Rajshahi University of Engineering & Technology
Rajshahi-6204

Department of Computer Science & Engineering

COURSE NO.:CSE 600

COURSE NAME: Software Development Project II

SUBMITTED BY:	**SUBMITTED TO:**
ROLL: 113007,113014,113015 **YEAR: 3RD YEAR, 6TH SEMESTER** **SESSION: 2011-2012**	**BIPRODIP PAL** **Assistant Professor** **Dept. of Computer Science & Engineering**

Code (In Java Language + Using MySQL Database)

Library : MySQL Connector ,Text to Voice convertor

```
import java.beans.Statement;

import java.sql.Connection;

import java.sql.DriverManager;

import java.sql.PreparedStatement;

import java.lang.Math;

import java.io.File;

import java.io.*;

import java.util.*;

import java.lang.*;

import java.lang.String;

import java.sql.ResultSet;

import java.util.Random;

import java.util.Scanner;

import java.util.StringTokenizer;

import javax.swing.JOptionPane;

import java.util.Locale;

import javax.speech.Central;

import javax.speech.synthesis.Synthesizer;

import javax.speech.synthesis.SynthesizerModeDesc;


public class Natural_Language_Processing

{

   // POS variable

   String noun[] = new String[200];

   String pronoun[] = new String[200];

   String adjective[] = new String[200];

   String verb[] = new String[200];
```

```
String adverb[] = new String[200];
String preposition[] = new String[200];
String conjunction[] = new String[200];
String interjection[] = new String[200];
String article[] = new String[200];
String negative[]=new String[200];
int num_of_noun = 0, num_of_pronoun = 0, num_of_adjective = 0, num_of_verb = 0, num_of_adverb = 0;
int num_of_prepostion = 0, num_of_conjunction = 0, num_of_interjection = 0, num_of_article = 0,num_of_negative=0;

//database connection
static final String JDBC_DRIVER = "com.mysql.jdbc.Driver";
static final String DATABASE_URL = "jdbc:mysql://localhost/NLP";
com.mysql.jdbc.Connection connection = null; // manages connection
com.mysql.jdbc.Statement statement = null; // query statement*/
ResultSet res = null;

// Get the POS from dictionary
public Natural_Language_Processing() {
   String query1 = "";
   String query2 = " ";
   String query3 = "";
   String query4 = " ";
   String query5 = "";
   String query6 = " ";
   String query7 = "";
   String query8 = " ";
   String query9= " ";
try {
      Class.forName(JDBC_DRIVER); // load database driver class
     // establish connection to database
      connection = (com.mysql.jdbc.Connection) DriverManager.getConnection(DATABASE_URL, "root", "");
```

```
// create Statement for querying database
  statement = (com.mysql.jdbc.Statement) connection.createStatement();
  query1 = "select * from dictionary where parts_of_speech='noun'";
  query2 = "select * from dictionary where parts_of_speech='pronoun'";
  query3 = "select *from dictionary where parts_of_speech='verb'";
  query4 = "select * from dictionary where parts_of_speech='adverb'";
  query5 = "select * from dictionary where parts_of_speech='adjective'";
  query6 = "select *from dictionary where parts_of_speech='preposition'";
  query7="select * from dictionary where parts_of_speech='conjunction'";
  query8="select * from dictionary where parts_of_speech='article'";
  query9="select * from dictionary where parts_of_speech='negative'";

  res = statement.executeQuery(query1);
  while (res.next()) {
      noun[++num_of_noun] = res.getString(1);
  }
res = statement.executeQuery(query2);
  while (res.next()) {
      pronoun[++num_of_pronoun] = res.getString(1);
  }
res = statement.executeQuery(query3);
  while (res.next()) {
      verb[++num_of_verb] = res.getString(1);
          }
  res = statement.executeQuery(query4);
