

A Mini project report on
HANGMAN GAME



**Department of Computer Science and Engineering,
Galgotia's College of Engineering and Technology**

SUBMITTED TO

Mrs. Renu Mishra

Assistant Professor

SUBMITTED BY

SHIVANI SINGH – 1900970100111

SPARSH VERMA- 1900970100115

TANISHA JAIN – 1900970100119

TARUN PAL – 1900970100121

Acknowledgement

The satisfaction that accompanies with the completion of this project would be incomplete without the mention of the people who made it possible, without whose constant guidance and encouragement would have made efforts go in vain. We consider ourselves privileged to express gratitude and respect towards all those who guided us through the completion of this project.

We convey thanks to our project guide Mrs. Renu Mishra, Department of Computer Science and Engineering for providing encouragement and constant support and guidance which is of great help to complete this project successfully.

We would like to thank our parents for financing our studies in this college as well as for constantly encouraging us to learn Engineering. Their personal sacrifice in providing this opportunity to learn Engineering is gratefully acknowledged.

List of Figures

	<u>Page no.</u>
*4.6.3-Swing user interface	
Fig-4.6.3.1	16
Fig-4.6.3.2	16
Fig-4.6.3.3	17
Fig-4.6.3.4	18
Fig-4.6.3.5	18
Fig-4.6.3.6	19
Fig-4.6.3.7	19
*6-System design	
Fig-6.1	26
Fig-6.2	27
*7-Module	
Fig-7.1.1	28
Fig-7.1.2	29
Fig-7.1.3	30
Fig-7.1.4	32
Fig-7.1.5	33
*7.2-Data Base Diagram	
Fig-7.2	34
*7.3-Data Flow Design	
Fig-7.3	35
*8-Testing/Evaluation	
Fig-8.1	37

*10-Appendix (coding screenshots)

Fig-10.1	39
Fig-10.2	39
Fig-10.3	40
Fig-10.4	40
Fig-10.5	41

List of Tables	Page no
1-Abstract.....	6
2-Introduction.....	7
2.1-How to play.....	8
3-Language.....	9
4-System Analysis.....	10
4.1-Proposed system.....	10
4.2-Hangman Functional Requirement.....	11
4.3-Word Server Functional Requirement.....	12
4.4-Hangman Non-Functional Requirement.....	12
4.5-Operation Constraints.....	12
4.6-Hangman User Interface Prototype.....	14
4.6.1-Command line.....	14
4.6.2-Simple UI.....	14-16
4.6.3-Swing UI.....	16-20
4.6.4-Gallows UI.....	21-22
5-System Specification.....	23
5.1-Software Requirement.....	23
5.2-Hardware Requirement.....	23
5.3-Strategy.....	23-25
6-System Design.....	26-27
7-Detailed Design.....	28
7.1-Module.....	28-33
7.2-Data Base Design.....	34
7.3-Data Flow Design.....	35-36

8-Testing/ Evaluation	37
9-Conclusion & Utility.....	38
9.1-Future Scope & extension of this project.....	38
10-Appendix	39-41
11-References.....	42

1-Abstract

Hangman is a guessing game for two or more players. One player thinks of a word and the other tries to guess it by suggesting the letters. The word to be guessed is represented through a word which acts as the foremost hint and a row of blanks, disclosing the number of letters in the word as another hint. If the guessing player suggests a letter which occurs in the word, the program replaces all the respective blanks with the correct letter guessed. If the suggested letter does not occur in the word, the other player draws one element of the hangman diagram as a tally mark. The game is over when, either the full word has been guessed correctly or the player has no more chances left to make a guess while the word is still incomplete. In the latter case, the whole man hanging from the rope will be displayed!

The primary motivation behind our project was the original 'pencil and paper' guessing game with the same name i.e., The Hangman Game. The original game was played between, not more than two players; however, with our virtualization of the famous game, more than two players are allowed to indulge in this superb entertainment pastime!

Hangman is often used by teachers to practice spelling, vocabulary with a tinge of fun. The most popular way to play hangman game is to draw blank letters for the chosen word on a paper or on the blackboard and let the players guess the letters. Although in this era of digitalization, everything has been virtualized to an extent and something is hardly unavailable online. Thus, the hangman game also proves to be a good entertainer now that it is available online.

We have decided to use **JAVA** programming language and specifically **NetBeans** toolkit for adding graphics (GUI package) to the game. It will mainly iterate around all the 26 letters of the English Alphabet. Hence our input data will just be letters and as output the player comes to know whether he has guessed the right letter and lastly, won or not. In the end, in case the user fails, the program will automatically display the correct word.

2-Introduction

Hangman is a popular word guessing game where the player attempts to build a missing word by guessing one letter at a time.

The player is provided with certain number of chances and has to guess wisely the letter with the total number of letters in that word being the only hint provided to him. The game ends if a certain number of chances are exploited and the word is still incomplete or fully missing.

The game also gets over when the whole word has been guessed correctly by the player.

Traditionally, chances are tracked using a stick figure drawing of a person being hanged from a gallows. The figure is drawn one body part at a time, and the guesser loses when the entire figure has been drawn. This game is also the basis for the TV game show Wheel of Fortune.

2.1 How to Play

Our game provides the player with a choice of playing as a single user or a multi user.

In a single player game, a word is automatically generated from among a pre-loaded set of strings.

In a multi-user game, a friend (or player 2) is involved. This player enters the word to be guessed by the player one.

A row of blanks will be displayed in the frontend screen where the player types a letter to be guessed. If that letter is in the word then the code will generate the letter at all the correct places. If the letter isn't in the word then we cross out the lifelines (which are usually a finite no. of chances) from the list. The player will continue guessing the letters until he guesses all the correct letters or he will end up losing all the lifelines and he will be declared a **LOSER**.

3-Language Choice

We have chosen JAVA programming language to code the backend of our gaming software for its robust nature and the advantage that it comes with many pre-loaded functions.

We have chosen NetBeans for programming the frontend of our game.

4-SYSTEM ANALYSIS

4.1 PROPOSED SYSTEM :-

Product Perspective

The benefit of this program is primarily that it allows someone to play as a solitaire game without requiring another person to participate.

Product Features

The application's purpose is to simulate the word game "Hangman." The major features are:

Obtain a hidden word from a remote computer on the internet.

Manage the game play: getting the player guess, evaluating it, and updating the game board.

Determining if the game is won or lost.

Allow the user to specify an alternate user interface.

User Characteristics

Hangman is designed for personal computer users who enjoy simple solitaire word games. Hangman is purely recreational software with little or no utilitarian value. The user is assumed to be competent using a WIMP interface (windows, icons, menus, pointers) to operate simple computer games such as Minesweeper or Klondike Solitaire.

The user is assumed to be familiar with the rules of Hangman. It is assumed the user knows how to start Java applications from the command prompt and provide command line parameters.

Assumptions and dependencies

Hangman is dependent upon the Hangman Word Server application. Word Server must be running in order for Hangman to obtain a hidden word to use for the game. Word Server can be running on the same computer, or on a remote computer in which case the computer running Hangman will need an active network connection to the remote computer. Hangman uses hidden words in English.

Hangman assumes the words obtained from the Word Server are valid.

Scheduler will run on Windows 9x/XP and Linux platforms.

General Constraints

none.

4.2 Hangman Functional Requirements

A. Starting the Application

Allow the user to enter on the command line the IP address of a remote computer that is running a Hangman Word Server. If the user doesn't provide an IP address, use as a default the address "localhost."

At the start of each game the application will request a word from the Hangman Word Server specified by IP address.

The word obtained from the Word Server will be designated as the "hidden" word the player tries to guess.

If no connection can be made to a Word Server on the given IP address display an error message on the console "Apparently no hangman word server is running at <IP address>."

If a word is obtained from the Word Server, game play begins by offering the player their first turn.

B. Playing a game

At each turn the application will display a visual indicator of how many letters are in the hidden word and if any of the letters have been correctly guessed they are shown in the proper position in which they appear in the word.

The application will display a "guess count" which shows how many incorrect guesses the player has made. An incorrect guess is guessing a letter which is not in the hidden word.

The application will allow the player to enter a letter.

If the letter entered is not between A and Z display a message "Invalid move" and allow the player to guess again (without penalty).

When the player enters a valid letter the application will check to see if the game is over and if not will continue to the next turn.

C. Ending a game

The player wins by correctly guessing all the letters in the hidden word.

The player loses if he/she makes seven incorrect guesses.

If the player wins the application will display a congratulatory message.

If the player loses the application will display a consolation message and will reveal the hidden word.

When the game is over (either win or loss) the application will offer the player an opportunity to begin another game.

If the player indicates they want to play again, a new game is started.

If the player indicates they do not want to play again, the application is terminated.

Note: there is no way for a player to request termination of play during the middle of a game.

4.3 Word Server Functional Requirements

Hangman Word Server runs as a stand alone application independently from the Hangman Solitaire game.

When a client program makes a network connection to the server's port the server will respond by sending a character string then closing the connection.

The character string will be randomly selected from a file of English words.

User Interface Requirement

When started, Word Server must display a message indicating the host address on which it is running, e.g.:

4.4 Hangman Non-Functional Requirements

General Guidelines: Performance and reliability are not very important. Priority should be given to adaptability, maintainability, and usability.

4.5 Operating Constraints

The program requires the JRE v1.4 with Swing from Sun Microsystems.

Platform constraints

The program requires a 486 or higher processor with 16 Megabytes of RAM and 5 Megabytes of available hard drive space.

Modifiability

If it is desired to change the number of turns in a game, the developer will be able to make the required changes in < 1 person-hours.

Adaptability

The user must be able to specify an alternate user interface at execution time on the command line.

Any alternate user interface must be able to "plug in" at run time without recompiling the application.

Any alternate user interface must implement the HangmanUI interface provided by Dr. Dalbey (see External Interfaces).

Documentation

A short (< 200 words) message explaining command line syntax should be displayed if the user enters no command line parameters.

Portability

The program will run on Win 95/98/XP and Red Hat Linux 9.

Reliability

Since the program is purely for recreation and involves no user data, reliability is of low importance.

Security

The program will not access any user data files or programs.

The program will not alter or replace any system files.

Usability

A new user should be able to play a complete game of hangman in less than ten minutes.

A new user should commit less than one error in use of the game (e.g. selecting the wrong letter) every ten minutes.

A user who is familiar with the rules of Hangman be able to correctly operate the program without any written documentation.

