CS 461 - Artificial Intelligence
Prof. Varol Akman

Final Report
10th May 2018

Group: *Bilkent_Colts*

Yrsyskul Turduev
Alemdar Salmoor
Alperen Kaya
Hamza Saeed Khan

Introduction

The purpose of the program is to solve the New York Times daily mini- crossword puzzle. The program initially, downloads the puzzle form the NYT website. The puzzle and the clues are then displayed just as they appear on the website. In order to solve the puzzle, the program searches google and wikipedia with each clue and scraps data from both of them. The next step for the program is to remove from the list of candidates, words that do not meet the word length requirement. Afterwards, the length filtered words are compared to the answers taken from the website to choose one candidate that is a definite answer. In the penultimate step, different combinations of the other candidates are tried with this definite candidate to see which combination leads to the most answers found. Finally, any word partially unfilled is plugged in, after searching with letter constraints from a database of 4o,ooo words that is stored locally. The answer found are displayed on GUI rendered by the program.

Implementation

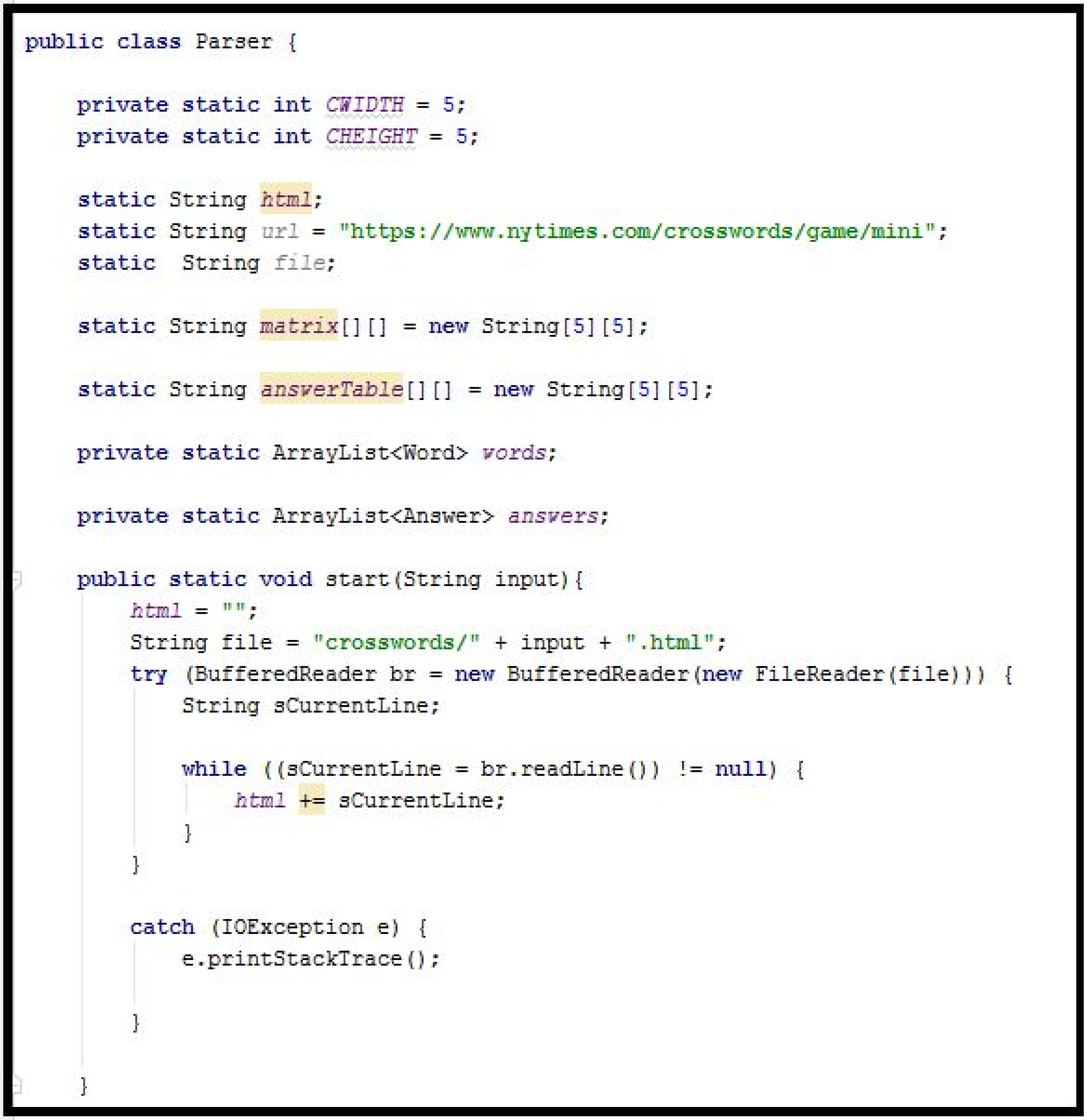
The implementation of the program is divided into the GUI components and the backend components. The GUI classes are responsible for the visual representation of the crossword puzzle, the hints, answers and the log. While the backend classes download the puzzle, solve it and as well as control gui classes.

System decomposition:

As mentioned above our application consists of two subsystems:
1. Subsystem is for the logic of the application which includes such classes as Word.java, Answer.java, Parser.java
2. Graphical user interface consisting GridCell.java, AnswerCell.java, CrosswordGrid.java

Logical subsystem:

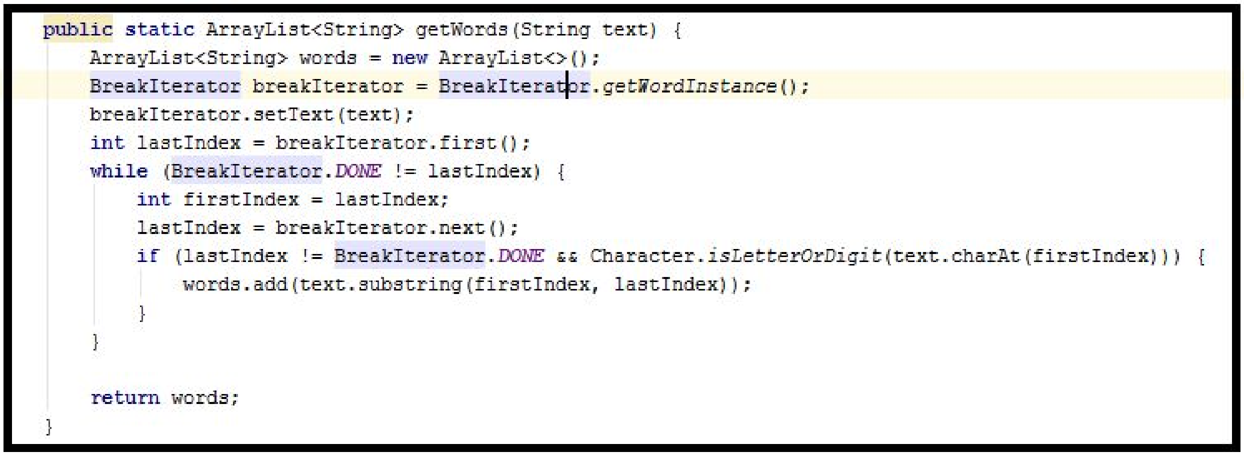
Parser.java:



```
public class Parser {

    private static int CWIDTH = 5;
    private static int CHEIGHT = 5;

    static String html;
    static String url = "https://www.nytimes.com/crosswords/game/mini";
    static  String file;

    static String matrix[][] = new String[5][5];

    static String answerTable[][] = new String[5][5];

    private static ArrayList<Word> words;

    private static ArrayList<Answer> answers;

    public static void start(String input){
        html = "";
        String file = "crosswords/" + input + ".html";
        try (BufferedReader br = new BufferedReader(new FileReader(file))) {
            String sCurrentLine;

            while ((sCurrentLine = br.readLine()) != null) {
                html += sCurrentLine;
            }
        }

        catch (IOException e) {
            e.printStackTrace();

        }

    }
```

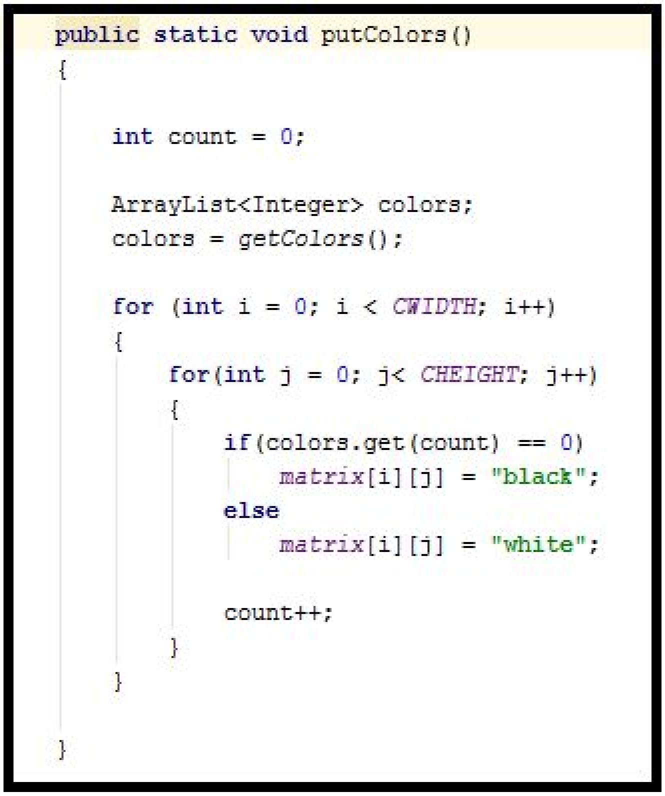
-Above is our data members along with the initialization of the required crossword.

-There are some methods such as getClues(), getNumbers(), getColors() and getAnswers() that scrap respective information from the requested NY times mini crossword webpage using the corresponding methods of jsoup library like getElementsbyTag and getElementsbyClass, screenshots of these classes are omitted here to avoid redundancy.



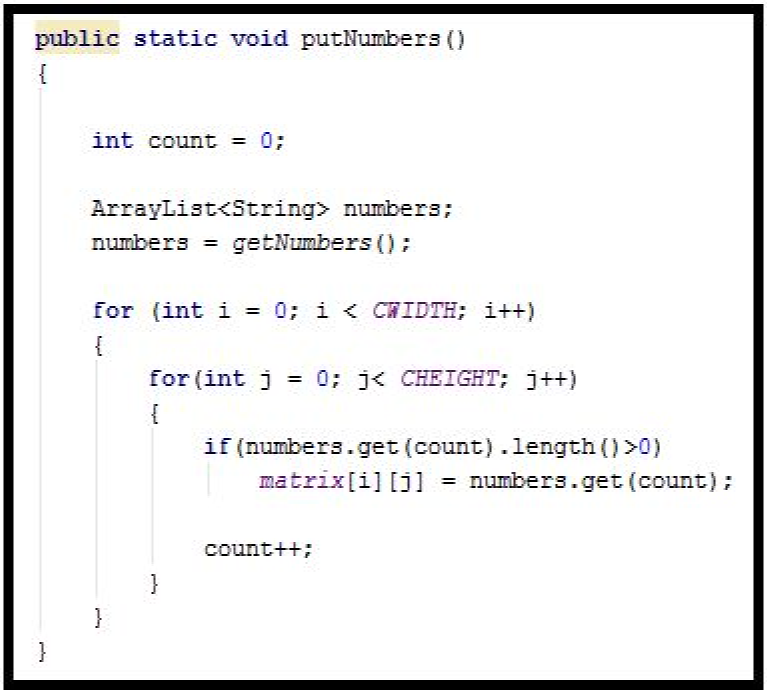
```
public static ArrayList<String> getWords(String text) {
    ArrayList<String> words = new ArrayList<>();
    BreakIterator breakIterator = BreakIterator.getWordInstance();
    breakIterator.setText(text);
    int lastIndex = breakIterator.first();
    while (BreakIterator.DONE != lastIndex) {
        int firstIndex = lastIndex;
        lastIndex = breakIterator.next();
        if (lastIndex != BreakIterator.DONE && Character.isLetterOrDigit(text.charAt(firstIndex))) {
            words.add(text.substring(firstIndex, lastIndex));
        }
    }

    return words;
}
```

getWords(): This is the helper method that separates the words from a stream that was acquired from parsing the elements.



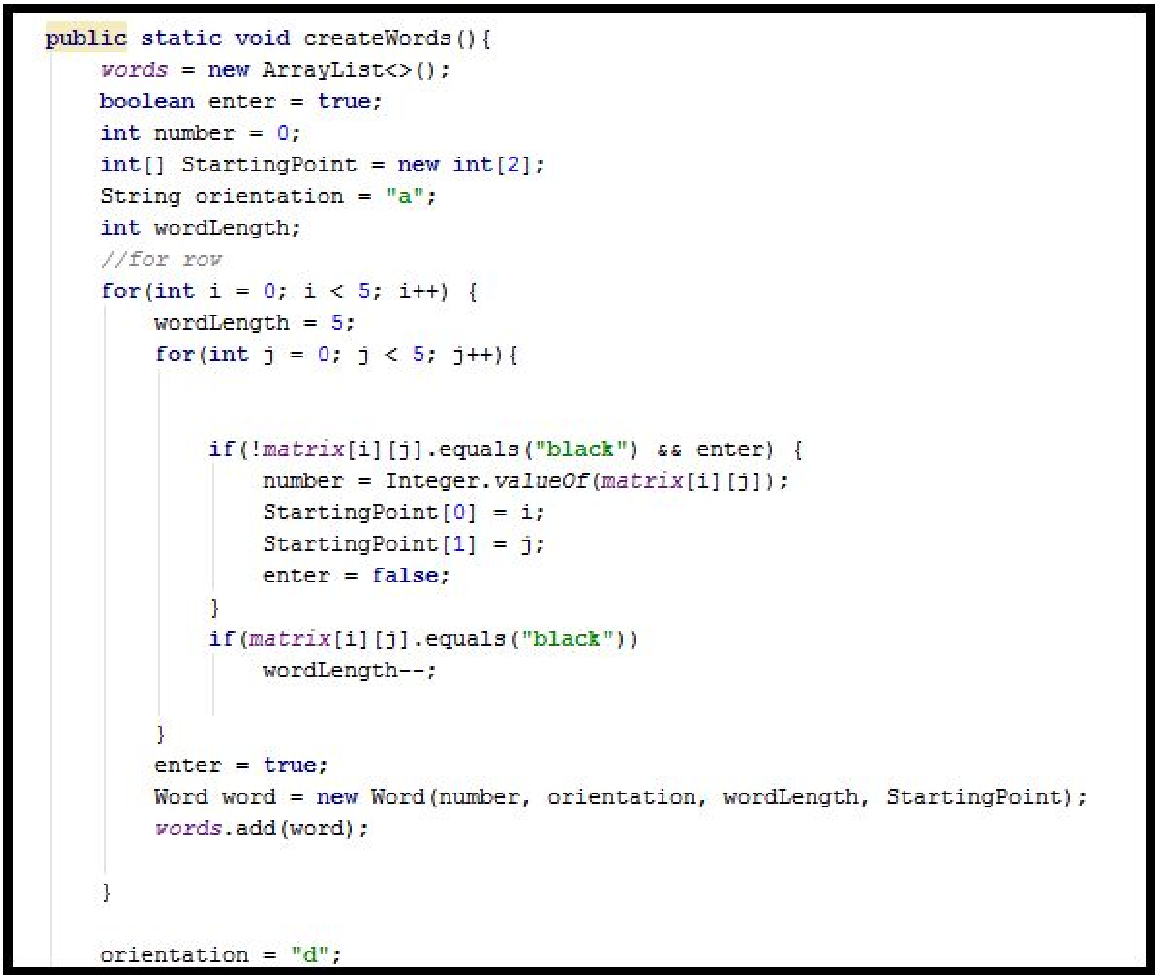
```
public static void putColors()
{


    int count = 0;

    ArrayList<Integer> colors;
    colors = getColors();

    for (int i = 0; i < CWIDTH; i++)
    {
        for(int j = 0; j< CHEIGHT; j++)
        {
            if(colors.get(count) == 0)
                matrix[i][j] = "black";
            else
                matrix[i][j] = "white";

            count++;
        }
    }

}
```

putColors(): This method that initializes the main matrix with proper coloring. It is used to locate the start end of the each words. The details coming in further explanations to the screenshots.



