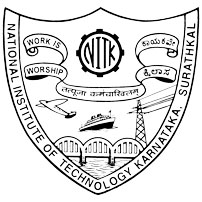
**NATIONAL INSTITIUE OF TECHNOLOGY, SURATHKAL**

****

**DE PARTMENT OF INFORMATION TECHNOLOGY**

**Paradigms of Programming-I**

**(IT206)**

**ChatBot With Web Scraping**

**Project Report**

**Submitted to:**

**Miss. R. Priyadarshini**

**Submitted by:**

**Ram Aditya .S. (16IT131)**

**Anumeha Agrawal (16IT107)**

**Rosa Anil George (16IT134)**

**1. SYNOPSIS**

This project was undertaken as part of the fulfillment of requirements of the course, **IT206 – Paradigms of Programming** and aims to create a simple ChatBot that has been named POP Bot by using the concepts of Inheritance, Data Abstraction, Interfaces, Exception Handling, Web Scraping and string matching algorithms.

The application has been designed to facilitate interaction between the user and the ChatBot. The ChatBot designer defines the ChatBot personality, the questions that will be asked to the users, and the overall interaction. It can be viewed as a subset of the conventional design. An important part of the ChatBot design is also centered around user testing. User testing can be performed following the same principles that guide the user testing of graphical interfaces.

This project aims to detail the methods and practices undertaken to complete this project, as well as serve as a record of our work and possible motivation for extension of this work into future projects.

**2. INTRODUCTION**

In today’s connected world, ChatBot applications are inescapable. ChatBots have become a source of a wide variety of information. ChatBots can schedule meetings, can tell you the weather, provide movie reviews as well as provide customer support.

Our project is an example of a simple ChatBot – POP bot. POP bot is a ChatBot with which one can have common day to day conversations with, it will listen to everything you say, it can provide you with movie reviews, it will also show you the corresponding image of any word you enter and make you laugh by saying a funny joke. POP bot is a Java based project based on inheritance and web scraping. This project creates a ChatBot that gives apt responses to the various queries of the users.

The process of building a ChatBot can be divided into two main tasks: understanding the user's intent and producing the correct answer. The first task involves understanding the user input. The second task may involve different approaches depending on the type of the response that the ChatBot will generate by checking the keyword ranking of the word in the knowledge base. Then giving the correct corresponding response of the matched keyword instantaneously.

**3. PROBLEM STATEMENT**

To create a ChatBot application using concepts of Inheritance, Data Abstraction, Interfaces, Exception Handling, Web Scraping and string matching algorithms which is capable of the following:

* Cleaning the string which is taken as an input from the user and ranking the important keywords.
* Giving an accurate response to the query by matching the keyword to the queries in the knowledge base.
* Generating the movie review for the queried movie or recommended movie by extracting the required information from the movie database.
* Checking if a word exists in the English dictionary by searching for it in the list of word in the dictionary.
* Displaying the required images of the person or thing that is queried about using the concept of Web Scraping.
* Displaying information about anything by scraping content from Wikipedia.

**4. OBJECTIVE**

The primary objective of this project is to build a ChatBot application that satisfies all the conditions mentioned in the problem statement by employing the concepts of Object Oriented Programming in Java. Furthermore, it is also our aim to apply all the concepts that have been taught in the theory classes and gain further knowledge and experience by doing so. This must serve as foundation for future works in building applications using Object Oriented Programming in languages other than Java as well.

**5. METHODOLOGY**

The complete development of the project was done systematically in different stages. Each stage was debugged and tested before moving on to the next one. Each functionality and its related data was encapsulated in a separate class. This enabled smooth workflow and distribution of work among the project members.

5.1 Class-Wise division of work

The following is the list of essential classes and the methodology involved in constructing each.

5.1.1 ChatKnow

This contains a 3-D array which arranges the various possible user inputs and the possible responses that the bot must pick. This data was constantly updated to enable better input-response match and adding possible user-inputs for requests for web scraping, jokes and movie details. For normal conversation, the response picked is a normal human-like response. For special features like jokes and web scraping, the response is a special flag value that the main program detects to execute the necessary routine.

5.1.2 ChatBot

This class is the main class which contains the *main()* method. It also contains methods that are responsible for receiving the text input from the user and producing an apt response. It contains a HashTable that stores the previous user-inputs and the responses given to that input.

* First, the user input string is cleaned. This involves removing all extra spaces, punctuation marks, etc to make it fit for using as a key to search in the Knowledge-Base.
* Next, the key which is the cleaned user-input string is used to search in the Knowledge-Base. There are two cases while searching for a key.

Case 1) :

If a perfect match is found, then a random response from the given list of possible responses for that key which is also given in the Knowledge-Base,is chosen. If the user-input/question has already been asked and if the currently selected response for that particular input already exists in the HashTable then this step is repeated once again to pick a new random response. Else, the response is picked as the correct response and finally added to the HashTable.

Case 2) :

If a perfect match is not found, then a keyword matching algorithm is used. This algorithm matches the key with the one in the Knowledge-Base which has the maximum number of words (as substrings) that are matching with the user input.

After this matching is done, the response is chosen in a similar fashion as explained above.

5.1.3 Dictionary

This class is used to search a dictionary database and find whether a certain word exists or not. A dictionary text file is used which contains all the words in the English language. This is added into an ArrayList. The key is first received from the main program and the existence of this key is checked in the ArrayList. The Boolean value corresponding to its existence is returned to the user.

5.1.4 GetFunData

The methods and data in this class are used to implement the features of retrieving movie details and jokes upon user request. There are two CSV files movieDB.csv and jokes.csv, which contain a dataset of movies and a list of over a thousand jokes respectively. These are parsed and loaded into a 2-D array. When the main program calls an object of this class by passing the key through the constructor and specifying whether it is a movie detail or joke that is needed, the appropriate database is scanned and the result is sent back.

5.1.5 Wiki

This class contains the code required to scrape data from the Wikipedia pages for the given user input. The java.net package is made use of for implementing the features. This class contains methods that first receive a keyword that is to be searched on the internet. A HTTP request is sent using the Wikipedia URL to fetch contents for the given keyword. The entire html source file is retrieved and stored in a text file. An ExecuteShellCommand object is created for further processing.

5.1.6 WikiImages

This class encapsulates the data and methods required for searching and opening the images of a given key on the Mozilla Firefox Web Browser. The key or image to be searched is received through the constructor from the main program. The website [www.dreamstime.com](http://www.dreamstime.com) is used for searching for the images. Using the Jsoup package commands an URL with the given key attached is opened and the content fetched is further processed to extract only the image content. The URL of each image is then passed on to the ExecuteShellCommand object which then uses UNIX commands to open the given URL on the Web Browser.