Testability

(optional) If the user provides an IP address of zero then skip getting a word from the Word Server and use a default word "calpoly" as the hidden word.

Deployment

The program is to be deployed as executable java bytecode files (".class" files) that must fit on a 1.2M flexible disk. They must use no specific java package.

Performance

Desired Response Times (not critical) : At game start: less than five seconds

After each turn: less than two seconds

Word Server Non-Functional Requirements

Performance

After receiving a request, the word server must respond in less than half a second.

Maintainability

It is desirable that the game administrator be able to modify the words data file using a simple text editor.

Adding a new word to the word data file should take less than ten minute

4.6 Hangman User Interface Prototype

[[Command line](#)] [[Simple UI](#)] [[Swing UI](#)] [[Gallows UI](#)]

4.6.1-Command Line Parameters

When invoked with no command line parameters, display an explanation of the parameters:

javaHangmanApp

Usage: java HangmanApp [ui class] [-t]

[ui class] is name of user interface class (default is 'SimpleUI').

[-t] optional flag to turn on test mode (for developers'). Uses wordlist as word source.

Welcome to Hangman!

Guesses: 0

4.6.2-SimpleUI

For the SimpleUI version of the user interface, the console will be used for input and output. The hidden word is shown with an underscore in the place of each letter. Correctly guessed letters are filled in placed in the proper position. In the example game below, the user runs out of guesses and loses the game. The user input is shown in bold face.

java

HangmanAppSimpleUI

Welcome to Hangman!

Guesses:	-	-	-	-	0
Enter					move: e
Guesses:	E	-	-	-	E
Enter					0
Guesses:	E	-	-	-	E
Enter					1
Guesses:	E	-	-	-	E
Enter					2
Guesses:	E	-	-	-	E
Enter					3
Guesses:	E	-	-	-	E
Enter					4
Guesses:	E	-	-	-	E
Enter					5
Guesses:	E	-	-	-	E
Enter					6
Guesses:	E	A	-	-	E
Enter					6
Guesses:	E	A	-	-	E
Enter					7

You		have		Lost!
The	word		was:	PEACE
Play				Again?
y				

In the example game below, the user guesses the word and wins the game.

Guesses:	-	-	-	-	-	0
Enter						move: e
Guesses:	-	E	-	-	-	0
Enter						move: a
Guesses:	-	E	-	-	A	0
Enter						move: o
Guesses:	-	E	-	-	A	1
Enter						move: i

You have Won!
 Play Again?
n

Welcome to Hangman!

[illegible]

4.6.3-SwingUI

To run the program with the Swing user interface, provide the SwingUI class name on the command line.

java HangmanAppSwingUI

When the game begins, a window appears displaying the game board, asking for the players choice to play as a SINGLE USER or to play with his friends (MULTI-USER).

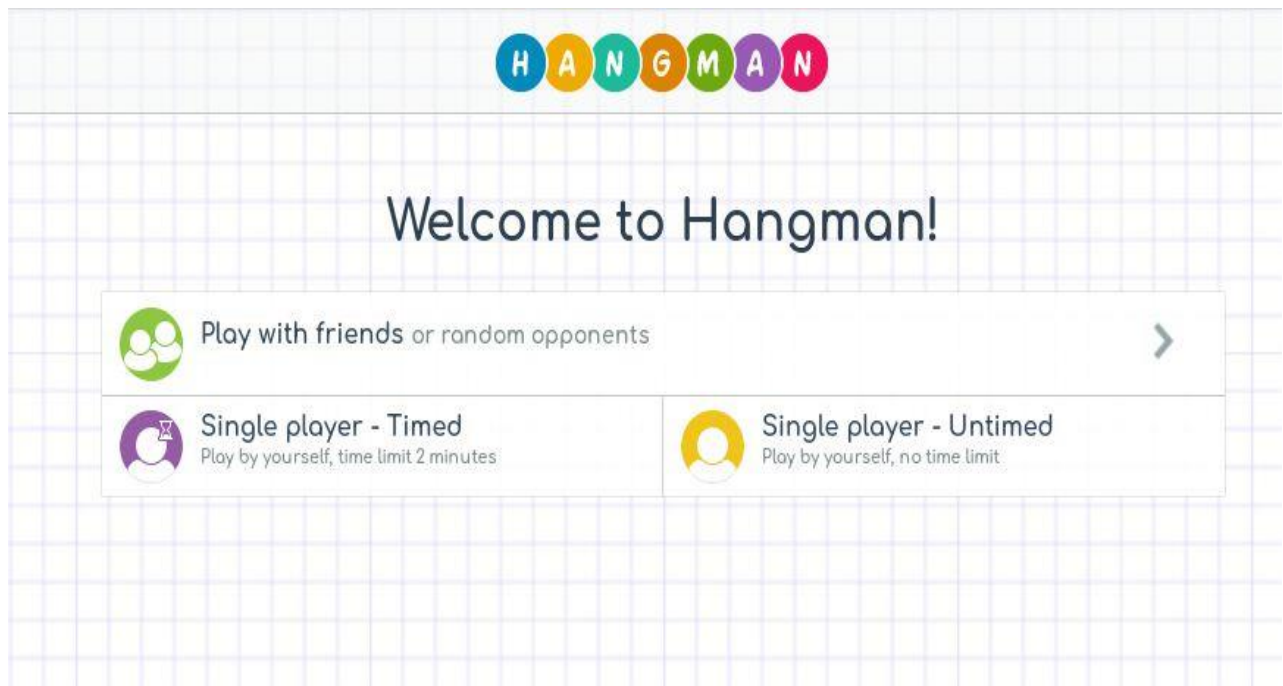


Fig-4.6.3.1

After choosing the options displayed....The game begins. Here the user has chosen to play as a single player. The following screen will appear .

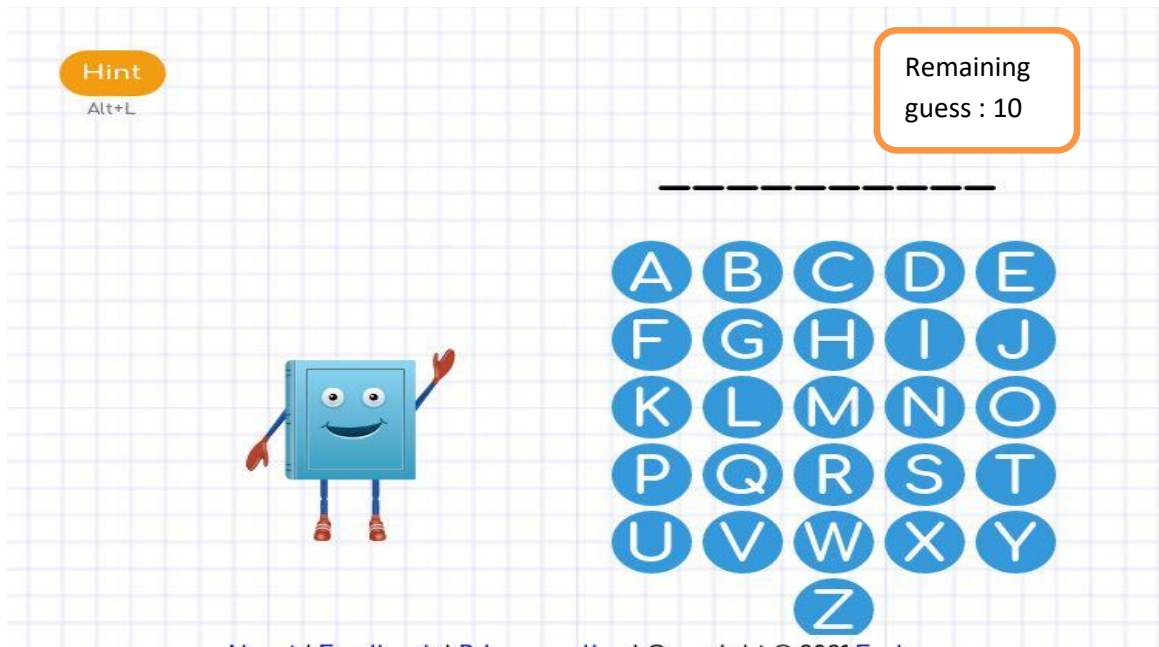


Fig-4.6.3.2

During play, the player uses the mouse or keyboard to select buttons for the letter they want to guess. Here the user guesses the letter G, which was incorrect, so it was colored brown and the remaining guess is reduced by 1. In the second guess the player guesses the letter N which was correct so it was filled in the blanks where it appears in the word.

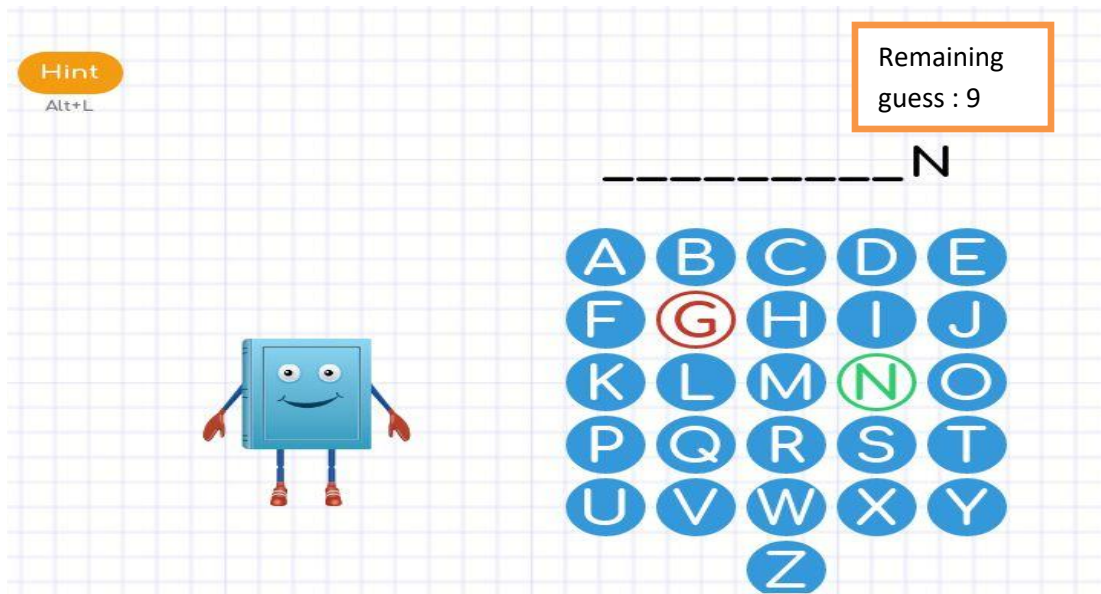


Fig-4.6.3.3

Again the user Guesses the letter D incorrectly. And the pillar of our hangman starts building.

