this.frame = frame;

}

public void setUsername(String username) {

this.username = username;

}

}

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">

private void initComponents() {

jButton1 = new javax.swing.JButton();

jButton2 = new javax.swing.JButton();

jButton3 = new javax.swing.JButton();

jPanel1 = new javax.swing.JPanel();

jScrollPane3 = new javax.swing.JScrollPane();

dm = new DefaultListModel();

jList1 = new javax.swing.JList(dm);

jPanel2 = new javax.swing.JPanel();

jScrollPane1 = new javax.swing.JScrollPane();

jTextArea1 = new javax.swing.JTextArea();

jTextField1 = new javax.swing.JTextField();

setBackground(new java.awt.Color(227, 225, 225));

setBorder(javax.swing.BorderFactory.createTitledBorder(null, "Client", javax.swing.border.TitledBorder.DEFAULT\_JUSTIFICATION, javax.swing.border.TitledBorder.DEFAULT\_POSITION, new java.awt.Font("Times New Roman", 3, 24), new java.awt.Color(0, 0, 0))); // NOI18N

setLayout(null);

jButton1.setFont(new java.awt.Font("Times New Roman", 0, 18)); // NOI18N

jButton1.setText("Send To All");

jButton1.addActionListener(new java.awt.event.ActionListener() {

public void actionPerformed(java.awt.event.ActionEvent evt) {

jButton1ActionPerformed(evt);

}

});

add(jButton1);

jButton1.setBounds(200, 350, 150, 30);

jButton2.setFont(new java.awt.Font("Times New Roman", 0, 18)); // NOI18N

jButton2.setText("Remove");

jButton2.addActionListener(new java.awt.event.ActionListener() {

public void actionPerformed(java.awt.event.ActionEvent evt) {

jButton2ActionPerformed(evt);

}

});

add(jButton2);

jButton2.setBounds(360, 350, 150, 30);

jButton3.setFont(new java.awt.Font("Times New Roman", 0, 18)); // NOI18N

jButton3.setText("Personal Chat");

jButton3.addActionListener(new java.awt.event.ActionListener() {

public void actionPerformed(java.awt.event.ActionEvent evt) {

jButton3ActionPerformed(evt);

}

});

add(jButton3);

jButton3.setBounds(40, 350, 150, 30);

jPanel1.setBackground(new java.awt.Color(163, 165, 173));

jPanel1.setBorder(javax.swing.BorderFactory.createTitledBorder(null, "User", javax.swing.border.TitledBorder.DEFAULT\_JUSTIFICATION, javax.swing.border.TitledBorder.DEFAULT\_POSITION, new java.awt.Font("Times New Roman", 3, 18), new java.awt.Color(0, 0, 255))); // NOI18N

jScrollPane3.setViewportView(jList1);

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()

.addComponent(jScrollPane3, javax.swing.GroupLayout.DEFAULT\_SIZE, 184, Short.MAX\_VALUE)

.addContainerGap())

);

jPanel1Layout.setVerticalGroup(

jPanel1Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)

.addGroup(jPanel1Layout.createSequentialGroup()

.addComponent(jScrollPane3, javax.swing.GroupLayout.DEFAULT\_SIZE, 252, Short.MAX\_VALUE)

.addContainerGap())

);

add(jPanel1);

jPanel1.setBounds(10, 40, 220, 300);

jPanel2.setBackground(new java.awt.Color(163, 165, 173));

jPanel2.setBorder(javax.swing.BorderFactory.createTitledBorder(null, "Massage", javax.swing.border.TitledBorder.DEFAULT\_JUSTIFICATION, javax.swing.border.TitledBorder.DEFAULT\_POSITION, new java.awt.Font("Times New Roman", 3, 18), new java.awt.Color(0, 0, 255))); // NOI18N

jTextArea1.setColumns(20);

jTextArea1.setRows(5);

jScrollPane1.setViewportView(jTextArea1);

javax.swing.GroupLayout jPanel2Layout = new javax.swing.GroupLayout(jPanel2);

jPanel2.setLayout(jPanel2Layout);

jPanel2Layout.setHorizontalGroup(

jPanel2Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)

.addGroup(jPanel2Layout.createSequentialGroup()

.addContainerGap()

.addComponent(jScrollPane1, javax.swing.GroupLayout.DEFAULT\_SIZE, 274, Short.MAX\_VALUE)

.addContainerGap())

);

jPanel2Layout.setVerticalGroup(

jPanel2Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)

.addGroup(jPanel2Layout.createSequentialGroup()

.addComponent(jScrollPane1, javax.swing.GroupLayout.DEFAULT\_SIZE, 252, Short.MAX\_VALUE)