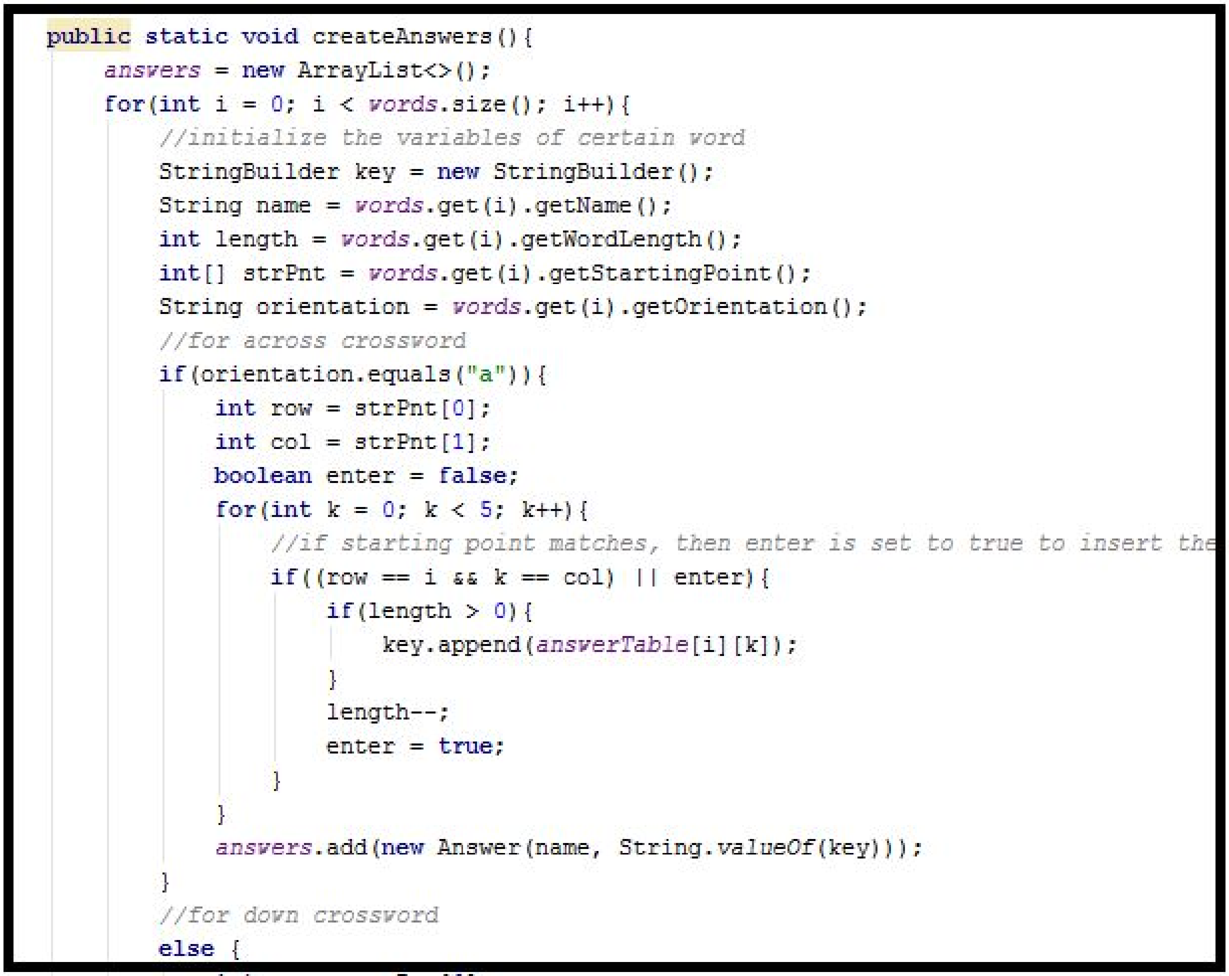
```
public static void putNumbers()
{

    int count = 0;

    ArrayList<String> numbers;
    numbers = getNumbers();

    for (int i = 0; i < CWIDTH; i++)
    {
        for(int j = 0; j< CHEIGHT; j++)
        {
            if(numbers.get(count).length()>0)
                matrix[i][j] = numbers.get(count);

            count++;
        }
    }
}
```

putNumbers(): it passes the numbering to the main matrix. This as well aids in locating the start of the words. Will be more clear in prior screenshot explanations.



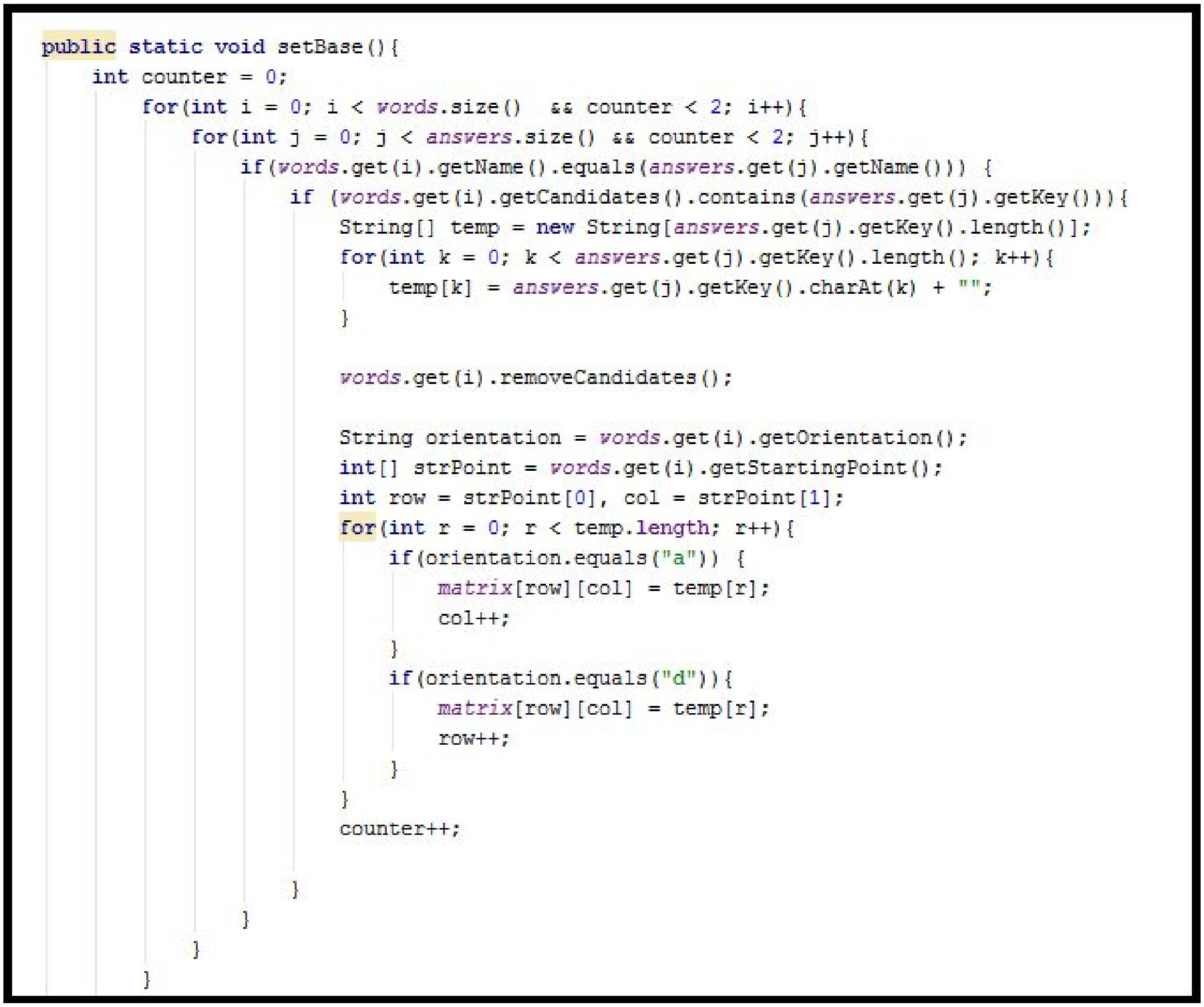
```
public static void createWords(){
    words = new ArrayList<>();
    boolean enter = true;
    int number = 0;
    int[] StartingPoint = new int[2];
    String orientation = "a";
    int wordLength;
    //for row
    for(int i = 0; i < 5; i++) {
        wordLength = 5;
        for(int j = 0; j < 5; j++){


            if(!matrix[i][j].equals("black") && enter) {
                number = Integer.valueOf(matrix[i][j]);
                StartingPoint[0] = i;
                StartingPoint[1] = j;
                enter = false;
            }
            if(matrix[i][j].equals("black"))
                wordLength--;

        }
        enter = true;
        Word word = new Word(number, orientation, wordLength, StartingPoint);
        words.add(word);

    }

    orientation = "d";
```

createWords(): This method that creates 10 word objects that correspond to specific crosswords words, without answers. It has number that corresponds to the number that it starts from, orientation whether it is across or down. Orientation is required since the same numbered word may may be across or down. Name and orientation make a unique name of a word object. It has starting point and length as well. Along with the orientation our program will be able to locate every word on the matrix. Explanation of the Word Object is after the Parser class explanation.



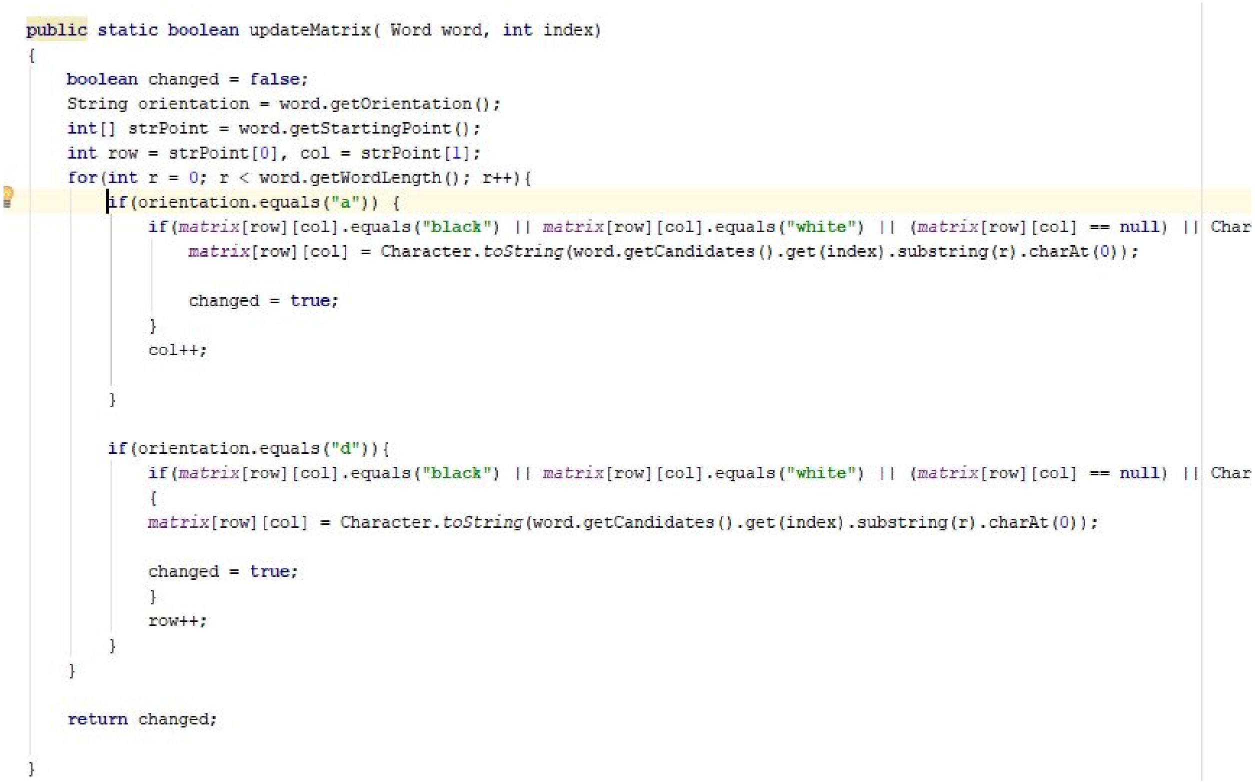
```
public static void createAnswers(){
    answers = new ArrayList<>();
    for(int i = 0; i < words.size(); i++){
        //initialize the variables of certain word
        StringBuilder key = new StringBuilder();
        String name = words.get(i).getName();
        int length = words.get(i).getWordLength();
        int[] strPnt = words.get(i).getStartingPoint();
        String orientation = words.get(i).getOrientation();
        //for across crossword
        if(orientation.equals("a")){
            int row = strPnt[0];
            int col = strPnt[1];
            boolean enter = false;
            for(int k = 0; k < 5; k++){
                //if starting point matches, then enter is set to true to insert the l
                if((row == i && k == col) || enter){
                    if(length > 0){
                        key.append(answerTable[i][k]);
                    }
                    length--;
                    enter = true;
                }
            }
            answers.add(new Answer(name, String.valueOf(key)));
        }
        //for down crossword
        else {
```

createAnswers(): here is part of the method that initializes an answerTable matrix because our words can not interfere with the actual answers. This matrix is used to make sure that we are using correct base words and have something to bound from. More to come in explanation of the SetBase() method.



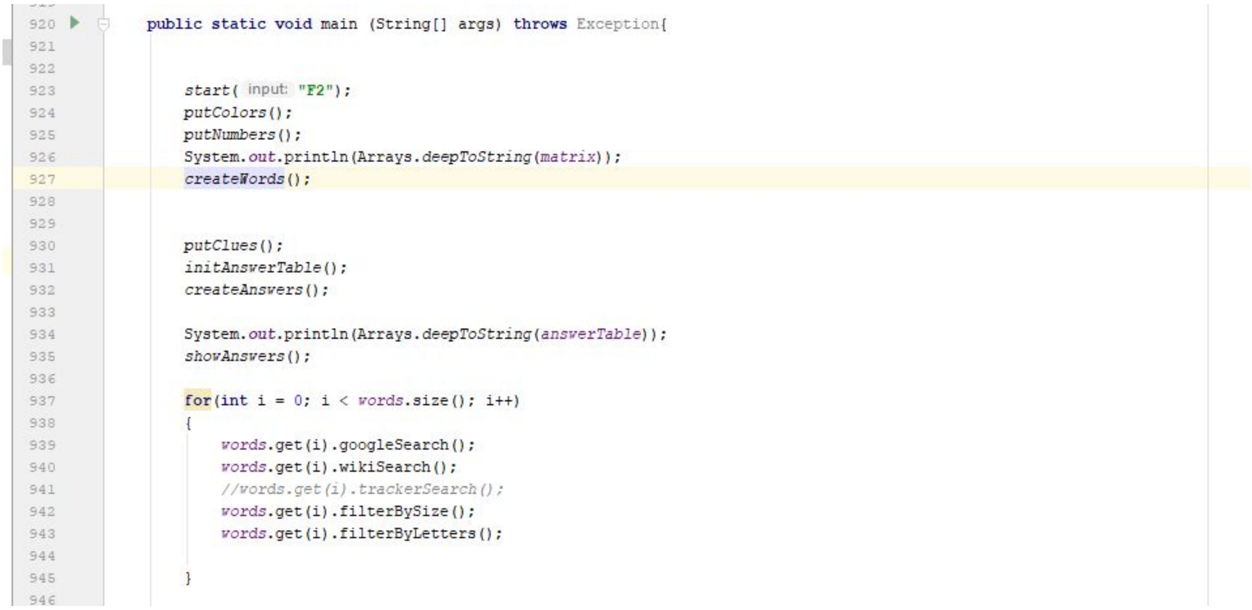
```
public static void setBase(){
    int counter = 0;
        for(int i = 0; i < words.size()  && counter < 2; i++){
            for(int j = 0; j < answers.size() && counter < 2; j++){
                if(words.get(i).getName().equals(answers.get(j).getName())) {
                    if (words.get(i).getCandidates().contains(answers.get(j).getKey())){
                        String[] temp = new String[answers.get(j).getKey().length()];
                        for(int k = 0; k < answers.get(j).getKey().length(); k++){
                            temp[k] = answers.get(j).getKey().charAt(k) + "";
                        }

                        words.get(i).removeCandidates();

                        String orientation = words.get(i).getOrientation();
                        int[] strPoint = words.get(i).getStartingPoint();
                        int row = strPoint[0], col = strPoint[1];
                        for(int r = 0; r < temp.length; r++){
                            if(orientation.equals("a")) {
                                matrix[row][col] = temp[r];
                                col++;
                            }
                            if(orientation.equals("d")){
                                matrix[row][col] = temp[r];
                                row++;
                            }
                        }
                        counter++;

                    }
                }
            }
        }
```

setBase(): This is the method that checks whether the words in the found candidates pool correspond the the actual answer. We return once we have a base of two words. Notice that we do not explicitly take two words from answers but rather check whether the words we found correspond to the actual answers. After we locate and put the answers as our base we remove the candidate list for the corresponding word. The object Answer which is separate from the word object that stores the “word so far” is used to check with actual answer

