



PROGRAMMING MODELS



A coursework completed as part of the requirement for

SUBJECT NAME: PROGRAMMING MODELS

SUBJECT CODE: CC303

LECTURER: DR. KYAW KYAW HTIKE

Entitled

ASSIGNMENT TITLE: ASSIGNMENT 1

Submitted on

DATE OF SUBMISSION: 27th MARCH 2018

Produced by

Name	Student ID	Course
LOUIS JULIENDO	1001540916	BSc (Hons) Computing
JOHN LOH ERN-RONG	1001439193	BSc (Hons) Computing
CHEW CHENG JIN	1001540205	BSc (Hons) Computing
CHAI JUN SHENG	1001540249	BSc (Hons) Computing

1.0 INTRODUCTION

Programming languages in general are a set or a collection of instructions that is constructed by a programmer for a computer to perform a certain task. PROLOG is also a programming language designed by Alain Colmerauer. Unlike most programming languages that are imperative languages which are based on sequence of commands, PROLOG is a formal logic-based language or it is also called as a declarative language since it contains facts and rules in a knowledge base.

PROLOG is also a programming language that is widely used in the field of artificial intelligence, such as the creation of expert systems and the processing of natural languages. PROLOG can be great solution for programmers that need to solve solutions that are declarative, for example, designing a family tree program. In addition to that, PROLOG can be integrated with other imperative languages like Java through a library known as JPL.

PROLOG derives new rules from the existing rules contained within the knowledge base. PROLOG contains three basic constructs in which are facts and rules (as mentioned above) and queries. Hence, these collections of facts and rules are known as the knowledge base (that functions like a database). PROLOG programming is precise on writing knowledge bases.

Programming languages can be divided into two types: that are imperative languages and relational languages. The most important difference between imperative languages like Java and relational languages like PROLOG is that PROLOG can run the program backwards which means that users are able to give the program an output and an input corresponding to it will be returned. This however is not the case in imperative languages.

While PROLOG provides many advantages to many different kinds of problems, it is never said that PROLOG is more important than traditional programming with imperative languages like Java or vice versa. Therefore, it is crucial for us to weigh and observe the needs and requirements of developing a program and then only decide which programming language is more suitable.

2.0 BACKGROUND

The system that we have developed is a Crossword Puzzle Solver System. The aim of this system is to solve a crossword puzzle with 6 different given English words as input. Two programming languages will be used in this system, which are Java and PROLOG. PROLOG is used as the back-end programming and Java is used as the front end. PROLOG is used to process the input from the user and then produce an output. Java is be used as the front end to develop the GUI of the system.

The overall workaround of this system is to allow users to input 6 different English words, 7 letters of each word. Some of the other functions of this system include:

i) Allow users to input 6 different English words. These 6 different words in the English language will be used as input to generate the crossword puzzle of all words. The validity of the words depends on the dictionary (a notepad file that contains 32,909 words exported from bestwordlist.com). Words that do not exist in the dictionary will not be accepted as solutions of a crossword puzzle as it is not a valid English word.

ii) Restrict users to input invalid characters. The system has the function of using DocumentFilter to restrict users from inputting special characters like symbols (e.g !, @, #, \$, %, ^, &, *, (,)) and numbers (e.g 1, 2, 3, 4, 5, 6, 7, 8, 9). Only alphabets, backspace and the delete button will be accepted by the system. The copying and pasting from the clipboard will also work. For example, if a user copies a phrase “123abalone” and pastes it in the JTextField, the DocumentFilter function is able to filter out the unaccepted characters (in our case, numbers), so the result of the pasting will only be “abalone”.

3.0 METHODOLOGY

3.1 Connection from SWI-PROLOG to NetBeans IDE

We have chosen the JPL library (jpl.jar) to connect SWI-PROLOG to Java using the NetBeans IDE. The JPL file was imported for the “lib” folder the the PROLOG directory. However, to ensure that NetBeans IDE recognises the JPL file and its path, several measures have to be done in order to make the whole process work.

There are 3 important things that have to be done, that is to make changes to our computer’s Environment Variables. In order to make changes, we have to open our Computer’s directory’s properties (named as This PC on Windows 10). Under the system properties, we have to access the advanced system settings that is listed which we will able to make changes to the environment variables.