5.1.7 ExecuteShellCommand

This class mainly utilizes the ProcessBuilder class defined in java.lang package, to execute some OS routines. It is used by the Wiki and WikiImages classes.

* It contains a small bash script that was written to filter the Wikipedia content that is passed on to this class by the Wiki class. First, all content within the anchor brackets is removed. Next, all kind of special characters are removed which are part of the main HTML code and do not correspond to the actual content displayed on the Wikipedia page. Then this filtered content is scanned to finally extract a paragraph which contains the information about the key being searched for.
* It is also used to open the image URLs passed on by the WikiImages class on the Mozilla Browser by using the xdg-open package commands.

5.2 Employing OOPs Concepts

A few instances of how OOPs concepts were used is as follows:

* ChatKnow mainly contains data that should be globally available for all the classes. Thus the data it contains, 3-D array which contains the Knowledge-Base and flag values corresponding to whether user wants movie details or a joke or Wikipedia content, etc. are all declared as static variables.
* At the beginning of the project it was not yet completely clear what the functionalities of the ChatBot would be and the implementation for cleaning the string and matching it. Thus only an interface was created for the class ChatBot which was then implemented by the class later on. Also, all methods in this class needed to be static as this class was not required to have any objects.
* Since the Dictionary class could not have a possible extension it was declared as final.
* The ExecuteShellCommand contains methods that do similar tasks but are not exactly the same. For example, the executecommand() method is used for both executing bash scripts and also to open URLs on the web browser. Thus this method was overloaded, so that the same method name is invoked to perform similar tasks.
* An abstract parent class was first written for web scraping. This was extended and implemented by the Wiki and WikiImage classes, both of which use the general methods of web scraping but in their own specialized ways.
* Exception Handling was necessary especially with ProcessBuilder which is used to execute OS routines and with getFunData and Dictionary as access to files was required. The first class can lead to errors if there are system errors like SystemException and in the second there can be exceptions while accessing files like FileNotFoundException. Thus Java`s exception handling features were used to handle such exceptions.
* ‘finalize’ method was used so that when the program terminates and system`s garbage collector destroys the objects, the system resources like the CSV files are freed.

**6. SYSTEM REQUIREMENTS:**

This ChatBot application will work on any system that contains the following software/packages.

* Java SE 6 or higher.
* Ubuntu 16.04 or higher.
* Jsoup 1.11.2 or higher.
* java.net Java package.
* Mozilla Firefox Web Browser version-22 or higher.
* xdg-open package for LINUX system.
* Internet connectivity with reasonable speed.

**7. WORKS DONE**

We have built a fully functional ChatBot with certain distinct features. The functionalities of the chat bot include dictionary checker, movie details, jokes, data scraping from Wikipedia and image scraping. The ChatBot is capable of making a general conversion with the user.

The user can ask the bot any number of general questions like how the bot is, the time, its name etc . This information is stored in a Knowledge-Base and keyword matching is used to generate answers based on prediction from data already present in the Knowledge-Base. The conversation ends when the user says bye and the ChatBot terminates.

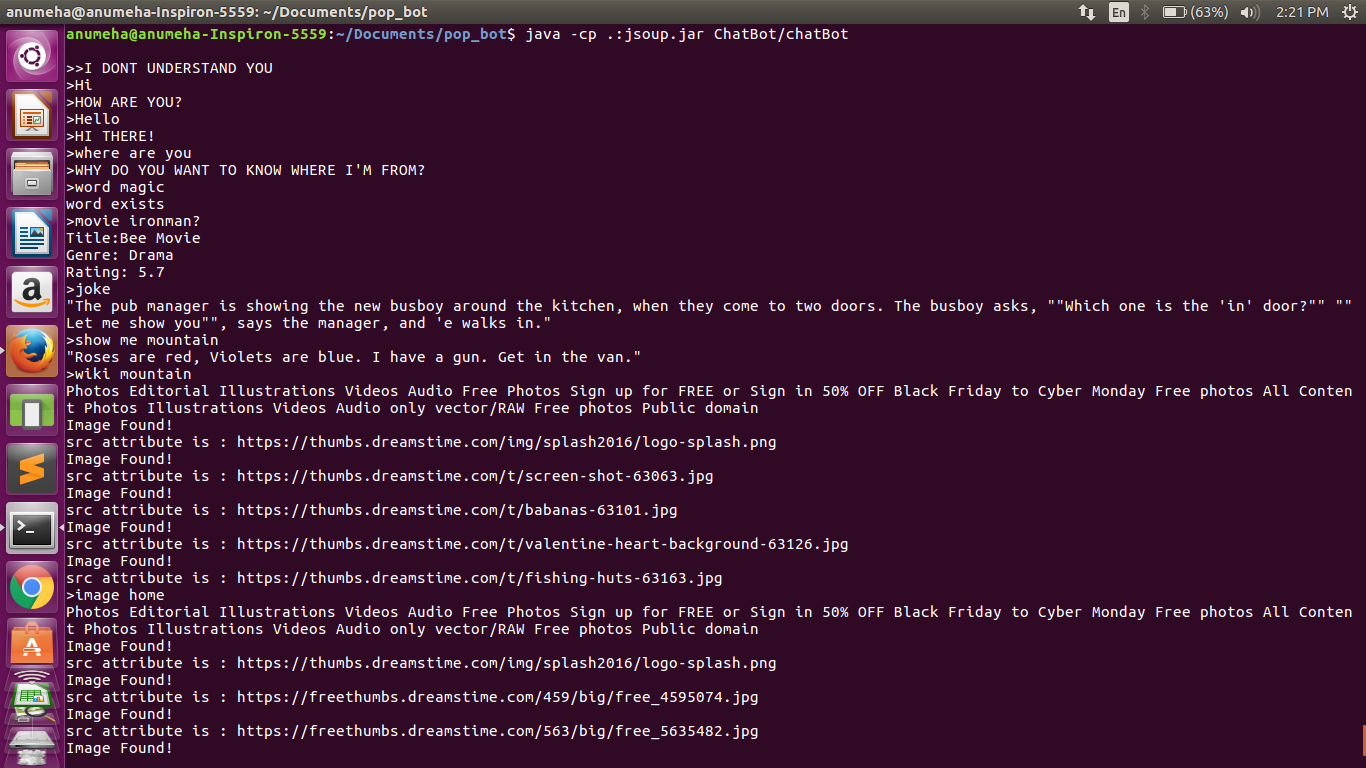


Figure 7.1

7.1Special Features

7.1.1 Dictionary Checker

This feature of the bot lets the user confirm the existence of a word in the dictionary. We have used a text file with a list of dictionary words. The user needs to enter the word and if the word exists the bot says word exists else it says that the word doesn`t exist.