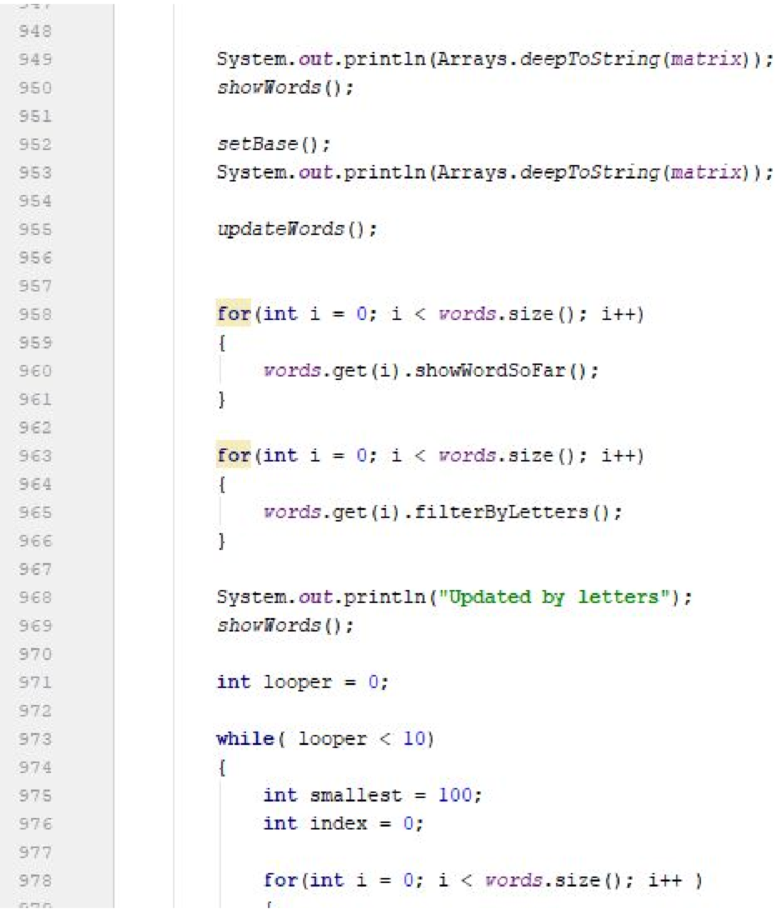


```
public static boolean updateMatrix( Word word, int index)
{
    boolean changed = false;
    String orientation = word.getOrientation();
    int[] strPoint = word.getStartingPoint();
    int row = strPoint[0], col = strPoint[1];
    for(int r = 0; r < word.getWordLength(); r++){
        if(orientation.equals("a")) {
            if(matrix[row][col].equals("black") || matrix[row][col].equals("white") || (matrix[row][col] == null) || Char
                matrix[row][col] = Character.toString(word.getCandidates().get(index).substring(r).charAt(0));

                changed = true;
            }
            col++;

        }

        if(orientation.equals("d")){
            if(matrix[row][col].equals("black") || matrix[row][col].equals("white") || (matrix[row][col] == null) || Char
            {
            matrix[row][col] = Character.toString(word.getCandidates().get(index).substring(r).charAt(0));

            changed = true;
            }
            row++;
        }
    }

    return changed;

}
```

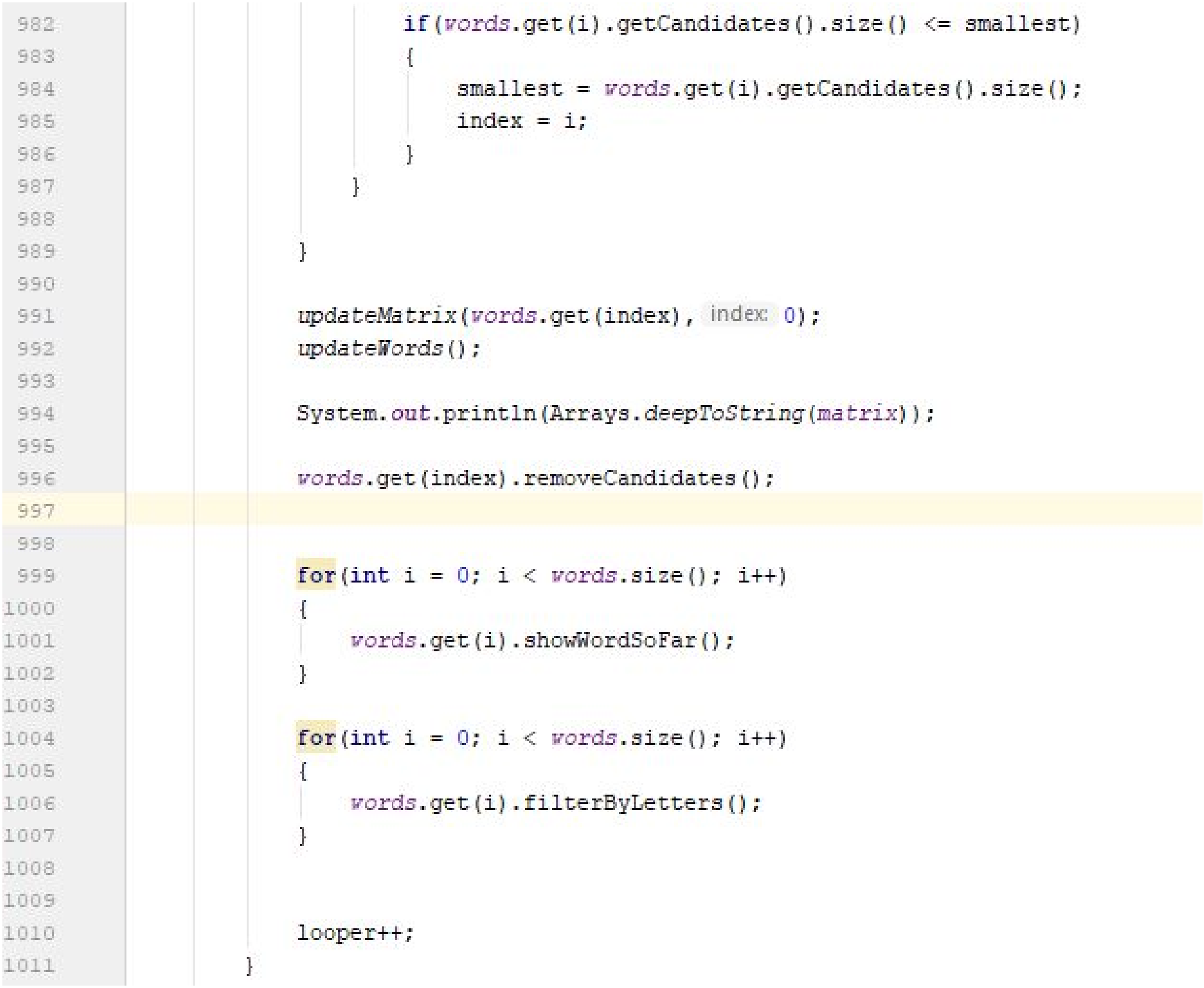
updateMatrix(): This is the method that updates the main matrix with the letters found so far using the specified word. As can be seen it takes the values such as orientation from the word object, starting point of this word and puts the the given candidate according to those values and given constraints. This method is used to proceed with the selected candidates.



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


    start( input: "F2");
    putColors();
    putNumbers();
    System.out.println(Arrays.deepToString(matrix));
    createWords();


    putClues();
    initAnswerTable();
    createAnswers();

    System.out.println(Arrays.deepToString(answerTable));
    showAnswers();

    for(int i = 0; i < words.size(); i++)
    {
        words.get(i).googleSearch();
        words.get(i).wikiSearch();
        //words.get(i).trackerSearch();
        words.get(i).filterBySize();
        words.get(i).filterByLetters();

    }
```



```java

System.out.println(Arrays.deepToString(matrix));
showWords();

setBase();
System.out.println(Arrays.deepToString(matrix));

updateWords();


for(int i = 0; i < words.size(); i++)
{
    words.get(i).showWordSoFar();
}

for(int i = 0; i < words.size(); i++)
{
    words.get(i).filterByLetters();
}

System.out.println("Updated by letters");
showWords();

int looper = 0;

while( looper < 10)
{
    int smallest = 100;
    int index = 0;

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



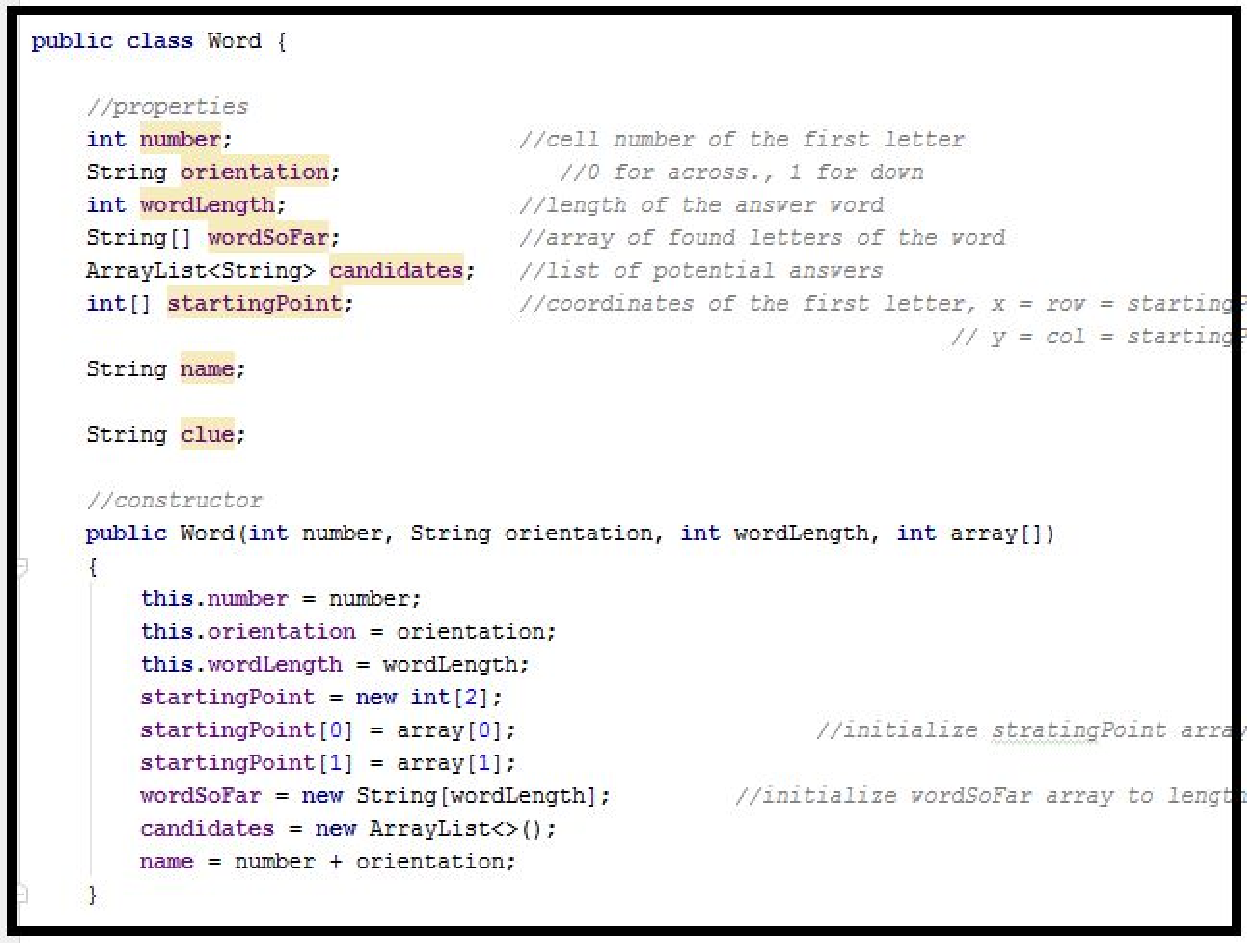
```
                    if(words.get(i).getCandidates().size() <= smallest)
                    {
                        smallest = words.get(i).getCandidates().size();
                        index = i;
                    }
                }

            }

            updateMatrix(words.get(index), index: 0);
            updateWords();

            System.out.println(Arrays.deepToString(matrix));

            words.get(index).removeCandidates();


            for(int i = 0; i < words.size(); i++)
            {
                words.get(i).showWordSoFar();
            }

            for(int i = 0; i < words.size(); i++)
            {
                words.get(i).filterByLetters();
            }


            looper++;
        }
```

The last three screenshots are the central to actually solving the puzzle. We will review it sequentially:

1. It first initializes the required crossword by choosing from the ones stored on the machine.
2. Then it puts numbers, colors to the main matrix, which can be referred as progress matrix from now on. Also the main method creates word objects, and put the corresponding clues in them.
3. Then it proceeds to initialize the answer table that is used to set base case for our crossword by creating Answer objects
4. Further, when we enter the for loop we initiate the searches for the candidates of each words. While googleSeacrh() and wikiSearch() are descriptive in the naming, trackerSearch() is obtains the candidates by querying the database that has associations for the words in the clue, once again to underline we do not use this method initially and in our test cases we indicate when we use this search and when we do not, sometimes it

gives fruitful results especially when we are in need of extra candidates where our filterByLetter() method removes all of the candidates since they do not satisfy the constraint, but still we do not include it in a general case but rather present it as an additional case. It should be seen in the examples that sometimes trackerSearch() actually does not create better results and actually messes with the potentially correct candidates by introducing large amount of candidates.

5. Further in a "for loop" words which contain corresponding candidates get filtered out from the unsatisfying words.
6. Then we set the base two words by checking whether the words we found actually correspond to the answer, note that at no time the user can see the actual answer and nothing gets hard coded in terms of actual answers. It is implemented in a way of black box where even Word object do not store the information about its actual answer.
7. Following two for loops are for the progress acknowledgement and to further filter out the words by unsatisfying letters.
8. Then we enter a for loop where we continuously choose the next word to be discovered to be the one with least amount of candidates. When it locates such word it picks up the upper candidate and proceeds by updating the matrix accordingly and the words' "wordSoFar" gets updated as well. This process continues 10 times at the longest since we only have 10 words to be discovered in crossword.



```
public class Word {

    //properties
    int number;                        //cell number of the first letter
    String orientation;                   //0 for across., 1 for down
    int wordLength;                    //length of the answer word
    String[] wordSoFar;                //array of found letters of the word
    ArrayList<String> candidates;      //list of potential answers
    int[] startingPoint;               //coordinates of the first letter, x = row = startingP
                                                                // y = col = startingP

    String name;

    String clue;

    //constructor
    public Word(int number, String orientation, int wordLength, int array[])
    {
        this.number = number;
        this.orientation = orientation;
        this.wordLength = wordLength;
        startingPoint = new int[2];
        startingPoint[0] = array[0];                //initialize stratingPoint arra
        startingPoint[1] = array[1];
        wordSoFar = new String[wordLength];        //initialize wordSoFar array to lengt
        candidates = new ArrayList<>();
        name = number + orientation;
    }
```

Word: This is the object aimed to hold the every item of crossword with its properties such as number, orientation, word's length, starting coordinate, candidates, name and word so far.