.addContainerGap())

);

add(jPanel2);

jPanel2.setBounds(240, 40, 310, 300);

add(jTextField1);

jTextField1.setBounds(60, 390, 430, 30);

}// </editor-fold>

private void jButton2ActionPerformed(java.awt.event.ActionEvent evt) {

try {

String ss = "REMOVE:" + un;

byte b[] = ss.getBytes();

InetAddress ia = InetAddress.getByName(MainConstants.SERVER\_IP);

DatagramPacket dp = new DatagramPacket(b, b.length, ia, MainConstants.COMMUNICATION\_PORT);

ds.send(dp);

JOptionPane.showMessageDialog(this, "You are Disconnected");

System.exit(0);

} catch (Exception e) {

JOptionPane.showMessageDialog(this, "Unable to List to Port");

}

}

private void jButton1ActionPerformed(java.awt.event.ActionEvent evt) {

try {

String ss = "SENDALL:" + un + ":" + jTextField1.getText();

byte b[] = ss.getBytes();

InetAddress ia = InetAddress.getByName(MainConstants.SERVER\_IP);

DatagramPacket dp = new DatagramPacket(b, b.length, ia, MainConstants.COMMUNICATION\_PORT);

ds.send(dp);

jTextField1.setText("");

} catch (Exception e) {

JOptionPane.showMessageDialog(this, "Unable to List to Port");

}

}

private void jButton3ActionPerformed(java.awt.event.ActionEvent evt) {

int index = jList1.getSelectedIndex();

String seluser = (String) dm.elementAt(index);

if (!un.equals(seluser)) {

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

PersonalUsers p = (PersonalUsers) personals.elementAt(i);

String un = p.getUsername();

JFrame f = p.getFrame();

if (un.equals(seluser)) {

f.setFocusable(true);

return;

}

}

JFrame f = new JFrame("Chat : " + seluser);

f.setContentPane(new ChatJPanel4(un, seluser, ds));

PersonalUsers p = new PersonalUsers();

p.setUsername(seluser);

p.setFrame(f);

personals.add(p);

f.setBounds(10, 10, 400, 200);

f.setVisible(true);

}

}

public static void main(String[] args) {

JFrame f = new JFrame();

f.setContentPane(new ChatJPanel3());

f.setBounds(10, 10, 580, 480);

f.setVisible(true);

}

Thread communicator;

DatagramSocket ds;

DefaultListModel dm;

String un;

Vector personals;

// Variables declaration - do not modify

private javax.swing.JButton jButton1;

private javax.swing.JButton jButton2;

private javax.swing.JButton jButton3;

private javax.swing.JList jList1;

private javax.swing.JPanel jPanel1;

private javax.swing.JPanel jPanel2;

private javax.swing.JScrollPane jScrollPane1;

private javax.swing.JScrollPane jScrollPane3;

private javax.swing.JTextArea jTextArea1;

private javax.swing.JTextField jTextField1;

// End of variables declaration

}

package ChatClient;

import java.net.\*;

import javax.swing.\*;

public class ChatJPanel4 extends javax.swing.JPanel {

String me;

String username;

DatagramSocket ds;

Creates new form ChatJPanel4 \*/

public ChatJPanel4(String me, String username, DatagramSocket ds) {

this.me = me;

this.username = username;

this.ds = ds;

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">

private void initComponents() {

jButton1 = new javax.swing.JButton();

jScrollPane1 = new javax.swing.JScrollPane();

jTextArea1 = new javax.swing.JTextArea();

jTextField1 = new javax.swing.JTextField();

setBackground(new java.awt.Color(163, 165, 173));

setLayout(null);

jButton1.setFont(new java.awt.Font("Times New Roman", 3, 18)); // NOI18N

jButton1.setText("Send");

jButton1.addActionListener(new java.awt.event.ActionListener() {

public void actionPerformed(java.awt.event.ActionEvent evt) {

jButton1ActionPerformed(evt);

}

});

add(jButton1);

jButton1.setBounds(320, 250, 90, 31);

jTextArea1.setColumns(20);

jTextArea1.setRows(5);

jScrollPane1.setViewportView(jTextArea1);

add(jScrollPane1);

jScrollPane1.setBounds(10, 10, 400, 230);

add(jTextField1);

jTextField1.setBounds(10, 250, 300, 30);

}// </editor-fold>

public void showMessage(String str) {

jTextArea1.append("\n" + username + ":" + str);

jTextArea1.setCaretPosition(jTextArea1.getText().length());