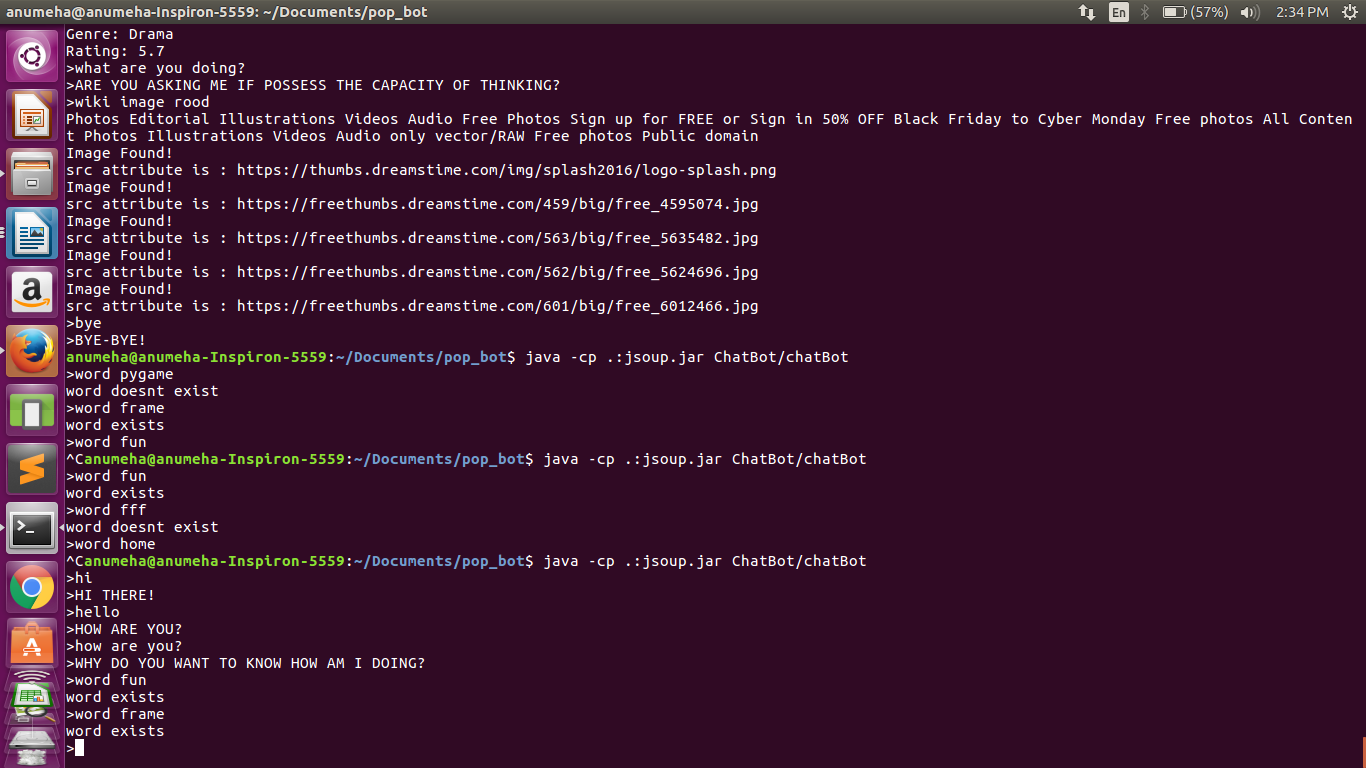


Figure 7. 2

7.1.2 Movie Details

This feature of the bot helps the user know the genre, title and rating of any movie. This data is fetched from a CSV file on the IMDB Database(movieDB.csv). The user needs to enter the movie name and the details will be displayed.

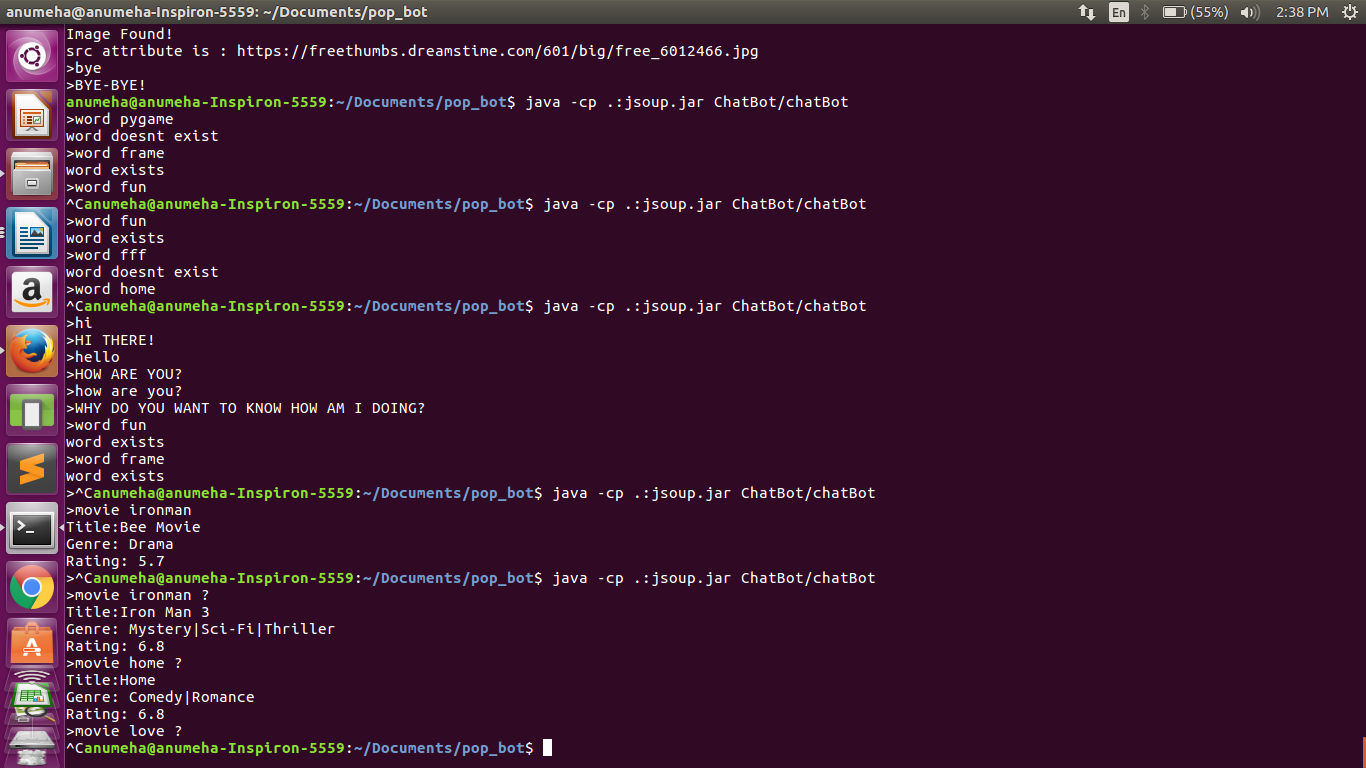


Figure 7.3

7.1.3 Joke Generator

This feature of the bot adds some humor. It generates jokes randomly when the user asks for something funny or something to laugh at. The data is present in a CSV file which is being parsed and a random joke is selected and displayed. Each time the user asks for a joke the bot displays a new joke for the user.