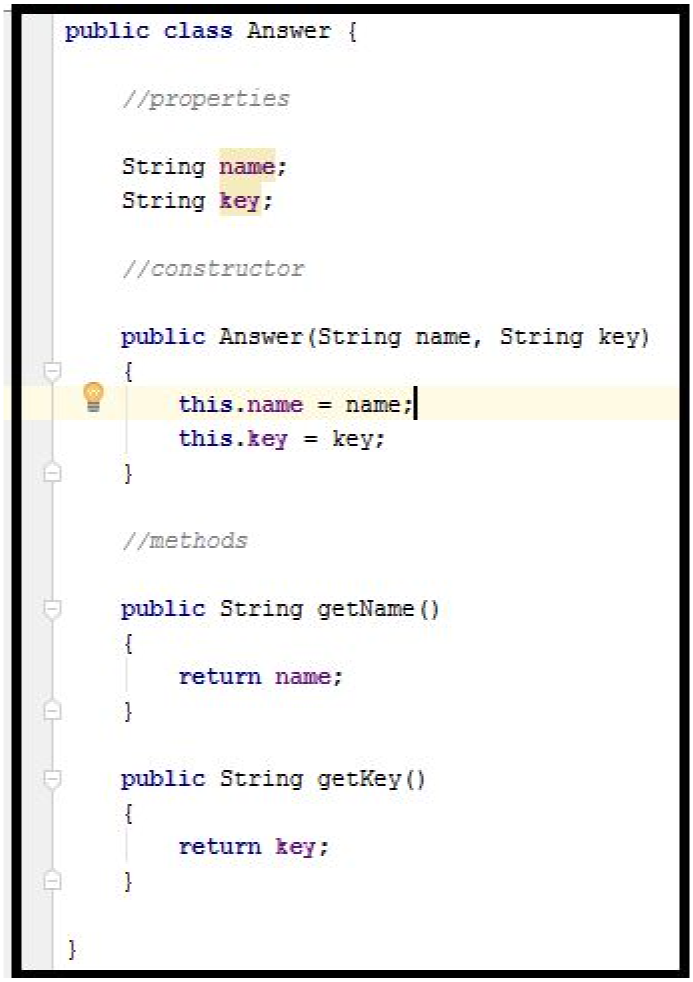
googleSearch(): This method gives the all the candidates from google webpage for its word object. Additionally, it filters the candidates according to their size which has to be smaller than six and removes duplicates.

wikiSearch(): This method gives the all the candidates from wikipedia webpage for its word object. Additionally, it filters the candidates according to their size which has to be smaller than six and removes Duplicates.

trackerSearch(): This method gives the all the candidates from crosswordtracker webpage for its word object. Additionally, it filters the candidates according to their size which has to be smaller than six and removes duplicates.

filterByLetters(): This method filters the corresponding word's candidates according to words are find so far in crossword by checking every word's founded letters.

getWords(): This method returns the array of string which holds the letters of given input parameter string.



```
public class Answer {

    //properties

    String name;
    String key;

    //constructor

    public Answer(String name, String key)
    {
        this.name = name;
        this.key = key;
    }

    //methods

    public String getName()
    {
        return name;
    }

    public String getKey()
    {
        return key;
    }

}
```

Answer: This object holds the actual answers.

-name: it is for finding its proper crossword

-key: the actual answer.

- **GUI Components**

In total there are three gui classes that handle the graphical side of the program. They are namely, GridCell, AnswerCell and CrosswordGrid.

GridCell

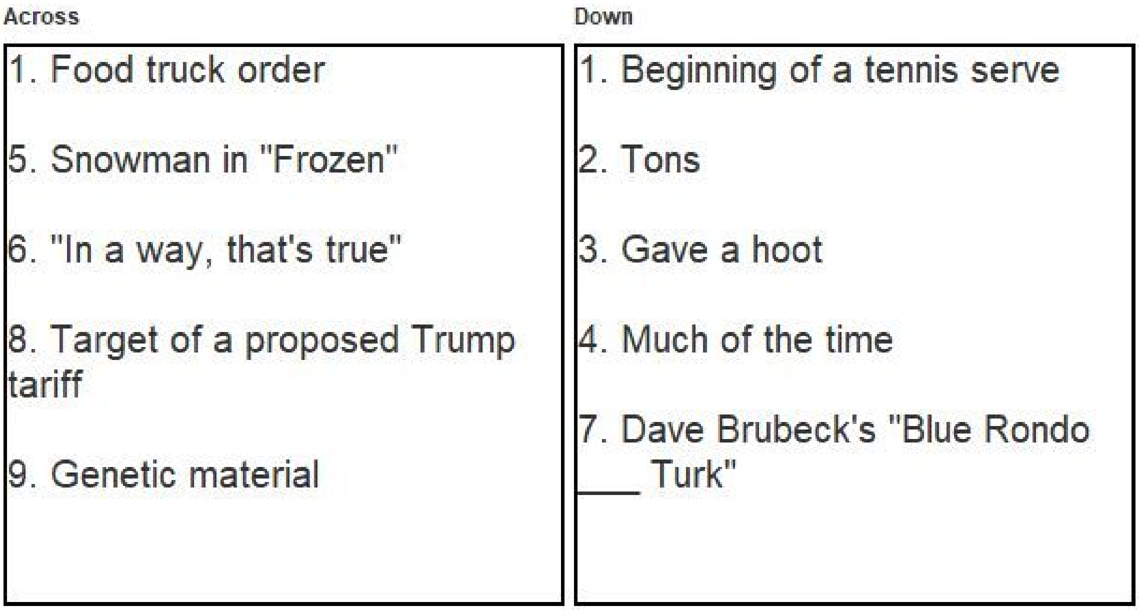
GridCell corresponds to the white and black, numbered and unnumbered boxes of the puzzle. In total, the program initializes twenty five grid cells just as there are twenty five cells in the crossword. The GridCell class extends JPanel, which is composed of JLabel and JTextField. The JLabel is used for the number on the cell, if a number is present on the cell while JTextField is used to take an input from the users and as well as output the correct letters when the program manages to find them. GridCell also has the data member int isBlack which informs the program that this GridCell background should be black, the setEditable() should be set to “false” and no numbers should be displayed if isBlack is 0.

AnswerCell

AnswerCell is essentially a stripped down version of GridCell. Like GridCell twenty five of them are initialised and they are used to make the answer grid. The answer grid contains the answer that have been taken from the NYT website. Answer grid is identical to the crossword puzzle except that the answers are shown and the answer cells are uneditable.

CrosswordGrid

CrosswordGrid is by far the most important class out of all gui classes. It is responsible for making the crossword puzzle from the GridCell objects and the answer grid from the AnswerCell objects. In addition to this, CrosswordGrid is responsible for displaying the clues and as well as the log. The log is a continuous stream of messages that details what the program has been doing at the moment. When the program finally finds a solution, an array of answers is sent to CrosswordGrid class. The CrosswordGrid displays these answers on the puzzle.



Across

1. Food truck order

5. Snowman in "Frozen"

6. "In a way, that's true"

8. Target of a proposed Trump tariff

9. Genetic material

Down

1. Beginning of a tennis serve

2. Tons

3. Gave a hoot

4. Much of the time

7. Dave Brubeck's "Blue Rondo ___ Turk"

Fig. Clues as visible in the program

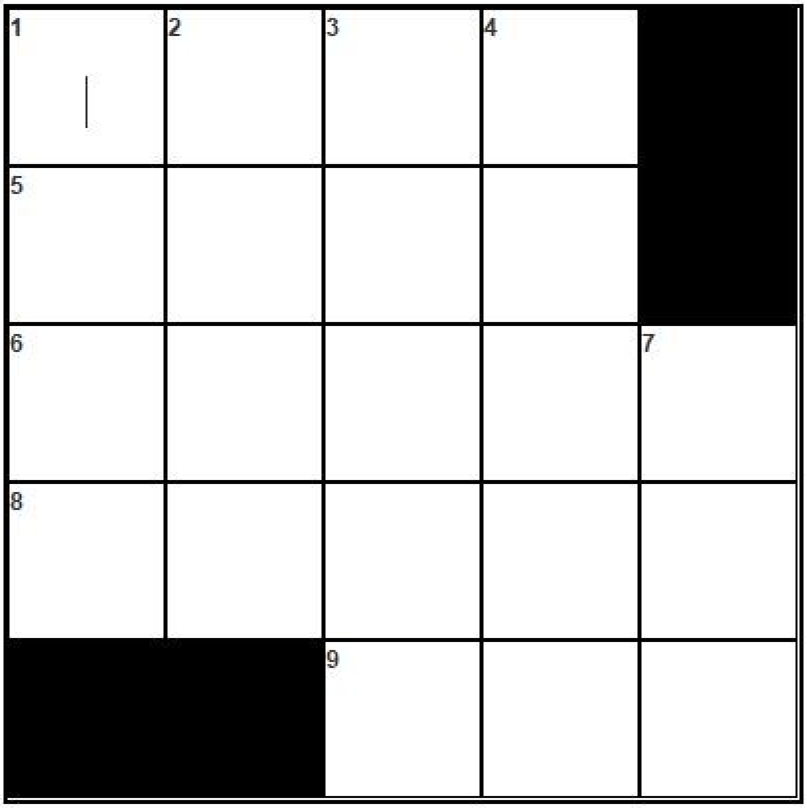
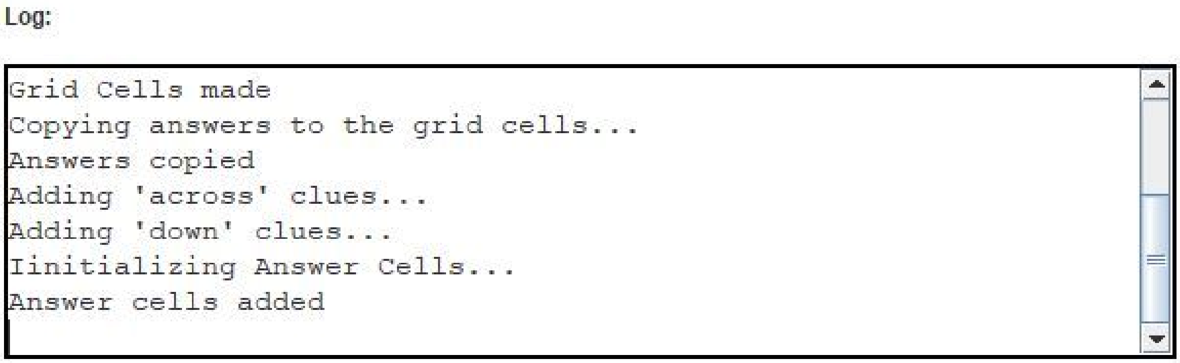
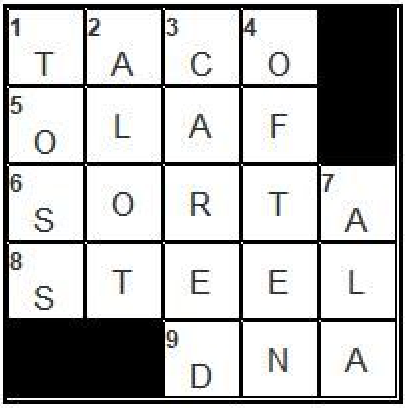


Fig. The crossword puzzle



```
Grid Cells made
Copying answers to the grid cells...
Answers copied
Adding 'across' clues...
Adding 'down' clues...
Iinitializing Answer Cells...
Answer cells added
```

Fig. The log as it appears during program run



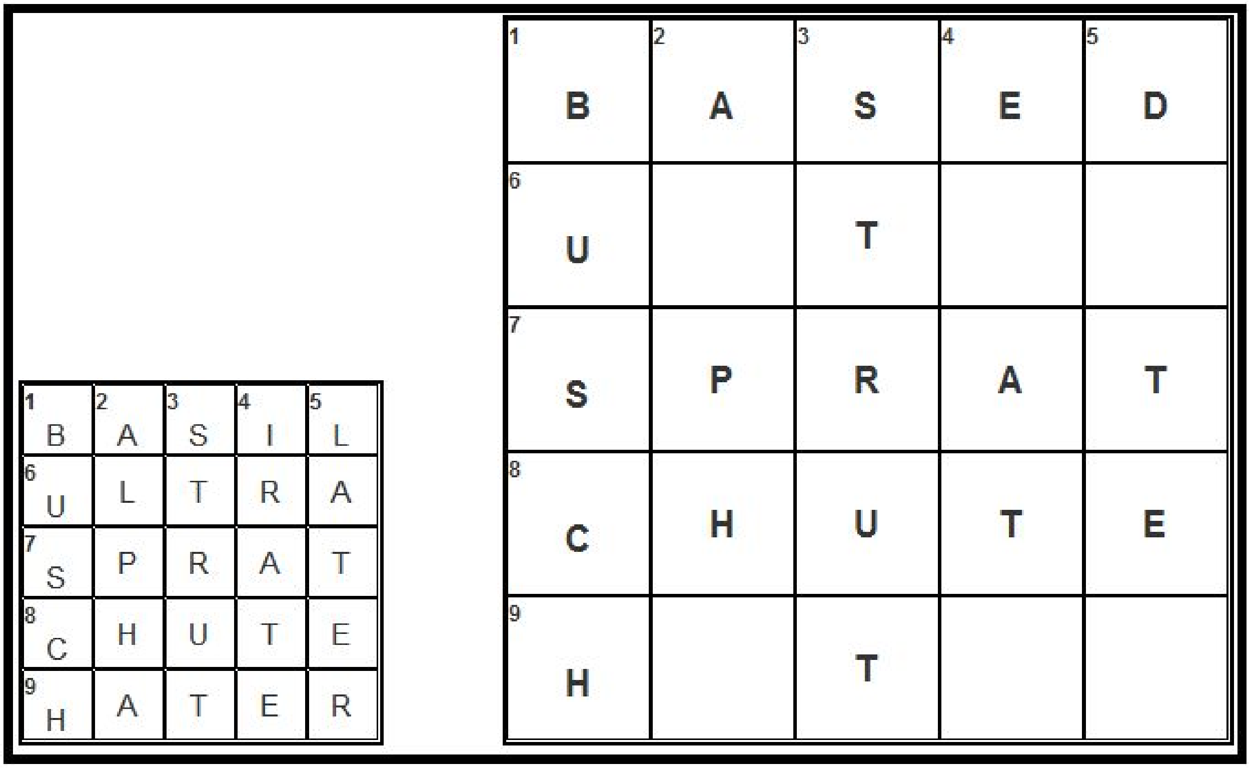
1 T	2 A	3 C	4 O	■
5 O	L	A	F	■
6 S	O	R	T	7 A
8 S	T	E	E	L
■	■	9 D	N	A

Fig. Answer grid shown

Test results:

Below are some of the results that we found.