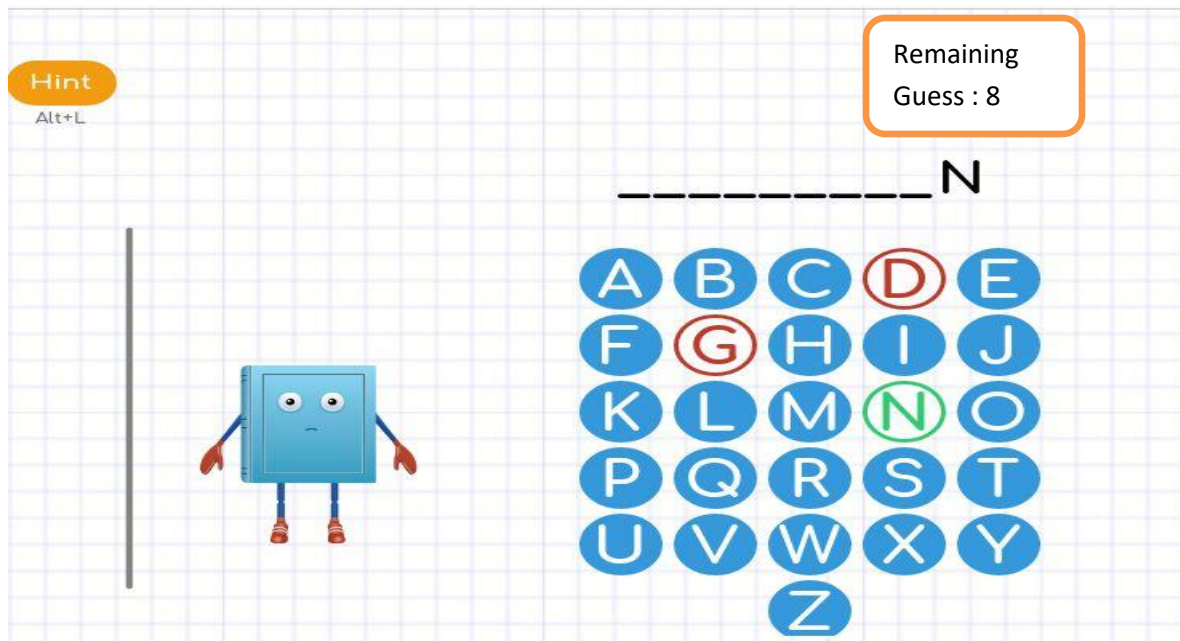


Fig-4.6.3.4

And the game Continues . If the user gets stuck somewhere, he / she can take some hint by clicking on the hint button on the left corner.

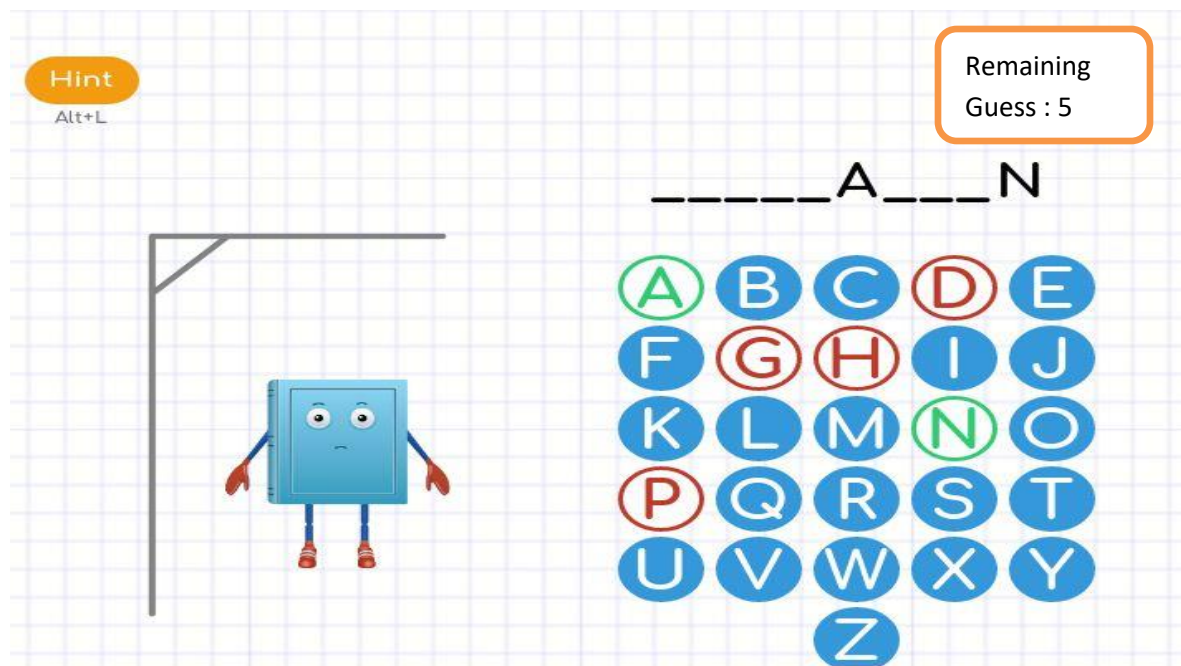


Fig-4.6.3.5

In the example below, the user has used all The guesses so they lost the game.

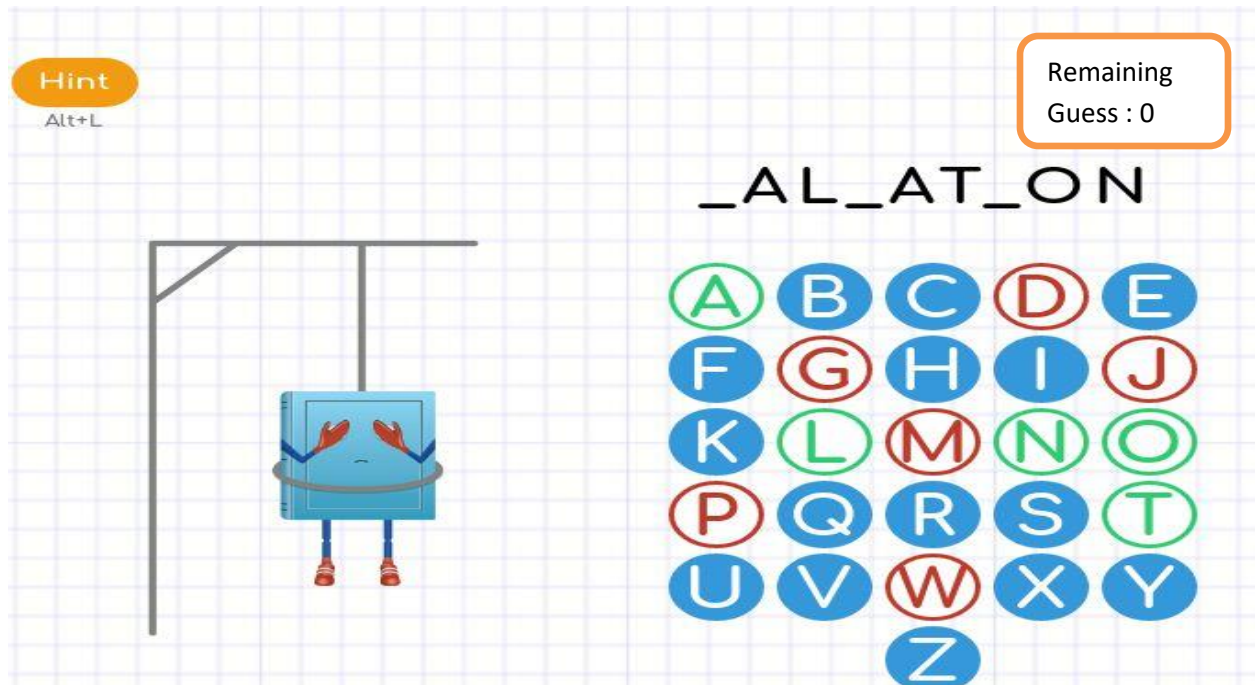


Fig-4.6.3.6

The picture below shows the Play Again dialog which appears when the game is over. Note that the buttons have keyboard shortcuts and that the default button is Yes.

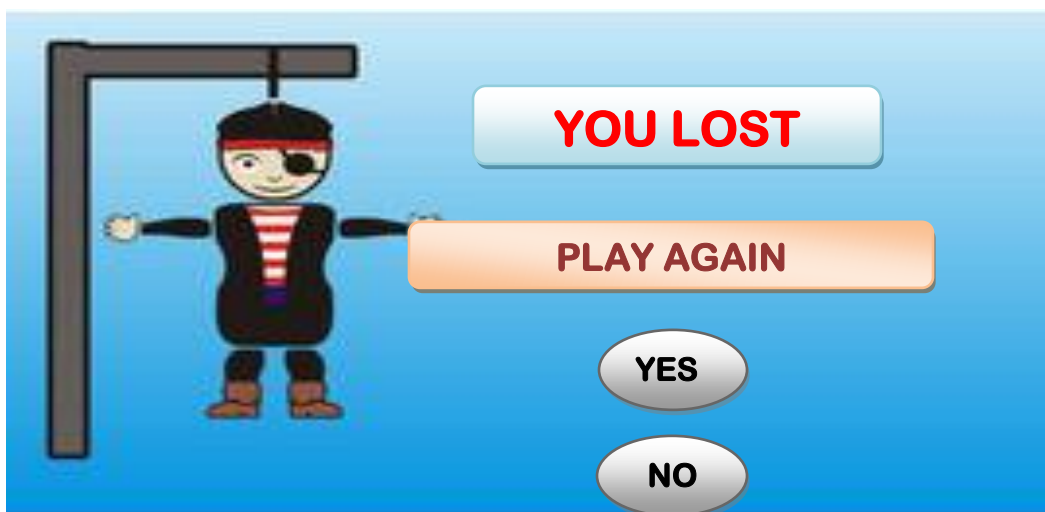


Fig-4.6.3.7

This interface is also console based but features more clever ASCII graphics of a stick figure hanging from a gallows. It also shows available letters.