  while (res.next()) {
      adverb[++num_of_adverb] = res.getString(1);
          }
  res = statement.executeQuery(query5);
  while (res.next()) {
      adjective[++num_of_adjective] = res.getString(1);          }
```

```
    res = statement.executeQuery(query6);
    while (res.next()) {
    preposition[++num_of_prepostion] = res.getString(1); }
    res = statement.executeQuery(query7);
    while (res.next()) {
    conjunction[++num_of_conjunction] = res.getString(1);}
    res = statement.executeQuery(query8);
    while (res.next()) {
     article[++num_of_article] = res.getString(1);  }
    res = statement.executeQuery(query9);
     while (res.next()) {
     negative[++num_of_negative] = res.getString(1);  }        }
   catch (Exception e) {
   e.printStackTrace();      }    }


  /**
* @param args the command line arguments
*/
   public static void main(String[] args) throws IOException
   {
       Natural_Language_Processing obj=new Natural_Language_Processing();     // object declaration


       //get input sentence, declaring variables
      String Parts_Of_Speech[]=new String[20];
      String position[]=new String[20];
      String noun_phrase[]=new String[20];
      String vp[]=new String[20];
     String verbphrase[][]=new String[20][20];
      String question;
      String structure_of_question[]=new String[20];
```

```
int input_sentence_structure[]=new int[20];
int  valid_sentence_structure[][]={{1,5,7,4,1},{2,5,5,6},{7,1,5,7,1},{7,1,5,3,7,1},{2,5,7,4,6}
      ,{2,5,8,4,3,1},{1,5,5,2,1},{2,5,1,3,1},{1,5,3,7,1},{7,6,3,1,5,4}};
int lenght_of_each_structure[]={5,4,5,6,5,6,5,5,5,6};
int cn=0;
int dd[]=new int[20];
Scanner scanner= new Scanner( System.in);
JOptionPane.showMessageDialog(null,"SECTION ONE :: LEARNING MODE\n(PARSING & PARTS OF SPEECH TAGGING)");
System.out.println("SECTION ONE :: LEARNING MODE\n(PARSING & PARTS OF SPEECH TAGGING)");
System.out.println("================================\n================================");
System.out.println("Type a sentence and press Enter: ");
String sentence = scanner.nextLine();

//process user sentence
int a=0;
int f=0;
int vv=0;
StringTokenizer tokens= new StringTokenizer(sentence);
System.out.printf("Number of words: %d",tokens.countTokens() );
System.out.println();
System.out.print("The words are:\n");
while(tokens.hasMoreTokens())
{
   Parts_Of_Speech[a]=tokens.nextToken();
   //System.out.println(tokens.nextToken());
   a++;
}
```

```
//Sentence structure and Parts of Speech Tagging
boolean flag=false;
boolean lexicon=false;
int b=0;
Scanner abc;
for(int i=0;i<a;i++)
{
   lexicon=false;
   System.out.println(Parts_Of_Speech[i]);
   for(int j=0;j<obj.noun.length;j++)
   {
      if(Parts_Of_Speech[i].equalsIgnoreCase(obj.noun[j]))
      {
         lexicon=true;
         input_sentence_structure[b]=1;
         b++;
         break;
      }
   }
   for(int j=0;j<obj.pronoun.length;j++)
   {
      if(Parts_Of_Speech[i].equalsIgnoreCase(obj.pronoun[j]))
      {
         lexicon=true;
         input_sentence_structure[b]=2;
         b++;
         break;
      }
   }
   for(int j=0;j<obj.preposition.length;j++)  {
  if(Parts_Of_Speech[i].equalsIgnoreCase(obj.preposition[j]))
```

```
   {
      lexicon=true;
      input_sentence_structure[b]=3;
      b++;break;
   }            }

for(int j=0;j<obj.adjective.length;j++)
{
   if(Parts_Of_Speech[i].equalsIgnoreCase(obj.adjective[j]))
   {
      lexicon=true;
      input_sentence_structure[b]=4;
      b++;
      break;
   }
}
for(int j=0;j<obj.verb.length;j++)
{
   if(Parts_Of_Speech[i].equalsIgnoreCase(obj.verb[j]))
   {
      lexicon=true;
      input_sentence_structure[b]=5;