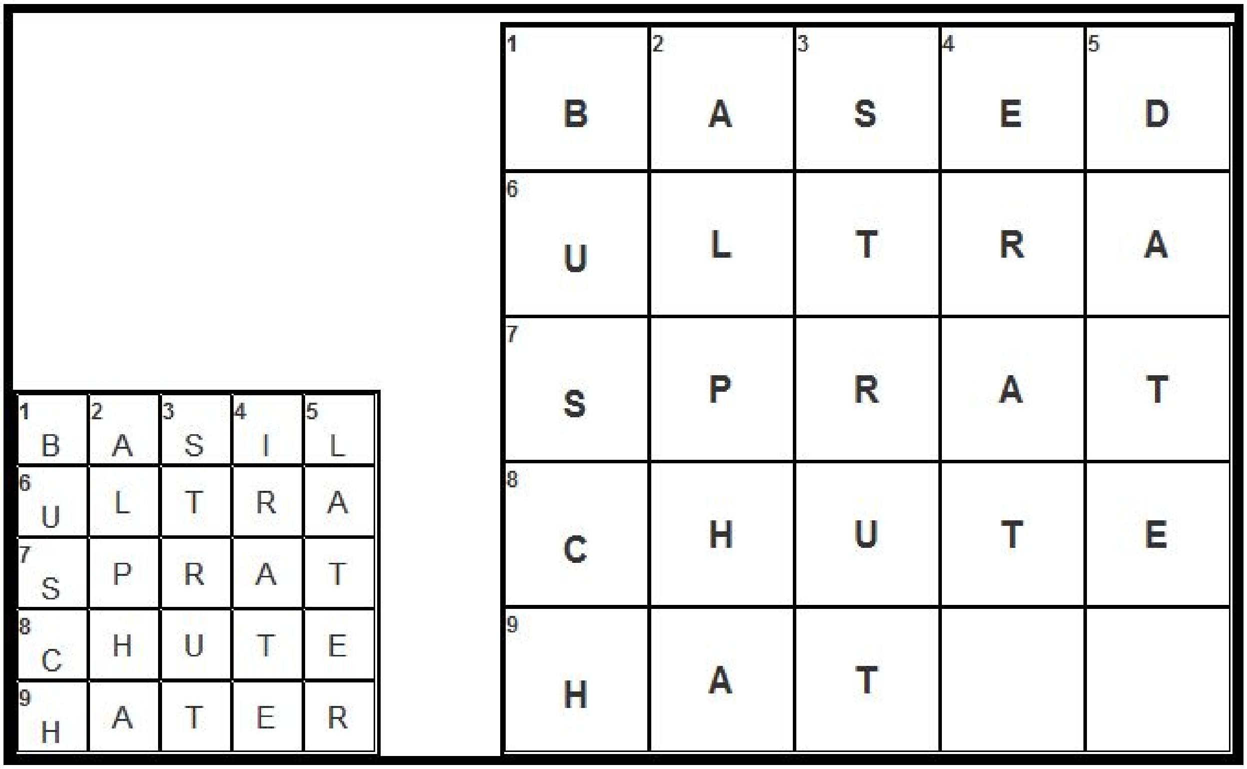
7th February-2018:



1 B	2 A	3 S	4 I	5 L
6 U	L	T	R	A
7 S	P	R	A	T
8 C	H	U	T	E
9 H	A	T	E	R

1 B	2 A	3 S	4 E	5 D
6 U		T		
7 S	P	R	A	T
8 C	H	U	T	E
9 H		T		

Attempt # 1: The bigger grid shows the crossword puzzle while the smaller one shows the answers that were pulled down from the NYT website. The program managed to find around four correct answers in this instance. It should be noted that the program did not fail to identify answers from the candidate list but rather the google searches did not lead to the correct answers and hence, the correct answers were not available in the candidate list.

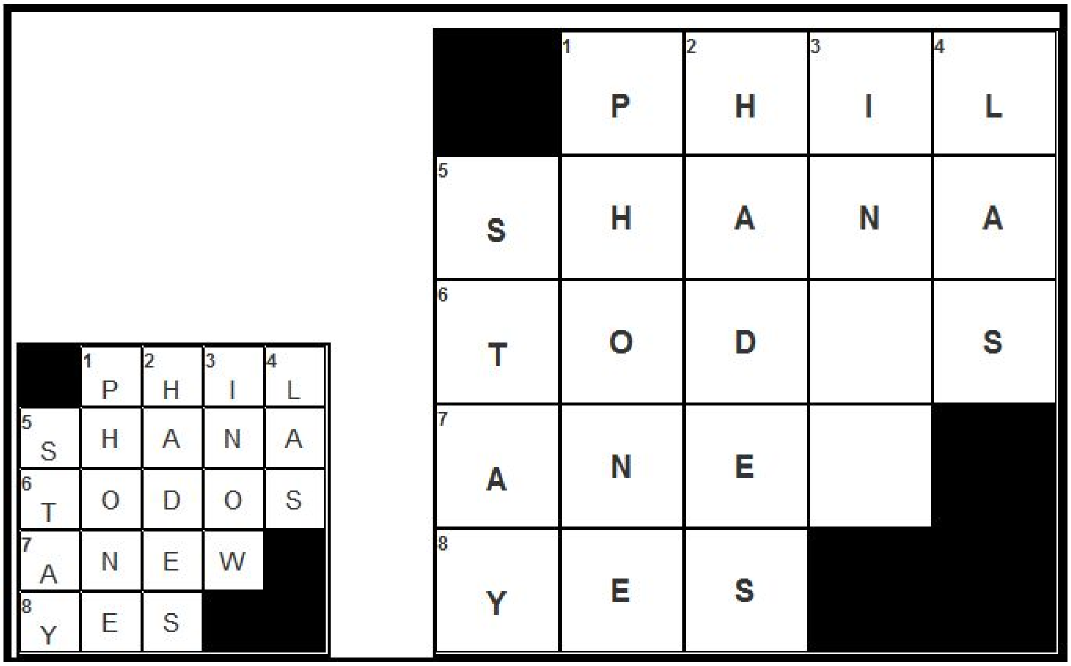


1 B	2 A	3 S	4 I	5 L
6 U	L	T	R	A
7 S	P	R	A	T
8 C	H	U	T	E
9 H	A	T	E	R

1 B	2 A	3 S	4 E	5 D
6 U	L	T	R	A
7 S	P	R	A	T
8 C	H	U	T	E
9 H	A	T		

Attempt # 2: The above result are again for 7th february-2018. This time however, the search includes trackerSeacrh(), trackerSearch() as mentioned earlier is a database of associations of words in the clues. Addition of tracker search increases the success rate from 4 correct to 6 correct answers.

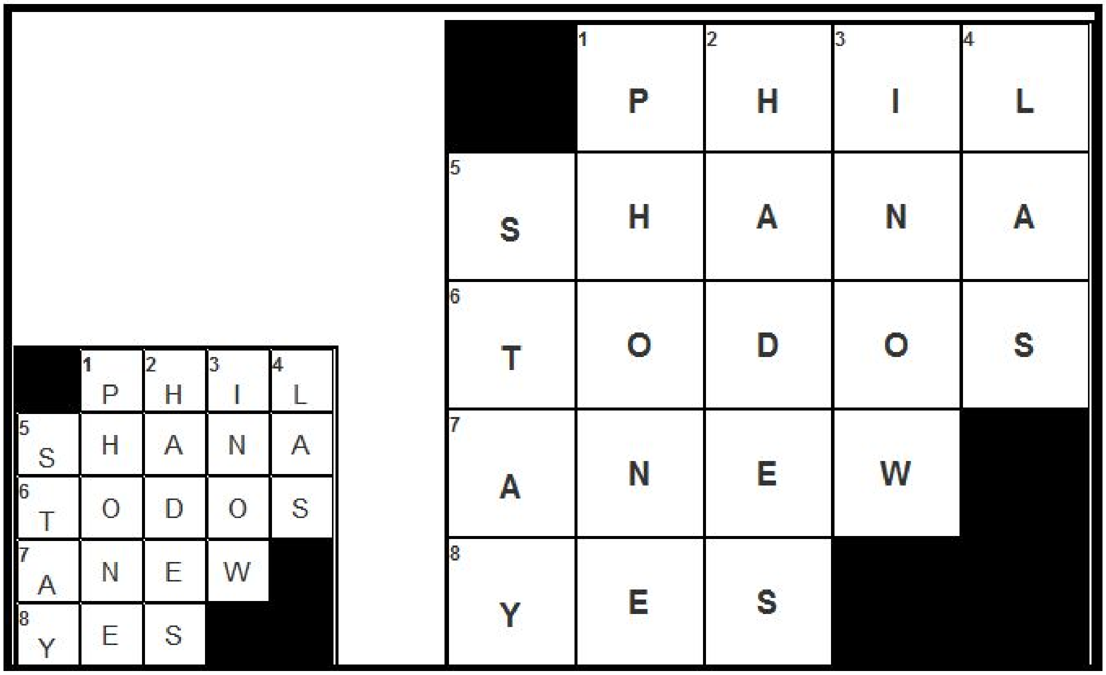
2nd February-2018



	1 P	2 H	3 I	4 L
5 S	H	A	N	A
6 T	O	D	O	S
7 A	N	E	W	
8 Y	E	S		

	1 P	2 H	3 I	4 L
5 S	H	A	N	A
6 T	O	D		S
7 A	N	E		
8 Y	E	S		

Attempt # 1: The 2nd February try has been the program's most successful one, as it managed to get 7 correct answers. The only roadblock happened to be a sentence "I Know" which our algorithms would treat as a separate sentence. Important to note, this was done without tracker search.

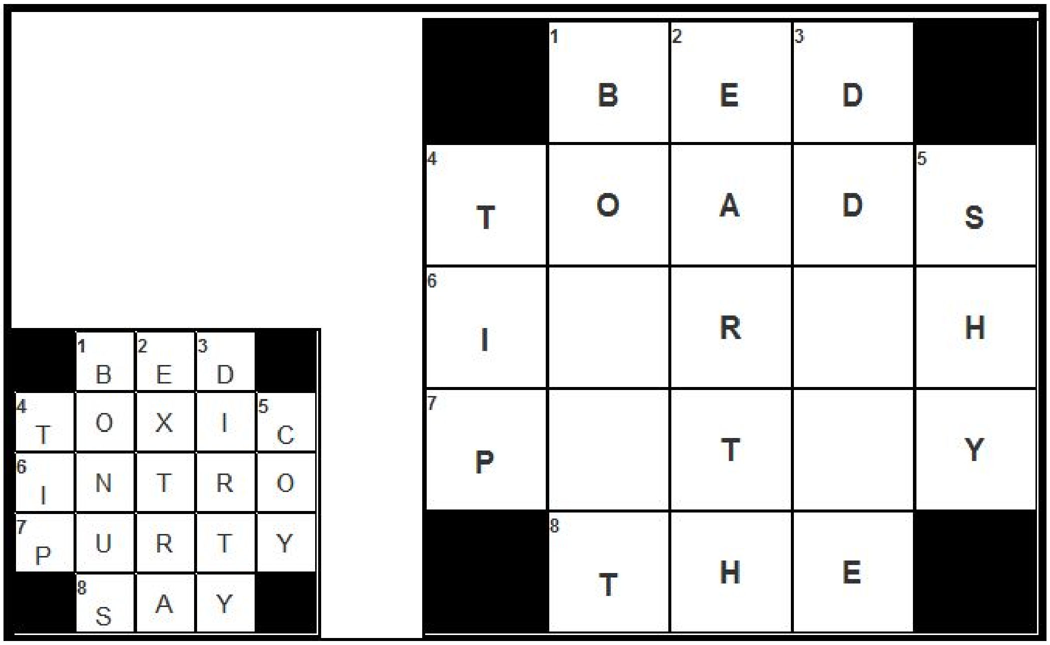


	1 P	2 H	3 I	4 L
5 S	H	A	N	A
6 T	O	D	O	S
7 A	N	E	W	
8 Y	E	S		

	1 P	2 H	3 I	4 L
5 S	H	A	N	A
6 T	O	D	O	S
7 A	N	E	W	
8 Y	E	S		

Attempt # 2: With the scraping of tracker search, the success rate jumps to 10 out of 10. Although tracker search introduces a lot of new candidates, because the algorithm is able to solve the puzzle successfully it thus imposes better constraints to choose from the candidates returned by the trackerSearch(), The attempts shows that the program does not misses the correct answer if it is present in the list of correct business

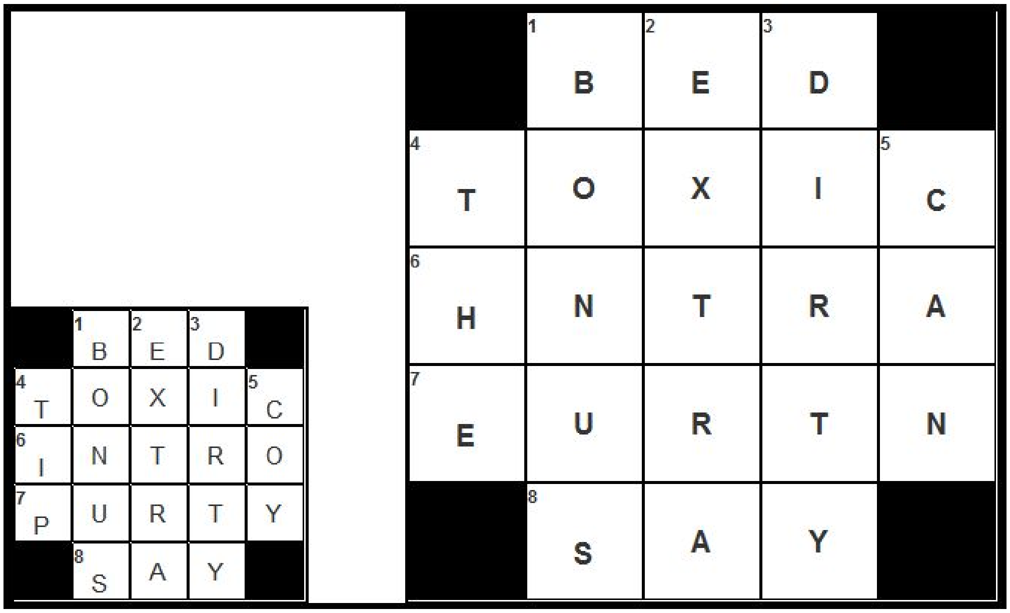
3rd January-2018



■	1 B	2 E	3 D	■
4 T	O	X	I	5 C
6 I	N	T	R	O
7 P	U	R	T	Y
■	8 S	A	Y	■

■	1 B	2 E	3 D	■
4 T	O	A	D	5 S
6 I		R		H
7 P		T		Y
■	8 T	H	E	■

Attempt # 1: The 3rd of January try is the worst result we have gotten so far. The program only managed to get two correct words.

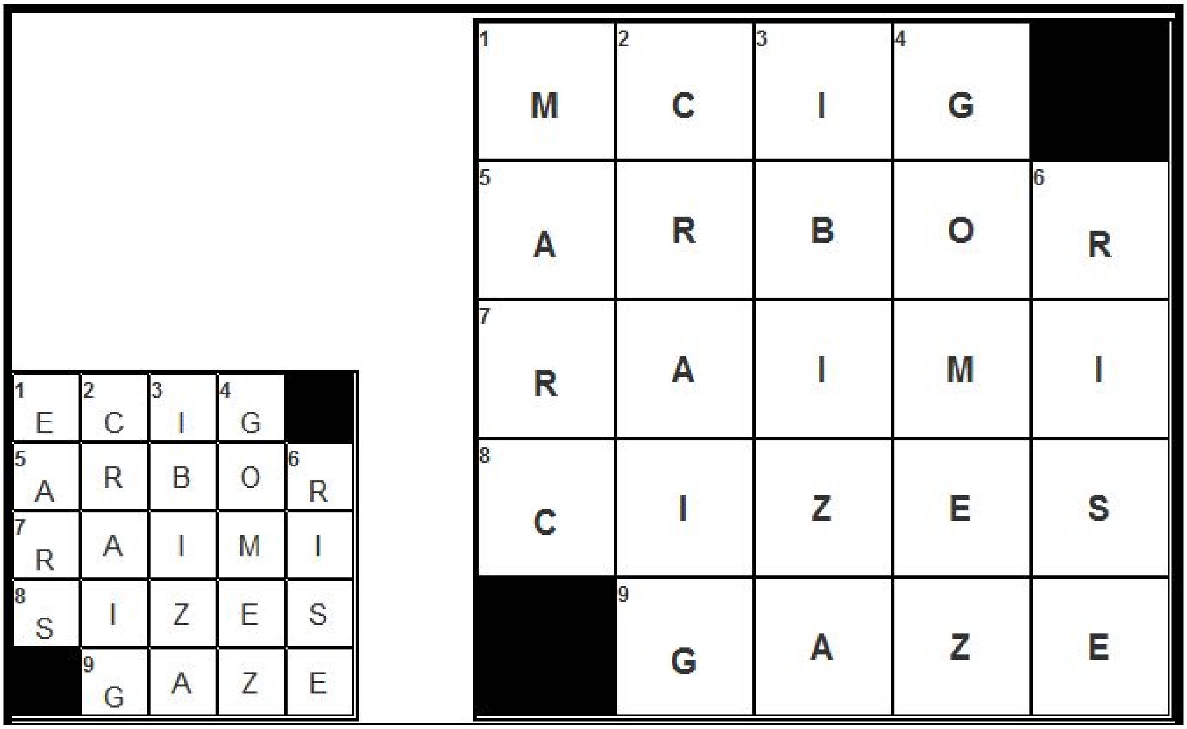


■	1 B	2 E	3 D	■
4 T	O	X	I	5 C
6 I	N	T	R	O
7 P	U	R	T	Y
■	8 S	A	Y	■

■	1 B	2 E	3 D	■
4 T	O	X	I	5 C
6 H	N	T	R	A
7 E	U	R	T	N
■	8 S	A	Y	■

Attempt # 2: With the tracker search the success rate increases threefold. Instead of two we get 6 correct answers. Explanation is that our algorithm that choose the candidates with the least number of candidates chose the wrong word here or simply our pool did not contain correct candidates and just proceeded with what it had - poor candidate values.