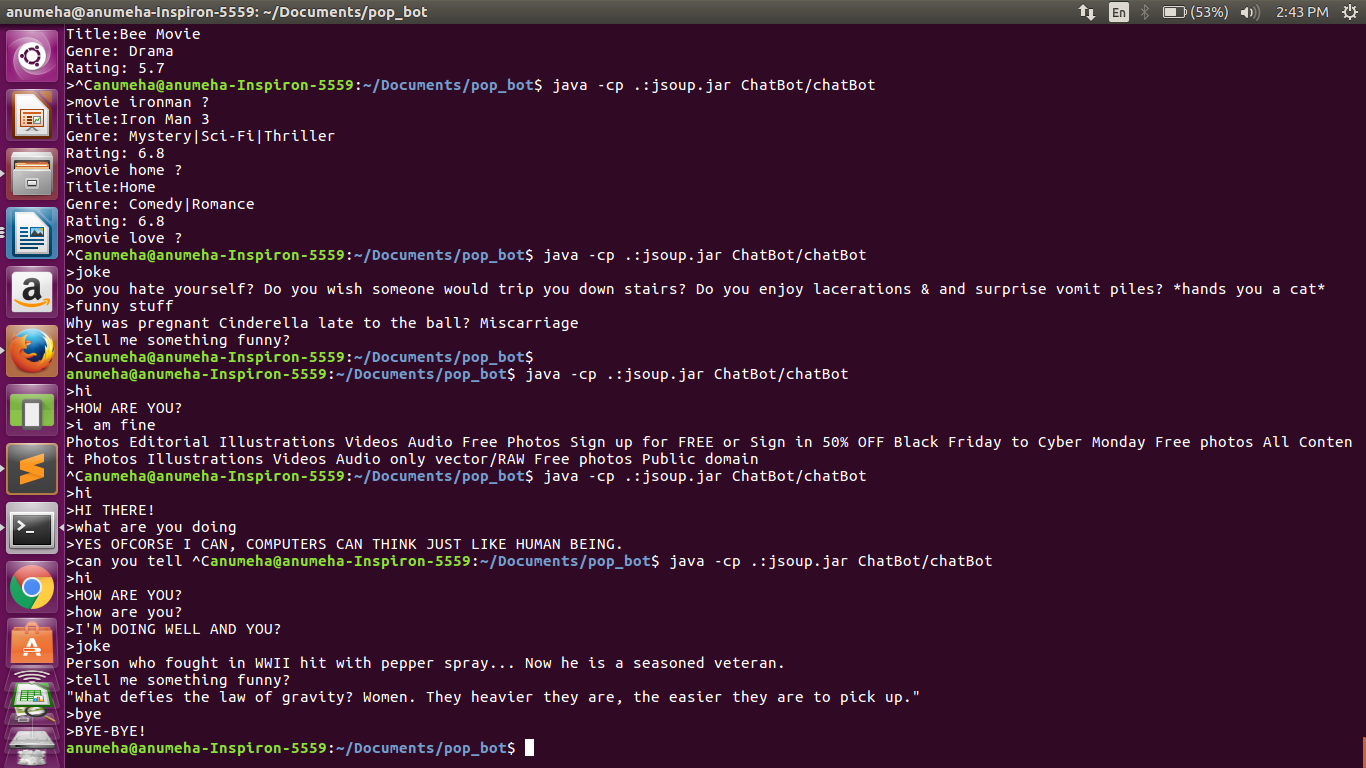


Figure 7.4

7.1.4 Image Scraping

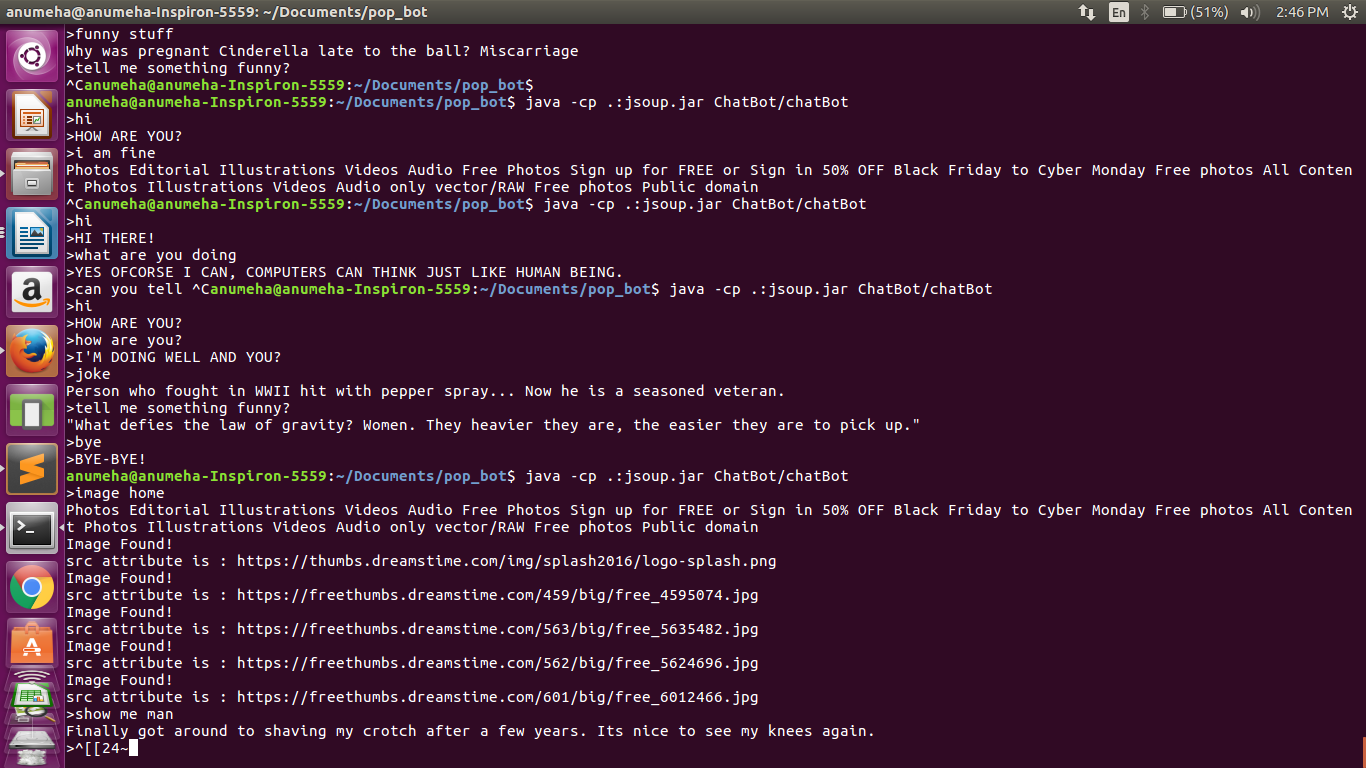
This feature of the bot gives a few links of the image the user requests for. It also opens up those links on the Web Browser and the user can directly get those pictures. This feature is highly useful when the user wants number of links for a particular image. It also saves time. We have used web scraping (using jsoup) to implement this feature.