      b++;
      break;
   }
}
for(int j=0;j<obj.adverb.length;j++)
{
   if(Parts_Of_Speech[i].equalsIgnoreCase(obj.adverb[j]))
   {
      lexicon=true;
```

```
            input_sentence_structure[b]=6;
            b++;
            break;   }    }
    for(int j=0;j<obj.article.length;j++)     {
       if(Parts_Of_Speech[i].equalsIgnoreCase(obj.article[j]))  {
            lexicon=true;
            input_sentence_structure[b]=7;
            b++;
            break;   }  }
    for(int j=0;j<obj.negative.length;j++)      {
       if(Parts_Of_Speech[i].equalsIgnoreCase(obj.negative[j]))   {
            lexicon=true;
            input_sentence_structure[b]=8;
            b++;
            break;      }   }
    for(int j=0;j<obj.conjunction.length;j++)  {
       if(Parts_Of_Speech[i].equalsIgnoreCase(obj.conjunction[j]))  {
            lexicon=true;
            input_sentence_structure[b]=9;
            b++;
            break;  }  }
       if(lexicon==false){
        input_sentence_structure[b]=0;
        b++;
        continue;  }  }


boolean wfound=false;
System.out.println("The structure of the sentence is: ");
for(int z=0;z<b;z++)   {
    if(input_sentence_structure[z]==1)  {
        System.out.print("NOUN+");
```

```
    continue;    }
  if(input_sentence_structure[z]==2)   {
    System.out.print("PRONOUN+");
    continue;  }
  if(input_sentence_structure[z]==3){
  System.out.print("PREPOSITION+");
    continue;   }
  if(input_sentence_structure[z]==4)  {
    System.out.print("ADJECTIVE+");
    continue;  }
  if(input_sentence_structure[z]==5)   {
    System.out.print("VERB+");
    continue;}
  if(input_sentence_structure[z]==6)  {
    System.out.print("ADVERB+");
    continue;   }
  if(input_sentence_structure[z]==7){
    System.out.print("ARTICLE+");
    continue;}
  if(input_sentence_structure[z]==8){
    System.out.print("NEGETIVE+");
    continue;  }
  if(input_sentence_structure[z]==9)  {
    System.out.print("CONJUNCTION+");
    continue;}
  if(input_sentence_structure[z]==0)  {
for(int str=0;str<10;str++)   {
        int mchcn=0;
        int newpos=0;
        int extra=0;
```

```
    // matching the sentence structure
    for(int mch=0;mch<lenght_of_each_structure[str];mch++)  {
       if(input_sentence_structure[mch]==valid_sentence_structure[str][mch])       {
          mchcn++;    }


else    {
          extra++;
          newpos=valid_sentence_structure[str][mch];    } }
    if(mchcn==((lenght_of_each_structure[str])-1))   {
       if(newpos==1){
          try{
      String insert="insert into dictionary values('"+Parts_Of_Speech[z]+"','noun','')";
      obj.statement.execute(insert);
      JOptionPane.showMessageDialog(null,Parts_Of_Speech[z]+" added to the lexicon!!\n");     }
      catch(Exception e3)    {
      JOptionPane.showMessageDialog(null,Parts_Of_Speech[z]+" can't be added to the lexicon!!\n");}
      System.out.print("NOUN+");}
    if(newpos==2)   {
          System.out.print("PRONOUN+");
          try{
  String insert="insert into dictionary values('"+Parts_Of_Speech[z]+"','pronoun','')";
   obj.statement.execute(insert);
   JOptionPane.showMessageDialog(null,Parts_Of_Speech[z]+" added to the lexicon!!\n");    }
  catch(Exception e3)  {
  JOptionPane.showMessageDialog(null,Parts_Of_Speech[z]+" can't be added to the lexicon!!\n");   }    }
  if(newpos==3    {
   System.out.print("PREPOSITION+");
  try{
  String insert="insert into dictionary values('"+Parts_Of_Speech[z]+"','preposition','')";
  obj.statement.execute(insert);
  JOptionPane.showMessageDialog(null,Parts_Of_Speech[z]+" added to the lexicon!!\n");    }
```

```