Possible moves:

A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	Q	R	S	T	U	V	W	X	Y	Z
---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---

Enter move: **o**

[illegible]

The
Play Again?

/
word

|\
was:

SPECIAL

5- System Specifications

5.1-SOFTWARE REQUIREMENTS

1. Java
2. Android Studio
3. Android SDK

5.2-HARDWARE REQUIREMENTS

1. Microsoft® Windows® 7/8/10 (32- or 64-bit)
2. 2 GB RAM minimum, 8 GB RAM recommended
3. 2 GB of available disk space minimum, 4 GB Recommended (500 MB for IDE + 1.5 GB for Android SDK and emulator system image)
4. 1280 x 800 minimum screen resolution
5. For accelerated emulator: 64-bit operating system and Intel® processor with support for Intel® VT-x, Intel® EM64T (Intel® 64), and Execute Disable (XD) Bit functionality [12].

5.3-STRATEGY

In English words, the twelve most commonly occurring letters are this (in descending order): e-t-a-o-i-n-s-h-r-d-l-u. This and other letter lists are used by the guessing player to increase the odds when it is their turn to guess. On the other hand, the same lists can be used by the puzzle setter to stump their opponent by choosing a word you planned avoids common letters (e.g. hangman or apple) or one that contains rare letters (e.g. mango)

Another common strategy is to guess vowels first, as English only has five vowels (a, e, i, o, and u, while y may sometimes, but rarely, be used as a vowel) and almost every word has at least one.

Example game

The following example will explain a player how to guess the word hangman using a strategy based solely on letter frequency.



Word: _ _ _ _ _
Guess: E
Misses:



Word: _ _ _ _ _
Guess: T
Misses: E



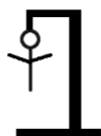
Word: _ _ _ _ _
Guess: A
Misses: E, T



Word: _ A _ _ _ A _
Guess: O
Misses: E, T



Word: _ A _ _ _ A _
Guess: I
Misses: E, T, O



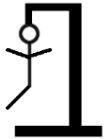
Word: _ A _ _ _ A _
Guess: S
Misses: E, T, O, I



Word: _ A _ _ _ A _

Guess: N

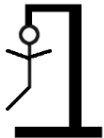
Misses: E, T, O, I, S



Word: _ A N _ _ A N

Guess: H

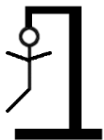
Misses: E, T, O, I, S



Word: HA N _ _ A N

Guess: H

Misses: E, T, O, I, S



Word: HA N _ _ A N

Guess: R

Misses: E, T, O, I, S



Word: HA N _ _ A N

Guess:

Misses: E, T, O, I, S, R

Guesser Loses - The answer was HANGMAN.

6- System Design (Low Level)

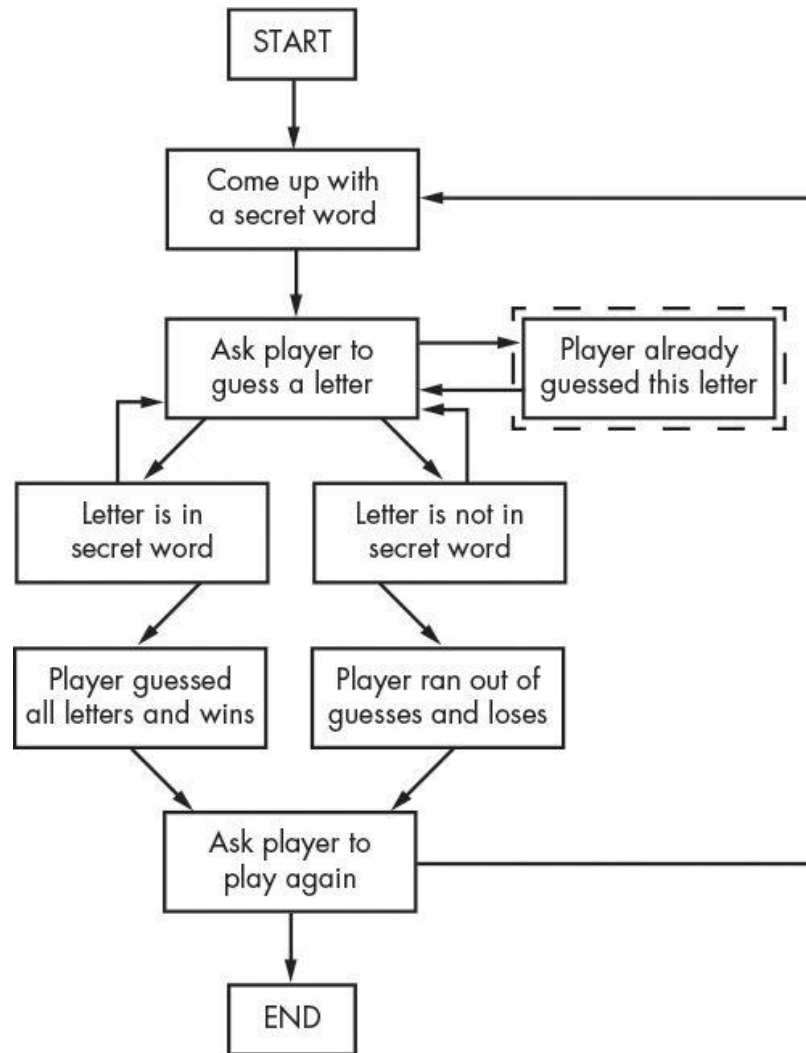


Fig-6.1



Fig-6.2

7-Detailed Design

7.1 Module

The underlining data model should consist of the following linear recursive structures.

An abstract *IWord* interface to represent the input word. An *IWord* can be empty or non-empty. When it is empty it contains nothing. When it is non-empty, it contains a character and another substructure *IWord* called its rest. A non-empty *IWord* can be hidden or visible. When it is hidden, its String representation is "-" followed by the String representation of its rest. When it is visible, its String representation is the String representation of the character itself followed by the String representation of its rest.. *IWord* should have enough intelligence to interact with the rest of the system as described below.

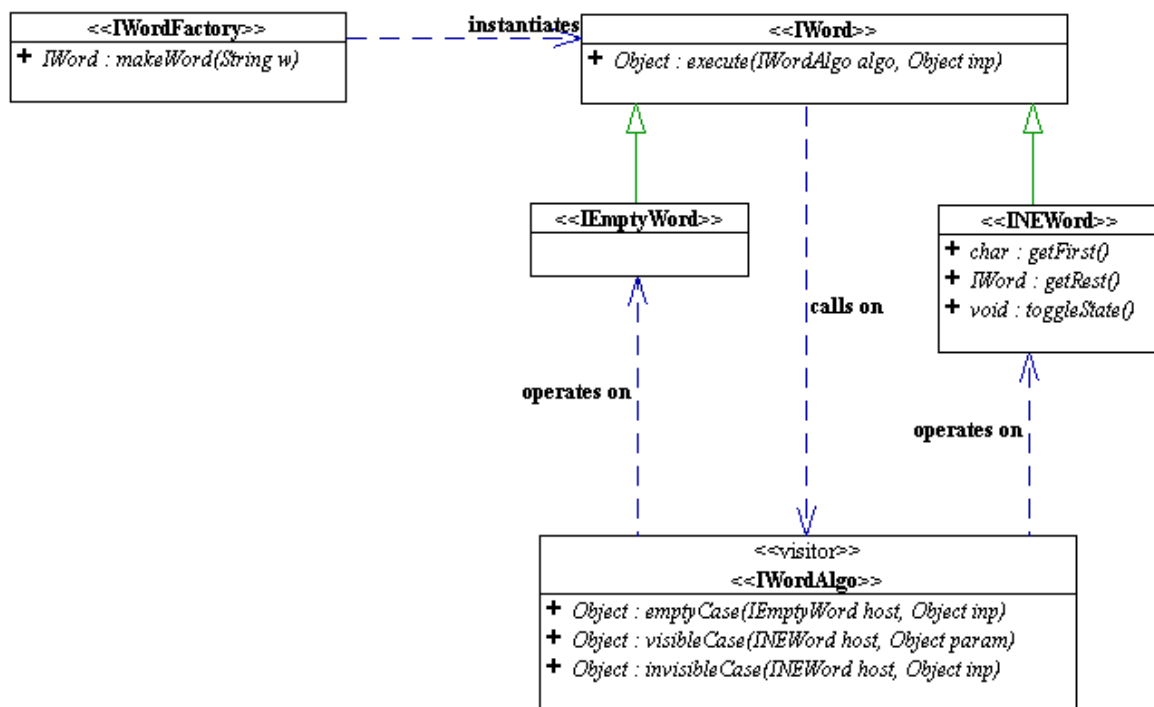


Fig:-7.1.1

An *IBody* can be empty or non-empty. When it is empty it contains nothing. When it is not empty, it contains another substructure *IBody* called *rest*. An *IBody* can change state from invisible to visible. *IBody* should have enough intelligence to interact with the rest of the system as described below.

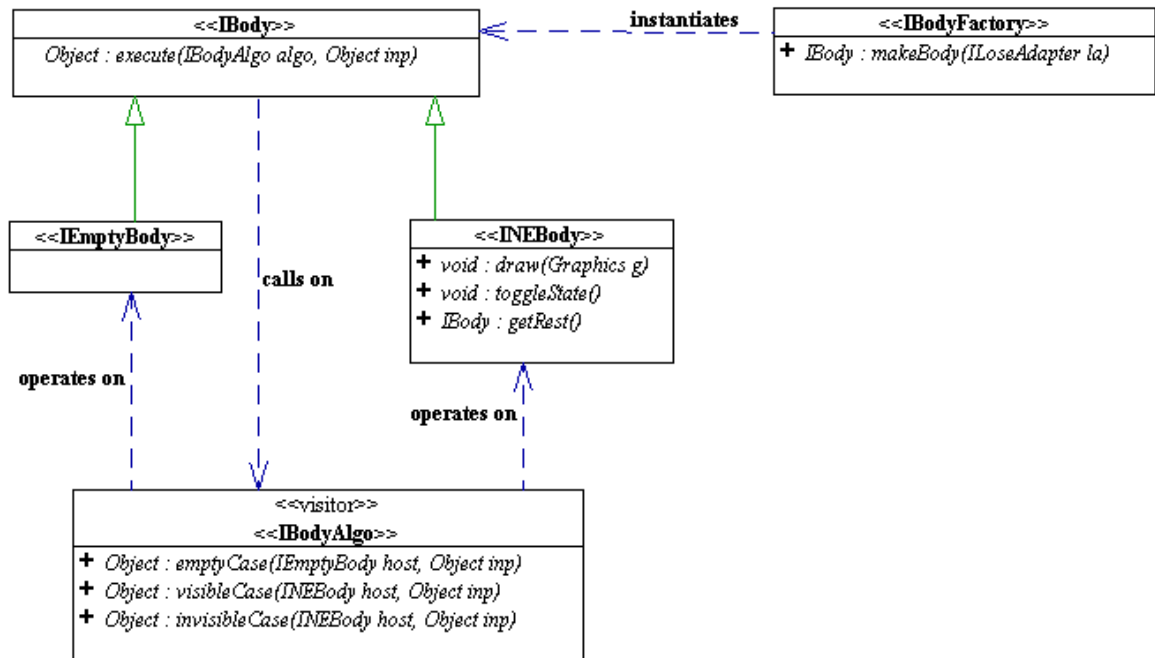


Fig:-7.1.2

The above data models are encapsulated into a single class that represents the operational model of the Hangman game, called (duh) Hangman Game. This class can initialize the game, allow guesses of characters, show what characters in the answer have been correctly guessed, draw onto a supplied graphics surface the body parts associated with incorrect guesses, and finally to indicate whether the game has been won or lost.