Figure 7.5

**8. FUTURE WORKS**

The current version of this application can be well-extended to add several useful functionalities. The following are the broad areas which have greater scope for enhancement:

8.1 Keyword Match

The current version finds the best match for a given user input by implementing a keyword match algorithm that finds the response in the Knowledge-Base which has the maximum number of words ( as substrings) that are matching with the user input. The other possibility is much more complex which uses the concept of **Fuzzy String Search**. To apply this method, the input and the current keyword is broken into separate words, after which two different vectors are used. The first one is used to store the words for the input and the other one would store the words for the current keyword. The **Levenshtein distance** is then used for measuring the distance between the two, word vectors. The **Levenshtein distance** is a measure of the similarity between two strings, the source string (s) and the target string (t). The distance is the number of deletions, insertions, or substitutions required to transform s into t.

8.2 Web Scraping

The web scraping tool of the ChatBot can be further enhanced. The current version only supports scraping content from websites such as Wikipedia which allow bots to access the content. However, other websites like Amazon, YouTube, Flipkart, Zomato can also be scraped by using proxy servers. This can for example, be used for fetching relevant information when user request details about prices of items on popular shopping websites, suggestions for places to have dinner, etc.

8.3 Making the conversation more interesting

To keep the user interested some features like providing movie details such as ratings,genre providing humorous content and definitions for words were added by including a movie, jokes and dictionary database which the bot would access to fetch content. This can be extended to playing audio files from the system, providing fun facts and creating small quizzes for the user.

8.4 Multi-Threading to enable faster querying

Many operations like scanning through the movie database to fetch movie details and scarping content from Wikipedia pages is currently handled in a sequential manner, i.e., by only the single thread. There is lot of scope to multi-thread the search as each thread can be made to search different portions of the database which would speed up the process by a good margin. This would enable smoother conversations with minimum delay.

**9. RESULT AND DISCUSSION**

The final result is a fully functional ChatBot . We have included some important jsoup features in our project. First we use Jsoup.connect with timeout parameter which increases the effectiveness of parser since some websites can be quite slow. We also use ignoreContentType since it helps ignoring false positives for URLs.

After that we extracts all the href links from a website. We have also used a shell script which uses sed and grep for the data scraping. Thus our scraping is hardcoded in a way as we have implemented out own function for selecting specific tags.

Thus our final project looks like this :-

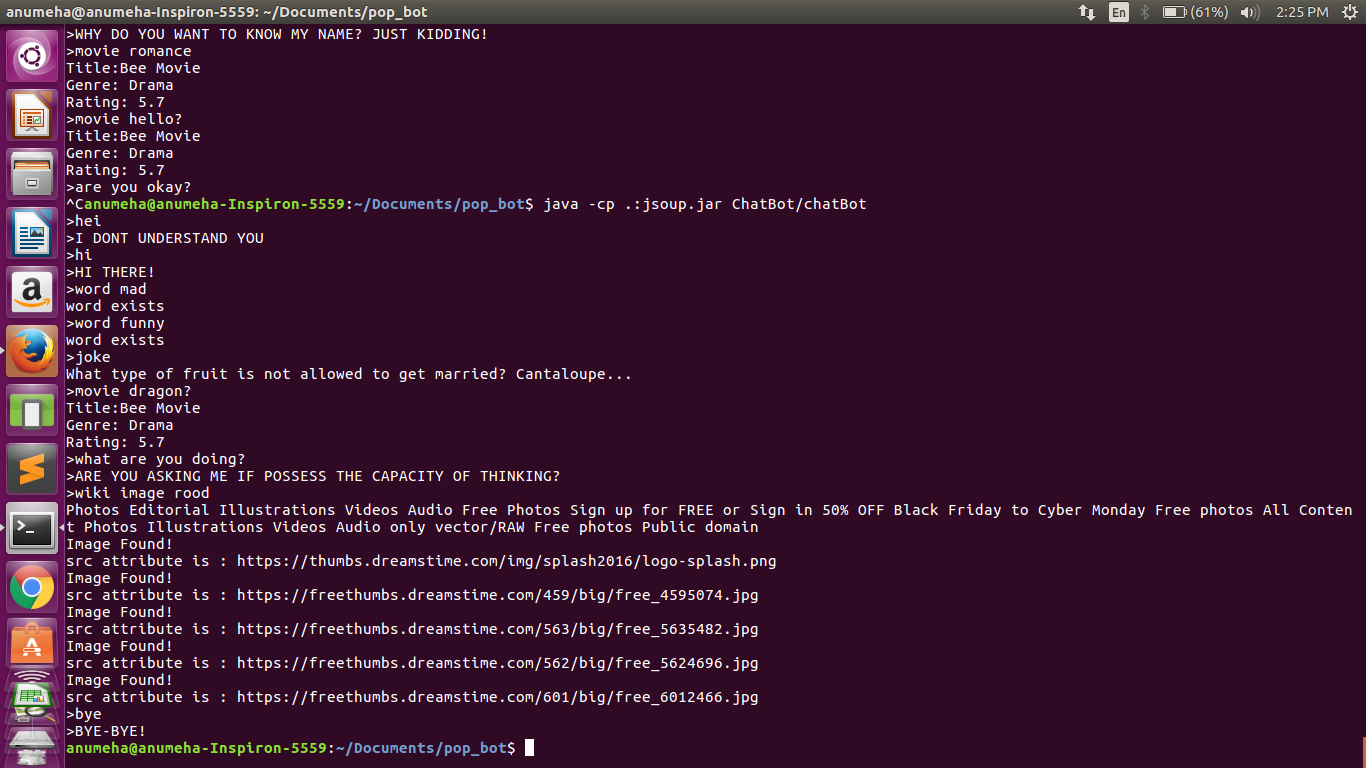


Figure 9.1

**10. CONCLUSION**

We have achieved the goals of this project which we had taken up as a part of Paradigms of Programming course. We have successfully inculcated the object oriented programming concepts which we had learnt under this course. OOPs has helped us modularize our code and thus achieve the end target.