 catch(Exception e3)     {
JOptionPane.showMessageDialog(null,Parts_Of_Speech[z]+" can't be added to the lexicon!!\n");    }  }
if(newpos==4)  {
            System.out.print("ADJECTIVE+");


            try{
      String insert="insert into dictionary values('"+Parts_Of_Speech[z]+"','adjective','')";
        obj.statement.execute(insert);
        JOptionPane.showMessageDialog(null,Parts_Of_Speech[z]+" added to the lexicon!!\n");   }
        catch(Exception e3)    {
       JOptionPane.showMessageDialog(null,Parts_Of_Speech[z]+" can't be added to the lexicon!!\n");    }  }
        if(newpos==5)    {
            System.out.print("VERB+");
             try{
      String insert="insert into dictionary values('"+Parts_Of_Speech[z]+"','verb','')";
      obj.statement.execute(insert);
        JOptionPane.showMessageDialog(null,Parts_Of_Speech[z]+" added to the lexicon!!\n");      }
        catch(Exception e3)  {
      JOptionPane.showMessageDialog(null,Parts_Of_Speech[z]+" can't be added to the lexicon!!\n");    }      }
     if(newpos==6)    {
            System.out.print("ADVERB+");
            try{
      String insert="insert into dictionary values('"+Parts_Of_Speech[z]+"','adverb','')";
      obj.statement.execute(insert);
      JOptionPane.showMessageDialog(null,Parts_Of_Speech[z]+" added to the lexicon!!\n");    }
     catch(Exception e3)       {
     JOptionPane.showMessageDialog(null,Parts_Of_Speech[z]+" can't be added to the lexicon!!\n");  }    }
      if(newpos==7)   {
            System.out.print("ARTICLE+");
            try{
      String insert="insert into dictionary values('"+Parts_Of_Speech[z]+"','article','')";
```

```
            obj.statement.execute(insert);
            JOptionPane.showMessageDialog(null,Parts_Of_Speech[z]+" added to the lexicon!!\n");    }
            catch(Exception e3)    {
          JOptionPane.showMessageDialog(null,Parts_Of_Speech[z]+" can't be added to the lexicon!!\n");     }      }


            if(newpos==8)  {
            System.out.print("NEG+");
             try{
            String insert="insert into dictionary values('"+Parts_Of_Speech[z]+"','negative','')";
             obj.statement.execute(insert);
        JOptionPane.showMessageDialog(null,Parts_Of_Speech[z]+" added to the lexicon!!\n");  }
        catch(Exception e3)     {
        JOptionPane.showMessageDialog(null,Parts_Of_Speech[z]+" can't be added to the lexicon!!\n");    }   }
        if(newpos==9)  {
               System.out.print("CONJUNCTION+");
                try{
         String insert="insert into dictionary values('"+Parts_Of_Speech[z]+"','conjunction','')";
            obj.statement.execute(insert);
            JOptionPane.showMessageDialog(null,Parts_Of_Speech[z]+" added to the lexicon!!\n");     }
            catch(Exception e3)   {
         JOptionPane.showMessageDialog(null,Parts_Of_Speech[z]+" can't be added to the lexicon!!\n");     }    }      }
      }     }       }
   System.out.println();


//PRACTICE MODE
JOptionPane.showMessageDialog(null,"SECTION TWO:: PRACTICE MODE\n(QUESTION AND ANSWER)");
System.out.println("\n\nSECTION TWO:: PRACTICE MODE\n(QUESTION AND ANSWER)");
System.out.println("============================\n==============================");
Scanner msg= new Scanner( System.in );
int m;
String context[]=new String[20];
```

```
System.out.println("How many sentences will be in the combined context??");
int sentencenum= msg.nextInt();
System.out.println();
System.out.println("Enter a combined context: ");
for(m=0;m<sentencenum+1;m++)  {
   context[m] = msg.nextLine();
   int u=0;
   StringTokenizer tokenss= new StringTokenizer(context[m]);
   System.out.println();