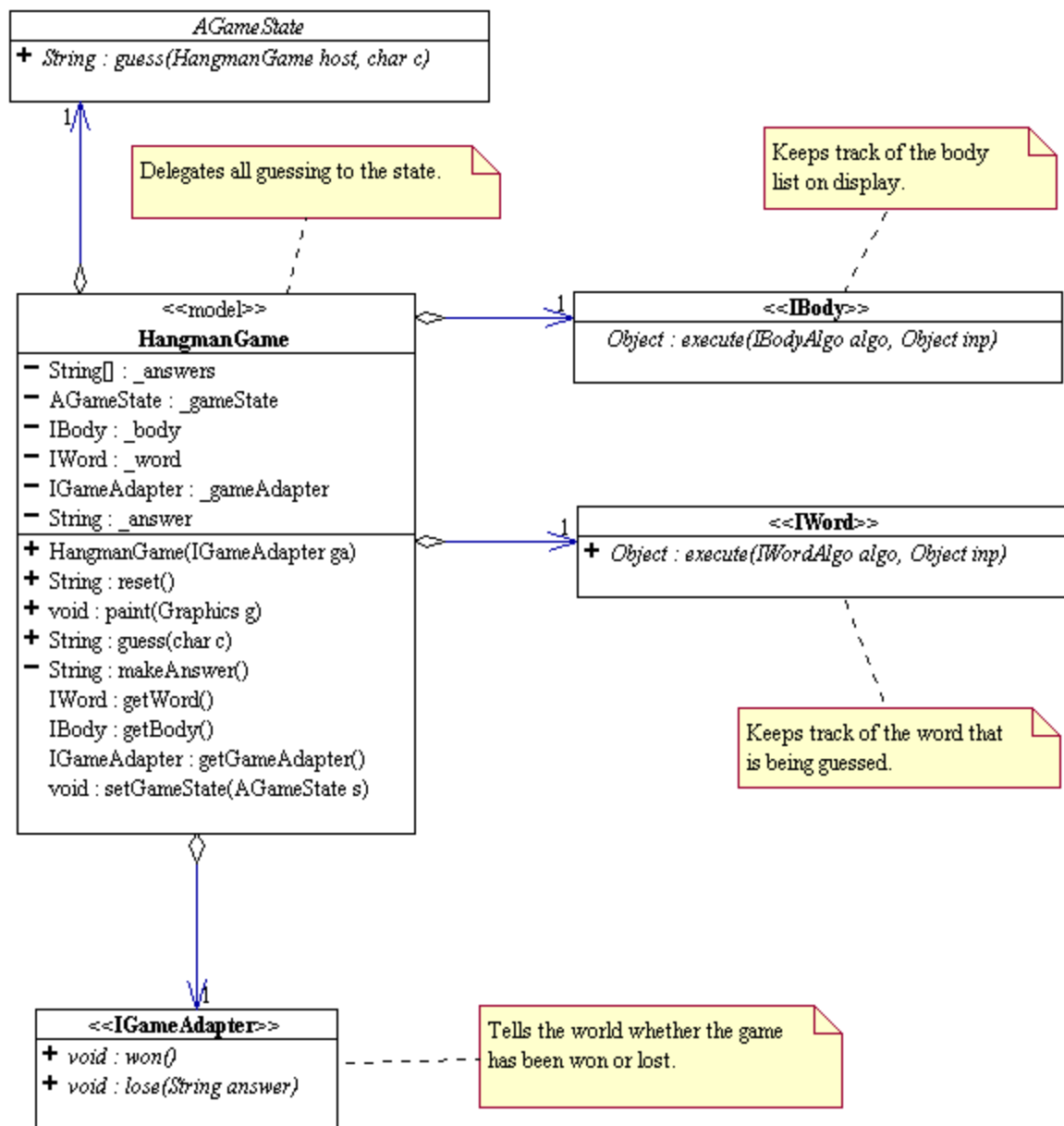


Fig:-7.1.3

The View

The GUI should have the following features:

A GUI component to display the answer as a series of dashes ("-") and letters (if they have been correctly guessed).

A GUI component for the program to draw the various parts of a body.

A GUI component for you to enter a character as a guess and obtain a response from the program. If your guess does not match any character in the original word, one part of the body will be drawn. If there is a match, the character will fill the appropriate dashes (displayed in item 1). The game ends when there is no more body parts to draw, in which case you lose, or when all the dashes are replaced by the actual characters in the word, in which case you win. It is the job of the model to figure out if a guess is correct or not and to provide the proper String and graphical representations of the current state of the game. All the view needs to do is to translate the user's keystrokes and mouse clicks into the correct calls to the model and to perform any commands issued by the model.

A GUI component to indicate if the game has been won or lost. Optionally, the view can indicate winning or losing by playing an audio clip.

A GUI component for you to reset the game and start over.

As the model object is unaware of the GUI (view), so the view object should not know anything about the model that it is displaying. All communication from the view to the model is through 2 interfaces: IPaintAdapter and IGameControlAdapter.

Anonymous inner classes will be used extensively to connect the view (HangmanGUI) to its GUI components (buttons, text fields, labels, etc.);

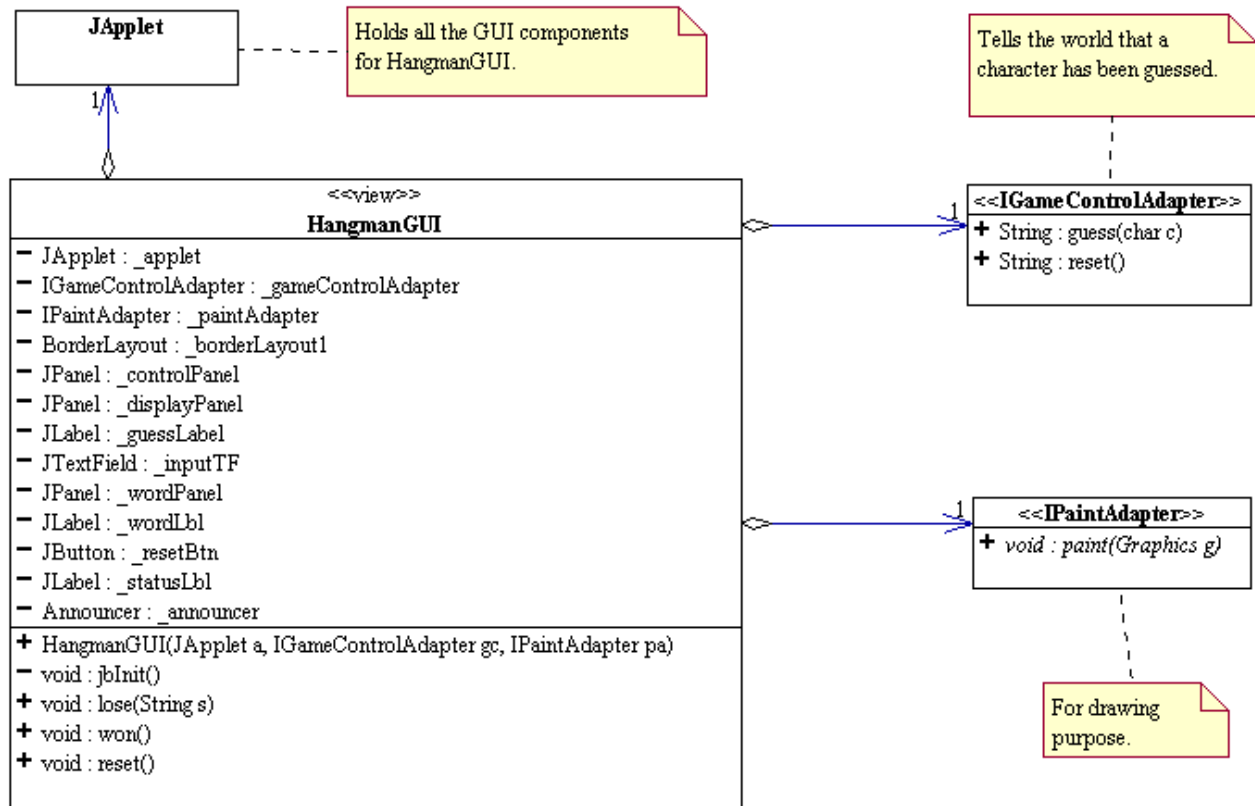


Fig:-7.1.4

The Controller

The controller's job is to

instantiate the view.

instantiate the model

instantiate the 3 interfaces needed for the model and view to communicate.

install the 3 objects implementing those 3 interfaces into the model and view, thus establishing the ability to communicate between them.

ensure that the system starts up properly.

As in the view, anonymous inner classes will be used to instantiate the 3 interface adapters.