**11. REFERENCES**

**11.1 Web links:**

1. <https://www.tutorialspoint.com/java/java_inheritance.htm>

2. <https://www.tutorialspoint.com/java/java_abstraction.htm>

3. <https://ksah.in/introduction-to-web-scraping-with-java/>

4. <https://www.tutorialspoint.com/java/java_data_structures.htm>

5. <http://tutorials.jenkov.com/java/interfaces.html>

**11.2 Printed Material:**

1. Thinking in JAVA, Bruce Eckel, Pearson Education

2. Computer Science with Java , Sumita Arora

**12. APPENDIX (Code)**

**12.1 To execute shell commands (Class:ExecuteShellCommand)**

String key="";

public ExecuteShellCommand(String key)

{

this.key=key;

}

public void executeCommands() throws IOException, InterruptedException {

File tempScript = createTempScript();

try {

ProcessBuilder pb = new ProcessBuilder("bash", tempScript.toString(),key);

pb.inheritIO();

Process process = pb.start();

process.waitFor();

} finally {

tempScript.delete();

}

}

public String executeCommand(String command)

{

{

StringBuffer output = new StringBuffer();

Process p;

try {

p = Runtime.getRuntime().exec(command);

p.waitFor();

BufferedReader reader =

new BufferedReader(new InputStreamReader(p.getInputStream()));

String line = "";

while ((line = reader.readLine())!= null) {

output.append(line + "\n");

}

} catch (Exception e) {

e.printStackTrace();

}

return output.toString();

}

}

public File createTempScript() throws IOException {

File tempScript = File.createTempFile("script", null);

Writer streamWriter = new OutputStreamWriter(new FileOutputStream(

tempScript));

PrintWriter printWriter = new PrintWriter(streamWriter);

printWriter.println("cp /home/ram\_aditya/eclipse-workspace/ChatBot/src/ChatBot/new /home/ram\_aditya/eclipse-workspace/ChatBot/temp");

printWriter.println("sed 's/<[^>]\\+>/ /g' temp > new1");

printWriter.println("grep -i $1.\* new1 | sed '/{/d' - |tr -s '\t ' | sed -n -f instr | head -n4 > new2");

printWriter.println("sed -i '/disambiguation/d' new2");

printWriter.println("sed -i -f instr2 new2");

printWriter.println("cat new2");

printWriter.close();

return tempScript;

}

**12.2 Web scraping Wikipedia (Class:Wiki)**

public void search(String key) throws InterruptedException {

URL url;

try {

// get URL content

String a="https://en.wikipedia.org/wiki/"+key;

url = new URL(a);

URLConnection conn = url.openConnection();

// open the stream and put it into BufferedReader

BufferedReader br = new BufferedReader(

new InputStreamReader(conn.getInputStream()));

Path dest=Paths.get("src/ChatBot/new");

String inputLine;

BufferedWriter writer=Files.newBufferedWriter(dest);

while ((inputLine = br.readLine()) != null) {

writer.write(inputLine);

writer.newLine();

}

br.close();

ExecuteShellCommand ex=new ExecuteShellCommand(key);

ex.executeCommands();

} catch (MalformedURLException e) {

e.printStackTrace();

} catch (IOException e) {

e.printStackTrace();

}

}

**12.3 To obtain images (Class:WikiImages)**

String word="";

public WikiImage(String word)

{

this.word=word;

}

public void imagesearch() throws Exception

{

String url="https://www.dreamstime.com/free-photos-images/"+word+".html";

Document document = Jsoup.connect(url).get();

String text = document.select("div").first().text();

System.out.println(text);

Elements img = document.getElementsByTag("img");

int count=0;

for (Element el : img) {

count++;

if(count>5) break;

//for each element get the src url

String src = el.absUrl("src");

System.out.println("Image Found!");

System.out.println("src attribute is : "+src);

}

}

**12.4 Checking Dictionary(Class:Dictionary)**

static ArrayList<String> words;

public static void main(String args[])

{

Scanner in = new Scanner(System.in);

System.out.println("Enter the string");

String s = in.nextLine();

Dictionary obj = new Dictionary();

s=s.toLowerCase();

if(obj.wordExists(s) == true)

{

System.out.println(message1);

}

else

System.out.println(message2);

}

public Dictionary()

{

words = new ArrayList<String>();

readInDictionaryWords();

}

public void readInDictionaryWords()

{

File dictionaryFile = new File("words.txt");

if( ! dictionaryFile.exists()) {

System.out.println("\*\*\* Error \*\*\* \n" +

"Your dictionary file has the wrong name or is " +

"in the wrong directory. \n" +

"Aborting program...\n\n");

System.exit( -1); // Terminate the program

}

try

{

Scanner inputFile = new Scanner( dictionaryFile);

while( inputFile.hasNext()) {

String line =inputFile.nextLine();

line = line.trim();

words.add(line.toLowerCase() );

}

}

catch(FileNotFoundException e)

{

System.out.println(e);

}

}

public boolean wordExists(String wordToLookup)

{

if( words.contains( wordToLookup)) {

return true; // words was found in dictionary

}

else {

return false; // word was not found in dictionary

}

}

**12.5 Fetching movie details and jokes(Class:GetFunData)**

private List<movieData> movieFile=new ArrayList<movieData>();

private List<String> jokeList= new ArrayList<String>();

public getFunData() throws IOException

{

Path movies=Paths.get("movieDB.csv");

BufferedReader br= Files.newBufferedReader(movies);

String li=br.readLine();

while(li!=null)

{

String[] columnData=li.split(",");

movieData temp=new movieData(columnData);

movieFile.add(temp);

li=br.readLine();

}

Path jokes=Paths.get("jokes.csv");

br= Files.newBufferedReader(jokes);

li=br.readLine();

while(li!=null)