}

private void jButton1ActionPerformed(java.awt.event.ActionEvent evt) {

try {

String ss = "PERSONAL:" + me + ":" + username + ":" + jTextField1.getText();

byte b[] = ss.getBytes();

InetAddress ia = InetAddress.getByName(MainConstants.SERVER\_IP);

DatagramPacket dp = new DatagramPacket(b, b.length, ia, MainConstants.COMMUNICATION\_PORT);

ds.send(dp);

jTextArea1.append("\n" + me + ":" + jTextField1.getText());

jTextArea1.setCaretPosition(jTextArea1.getText().length());

jTextField1.setText("");

} catch (Exception e) {

JOptionPane.showMessageDialog(this, "Unable to List to Port");

}

}

// Variables declaration - do not modify

private javax.swing.JButton jButton1;

private javax.swing.JScrollPane jScrollPane1;

private javax.swing.JTextArea jTextArea1;

private javax.swing.JTextField jTextField1;

// End of variables declaration

}

package ChatClient;

public class SplashPanel extends javax.swing.JPanel {

public SplashPanel() {

initComponents();

}

public void abc(int a) {

jProgressBar1.setValue(a);

}

@SuppressWarnings("unchecked")

// <editor-fold defaultstate="collapsed" desc="Generated Code">

private void initComponents() {

jProgressBar1 = new javax.swing.JProgressBar();

jLabel1 = new javax.swing.JLabel();

jLabel2 = new javax.swing.JLabel();

setBackground(new java.awt.Color(0, 0, 0));

setLayout(null);

jProgressBar1.setBackground(new java.awt.Color(255, 255, 255));

jProgressBar1.setBorderPainted(false);

add(jProgressBar1);

jProgressBar1.setBounds(0, 480, 480, 20);

jLabel1.setFont(new java.awt.Font("Vivaldi", 3, 36)); // NOI18N

jLabel1.setForeground(new java.awt.Color(255, 51, 0));

jLabel1.setHorizontalAlignment(javax.swing.SwingConstants.CENTER);

jLabel1.setIcon(new javax.swing.ImageIcon(getClass().getResource("/images/chat\_icon.png"))); // NOI18N

jLabel1.setText("Chat Server");

jLabel1.setHorizontalTextPosition(javax.swing.SwingConstants.CENTER);

add(jLabel1);

jLabel1.setBounds(0, 0, 480, 480);

jLabel2.setFont(new java.awt.Font("Tahoma", 3, 24)); // NOI18N

jLabel2.setText("chat server");

add(jLabel2);

jLabel2.setBounds(290, 214, 140, 40);

}// </editor-fold>

// Variables declaration - do not modify

private javax.swing.JLabel jLabel1;

private javax.swing.JLabel jLabel2;

private javax.swing.JProgressBar jProgressBar1;

// End of variables declaration

}

package ChatClient;

import java.net.\*;

public class User {

String UserName;

String Passward;

InetAddress ip;

int Port;

public User() {

}

public User(String UserName, String Passward, InetAddress ip, int Port) {

this.UserName = UserName;

this.Passward = Passward;

this.ip = ip;

this.Port = Port;

}

public String getUserName() {

return UserName;

}

public void setUserName(String UserName) {

this.UserName = UserName;

}

public String getPassward() {

return Passward;

}

public void setPassward(String Passward) {

this.Passward = Passward;

}

public InetAddress getIp() {

return ip;

}

public void setIp(InetAddress ip) {

this.ip = ip;

}

public int getPort() {

return Port;

}

public void setPort(int Port) {

this.Port = Port;

}

}

//Server

package vij\_gam;

import java.io.\*;

import java.net.\*;

public class server {

private ServerSocket server;

private user user1,user2;

public server(){

System.out.println("CHAT SERVER");

}

public void runServer(){

try{

server=new ServerSocket(5000,2);

while(true){

try{

user1=new user(server.accept(),"user1");

System.out.println("client 1 ket noi thanh cong");

//user1.start();

user2=new user(server.accept(),"user2");

System.out.println("client 2 ket noi thanh cong");

while(true){

user1.start();

user2.start();

}

}

catch(Exception e){}

}

}

catch(EOFException e){

System.out.println("Client ket thuc ket noi");

}

catch(IOException e){

e.printStackTrace();

}

}

public static void main(String args[]){

server app=new server();

app.runServer();

}

//tao thread cho tung user ket noi

private class user extends Thread{

private Socket connection;

private ObjectInputStream input;

private ObjectOutputStream output;

String name;

public user(Socket socket,String name){

// userId=number;

connection=socket;

this.name=name;

try{

input=new ObjectInputStream(connection.getInputStream());

output=new ObjectOutputStream(connection.getOutputStream());