13 April 2015



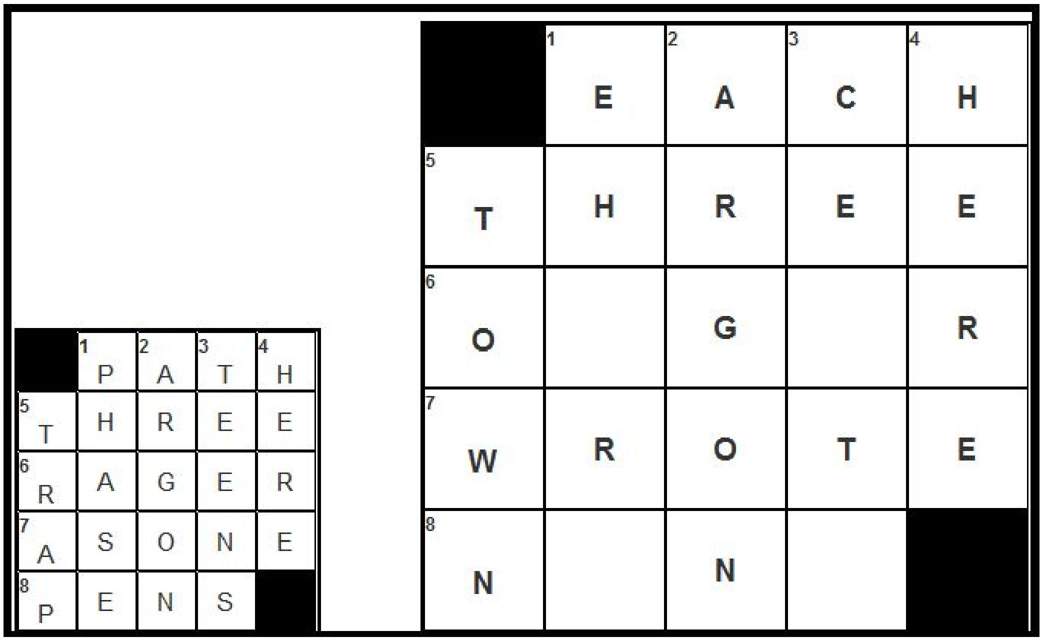
1 E	2 C	3 I	4 G	■
5 A	R	B	O	6 R
7 R	A	I	M	I
8 S	I	Z	E	S
■	9 G	A	Z	E

1 M	2 C	3 I	4 G	■
5 A	R	B	O	6 R
7 R	A	I	M	I
8 C	I	Z	E	S
■	9 G	A	Z	E

Attempt #1, 2: With the above two, switching to tracker search makes no difference. Even the wrong answers are the same. Possibly that Tracker Search did not introduce any candidates or they it had similar potentially good candidates

Tuesday August 16, 2016

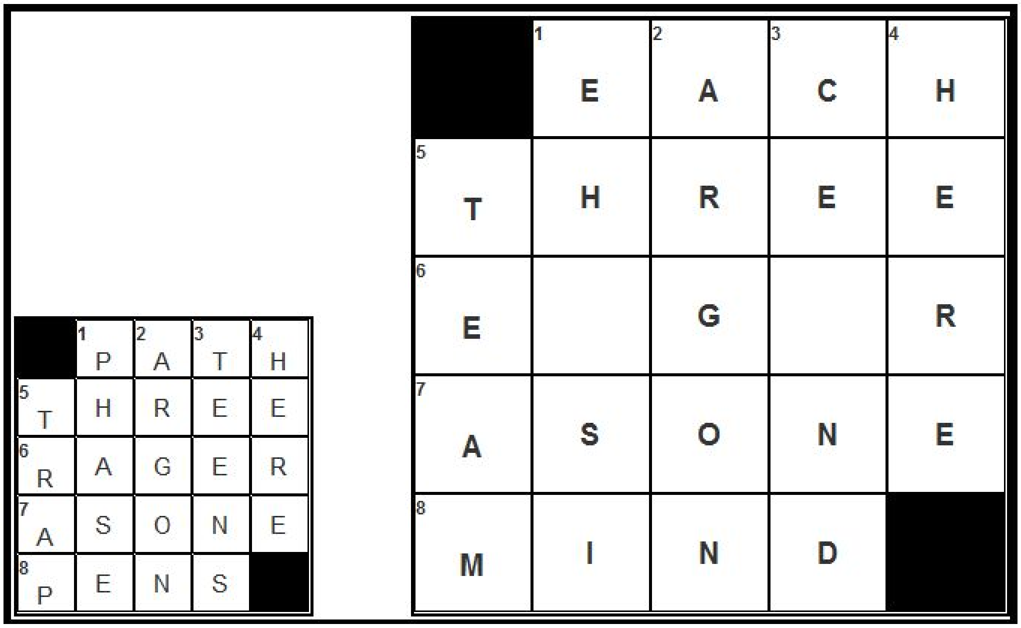
Attempt # 1: usual search



■	1 P	2 A	3 T	4 H
5 T	H	R	E	E
6 R	A	G	E	R
7 A	S	O	N	E
8 P	E	N	S	■

■	1 E	2 A	3 C	4 H
5 T	H	R	E	E
6 O		G		R
7 W	R	O	T	E
8 N		N		■

Attempt # 2: tracker search

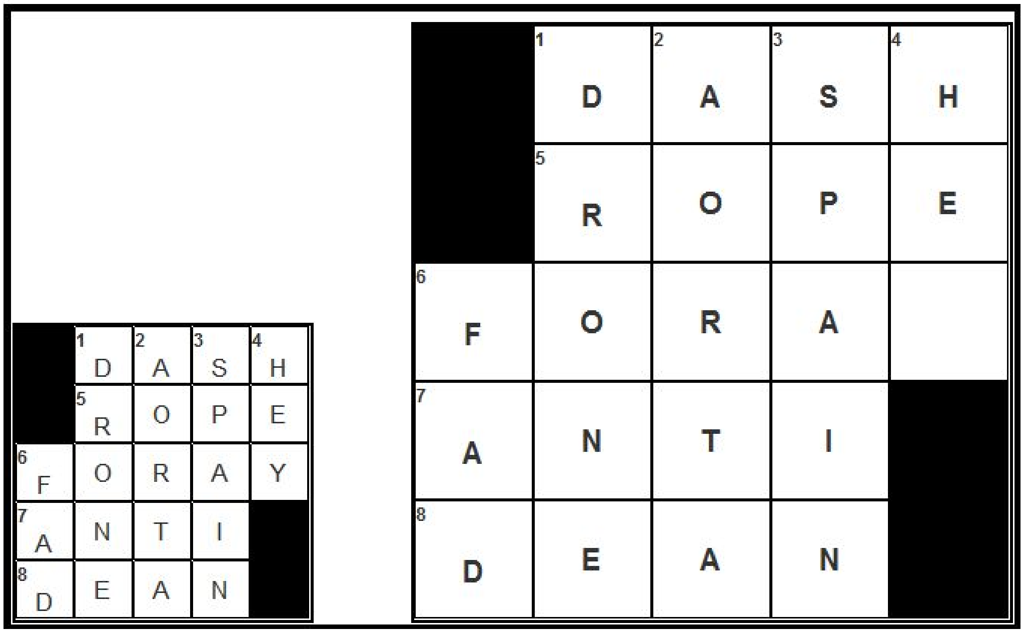


■	1 P	2 A	3 T	4 H
5 T	H	R	E	E
6 R	A	G	E	R
7 A	S	O	N	E
8 P	E	N	S	■

■	1 E	2 A	3 C	4 H
5 T	H	R	E	E
6 E		G		R
7 A	S	O	N	E
8 M	I	N	D	■

October 30, 2017

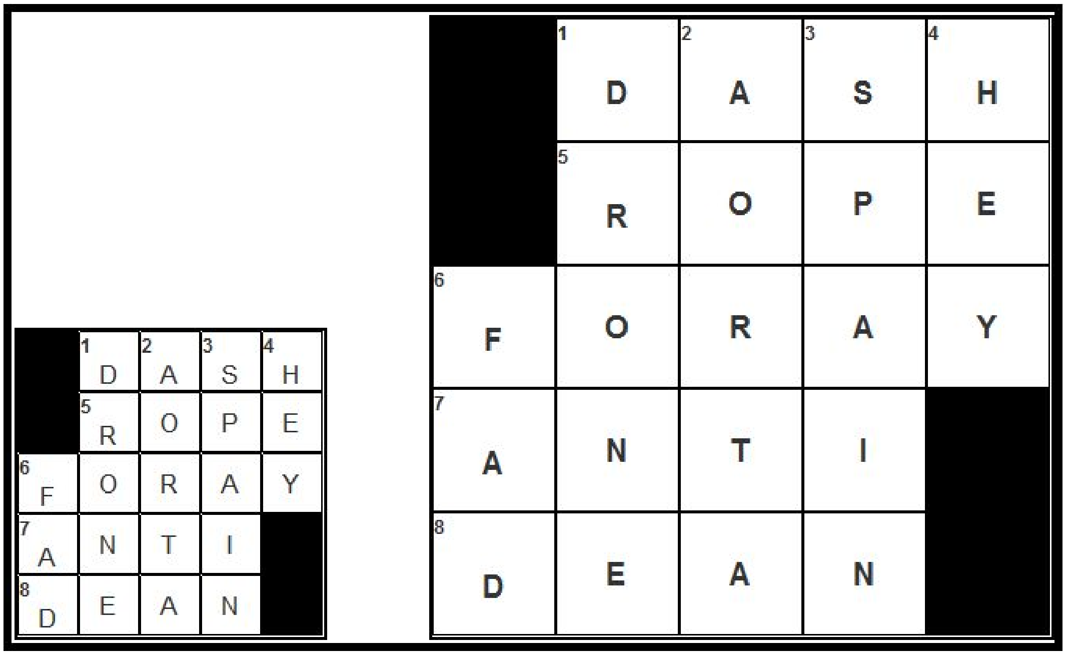
Attempt # 1: usual search



■	1 D	2 A	3 S	4 H
■	5 R	O	P	E
6 F	O	R	A	Y
7 A	N	T	I	■
8 D	E	A	N	■

■	1 D	2 A	3 S	4 H
■	5 R	O	P	E
6 F	O	R	A	
7 A	N	T	I	■
8 D	E	A	N	■

Attempt # 2: tracker search



■	1 D	2 A	3 S	4 H
■	5 R	O	P	E
6 F	O	R	A	Y
7 A	N	T	I	■
8 D	E	A	N	■

■	1 D	2 A	3 S	4 H
■	5 R	O	P	E
6 F	O	R	A	Y
7 A	N	T	I	■
8 D	E	A	N	■

Word Count:2043

This project reports work done in partial fulfillment of the requirements for the course CS 461 - Artificial Intelligence. The software developed in this project is, to a large extent, original (with borrowed code clearly identified). It was written solely by the members of the project group and is not simultaneously being submitted to another course.