{

jokeList.add(li);

li=br.readLine();

}

}

public String search(String key)

{

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

if(key.equalsIgnoreCase(movieFile.get(i).Title))

return movieFile.get(i).toString();

else if(movieFile.get(i).Title.toLowerCase().replaceAll(" ","").contains(key.toLowerCase().replaceAll(" ", "")))

return movieFile.get(i).toString();

return "That is not a popular movie.Sorry!";

}

public String getJoke()

{

Random rand=new Random();

int i=rand.nextInt(850);

return jokeList.get(i);

}

public static void main(String[] args) throws IOException

{

Scanner in=new Scanner(System.in);

String input=in.nextLine();

getFunData ob=new getFunData();

System.out.println(ob.search(input));

in.close();

}

}

class movieData {

public String Title;

public String Revenue;

public String Genre;

public String Rating;

public movieData(String[] columnData)

{

Revenue=columnData[0];

Title=columnData[1];

Genre=columnData[3];

Rating=columnData[2];

}

@Override

public String toString()

{

return "Title:"+Title+"\nGenre: "+Genre+"\nRating: "+Rating;

}

**12.6 Trimmed version Knowledge Base of the ChatBot (Class:ChatKnow)**

static String MOVIE="1";

static String JOKE="2";

static String IMAGEL="3";

static String INFO="4";

static String WIKIIM="5";

static String DICT="6";