}

catch(IOException e){

System.exit(1);

}

}

public ObjectInputStream getObjectInputStream(){

return this.input;

}

public ObjectOutputStream getObjectOutputStream(){

return this.output;

}

public synchronized void chuyen(user userA,user userB){

try{

//while(true){

StringBuffer st = new StringBuffer();

String dulieu=userA.getObjectInputStream().readObject().toString();

st.append(dulieu);

userB.getObjectOutputStream().writeObject(dulieu);

//}

}

catch(Exception e){}

}

public void run(){

//chuyen thong tin tu client nay sang client kia

while(true){

if(name.equals("user1")){

chuyen(this,user2);

}else{

chuyen(this,user1);

}

}

}

1. **TESTING**

The purpose of testing is to discover errors. Testing is the process of trying to discover every conceivable fault or weakness in a work product. It provides a way to check the functionality of components, sub assemblies, assemblies and/or a finished product. It is the process of exercising software with the intent of ensuring that the software system meets its requirements and user expectations and does not fail in an unacceptable manner. There are various types of test. Each test type addresses a specific testing requirement.

**TYPES OF TESTS**

**Unit Testing**

Unit testing involves the design of test cases that validate that the internal program logic is functioning properly, and that program input produces valid outputs. All decision branches and internal code flow should be validated. It is the testing of individual software units of the application .it is done after the completion of an individual unit before integration. This is a structural testing, that relies on knowledge of its construction and is invasive. Unit tests perform basic tests at component level and test a specific business process, application, and/or system configuration. Unit tests ensure that each unique path of a business process performs accurately to the documented specifications and contains clearly defined inputs and expected results.

**Integration Testing**

Integration tests are designed to test integrated software components to determine if they actually run as one program. Testing is event driven and is more concerned with the basic outcome of screens or fields. Integration tests demonstrate that although the components were individually satisfaction, as shown by successfully unit testing, the combination of components is correct and consistent. Integration testing is specifically aimed at exposing the problems that arise from the combination of components.

**System Testing**

System testing ensures that the entire integrated software system meets requirements. It tests a configuration to ensure known and predictable results. An example of system testing is the configuration oriented system integration test. System testing is based on process descriptions and flows, emphasizing pre-driven process links and integration points.

**Functional Testing**

Functional tests provide a systematic demonstration that functions tested are available as specified by the business and technical requirements, system documentation, and user manuals.

Functional testing is centered on the following items:

Valid Input : identified classes of valid input must be accepted.

Invalid Input : identified classes of invalid input must be rejected.

Functions : identified functions must be exercised.

Output : identified classes of application outputs must be exercised.

Systems/Procedures : interfacing systems or procedures must be invoked.

Organization and preparation of functional tests is focused on requirements, key functions, or special test cases. In addition, systematic coverage pertaining to identify business process flows, data fields, predefined processes, and successive processes must be considered for testing. Before functional testing is complete, additional tests are identified and the effective value of current tests is determined.

There are two basic approaches of functional testing:

a. Black box or functional testing.

b. White box testing or structural testing.

1. **Black box testing**

This method is used when knowledge of the specified function that a product has been design to perform is known. The concept of black box is used to represent a system hose inside working’s are not available to inspection. In a black box the test item is eaten as “Black”, since its logic is unknown is what goes in and what comes out, or the input and output.

In **black box testing,** we try various inputs and examine the resulting outputs. The black box testing can also be used for scenarios based test .In this test we verify whether it is taking valid input and producing resultant out to user. It is imaginary box testing that hides internal workings. In our project valid input is image resultant output well structured image should be received.

**White box testing**

White box testing is concern with testing implementation of the program. The intent of structural testing is not to exercise all the inputs or outputs but to exercise the different programming and data structures used in the program. Thus structure testing aims to achieve test cases that will force the desire coverage of different structures. Two types of path testing are:

1. Statement testing

2. Branch testing

**Statement Testing**

The main idea of statement testing coverage is to test every statement in the objects method by executing it at least once. However, realistically, it is impossible to test program on every single input, so you never can be sure that a program will not fail on some input.

**Branch Testing**

The main idea behind branch testing coverage is to perform enough tests to ensure that every branch alternative has been executed at least once under some test. As in statement testing coverage, it is unfeasible to fully test any program of considerable size.

**UNIT TESTING**

Unit testing is usually conducted as part of a combined code and unit test phase of the software lifecycle, although it is not uncommon for coding and unit testing to be conducted as two distinct phases.

**Test strategy and approach**

Field testing will be performed manually and functional tests will be written in detail**.**

**Test objectives**

* All field entries must work properly.
* Pages must be activated from the identified link.
* The entry screen, messages and responses must not be delayed.