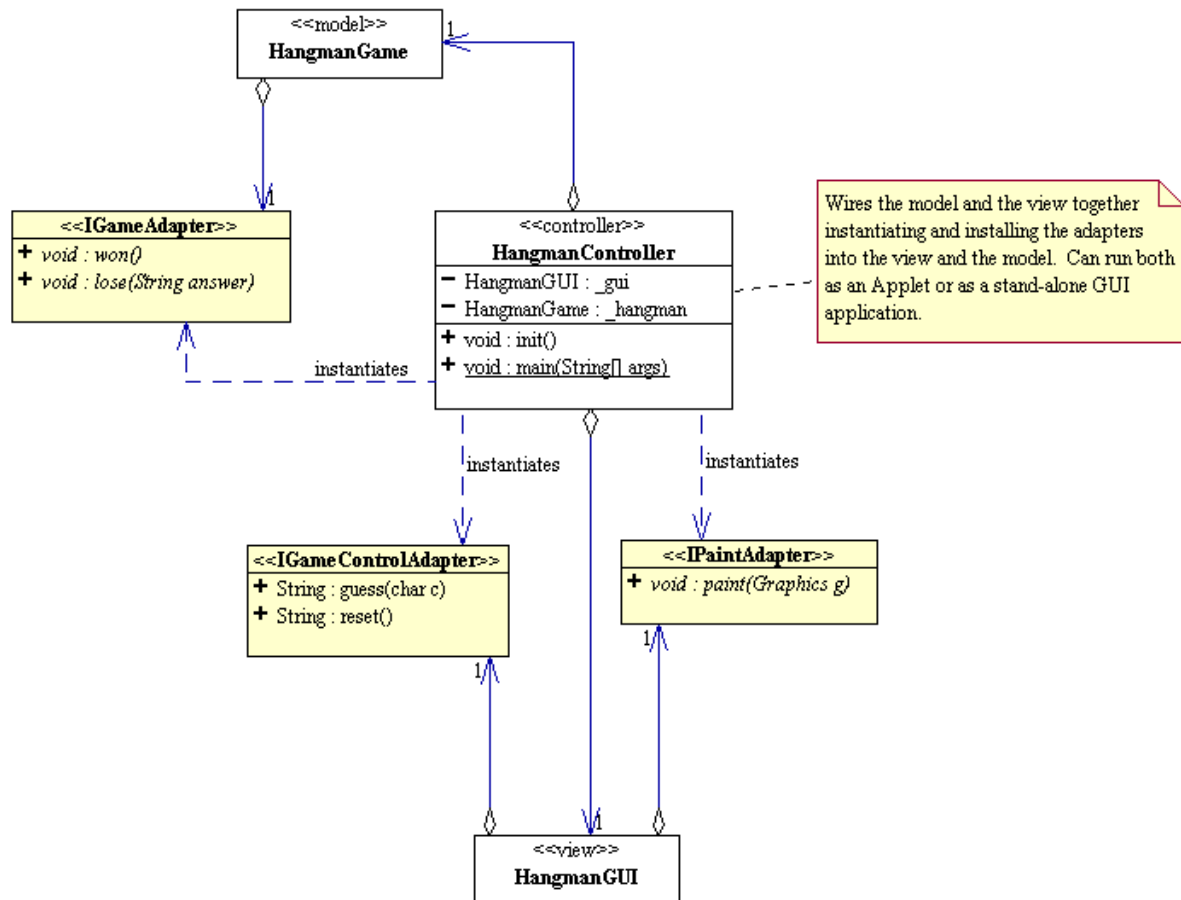


Fig:-7.1.5

7.2 Data base design

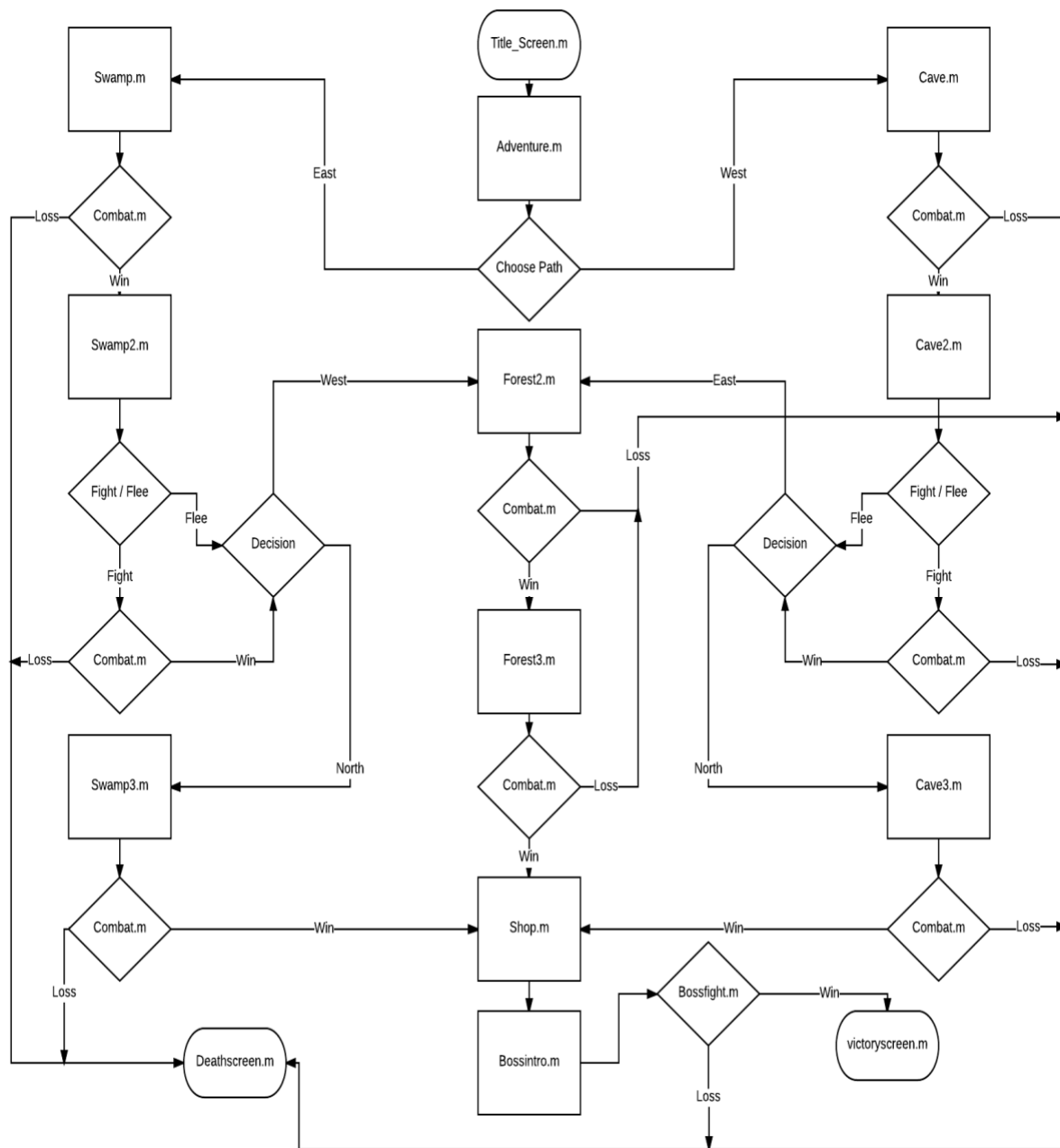


Fig-7.2

7.3-DFD MODELLING

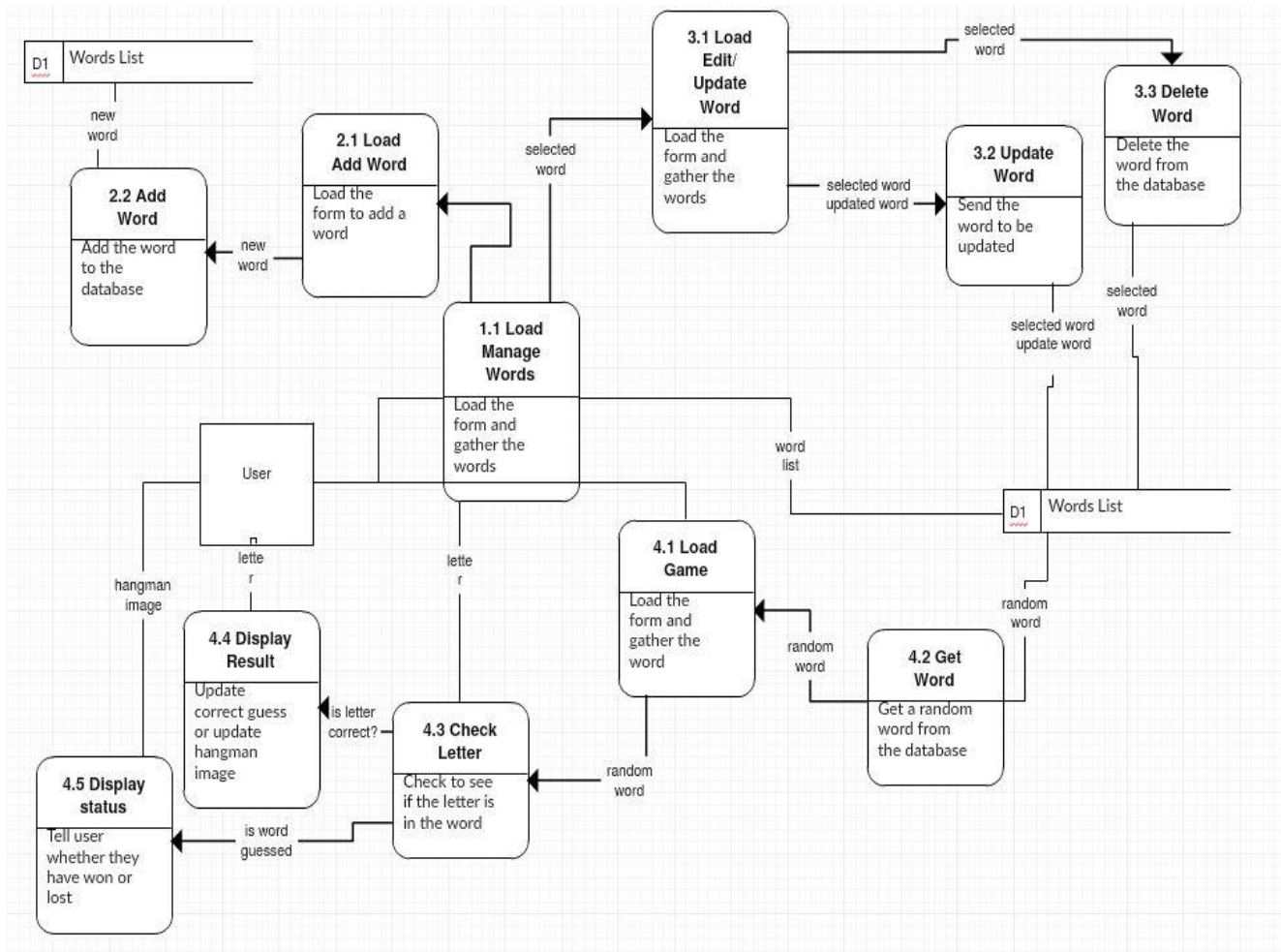


Fig-7.3

8-Testing/Evaluation Plan

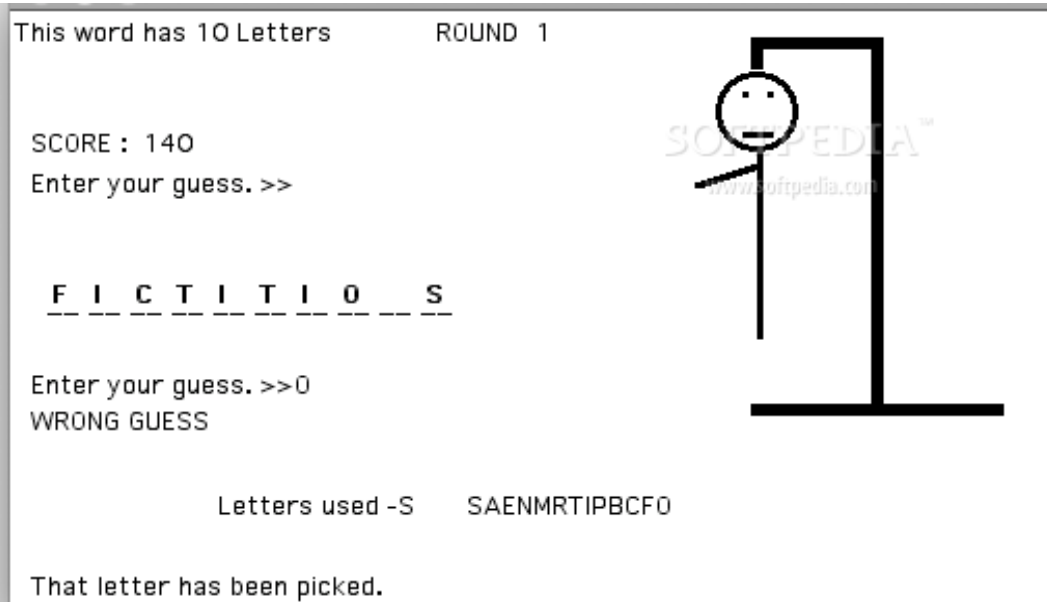


Fig-8.1

For this particular game , there will not be any specific test case except that , I will have to check whether the code works correctly or not for the words which are to be guessed. Also , by giving wrong letters more than the number of chances , the user is declared the loser.
For example:-

Word: _ A N _ _ A N
Guess: H
Misses: e,i,o,s,t

Word: H A N _ _ A N
Guess: R
Misses: e,i,o,s,t

Word: H A N _ _ A N
Guess: N/A
Misses: e,i,o,r,s,t

Guesser loses - the answer was HANGMAN.

9- Conclusion/Utility of Hangman Games

Hangman is used often by teachers to practice spelling, vocabulary and just for fun. The most popular way to play hangman games offline is to draw blank letters for the chosen word on a paper or on the blackboard and let the players guess the letters. For each incorrect guess, another part of the man is drawn. If the picture is complete before the word is revealed the hangman game is lost and the character is hanged, if the word is revealed before the execution the game is won.

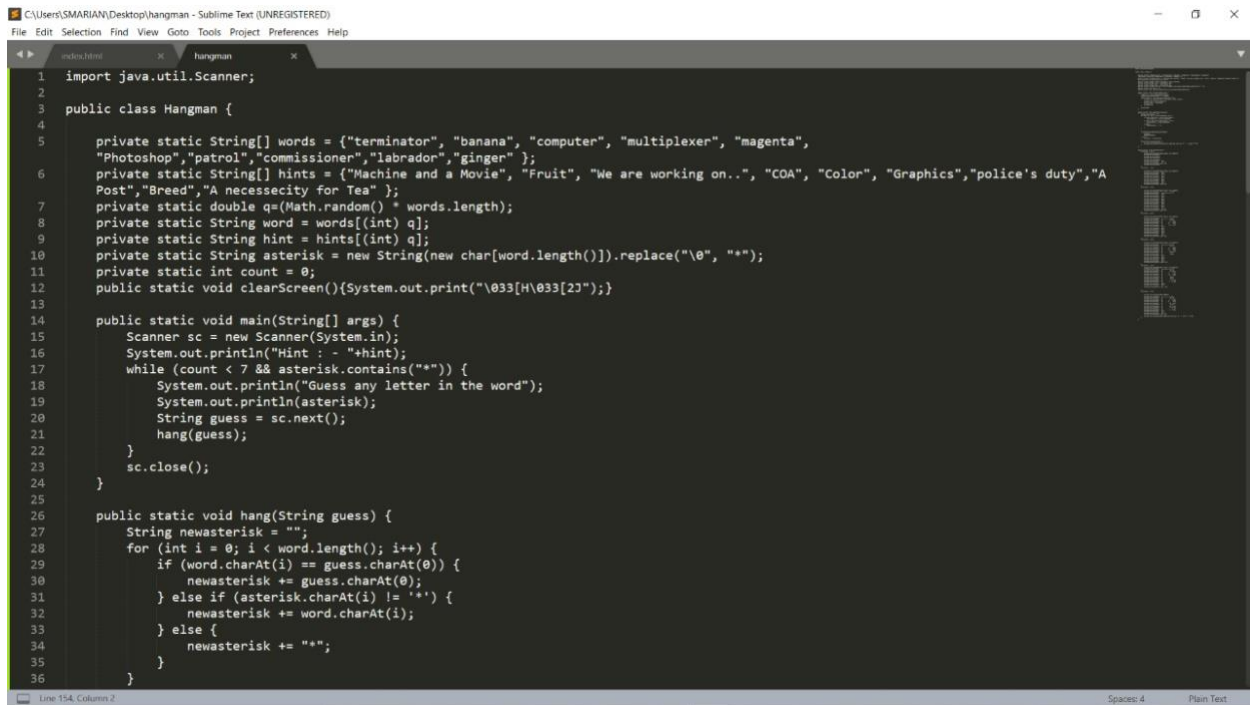
Justification:-