public static String[][][] Knowledge-Base = {

{ {"WHAT IS YOUR NAME"},

{"MY NAME IS POP-BOt!",

"YOU CAN CALL ME POP-BOT!",

"WHY DO YOU WANT TO KNOW MY NAME? JUST KIDDING!"}

},

{ {"WHICH MOVIE IS","DO YOU KNOW WHICH MOVIE","MOVIE","FILM","FAVOURTIE FILM"},

{MOVIE}

},

{

{"WIKIIMAGELINK","WIKI IMAGE","WIKI"},

{WIKIIM}

},

{

{"EXIST WORD","WORD-","WORD EXISTS"},

{DICT}

},

{

{"IMAGES","IMAGE","I WANT TO SEE","HAVE YOU SEEN","SEEN","PICTURE","PICTURES","PHOTOS"},

{IMAGEL}

},

{ {"A JOKE","SOMETHING FUNNY","MAKE ME LAUGH","I WANT TO LAUGH"},

{JOKE}

},

{ {"WHO IS THE","WHAT IS THE","DO YOU KNOW ANYTHING ABOUT","TELL SOMETHING ABOUT","INFORMATION ON"},

{INFO}

},

{{"HI", "HELLO"},

{"HI THERE!",

"HOW ARE YOU?",

"HI!"}

},

{{"I"},

{"SO, YOU ARE TALKING ABOUT YOURSELF",

"SO, THIS IS ALL ABOUT YOU?",

"TELL ME MORE ABOUT YOURSELF."},

},

{{"I WANT"},

{"WHY DO YOU WANT IT?",

"IS THERE ANY REASON WHY YOU WANT THIS?",

"IS THIS A WISH?",

"WHAT ELSE YOU WANT?"}

},

{{"I HATE"},

{"WHY DO YOU HATE IT?",

"WHY DO YOU HATE\*?",

"THERE MUST A GOOD REASON FOR YOU TO HATE IT.",

"HATERED IS NOT A GOOD THING BUT IT COULD BE JUSTIFIED WHEN IT IS SOMETHING BAD."}

},

{{"I LOVE CHATING"},

{"GOOD, ME TOO!",

"DO YOU CHAT ONLINE WITH OTHER PEOPLE?",

"FOR HOW LONG HAVE YOU BEEN CHATING?",

"WHAT IS YOUR FAVORITE CHATING WEBSITE?"}

},

{{"I MEAN"},

{"SO, YOU MEAN\*.",

"SO, THAT'S WHAT YOU MEAN.",

"I THINK THAT I DIDN'T CATCH IT THE FIRST TIME.",

"OH, I DIDN'T KNOW IT MEANT THAT."}

},

{{"I DIDN'T MEAN"},

{"OK, YOU DIDN'T MEAN\*.",

"OK, WHAT DID YOU MEAN THEN?",

"SO I GUESS THAT I MISSUNDESTOOD."}

},

{{"I GUESS"},

{"SO YOU ARE MAKING A GUESS.",

"AREN'T YOU SURE?",

"ARE YOU GOOD A GUESSING?",

"I CAN'T TELL IF IT IS A GOOD GUESS."}

},

{{"I'M DOING FINE", "I'M DOING OK"},

{"I'M GLAD TO HEAR IT!",

"SO, YOU ARE IN GOOD SHAPE."}

},

{{"CAN YOU THINK OF"},

{"YOU MEAN LIKE IMAGINING SOMETHING.",

"I DON'T KNOW IF CAN DO THAT.",

"WHY DO YOU WANT ME THINK OF IT?"}

},

{{"HOW ARE YOU", "HOW DO YOU DO"},

{"I'M DOING FINE!",

"I'M DOING WELL AND YOU?",

"WHY DO YOU WANT TO KNOW HOW AM I DOING?"}

},

{{"WHO ARE YOU"},

{"I'M AN A.I PROGRAM.",

"I THINK THAT YOU KNOW WHO I'M.",

"WHY ARE YOU ASKING?"}

},

{{"ARE YOU INTELLIGENT"},

{"YES,OFCOURSE.",

"WHAT DO YOU THINK?",

"ACTUALY,I'M VERY INTELLIGENT!"}

},

{{"ARE YOU REAL"},

{"DOES THAT QUESTION REALLY MATERS TO YOU?",

"WHAT DO YOU MEAN BY THAT?",

"I'M AS REAL AS I CAN BE."}

},

{{"MY NAME IS", "YOU CAN CALL ME"},

{"SO, THAT'S YOUR NAME.",

"THANKS FOR TELLING ME YOUR NAME USER!",

"WHO GIVE YOU THAT NAME?"}

},

{{"BYE", "GOODBYE"},

{"IT WAS NICE TALKING TO YOU USER, SEE YOU NEXT

"BYE USER!",

"OK, BYE!"}

},

{{"OK"},

{"DOES THAT MEAN THAT YOU ARE AGREE WITH ME?",

"SO YOU UNDERSTAND WHAT I'M SAYING.",

"OK THEN."},

},

{{"OK THEN"},

{"ANYTHING ELSE YOU WISH TO ADD?",

"IS THAT ALL YOU HAVE TO SAY?",

"SO, YOU AGREE WITH ME?"}

},

{{"ARE YOU A HUMAN BEING"},

{"WHY DO YOU WANT TO KNOW?",

"IS THIS REALLY RELEVENT?"}

},

{{"YOU ARE VERY INTELLIGENT"},

{"THANKS FOR THE COMPLIMENT USER, I THINK THAT YOU ARE INTELLIGENT TO!",

"YOU ARE A VERY GENTLE PERSON!",

"SO, YOU THINK THAT I'M INTELLIGENT."}

},

{{"YOU ARE WRONG"},

{"WHY ARE YOU SAYING THAT I'M WRONG?",

"IMPOSSIBLE, COMPUTERS CAN NOT MAKE MISTAKES.",

"WRONG ABOUT WHAT?"}

},

{{"ARE YOU SURE"},

{"OFCORSE I'M.",

"IS THAT MEAN THAT YOU ARE NOT CONVINCED?",

"YES,OFCORSE!"}

},

{{"WHO IS"},

{"I DON'T THINK I KNOW WHO.",

"I DON'T THINK I KNOW WHO\*.",

"DID YOU ASK SOMEONE ELSE ABOUT IT?",

"WOULD THAT CHANGE ANYTHING AT ALL IF I TOLD YOU WHO."}

},

{{"WHAT"},

{"SHOULD I KNOW WHAT\*?",

"I DON'T KNOW WHAT\*.",

"I DON'T KNOW.",

"I DON'T THINK I KNOW.",

"I HAVE NO IDEA."}

},

**12.7 Keyword Matching Algorithm (Class:ChatBot)**

static String find\_match(String str)

{

static HashMap<Integer,ArrayList<Integer>> counter=new HashMap<Integer,ArrayList<Integer>>();

int matchLength=-1;

String resp="";

String finalResp="";

for(int j=0;j<ChatKnow.Knowledge-Base.length;j++)

for(int t=0;t<ChatKnow.Knowledge-Base[j][0].length;t++)

{

int test=keywordMatch(ChatKnow.Knowledge-Base[j][0][t],str);

if(ChatKnow.Knowledge-Base[j][0][t].equalsIgnoreCase(str) || test!=-1)

{

Random rand=new Random();

int i=rand.nextInt(ChatKnow.Knowledge-Base[j][1].length); if(!counter.containsKey(j)){

ArrayList<Integer> count=new ArrayList<>();

if(ChatKnow.Knowledge-Base[j][0][t].equalsIgnoreCase(str))

count.add(i); counter.put(j,count);

resp=ChatKnow.Knowledge-Base[j][1][i];

}

else

{

ArrayList<Integer> temp=new ArrayList<>();

temp=counter.get(j);

if(! temp.contains(i))

{

if(ChatKnow.Knowledge-Base[j][0][t].equalsIgnoreCase(str))

{

temp.add(i);

counter.put(j,temp);

}

resp=ChatKnow.Knowledge-Base[j][1][i];

}

else

{

int newResp;

if(temp.size()!=ChatKnow.Knowledge-Base[j][1].length)

{

while(true)

{

newResp =rand.nextInt(ChatKnow.Knowledge-Base[j][1].length);

if(newResp!=i){

resp=ChatKnow.Knowledge-Base[j][1][newResp];break;}

}

}

else

{

newResp=rand.nextInt(ChatKnow.Knowledge-Base[j][1].length);

resp=ChatKnow.Knowledge-Base[j][1][newResp];

}

if(ChatKnow.Knowledge-Base[j][0][t].equalsIgnoreCase(str))

{

temp.add(newResp);

counter.put(j,temp);

}

}

}

}

if(ChatKnow.Knowledge-Base[j][0][t].equalsIgnoreCase(str) )

{

return resp;

}

else if(test!=-1)

{

if(test>matchLength)

{

matchLength=test;

finalResp=resp;

}

}

}

if(finalResp!=null)

for(int j=0;j<ChatKnow.Knowledge-Base.length;j++)

for(int i=0;i<ChatKnow.Knowledge-Base[j][1].length;i++)

{

if(ChatKnow.Knowledge-Base[j][1][i].equals(finalResp))

{ ArrayList<Integer> temp=new ArrayList<Integer>();

if(!counter.containsKey(j))

temp.add(i);

else

{ temp=counter.get(j);

temp.add(i);

}

counter.put(j,temp);

}

}

return finalResp;

}

static int keywordMatch(String s1,String s2)

{

String sub="";

String base="";

if(s1.length()> s2.length())

{

sub=s2;

base=s1;

}

else

{

sub=s1;

base=s2;

}

String[] stripSub=sub.split(" ");

String[] stripBase=base.split(" ");

int matchLength=stripSub.length;

for(;matchLength>0;matchLength--)

{

for(int j=0;j<stripSub.length-matchLength+1;j++)

{ String temp="";

for(int k=j;k<j+matchLength;k++)

temp+=(stripSub[k]+" ");

for(int t=0;t<stripBase.length-matchLength+1;t++)

{ String temp2="";

for(int x=t;x<t+matchLength;x++)

temp2+=(stripBase[x]+" ");

if(temp2.equalsIgnoreCase(temp)) return matchLength;

}

}

}

return -1;

}

**12.8 Main() method (Class:ChatBot)**

public static void main(String args[]) throws Exception{

Scanner scan=new Scanner(System.in);

while(true){

System.out.print('>');

String input=scan.nextLine();

if(input.equalsIgnoreCase("BYE")){

String response="BYE-BYE!"; System.out.println('>'+response);

break;

}

String temp=cleanString(input);

String response=find\_match(temp);

if(response.length()==0){

System.out.println(">"+"I DONT UNDERSTAND YOU");

}

else if (response.equals("1"))

{ String[] key=input.split(" ");

getFunData ob=new getFunData();

System.out.println(ob.search(key[key.length-2]));

}

else if (response.equals("2"))

{

getFunData ob=new getFunData();

System.out.println(ob.getJoke());

}

else if (response.equals("3")) {

String[] key=input.split(" ");

WikiImages wik=new WikiImages(key[key.length-2]);

wik.imagesearch();

}

else if(response.equals("4"))

{

String[] key=input.split(" "); Wiki wik=new Wiki();

wik.search(key[key.length-2]);

}

else if(response.equals("5")) {

{

String[] key=input.split(" ");

WikiImage wik=new WikiImage(key[key.length-2]); wik.imagesearch();

}

else if (response.equals("6"))

{

String[] key=input.split(" ");

String s = key[key.length-1];

Dictionary obj = new Dictionary();

s=s.toLowerCase();

if(obj.wordExists(s) == true)

{

System.out.println("word exists");

}

else

System.out.println("word doesnt exist");

}

else

System.out.println(">"+response);

}

scan.close();

}