**Lab Report on**

**Hangman Game**

**Project Name:** A hangman game.

**Description:** Hangman game is a word guessing game designed as a survival of a stickman or stick body or stick figure who is partially getting hanged by every wrong guessing. There are seven chances to make wrong. In every new game there is a given word, there are exactly as the same number of dash (-) as the number of the letters of the word that the player should guess.



Fig 1: The hangman game graphics.

In fig 1, a picture of hangman game is given. Player can put his input through the keyboard, or through the graphical alphabets given in the picture. Player can click on the characters and play the game. On the left side, there is a black frame. It is called hang frame, which is used to hang the stick person. If player made a mistake, At first the head appears, then for every mistake comes body, right hand, left hand, right leg, left leg. And when it is finished, a message appears called “You Lost”. And if the player can make the guess perfectly, then another message appears, called “You Win”.



Fig 2: A winning state in the Hangman game.

In fig 2, a winning state is shown. There we can see one leg of the stick body is missing. That means player did five wrong guess out of six wrong guess and did all the right guess. Also a button appears named “New game” which leads us in a new game.



Fig 3: A losing state in Hangman game.

In fig 3, a losing state is shown. Here, ---ch means we player made two correct guess out of five and six wrong guess. Also, the “New Game” button appears to hit a new game.

**Construction:** We used *Java* to make this language. We selected *Swing* framework of Java to create this game. It is a simple game with MouseListener and KeyListener feature. The internal part of code is divided into four part.

1. HangmanGame
2. HangmanPanel
3. HangmanFrame
4. HangmanCheckString

This four part are individual single classes, where the 2nd class maintain the running graphics like drawing the stick figure, maintaining the given input, checking the string, showing the result and most importantly, running the game.

The HangmanPanel class does almost all the gaming part, The HangManCheckString class just show the dashes and do the string check thing. HangmanFrame class maintains the sizes of the window, the internal frame and positioning them.

**Source code:** <https://goo.gl/wQyZ5W>

**Input:**

Senpkmiojuabl

**Output:** 

**Input:**

cdoughjk

**OutPut:**



**Conclusion:**

On the conclusion, we can hope that the game will be a great pleasure to the players and this game will let people be determined to learn new words.

**Lab Report on**

**Chat Box**

**Project Name:** A chat box using socket server.

**Description:** A chat box with socket server application is a desktop application which is designed to chat one with another within a common network. It is built on a socket server and for this it is a one to one paired connection. It is a simple program with a server and a client where a server is a user and a client is another user. We can send E-message through the app.

The server is connected through the IP address of the host computer. For testing purposes, we kept two user in one computer and thus the IP configuration is set as localhost, and IP is set to the local hosting computer IP. When we share a common network, we need to use IP of the network device such as router, wifi router, switches or other switch. And also we need to use a common port which is not used in both computers.

The chat box for server user and client user are almost identical, just the font color of server user chat box is given red and the font color of client user is given blue so that they don’t make any confusion. And this let us know who is server and who is client instantly with no extra attempt.

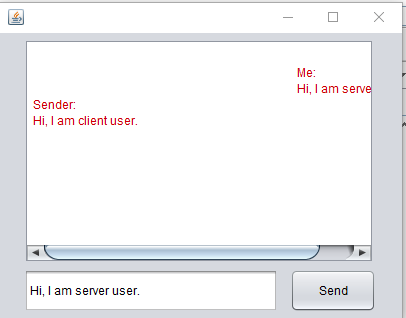


Fig 1: A server user interface.

A server user interface is shown in Fig 1, here we can see that the fonts are red as we mentioned, and what the sender texting appears in left side, and the text of user appears in right side. A class named chat\_server is created to maintain this graphics interface and hosting the connection.

The user just need to write down the message In the bottom text field and press send button to send message. Thus the receiver will get the message.

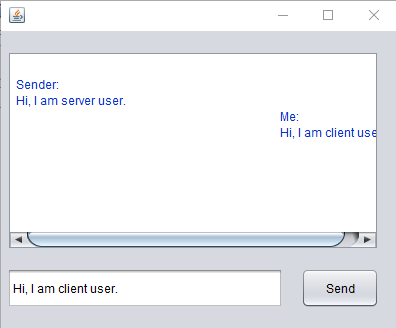


Fig 2: A client user interface.

**Construction:** We selected *Java* to make this application. We choose *Swing* framework to make this application. We used three class to make this class.

1. Chat
2. chat\_server
3. chat\_client

Chat class is made to run the main method. It is just made to maintain the main method nothing else. Class chat\_server is made for maintaining the server user as we made this app suing sever socket, so it will do pair to pair connection. Class chat\_client is made to maintain client user. At first we need a server to run the system, so we at first run the chat\_server. It will establish a connection through the IP of the server computer. And then we need to use the same IP and port into client computer to connect to the system. And then, just start chatting.

**Source Code:** <https://goo.gl/TXKy8C>

**Input and Output:**

As it is a running application, output of server will be the input of client, and the output of client will be the input of the server.

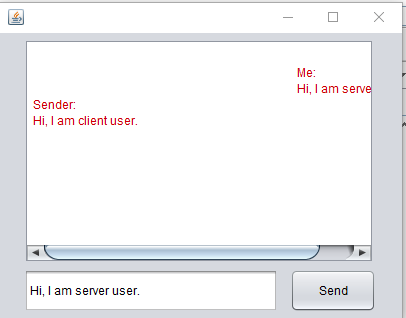


Fig: Server user.

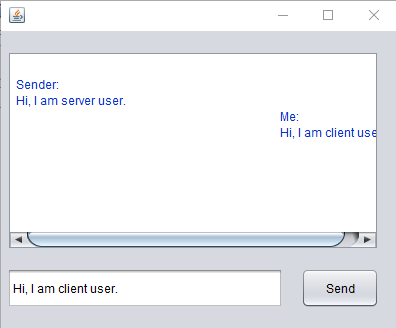


Fig: Client user.

**Conclusion:** We all know communication through network is very important now a days, it could be a start. It would be a great impact to understand server socket as it is a simple app. It can help to communicate like a pager as it is one to one pair connection.