The hangman game illustrates *how much with discretion one can make the best choice at each stage ...* strategies rely on obligations being fulfilled in the future(player strives to win). I picked up this topic as it was a simple but prudent game to check the intuition and tactics of a player in guessing the word in very less time.It subtly judges the efficient thought process of the player well.In the perspective of coding also, I feel it was a proper project for me to experience python GUI for the first time.

9.1Future Scopes and extensions of this project:-

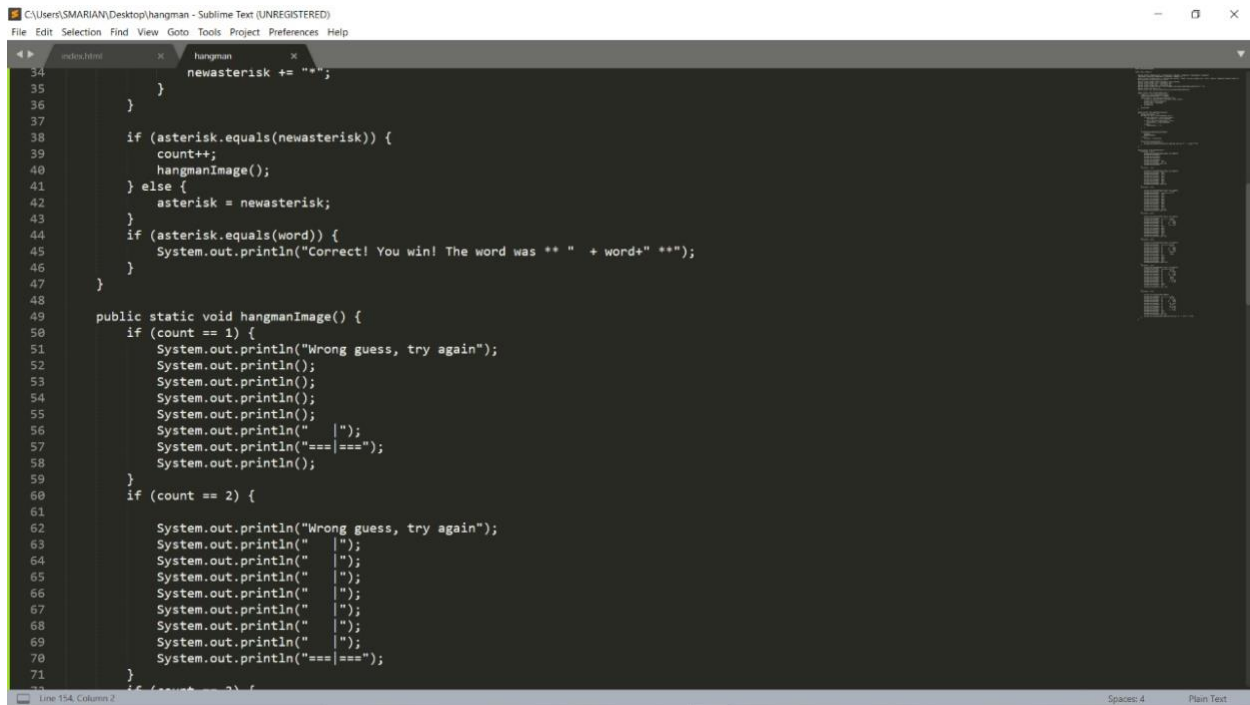
This game can have varied applications in the context of *word formations* and puzzles.Its knowledge can be valuable to many other games like CROSSWORD PUZZLES, WHEEL OF FORTUNE,SCRABBLE.We can also have an investigation of very popular and commonly used letters in most of the words. Make a frequency distribution in graph out of it. The underlying mathematical concepts are Data Collection and Analysis,Presentation and Interpretation which can have lot of implications in language processing and study of graphs and testing conjectures.Also, we can find out that the most popular letter in the English language is "e". The letter frequency of all letters in the English language is: e t a o i n s r h l d c u m f p g w y b v k x j q z.