Appendix - Code

```
import org.jsoup.Jsoup;
import org.jsoup.nodes.Document;
import org.jsoup.nodes.Element;
import org.jsoup.select.Elements;

import java.text.BreakIterator;
import java.util.*;


import javax.swing.*;
import java.awt.*;
import java.awt.event.ActionEvent;
import java.awt.event.ActionListener;
import java.io.*;
//import java.net.HttpURLConnection;
import java.net.URL;

//import java.net.URISyntaxException;



public class Parser {
```

```
private static int CWIDTH = 5;
private static int CHEIGHT = 5;

static String html;
static String url = "https://www.nytimes.com/crosswords/game/mini";
static  String file;

static String matrix[][] = new String[5][5];

static String answerTable[][] = new String[5][5];

private static ArrayList<Word> words;

private static ArrayList<Answer> answers;

public static void start(String input){
   html = "";
   String file = "crosswords/" + input + ".html";
   try (BufferedReader br = new BufferedReader(new FileReader(file))) {
      String sCurrentLine;

      while ((sCurrentLine = br.readLine()) != null) {
         html += sCurrentLine;
      }
   }

   catch (IOException e) {
      e.printStackTrace();

   }

}




public static ArrayList<String> getClues() {
    ArrayList<String> clues = new ArrayList<>();

    Document doc;


    try{

            doc = Jsoup.parse(html);


        Elements elems = doc.getElementsByClass("Clue-text--3lZl7");
        Elements elems2 = doc.getElementsByClass("Clue-label--2IdMY");
        for(int i = 0; i < elems.size(); i++){
            clues.add(elems2.get(i).text() + ". " + elems.get(i).text());
        }
    }
    catch (Exception e)
    {
        e.printStackTrace();
    }



    return clues;

}


public static ArrayList<Integer> getColors(){

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

    Document doc;

    try{

            doc = Jsoup.parse(html);

        Elements elems = doc.select("g[data-group=\"cells\"]");
        elems = elems.get(0).getElementsByTag("rect");
        for (int i = 0; i < elems.size(); i++) {

            if (elems.get(i).className().contains("Cell-cell--1p4gH" ))
            {
                colors.add(1);
            }
            else
            {
                colors.add(0);
            }
        }
    }
    catch (Exception e)
    {
        e.printStackTrace();
    }



    return colors;

}



public static ArrayList<String> getNumbers(){

    ArrayList<String> numbers = new ArrayList<>();

    Document doc;


    try
    {

            doc = Jsoup.parse(html);
```

```
  Elements elems = doc.select("g[data-group=\"cells\"]");

  elems = elems.get(0).getElementsByTag("g");
  for (int i = 1; i < elems.size(); i++) {
    Element el = elems.get(i).attr("text-anchor","start");
    if(el.text().length() > 1)
    {
      for(int j = 0; j < el.text().length(); j++)
      {
        if (Character.isDigit(el.text().charAt(j)))
        {
          numbers.add("" + el.text().charAt(j));
        }

      }
    }
    else if (el.text().length() == 1)
    {
      if (Character.isDigit(el.text().charAt(0)))
      {
        numbers.add("" + el.text().charAt(0));
      }
      else
      {
        numbers.add("");
      }
    }
    else
    {
      numbers.add(""+ el.text());
    }

  }
}
catch (Exception e)
{
  e.printStackTrace();
}
```

```
    return numbers;

}


public static ArrayList<String> getAnswers() {

    ArrayList<String> numbers = new ArrayList<>();

    Document doc = Jsoup.parse(html);
    Elements elems = doc.select("g[data-group=\"cells\"]");

    elems = elems.get(0).getElementsByTag("g");
    // System.out.println("Before iteration");
    for (int i = 1; i < elems.size(); i++) {
        Element el = elems.get(i).attr("text-anchor","start");
        if(el.text().length() > 1)
        {
            for(int j = 0; j < el.text().length(); j++)
            {
                if (Character.isLetter(el.text().charAt(j)))
                {
                    numbers.add(String.valueOf(el.text().charAt(j)));
                }

            }
        }
        else if (el.text().length() == 1)
        {
            if (Character.isLetter(el.text().charAt(0)))
            {
                numbers.add(String.valueOf(el.text().charAt(0)));
            }
            else
            {
                numbers.add("");
            }
        }
        else
        {
            numbers.add("");
        }
    }

        /*   } catch (IOException e) {
                e.printStackTrace();

            } */
    return numbers;
}

public static ArrayList<String> getWords(String text) {
    ArrayList<String> words = new ArrayList<>();
    BreakIterator breakIterator = BreakIterator.getWordInstance();
    breakIterator.setText(text);
    int lastIndex = breakIterator.first();
    while (BreakIterator.DONE != lastIndex) {
        int firstIndex = lastIndex;
        lastIndex = breakIterator.next();
        if (lastIndex != BreakIterator.DONE &&
Character.isLetterOrDigit(text.charAt(firstIndex))) {
            words.add(text.substring(firstIndex, lastIndex));
        }
    }

    return words;
}


public static void putColors()
{

    int count = 0;

    ArrayList<Integer> colors;
    colors = getColors();
    for (int i = 0; i < CWIDTH; i++)
    {
        for(int j = 0; j< CHEIGHT; j++)
        {
            if(colors.get(count) == 0)
                matrix[i][j] = "black";
            else
                matrix[i][j] = "white";

            count++;
        }
    }

}

public static void putNumbers()
{

    int count = 0;

    ArrayList<String> numbers;
    numbers = getNumbers();

    for (int i = 0; i < CWIDTH; i++)
    {
        for(int j = 0; j< CHEIGHT; j++)
        {
            if(numbers.get(count).length()>0)
                matrix[i][j] = numbers.get(count);

            count++;
        }
    }
}

public static void createWords(){
    words = new ArrayList<>();
    boolean enter = true;
    int number = 0;
```

```
int[] StartingPoint = new int[2];
String orientation = "a";
int wordLength;
//for row
for(int i = 0; i < 5; i++) {
    wordLength = 5;
    for(int j = 0; j < 5; j++){


        if(!matrix[i][j].equals("black") && enter) {
            number = Integer.valueOf(matrix[i][j]);
            StartingPoint[0] = i;
            StartingPoint[1] = j;
            enter = false;
        }
        if(matrix[i][j].equals("black"))
            wordLength--;

    }
    enter = true;
    Word word = new Word(number, orientation, wordLength, StartingPoint);
    words.add(word);

}

orientation = "d";
//for column
for(int i = 0; i < 5; i++) {
    wordLength = 5;
    for(int j = 0; j < 5; j++){

        if(!matrix[j][i].equals("black") && enter) {
            number = Integer.valueOf(matrix[j][i]);
            StartingPoint[0] = j;
            StartingPoint[1] = i;
            enter = false;

        }
        if(matrix[j][i].equals("black"))
```

```
                    wordLength--;


                }
                enter = true;
                Word word = new Word(number, orientation, wordLength, StartingPoint);
                words.add(word);
            }
        }

    public static void updateWords(){

        int[] StartingPoint;
        int wordLength;
        int k = 0;

            //for row
            for(int i = 0; i < 5; i++) {
                StartingPoint = words.get(k).getStartingPoint();
                wordLength = words.get(k).getWordLength();
                for(int j = StartingPoint[1]; j < wordLength + + StartingPoint[1]; j++){
                    if(!matrix[i][j].equals("black") && !matrix[i][j].equals("white") && !(matrix[i][j]
== null) && !Character.isDigit(matrix[i][j].charAt(0)) ) {
                        words.get(k).updateWordSoFar(matrix[i][j], j - StartingPoint[1]);
                    }
                }

                k++;
            }

            //for column
            for(int i = 0; i < 5; i++) {
                StartingPoint = words.get(k).getStartingPoint();
                wordLength = words.get(k).getWordLength();
                for(int j = StartingPoint[0]; j < wordLength + StartingPoint[0]; j++){
                    if(!matrix[j][i].equals("black") && !matrix[j][i].equals("white") && !(matrix[j][i]
== null)  && !Character.isDigit(matrix[j][i].charAt(0)) ) {
                        words.get(k).updateWordSoFar(matrix[j][i], j - StartingPoint[0]);
                        System.out.println("j is:" + j + ", value is" +  matrix[j][i]);
```

```
          }

        }
        k++;

      }

  }

  public static void showWords()
  {
    for(int i = 0; i < words.size(); i++)
    {
      int number = words.get(i).getNumber();
      String orientation = words.get(i).getOrientation();
      int wordLength = words.get(i).getWordLength();
      int startingPointx = words.get(i).getStartingPoint()[0];                    //initialize
stratingPoint array to length 2
      int startingPointy = words.get(i).getStartingPoint()[1];
      String clue = words.get(i).getClue();
      ArrayList<String> cans= words.get(i).getCandidates();
      //wordSoFar = new String[this.wordLength];        //initialize wordSoFar array to length =
wordLength
      //candidates = new ArrayList<String>();

      System.out.println("number: " + number + ", orientation : " + orientation + ", word
length: " + wordLength + ", stx: " + startingPointx + ", sty:" + startingPointy + ", clue: " + clue +
", length: " + words.get(i).getLength());
      for(int j = 0; j < cans.size(); j++)
      {
        System.out.println(cans.get(j));
      }
    }
  }

  public static void putClues(){
    ArrayList<String> clues = getClues();
    String orientation;
    for(int i = 0; i < 10; i++){
        for(int j = 0; j < 10; j++){
            if(j < 5){
                orientation = "a";
            }
            else {
                orientation = "d";
            }
            if(words.get(i).getName().equals(clues.get(j).charAt(0) + orientation)){
                words.get(i).setClue(clues.get(j).substring(2));
            }
        }
    }

}



public static void initAnswerTable()
{
    int count = 0;
    ArrayList<String> answers = getAnswers();

    for (int i = 0; i < CWIDTH; i++)
    {
        for(int j = 0; j< CHEIGHT; j++)
        {
            if(answers.get(count).length()>0)
                answerTable[i][j] = answers.get(count);
            count++;
        }
    }

}

public static void createAnswers(){
    answers = new ArrayList<>();
    for(int i = 0; i < words.size(); i++){
        //initialize the variables of certain word
        StringBuilder key = new StringBuilder();
```

```
      String name = words.get(i).getName();
      int length = words.get(i).getWordLength();
      int[] strPnt = words.get(i).getStartingPoint();
      String orientation = words.get(i).getOrientation();
      //for across crossword
      if(orientation.equals("a")){
        int row = strPnt[0];
        int col = strPnt[1];
        boolean enter = false;
        for(int k = 0; k < 5; k++){
          //if starting point matches, then enter is set to true to insert the letters till the length
of word is zero
          if((row == i && k == col) || enter){
            if(length > 0){
              key.append(answerTable[i][k]);
            }
            length--;
            enter = true;
          }
        }
        answers.add(new Answer(name, String.valueOf(key)));
      }
      //for down crossword
      else {
        int row = strPnt[0];
        int col = strPnt[1];
        boolean enter = false;
        for(int k = 0; k < 5; k++){
          //if starting point matches, then enter is set to true to insert the letters till the length
of word is zero
          if((col == (i - 5)&& k == row) || enter){
            if(length > 0){
              key.append(answerTable[k][i-5]);
            }
            length--;
            enter = true;
          }
        }
        answers.add(new Answer(name, String.valueOf(key)));
      }
    }

  }


  public static void showAnswers()
  {
    for(int i = 0; i < answers.size(); i++)
    {
      System.out.println("name " + answers.get(i).getName() + ", key : " +
answers.get(i).getKey());
    }
  }

  public static void setBase(){
    int counter = 0;
      for(int i = 0; i < words.size()  && counter < 2; i++){
        for(int j = 0; j < answers.size() && counter < 2; j++){
          if(words.get(i).getName().equals(answers.get(j).getName())) {
            if (words.get(i).getCandidates().contains(answers.get(j).getKey())){
              String[] temp = new String[answers.get(j).getKey().length()];
              for(int k = 0; k < answers.get(j).getKey().length(); k++){
                temp[k] = answers.get(j).getKey().charAt(k) + "";
              }

              words.get(i).removeCandidates();

              String orientation = words.get(i).getOrientation();
              int[] strPoint = words.get(i).getStartingPoint();
              int row = strPoint[0], col = strPoint[1];
              for(int r = 0; r < temp.length; r++){
                if(orientation.equals("a")) {
                  matrix[row][col] = temp[r];
                  col++;
                }
                if(orientation.equals("d")){
                  matrix[row][col] = temp[r];
                  row++;
                }
              }
              counter++;

            }
          }
        }
      }

  }

  public static boolean updateMatrix( Word word, int index)
  {
    boolean changed = false;
    String orientation = word.getOrientation();
    int[] strPoint = word.getStartingPoint();
    int row = strPoint[0], col = strPoint[1];
    for(int r = 0; r < word.getWordLength(); r++){
      if(orientation.equals("a")) {
        if(matrix[row][col].equals("black") || matrix[row][col].equals("white") ||
(matrix[row][col] == null) || Character.isDigit(matrix[row][col].charAt(0))) {

          matrix[row][col] =
Character.toString(word.getCandidates().get(index).substring(r).charAt(0));

          changed = true;
        }
        col++;

      }

      if(orientation.equals("d")){
        if(matrix[row][col].equals("black") || matrix[row][col].equals("white") ||
(matrix[row][col] == null) || Character.isDigit(matrix[row][col].charAt(0)))
        {
        matrix[row][col] =
Character.toString(word.getCandidates().get(index).substring(r).charAt(0));

        changed = true;
```

```
            }
            row++;
         }
      }

      return changed;

   }

   public ArrayList<String[]> getFromDictionary(String[] wordSoFar) throws
FileNotFoundException {
      ArrayList<String[]> resultList = new ArrayList<>();
      int count = 0;

      for (String aWordSoFar : wordSoFar)
         if (aWordSoFar != null)
            count++;

      Scanner scan = new Scanner(new File("newWord.txt"));
      while(scan.hasNext()){
         String line = scan.nextLine().toUpperCase();
         if(line.length() == wordSoFar.length) {
            int count2 = 0;
            String[] lineArray = new String[line.length()];
            for(int i = 0; i < line.length(); i++)
               lineArray[i] = String.valueOf((line.charAt(i)));
            for(int i = 0; i < line.length(); i++) {
               if(wordSoFar[i] != null && wordSoFar[i].equals(lineArray[i]))
                  count2++;
            }
            if(count == count2)
               resultList.add(lineArray);

         }

      }

      return resultList;
   }
```

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



       /* ArrayList<String> colors;
        colors = getNumbers(true);
        for(int i = 0; i < colors.size(); i++)
        {
            System.out.println(colors.get(i));
        } */

        Scanner reader = new Scanner(System.in);  // Reading from System.in
        System.out.println("1. for \"Today's, 8th March-2018\"\n" +
                "2. for \"7th March-2018\"\n3. for \"6th March-2018\"\n" +
                "4. for \"7th February-2018\"\n5. for \"2nd February-2018\"\n6. for \"3rd
January-2018\"\n" +
                "7. for \"5th December-2017\"\n ");
        int n = reader.nextInt(); // Scans the next token of the input as an int.

        reader.close();

        String input ="M8";

        boolean today = true;


        if(n == 1)
        {
            input = "M8";
            today = true;
        }
        else if (n == 2)
        {
            input = "M7";
            today = false;
        }
        else if (n == 3)
```

```
{
   input = "M6";
   today = false;
}
else if (n == 4)
{
   input = "F7";
   today = false;
}
else if (n == 5)
{
   input = "F2";
   today = false;
}
else if (n == 6)
{
   input = "J3";
   today = false;
}
else if (n == 7)
{
   input = "O30_17";
   today = false;
}


start(input);
putColors();
putNumbers();
System.out.println(Arrays.deepToString(matrix));
createWords();


putClues();
initAnswerTable();
createAnswers();

System.out.println(Arrays.deepToString(answerTable));
showAnswers();
```

```
for(int i = 0; i < words.size(); i++)
{
   words.get(i).googleSearch();
   words.get(i).wikiSearch();
   words.get(i).trackerSearch();
   words.get(i).filterBySize();
   words.get(i).filterByLetters();

}



System.out.println(Arrays.deepToString(matrix));
showWords();

setBase();
System.out.println(Arrays.deepToString(matrix));

updateWords();


for(int i = 0; i < words.size(); i++)
{
   words.get(i).showWordSoFar();
}

for(int i = 0; i < words.size(); i++)
{
   words.get(i).filterByLetters();
}

System.out.println("Updated by letters");
showWords();

int looper = 0;

while( looper < 10)
{
```

```
int smallest = 100;
int index = 0;

for(int i = 0; i < words.size(); i++ )
{
   if(words.get(i).getCandidates().size() > 0)
   {
      if(words.get(i).getCandidates().size() <= smallest)
      {
         smallest = words.get(i).getCandidates().size();
         index = i;
      }
   }

}


updateMatrix(words.get(index),0);
updateWords();

System.out.println(Arrays.deepToString(matrix));

words.get(index).removeCandidates();

showWords();

for(int i = 0; i < words.size(); i++)
{
   words.get(i).showWordSoFar();
}

for(int i = 0; i < words.size(); i++)
{
   words.get(i).filterByLetters();
}

showWords();

looper++;
```

```
        }


        ArrayList<Integer> colors;
        colors = getColors();

        ArrayList<String> numbers;
        numbers = getNumbers();

        ArrayList<String> clues;
        clues = getClues();

        ArrayList<String> answers;
        answers = getAnswers();


        CrosswordGrid cd = new CrosswordGrid(colors, numbers, clues, answers);
        cd.sendArray(matrix);
        cd.display();

    }


}

import org.jsoup.Jsoup;
import org.jsoup.nodes.Document;
import org.jsoup.select.Elements;

import java.net.URL;
import java.text.BreakIterator;
import java.util.ArrayList;
import java.util.Arrays;
import java.util.LinkedHashSet;

public class Word {

    //properties
    int number;                    //cell number of the first letter
```

```
  String orientation;               //0 for across., 1 for down
  int wordLength;                  //length of the answer word
  String[] wordSoFar;              //array of found letters of the word
  ArrayList<String> candidates;   //list of potential answers
  int[] startingPoint;            //coordinates of the first letter, x = row = startingPoint[0],
                                                      // y = col = startingPoint[1]
  String name;

  String clue;

  //constructor
  public Word(int number, String orientation, int wordLength, int array[])
  {
    this.number = number;
    this.orientation = orientation;
    this.wordLength = wordLength;
    startingPoint = new int[2];
    startingPoint[0] = array[0];                    //initialize stratingPoint array to length 2
    startingPoint[1] = array[1];
    wordSoFar = new String[wordLength];        //initialize wordSoFar array to length =
wordLength
    candidates = new ArrayList<>();
    name = number + orientation;
  }

  //copy constructor
  public Word( Word newWord)
  {
    this.number = newWord.getNumber();
    this.orientation = newWord.getOrientation();
    this.wordLength = newWord.getWordLength();
    this.startingPoint = new int[2];
    this.startingPoint[0] = newWord.getStartingPoint()[0];
    this.startingPoint[1] = newWord.getStartingPoint()[1];
    this.wordSoFar = new String[wordLength];
    for(int i = 0; i < wordSoFar.length; i++)
    {
      if(newWord.getWordSoFar()[i] != null)
        wordSoFar[i] = newWord.getWordSoFar()[i];
    }
    this.candidates = new ArrayList<>();
    for(int i = 0; i < newWord.getCandidates().size(); i++)
    {
            this.candidates.add(newWord.getCandidates().get(i));
    }
    this.name = newWord.getName();

}

//methods


public void removeCandidates()
{
    while(candidates.size()!= 0)
    {
        candidates.remove(0);
    }
}

public int getLength()
{
    return wordLength;
}

public void updateWordSoFar(String letter, int index)
{
    wordSoFar[index] = letter;
}

public String getName()
{
    return name;
}

public int getNumber()
{
    return number;
```

```
}

public String getOrientation()
{
    return orientation;
}

public int getWordLength()
{
    return wordLength;
}

public int[] getStartingPoint()
{
    return startingPoint;
}

public ArrayList<String> getCandidates()
{
    return candidates;
}

public String[] getWordSoFar()
{
    return wordSoFar;
}

public void googleSearch()  {



        ArrayList<String> words = getWords(getClue());