   while(tokenss.hasMoreTokens())  {
   position[u]=tokenss.nextToken();
      u++;  }
   flag=false;
   int v=0;

   //PROCESSING NOUN PHRASE (NOUN+PRONOUN+ADVERB)
   for(v=0;v<(Math.ceil(u/2));v++)      {
      flag=false;
      if(flag==true)  {
      break;  }
      for(int w=0;w<obj.adverb.length;w++)    {
      if(position[v].equalsIgnoreCase(obj.adverb[w]))   {
      noun_phrase[m]=obj.adverb[w];
            flag=true;
            break;   }   }
      for(int w=0;w<obj.noun.length;w++)  {
      if(position[v].equalsIgnoreCase(obj.noun[w]))  {
      noun_phrase[m]=obj.noun[w];
     flag=true;
     break;}    }
      for(int w=0;w<obj.pronoun.length;w++)  {
```

```
        if(position[v].equalsIgnoreCase(obj.pronoun[w]))  {
            noun_phrase[m]=obj.pronoun[w];
            break;        }    }
     for(int w=0;w<obj.verb.length;w++)    {

        if(position[v].equalsIgnoreCase(obj.verb[w]))    {
            noun_phrase[m]=obj.verb[w];
            flag=true;
            break;      }     }
    if(flag==false)    {
   for(int w=0;w<obj.verb.length;w++)   {
   if(position[v].equalsIgnoreCase(obj.verb[w]))    {
              noun_phrase[m]=obj.verb[w];
              flag=true;
              break;     }        }    }
if(flag==true){
        break;   }   }
 flag=false;
 cn=0;

 //VERB PHRASE PROCESSING (NOUN+ADJECTIVE+VERB+ADVERB+PREPOSITION)
 for(v=(u/2);v<u;v++)     {
    for(int w=0;w<obj.adjective.length;w++)    {
        if(flag==false){
            if(position[v].equalsIgnoreCase(obj.adjective[w]))   {
                vp[m]=obj.adjective[w];
                verbphrase[m][cn]=obj.adjective[w];
                cn++;
                flag=true;       }       }
        else    {
            if(position[v].equalsIgnoreCase(obj.adjective[w]))   {
```

```
            vp[m]=vp[m]+" "+obj.adjective[w];
            verbphrase[m][cn]=obj.adjective[w];
            cn++;
            flag=true;      }     }
 }
for(int w=0;w<obj.preposition.length;w++)    {
   if(flag==true)      {
      if(position[v].equalsIgnoreCase(obj.preposition[w]))      {
            vp[m]=vp[m]+" "+obj.preposition[w];
            verbphrase[m][cn]=obj.preposition[w];
            cn++;
            flag=true;   } }
   else   {
      if(position[v].equalsIgnoreCase(obj.preposition[w]))       {
            vp[m]=obj.preposition[w];
            verbphrase[m][cn]=obj.preposition[w];
            cn++;
            flag=true;   }  }      }
for(int w=0;w<obj.verb.length;w++)  {
   if(flag==true)   {
      if(position[v].equalsIgnoreCase(obj.verb[w]))    {
            vp[m]=vp[m]+" "+obj.verb[w];
            verbphrase[m][cn]=obj.verb[w];
            cn++;
            flag=true;    }      }
   else     {
      if(position[v].equalsIgnoreCase(obj.verb[w])){
            vp[m]=obj.verb[w];
            verbphrase[m][cn]=obj.verb[w];
            cn++;
            flag=true;    }     }     }
```

```
  for(int w=0;w<obj.adverb.length;w++)  {
    if(flag==true)        {
      if(position[v].equalsIgnoreCase(obj.adverb[w]))       {
        vp[m]=vp[m]+" "+obj.adverb[w];
        verbphrase[m][cn]=obj.adverb[w];
        cn++;
        flag=true;      }   }
    else    {
      if(position[v].equalsIgnoreCase(obj.adverb[w]))     {
        vp[m]=obj.adverb[w];
        verbphrase[m][cn]=obj.adverb[w];
        cn++;
        flag=true;    }             }          }
  for(int w=0;w<obj.noun.length;w++)       {
    if(flag==true)          {
      if(position[v].equalsIgnoreCase(obj.noun[w]))  {
        vp[m]=vp[m]+" "+obj.noun[w];
        verbphrase[m][cn]=obj.noun[w];
        cn++;
        flag=true;  }   }
    else     {