10-Appendix (coding screenshots)



```
1 import java.util.Scanner;
2
3 public class Hangman {
4
5     private static String[] words = {"terminator", "banana", "computer", "multiplexer", "magenta",
6     "Photoshop", "patrol", "commissioner", "labrador", "ginger"};
7     private static String[] hints = {"Machine and a Movie", "Fruit", "We are working on..", "COA", "Color", "Graphics", "police's duty", "A
8     Post", "Breed", "A necessecity for Tea"};
9     private static double q=(Math.random() * words.length);
10    private static String word = words[(int) q];
11    private static String hint = hints[(int) q];
12    private static String asterisk = new String(new char[word.length()]).replace("\0", "");
13    private static int count = 0;
14    public static void clearScreen(){System.out.print("\033[H\033[2J");}
15
16    public static void main(String[] args) {
17        Scanner sc = new Scanner(System.in);
18        System.out.println("Hint : - "+hint);
19        while (count < 7 && asterisk.contains("")) {
20            System.out.println("Guess any letter in the word");
21            System.out.println(asterisk);
22            String guess = sc.next();
23            hang(guess);
24        }
25        sc.close();
26    }
27
28    public static void hang(String guess) {
29        String newasterisk = "";
30        for (int i = 0; i < word.length(); i++) {
31            if (word.charAt(i) == guess.charAt(0)) {
32                newasterisk += guess.charAt(0);
33            } else if (asterisk.charAt(i) != '') {
34                newasterisk += word.charAt(i);
35            } else {
36                newasterisk += " ";
37            }
38        }
39    }
40}
```

Fig-10.1



```
34 newasterisk += " ";
35 }
36 }
37
38 if (asterisk.equals(newasterisk)) {
39     count++;
40     hangmanImage();
41 } else {
42     asterisk = newasterisk;
43 }
44 if (asterisk.equals(word)) {
45     System.out.println("Correct! You win! The word was ** " + word+ " **");
46 }
47 }
48
49 public static void hangmanImage() {
50     if (count == 1) {
51         System.out.println("Wrong guess, try again");
52         System.out.println();
53         System.out.println();
54         System.out.println();
55         System.out.println();
56         System.out.println(" |");
57         System.out.println("===|===");
58         System.out.println();
59     }
60     if (count == 2) {
61         System.out.println("Wrong guess, try again");
62         System.out.println(" |");
63         System.out.println(" |");
64         System.out.println(" |");
65         System.out.println(" |");
66         System.out.println(" |");
67         System.out.println(" |");
68         System.out.println(" |");
69         System.out.println(" |");
70         System.out.println("===|===");
71     }
72     if (count == 3) {
73         System.out.println("Wrong guess, try again");
74         System.out.println(" |");
75         System.out.println(" |");
76         System.out.println(" |");
77         System.out.println(" |");
78         System.out.println(" |");
79         System.out.println(" |");
80         System.out.println(" |");
81         System.out.println(" |");
82         System.out.println("===|===");
83     }
84 }
```

Fig-10.2

```

C:\Users\SMARIAN\Desktop\hangman - Sublime Text (UNREGISTERED)
File Edit Selection Find View Goto Tools Project Preferences Help

index.html hangman
71 }
72 if (count == 3) {
73
74     System.out.println("Wrong guess, try again");
75     System.out.println(" =====");
76     System.out.println(" |");
77     System.out.println(" |");
78     System.out.println(" |");
79     System.out.println(" |");
80     System.out.println(" |");
81     System.out.println(" |");
82     System.out.println(" |");
83     System.out.println(" |");
84     System.out.println(" |");
85     System.out.println(" |");
86     System.out.println("=====");
87 }
88 if (count == 4) {
89
90     System.out.println("Wrong guess, try again");
91     System.out.println(" =====");
92     System.out.println(" | _ _");
93     System.out.println(" | / \");
94     System.out.println(" | |");
95     System.out.println(" | \ _ /");
96     System.out.println(" |");
97     System.out.println(" |");
98     System.out.println(" |");
99     System.out.println(" |");
100    System.out.println(" |");
101    System.out.println(" |");
102    System.out.println("=====");
103 }
104 if (count == 5) {
105
106     System.out.println("Wrong guess, try again");
107     System.out.println(" =====");
108     System.out.println(" | _ _");
109     System.out.println(" | / \");

```

Fig-10.3

```

C:\Users\SMARIAN\Desktop\hangman - Sublime Text (UNREGISTERED)
File Edit Selection Find View Goto Tools Project Preferences Help

index.html hangman
108 System.out.println(" | _ _");
109 System.out.println(" | / \");
110 System.out.println(" | |");
111 System.out.println(" | \ _ /");
112 System.out.println(" |");
113 System.out.println(" |");
114 System.out.println(" |");
115 System.out.println(" |");
116 System.out.println(" |");
117 System.out.println(" |");
118 System.out.println("=====");
119 }
120 if (count == 6) {
121     System.out.println("Wrong guess, try again");
122     System.out.println(" =====");
123     System.out.println(" | _ _");
124     System.out.println(" | / \");
125     System.out.println(" | |");
126     System.out.println(" | \ _ /");
127     System.out.println(" |");
128     System.out.println(" |");
129     System.out.println(" | / \");
130     System.out.println(" |");
131     System.out.println(" |");
132     System.out.println(" |");
133     System.out.println("=====");
134 }
135 if (count == 7) {
136
137     System.out.println("GAME OVER!");
138     System.out.println(" =====");
139     System.out.println(" | _ _");
140     System.out.println(" | / \");
141     System.out.println(" | |");
142     System.out.println(" | \ _ /");
143     System.out.println(" |");
144     System.out.println(" |");
145     System.out.println(" | / \");

```

Fig-10.4


```
C:\Users\SMARIAN\Desktop\hangman - Sublime Text (UNREGISTERED)
File Edit Selection Find View Goto Tools Project Preferences Help

index.html x hangman x
108 System.out.println("
109 System.out.println("
110 System.out.println("
111 System.out.println("
112 System.out.println("
113 System.out.println("
114 System.out.println("
115 System.out.println("
116 System.out.println("
117 System.out.println("
118 System.out.println("===|===");
119 }
120 if (count == 6) {
121     System.out.println("Wrong guess, try again");
122     System.out.println("=====");
123     System.out.println("
124     System.out.println("
125     System.out.println("
126     System.out.println("
127     System.out.println("
128     System.out.println("
129     System.out.println("
130     System.out.println("
131     System.out.println("
132     System.out.println("
133     System.out.println("===|===");
134 }
135 if (count == 7) {
136     System.out.println("GAME OVER!");
137     System.out.println("=====");
138     System.out.println("
139     System.out.println("
140     System.out.println("
141     System.out.println("
142     System.out.println("
143     System.out.println("
144     System.out.println("
145     System.out.println("

Line 111, Column 55 Spaces: 4 Plain Text
```

Fig-10.5

References

1. www.google.com
2. www.youtube.com
3. <https://www.java.com/en/>
4. <https://docs.oracle.com/en/java/>
5. www.slideshare.net.in
6. [https://en.wikipedia.org/wiki/Hangman_\(game\)](https://en.wikipedia.org/wiki/Hangman_(game))
7. <https://www.hangmanwords.com/play>
8. <https://www.coolmathgames.com/0-hangman>
9. <https://www.cse.iitb.ac.in/~anwsha/cs699/mpStage1.html>
10. https://teals-introcs.gitbooks.io/introduction-to-computer-science-principles/content/project_4.html
11. <https://gist.github.com/saroj22322/aa2f0849f33736395544c2d341ab3722>
12. www.github.io.in
13. <https://docs.oracle.com/en/java/javase/15/>
14. <https://docs.oracle.com/javame/8.3/index.html>
15. <https://medium.com/swlh/creating-a-hangman-game-in-java-2c3088cb0d6d>
16. <https://stackoverflow.com/questions/4988143/simple-java-hangman-assignment>
17. <https://www.clear.rice.edu/comp202/07-fall/assignments/hangman/>
18. <https://www.clear.rice.edu/comp202/07-fall/assignments/hangman/mvc.shtml>
19. <https://cnx.org/contents/17cde6f7-ad54-4c69-a00f-04b6f02685cb@3>
20. <https://rskoura.wordpress.com/70-2/>
21. <https://www.clear.rice.edu/comp202/07-fall/assignments/hangman/wordlist.shtml>
22. <https://www.clear.rice.edu/comp202/07-fall/assignments/hangman/bodyparts.shtml>
23. <https://www.clear.rice.edu/comp202/07-fall/assignments/hangman/docs/>
24. <https://www.clear.rice.edu/comp202/07-fall/assignments/hangman/view.shtml>
25. <https://www.clear.rice.edu/comp202/07-fall/assignments/hangman/bodyparts.shtml#checklist>
26. <https://www.clear.rice.edu/comp212/07-spring/projects/hangman/>
27. <https://rskoura.wordpress.com/about/>
28. <https://rskoura.wordpress.com/call-by-value-call-by-reference-explained/>
29. <https://rskoura.wordpress.com/icse-java-definitions/>
30. <https://rskoura.wordpress.com/icse-class-10-java-theory/>
31. https://teals-introcs.gitbooks.io/introduction-to-computer-science-principles/content/project_4.html
32. <https://www.xspdf.com/resolution/53670936.html>
33. <https://codereview.stackexchange.com/questions/171369/text-based-hangman-game-in-java>
34. www.geeksforgeeks.org
35. <https://www.geeksforgeeks.org/java-program-for-word-guessing-game/>
36. <https://codereview.stackexchange.com/tags>
37. <https://www.rd.com/article/hardest-word-guess-hangman/>
38. <https://www.rd.com/article/most-complicated-word-in-english/>
39. <https://www.rd.com/list/fake-words-in-dictionary/>
40. <https://www.rd.com/article/words-that-arent-words/>
41. <https://www.lexico.com/>
42. <https://www.rd.com/list/words-added-to-the-dictionary/>
43. <https://www.lexico.com/definition/set>

44. <https://www.irishtimes.com/culture/deadline-2037-the-making-of-the-next-oxford-english-dictionary-1.1667328>
45. <https://www.dreamincode.net/forums/topic/95521-word-guessing-game/>
46. <https://www.rd.com/list/missing-word-rat-puzzles/>
47. <https://www.dictionary.com/>
48. <https://dictionary.cambridge.org/>
49. <https://www.oxfordlearnersdictionaries.com/>
50. <https://languages.oup.com/>