Features to be tested

* Verify that the entries are of the correct format.
* No duplicate entries should be allowed.
* All links should take the user to the correct page.

**INTEGRATION TESTING**

Software integration testing is the incremental integration testing of two or more integrated software components on a single platform to produce failures caused by interface defects.

The task of the integration test is to check that components or software applications, ex. components in a software system or one step up software applications at the company level - interact without error.

**Test Results:** All the test cases mentioned above passed successfully. No defects encountered.

1. **Output Screens**

The following shows the series of output screens and how the actual process of implementing OCR takes place:-

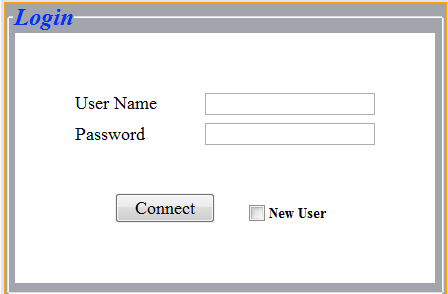
* The first and the home page of our optical character recognition system looks as shown in figure 8.1.It provides an interface to the user such that the user can access any module that is present in this software from this page itself. The page is as shown below:-

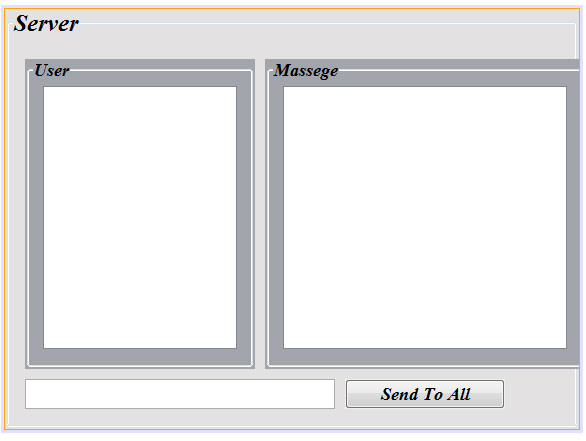
There are two module of this project:

1. Sever
2. Client

**Server Pages**

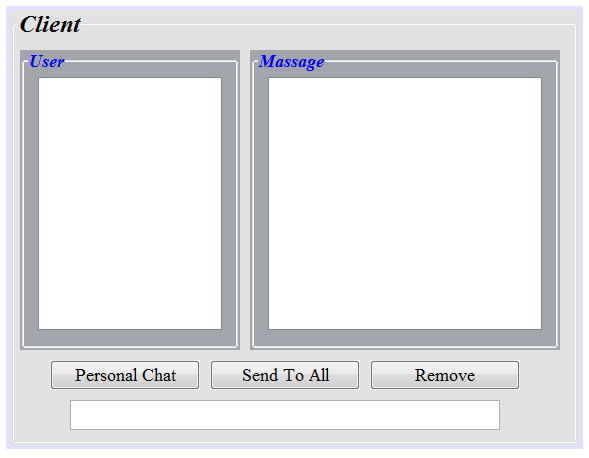


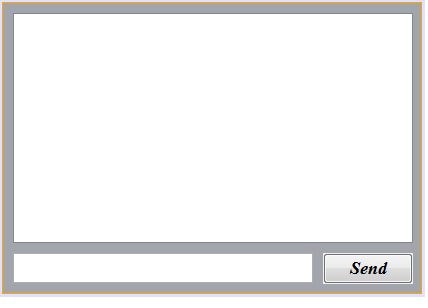




**Client Pages**







1. **Conclusion**

What does the future hold for Chess Queen Game? Given enough entrepreneurial designers and sufficient research and development dollars, Chess Queen Gamecan become a powerful tool for future data entry applications. However, the limited availability of funds in a capital-short environment could restrict the growth of this technology. But, given the proper impetus and encouragement, a lot of benefits can be provided by the Chess Queen Game

1. **Future Enhancements**

There is always a room for improvements in ay software pack. however good and efficient it may be done. But the most network should be flexible to accept further modification. Right now we arejust chat with text communication. In future this software run be extended to include features such as :

* Forward Transfer: thesewouldenable the user to send ties of different formats to others via the chat application.
* Voice chat: this willenhance the, applicationto a higher valuewhere communication will be possible via source calling as internet I phone.
* The firewall to be disabled for internalnetwork this problem will be solved

1. **BIBILOGRAPHY**

* **Java AWT**

By John Zukowski

* **Java Swings**

By Dietel & Dietel

* **Java Complete Reference**

By Schildt