      if(position[v].equalsIgnoreCase(obj.noun[w]))  {
        vp[m]=obj.noun[w];
        verbphrase[m][cn]=obj.noun[w];
        cn++;
        flag=true;    }  }      }   }
dd[m]=cn;          }
```

```
 System.out.println("NOUN PHRASE      VERB PHRASE");
System.out.println("================================");
for(int l=1;l<sentencenum+1;l++)    {
   System.out.println(noun_phrase[l]+"   "+vp[l]);   }


JOptionPane.showMessageDialog(null,"Your context is acknowledged.");
System.out.println("\nNow type your questions based on given context, enter \"stop\" for exit");
boolean q=false;
while(!q)   {
   Scanner wh= new Scanner( System.in );
   System.out.println("\nQuestion: ");
   question = scanner.nextLine();
   if(question.equalsIgnoreCase("stop"))   {
      q=true;
      System.exit(0);
      break;  }
   if(q==true)    {
      break;    }


  //tokenizer
   int h=0;
   StringTokenizer tokensss= new StringTokenizer(question);
  System.out.println();
 while(tokensss.hasMoreTokens())   {
structure_of_question[h]=tokensss.nextToken();
h++;}
   f=0;
   boolean qnoun=false;
   boolean qverb=false;
   for(int g=0;g<h;g++)    {
   for(int t=1; t<sentencenum+1; t++){
```

```
            if(structure_of_question[g].equalsIgnoreCase(noun_phrase[t])){
               qnoun=true;                    //question is in noun form
               f=t;                           //we found noun phrase in position t
               break;  }   }
          if(qnoun==true)   {
            break;       }
}


        if(qnoun==true)    {                          //question is in noun phrase form
          try       {
            System.setProperty("freetts.voices","com.sun.speech.freetts.en.us.cmu_us_kal.KevinVoiceDirectory");
            Central.registerEngineCentral("com.sun.speech.freetts.jsapi.FreeTTSEngineCentral");
            Synthesizer  synthesizer =Central.createSynthesizer(new SynthesizerModeDesc(Locale.US));
            synthesizer.allocate();
            synthesizer.resume();
            synthesizer.speakPlainText(vp[f], null);}
           catch(Exception e1){
           JOptionPane.showMessageDialog(null,"Could not convert the answer in speech");  }
           System.out.println("Answer: "+vp[f]);              }
           else //question is in verb phrase form    {
          for(int g=0;g<h;g++)   {
          for(int t=1; t<sentencenum+1; t++)  {
         for(int r=0;r<dd[t];r++)    {
        if(structure_of_question[g].equalsIgnoreCase(verbphrase[t][r]))    {
                    qverb=true; //question is in verb form
                    vv=t; //we found noun phrase in position t
                    break;   } }      }


            if(qverb==true) //question is in verb phrase form
```

```
                {
                   try  {
                      System.setProperty("freetts.voices","com.sun.speech.freetts.en.us.cmu_us_kal.KevinVoiceDirectory");
                      Central.registerEngineCentral("com.sun.speech.freetts.jsapi.FreeTTSEngineCentral");
                      Synthesizer  synthesizer =Central.createSynthesizer(new SynthesizerModeDesc(Locale.US));
                      synthesizer.allocate();
                      synthesizer.resume();
                      synthesizer.speakPlainText(noun_phrase[vv], null);                        }
                   catch(Exception e2)
                   {
                      JOptionPane.showMessageDialog(null,"Could not convert the answer in speech");
                   }
                   System.out.println("Answer: "+noun_phrase[vv]);
                   break;
                }
             }
          }
       }
    }
}
```