        ArrayList<String> localC = new ArrayList<>();

        String query = "http://www.google.com/search?q=";

        for (int i = 0; i < words.size(); i++) {
            if (i != (words.size() - 1)) {
                query = query + words.get(i).replaceAll("[-+.^:,']","") + "+";
            } else {
                query = query + words.get(i).replaceAll("[-+.^:,']","");
            }
        }

        System.out.println(query);

        try {

            Document doc = Jsoup.connect(query).userAgent("Chrome").get();


            Elements elems = doc.getElementsByClass("st" );



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

                // System.out.println(elems.get(i).text());

                localC.addAll(getWords(elems.get(i).text()));

            }
        } catch (Exception e) {
            e.printStackTrace();

        }

        candidates.addAll(localC);

        LinkedHashSet<String> localCset = new LinkedHashSet<>(candidates);

        candidates.clear();

        candidates.addAll(localCset);


        candidates.replaceAll(String::toUpperCase);
```

```
    removeDuplicates();


}

public void  trackerSearch()
{
  ArrayList<String> words = getWords(getClue());

  ArrayList<String> localC = new ArrayList<>();



    String query = "http://crosswordtracker.com/clue/";

    for (int i = 0; i < words.size(); i++)
    {
      if (i != (words.size() - 1))
      {
        query = query + words.get(i).toLowerCase() + "-";
      }
      else
      {
        query = query + words.get(i).toLowerCase() + "/";
      }
    }

    System.out.println(query);

    try {

      Document doc = Jsoup.parse(new URL(query), 10000);


      Elements elems = doc.getElementsByClass("answer");

      for(int i = 0; i < elems.size(); i++){
        localC.add(elems.get(i).text());
      }
```

```
        }
        catch (Exception e)
        {
            e.printStackTrace();
        }

    candidates.addAll(localC);

    LinkedHashSet<String> localCset = new LinkedHashSet<>(candidates);

    candidates.clear();

    candidates.addAll(localCset);


    candidates.replaceAll(String::toUpperCase);

    removeDuplicates();

}



public void wikiSearch() {

    ArrayList<String> words = getWords(getClue());

    ArrayList<String> localC = new ArrayList<>();

        String query = "https://en.wikipedia.org/w/index.php?search=";

        for (int i = 0; i < words.size(); i++) {
            if (i != (words.size() - 1)) {
                query = query + words.get(i).replaceAll("[-+.^:,']","") + "+";
            } else {
                query = query + words.get(i).replaceAll("[-+.^:,']","");
            }
        }
        System.out.println(query);
        try {
            Document doc = Jsoup.parse(new URL(query), 10000);

            Elements elems = doc.getElementsByClass("searchresult");
            for (int i = 0; i < elems.size(); i++) {
                localC.addAll(getWords(elems.get(i).text()));
            }
        } catch (Exception e) {
            e.printStackTrace();
        }

    candidates.addAll(localC);
    LinkedHashSet<String> localCset = new LinkedHashSet<>(candidates);
    candidates.clear();
    candidates.addAll(localCset);

    candidates.replaceAll(String::toUpperCase);
    removeDuplicates();
}
```

```
public void setClue(String clue)
{
   this.clue = clue;
}

public String getClue()
{
   return clue;
}

public void removeDuplicates()
{
   LinkedHashSet<String> localCset = new LinkedHashSet<>(candidates);

   candidates.clear();

   candidates.addAll(localCset);

}



public void filterByLetters()
{
   boolean removed;
   boolean ended;

   int counter;

   for(int i = 0; i < candidates.size(); i++)
   {
      removed = false;
      ended = false;
      counter = 0;

      while(!(removed || ended))
      {
         if(wordSoFar[counter]!= null)
```

```
            {
                if(candidates.get(i).charAt(counter) != wordSoFar[counter].charAt(0))
                {
                    candidates.remove(i);
                    i--;
                    removed = true;
                }

            }
            counter++;

            if(counter >= wordSoFar.length)
            {
                ended = true;
            }

        }
    }

}

public void addCandidates(ArrayList<String> newCand)
{
    this.candidates.addAll(newCand);
}

public void showWordSoFar()
{
    System.out.println(Arrays.deepToString(wordSoFar));
}


public void filterBySize()
{

    for(int i = 0; i < candidates.size(); i++)
            if(candidates.get(i).length() != wordLength) {
                candidates.remove(i);
                i--;
                }

    }

    public ArrayList<String> getWords(String text) {
        ArrayList<String> words = new ArrayList<>();
        BreakIterator breakIterator = BreakIterator.getWordInstance();
        breakIterator.setText(text);
        int lastIndex = breakIterator.first();
        while (BreakIterator.DONE != lastIndex) {
            int firstIndex = lastIndex;
            lastIndex = breakIterator.next();
            if (lastIndex != BreakIterator.DONE &&
Character.isLetterOrDigit(text.charAt(firstIndex))) {
                words.add(text.substring(firstIndex, lastIndex));
            }
        }

        return words;
    }


}

public class Answer {

    //properties

    String name;
    String key;

    //constructor

    public Answer(String name, String key)
    {
        this.name = name;
        this.key = key;
    }
    //methods

    public String getName()
    {
        return name;
    }

    public String getKey()
    {
        return key;
    }

}

import java.awt.*;
import java.awt.event.KeyAdapter;
import java.awt.event.KeyEvent;

import javax.swing.*;

public class AnswerCell extends JPanel {

    //properties
    JTextField field;
    JLabel superscript;
    String letter;
    int isBlack;

    String cellNo;

    public AnswerCell() {
        isBlack = 1;
        superscript = new JLabel("");
        field =  new JTextField("");
        cellNo = "";
        letter = null;

        makeCell();
    }
```

```
//methods
public void setIsBlack(int val) {
   isBlack = val;
}

public void setCellNo(String val) {
   cellNo = val;
}
public void setLetter(String str) {
   letter = str;
}
public String getLetter() {
   return letter;
}
public void makeCell() {
   setPreferredSize(new Dimension(35, 35));
   setLayout(new BorderLayout());


   if (isBlack == 0) {
      setBackground(Color.BLACK);
      //letter = "black";
      superscript.setVisible(false);
      field.setVisible(false);
   }
   else {
      // put a superscript if cell has a number

      String strCellNo = cellNo + "";
      superscript = new JLabel(strCellNo);
      superscript.setSize(new Dimension(1,1));
      superscript.setBackground(Color.WHITE);
      superscript.setOpaque(true);
      add(superscript, BorderLayout.NORTH);


      Font font = new Font("SansSerif", Font.PLAIN, 17);
      field = new JTextField();
```

```
            field.setText(letter);
            field.setBorder(BorderFactory.createEmptyBorder());
            field.setFont(font);
            field.setHorizontalAlignment(JTextField.CENTER);
            field.setEditable(false);
            field.setBackground(Color.WHITE);

            add(field, BorderLayout.CENTER);


            setBorder(BorderFactory.createLineBorder(Color.BLACK, 1));
        }
    }

}



import java.awt.*;

import java.util.ArrayList;

import java.awt.event.*;
import javax.swing.*;

public class CrosswordGrid {

    //properties
    private JFrame frame;
    private JPanel panel;
    private ArrayList<GridCell> cellList;
    private ArrayList<AnswerCell> answerList;
    //private ArrayList<String> inputLetters;
    private JButton button;
    private JTextArea across;
    private JTextArea down;
    private ArrayList<String> hintsAcross;
    private ArrayList<String> hintsDown;
    private JLabel acc;
    private JLabel dwn;
    private JPanel answerPanel;
    private ArrayList<Integer> colors;
    private ArrayList<String> numbers;
    private ArrayList<String> clues;
    private ArrayList<String> answers;
    private String output = "";
    private String[][] array;
    private ArrayList<String> tempArray;

    private JTextArea log;
    private JScrollPane scroll;
    private JLabel logTitle;
    final String[] arr = {"Unit of a train or roller coaster", "Where many modern files are stored,
with \"the\"","Apples that share a name with Japan's largest mountain","Hospital professionals,
for short","Political commentator Pfeiffer"};

    //constructor
    public CrosswordGrid(ArrayList<Integer> colors, ArrayList<String> numbers,
ArrayList<String> clues, ArrayList<String> answers) {
        //makeGrid();
        this.colors = colors;
        this.numbers = numbers;
        this.clues = clues;
        this.answers = answers;
        //setup the window
        frame = new JFrame();
        frame.setBounds(300, 120, 600, 600);
        //frame.setExtendedState(JFrame.MAXIMIZED_BOTH);
        //frame.setUndecorated(true);
        frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
        frame.getContentPane().setLayout(null);
        frame.getContentPane().setBackground(Color.WHITE);



    }
```

```
  public void update(ArrayList<Integer> colors, ArrayList<String> numbers, ArrayList<String>
clues, ArrayList<String> answers)
  {
    this.colors = colors;
    this.numbers = numbers;
    this.clues = clues;
    this.answers = answers;
    display();
  }

  public void display() {

    makeGrid();
    displayHints();
    displayAnswerGrid();
    makeLog();
    // selectCrossWord();

    //frame.repaint();

  }

  public void makeLog() {
    log = new JTextArea();
    //log.setBounds(755, 50, 600, 150);
    //log.setBorder(BorderFactory.createLineBorder(Color.BLACK, 2));
    log.setLineWrap(true);
    log.setWrapStyleWord(true);



    Font font = new Font("Courier New",Font.PLAIN, 15);
    log.setFont(font);
    log.setEditable(true);
    scroll = new JScrollPane(log, JScrollPane.VERTICAL_SCROLLBAR_ALWAYS,
JScrollPane.HORIZONTAL_SCROLLBAR_NEVER);
    scroll.setBounds(755, 70, 600, 150);
    scroll.setBorder(BorderFactory.createLineBorder(Color.BLACK, 2));
    //DefaultCaret caret = (DefaultCaret)log.getCaret();
    //caret.setUpdatePolicy(DefaultCaret.OUT_BOTTOM);
```

```
        logTitle = new JLabel("Log:");
        logTitle.setBounds(755, 0, 70, 90);
        frame.add(logTitle);
        frame.add(scroll);
    }

    public void updateLog(String text) {
        log.setCaretPosition(log.getDocument().getLength());
        log.append(text);
        log.setCaretPosition(log.getDocument().getLength());
        log.append("\n");
        //log.append("\n");
        log.setCaretPosition(log.getDocument().getLength());
    }



    public void makeGrid() {

        //make 25 grid cells
        cellList = new ArrayList<GridCell>(26);
        tempArray = new ArrayList<>();
        for (int i = 0; i < 25; i++) {
            cellList.add(new GridCell());

        }
        for (int i = 0; i < 25; i++) {
            cellList.get(i).setIsBlack(colors.get(i));
            cellList.get(i).setCellNo(numbers.get(i));

        }
        for (int i = 0; i < 5;i++) {
            for (int j = 0; j < 5;j++) {
                tempArray.add(array[i][j]);
            }
        }
        //write answers
        for (int i = 0; i < 25; i++) {
            if(cellList.get(i).getIsBlack() == 1 && cellList.get(i).getLetter() != null &&
!(tempArray.get(i).equals("white"))) {
                cellList.get(i).setLetter(tempArray.get(i));
            }


        }

        //setup the grid
        panel = new JPanel();
        panel.setBounds(280, 180, 400, 400);
        panel.setLayout(new GridLayout(5,5));
        panel.setBorder(BorderFactory.createLineBorder(Color.BLACK, 2));
        panel.setBackground(Color.WHITE);

        //add cells to the grid
        for (int i = 0; i < 25; i++) {
            panel.add(cellList.get(i));
            cellList.get(i).makeCell();
        }



        frame.add(panel);
    }

    public void displayHints() {
        across = new JTextArea();
        down = new JTextArea();

        across.setBounds(755, 280, 300,300);
        down.setBounds(1060,280,300,300);
        across.setBorder(BorderFactory.createLineBorder(Color.BLACK, 2));
        down.setBorder(BorderFactory.createLineBorder(Color.BLACK, 2));

        acc = new JLabel("Across");
        dwn = new JLabel("Down");
    acc.setBounds(755, 220, 70, 90);
    dwn.setBounds(1060,220,70,90);

    across.setLineWrap(true);
    across.setWrapStyleWord(true);
    down.setLineWrap(true);
    down.setWrapStyleWord(true);

    Font font = new Font("Arial",Font.PLAIN, 20);
    across.setFont(font);
    down.setFont(font);
    for (int i = 0 ; i < 5 ; i ++ ) {
        across.append(clues.get(i));
        across.append("\n");
        across.append("\n");
    }
    for (int i = 5 ; i < 10 ; i ++ ) {
        down.append(clues.get(i));
        down.append("\n");
        down.append("\n");
    }

    across.setEditable(false);
    down.setEditable(false);

    frame.add(acc);
    frame.add(dwn);
    frame.add(across);
    frame.add(down);

}

public void getVal (String value)
{
    output = value;
}

public String getOutput()
```

```
    {
       return output;

    }



    public void sendArray(String[][] array) {
       this.array = array;
    }

    public void displayAnswerGrid() {
       //setup the grid
       answerPanel = new JPanel();
       answerPanel.setBounds(15, 380, 200, 200);
       answerPanel.setLayout(new GridLayout(5,5));
       answerPanel.setBorder(BorderFactory.createLineBorder(Color.BLACK, 2));
       //panel.setBorder(BorderFactory.createStrokeBorder(new BasicStroke(5.0f)));
       answerPanel.setBackground(Color.WHITE);

       answerList = new ArrayList<AnswerCell>();
       //add cells to the grid
       for (int i = 0; i < 25; i++) {
          answerList.add(new AnswerCell());
          answerList.get(i).setLetter(answers.get(i));
          answerList.get(i).setCellNo(numbers.get(i));
          answerList.get(i).setIsBlack(colors.get(i));

       }
       for (int i = 0; i < 25; i++) {
          answerPanel.add(answerList.get(i));
          answerList.get(i).makeCell();
       }

       frame.add(answerPanel);
       frame.setVisible(true);
    }

}
```

```
import java.awt.*;
import java.awt.event.KeyAdapter;
import java.awt.event.KeyEvent;

import javax.swing.*;

public class GridCell extends JPanel {

   //properties
   JTextField field;
   JLabel superscript;
   String letter;
   int isBlack;

   String cellNo;

   public GridCell() {
      isBlack = 1;

      cellNo = "";
      letter = "";

      makeCell();
   }

   //methods
   public void setIsBlack(int val) {
      isBlack = val;

   }
   public int getIsBlack() {
      return isBlack;

   }
   public void setLetter(String str) {
      letter = str;
```

```
}

public void setCellNo(String val) {
   cellNo = val;
}
public String getLetter() {
   return letter;
}
public void makeCell() {
   setPreferredSize(new Dimension(50, 50));
   setLayout(new BorderLayout());
   //setBackground(Color.WHITE);

   if (isBlack == 0) {
      setBackground(Color.BLACK);
      superscript.setVisible(false);
      field.setVisible(false);
   }
   else {
      // put a superscript if cell has a number

      String strCellNo = cellNo;
      superscript = new JLabel(strCellNo);
      // superscript.setSize(new Dimension(2,2));
      superscript.setBackground(Color.WHITE);
      superscript.setOpaque(true);
      add(superscript, BorderLayout.NORTH);


      Font font = new Font("SansSerif", Font.BOLD, 20);
      field = new JTextField();

      //field.setText("A");r
      field.setBorder(BorderFactory.createEmptyBorder());
      field.setFont(font);
      field.setText(letter);
      field.setHorizontalAlignment(JTextField.CENTER);

      field.addKeyListener(new KeyAdapter() {
            public void keyTyped(KeyEvent e) {
               if (field.getText().length() >= 1 ) {//limit text to 1 characters
                  e.consume();
                  letter = field.getText();
                  printLetter();
               }
            }
         });

         add(field, BorderLayout.CENTER);

         //setBorder(BorderFactory.createStrokeBorder(new BasicStroke(5.0f)));
         setBorder(BorderFactory.createLineBorder(Color.BLACK, 1));
      }
   }
   public void printLetter() {
      System.out.println(letter);
   }
}
```