From that window, the following variables are the variables which we need to add and modify:

i. Add a new variable:-

Variable name: SWI_HOME_DIR

Variable value: Path to the location where SWI-PROLOG was being installed

ii. Add a new path:-

Variable name: PATH

Variable value: %SWI_HOME_DIR%\bin

iii. Add another new path under the same PATH variable created in part (ii):-

Variable name: PATH (in relation to path created in part(ii))

Variable value: %SWI_HOME_DIR%\lib\jpl.jar

Example:

SWI_HOME_DIR: C:\Program Files\swipl

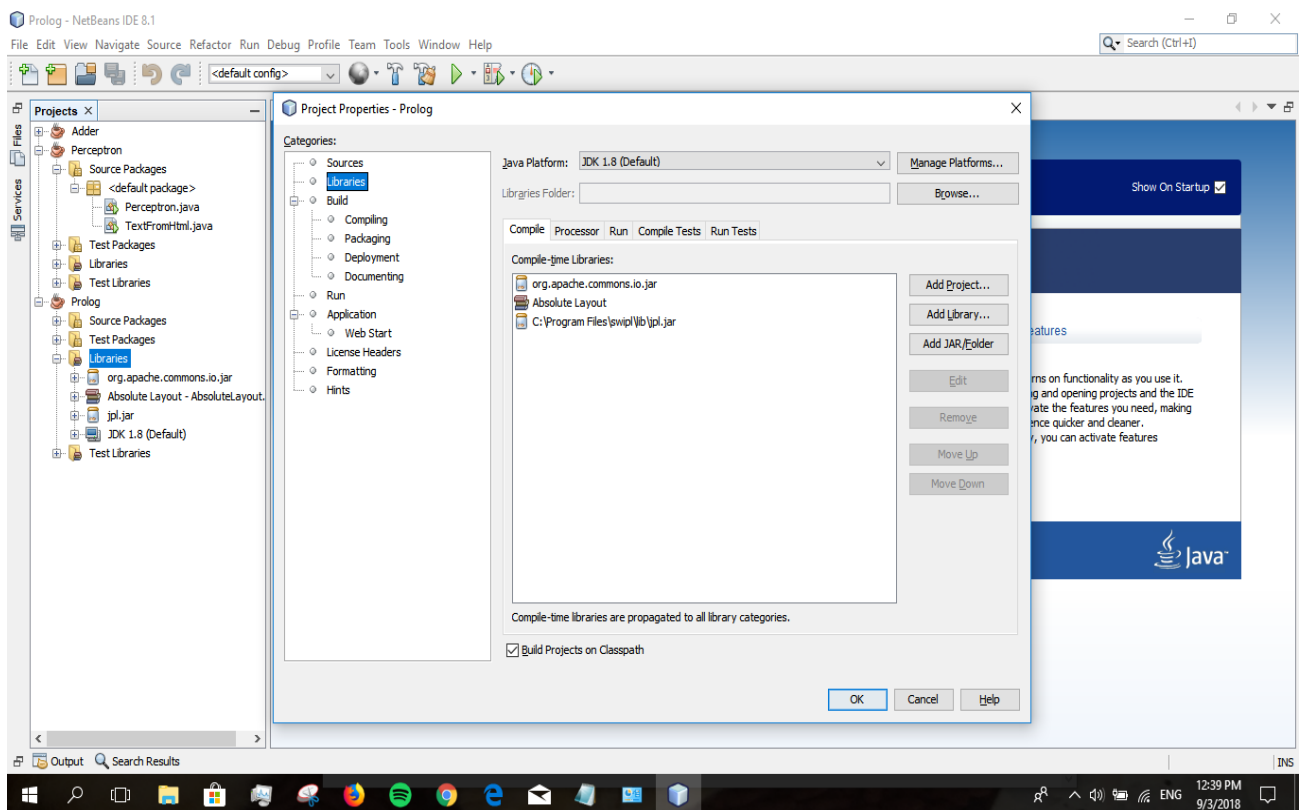
PATH: C:\Program Files\swipl%SWI_HOME_DIR%\bin%SWI_HOME_DIR%\lib

C:\Program Files\swipl

**Environment variables are created on a Windows 10 computer with a 64-bit version of SWI-PROLOG that is only compatible with a 64-bit version of JDK 1.8*

3.2 The import of SWI-PROLOG library into NetBeans IDE

In order to use the library (jpl.jar) in our IDE, we firstly have to import the .jar file into the IDE. Firstly, under the projects pane, right click on the Libraries folder and click on Properties. Navigate to “Libraries” under the Categories, click the button “Add JAR/Folder”.



Then, navigate again to the jpl.jar that is usually stored in the following path:

"C:\Program Files\swipl\lib\jpl.jar"

**Path only applies to a 64-bit version of Windows 10 and 64-bit version of SWI-PROLOG*

3.3 Methods

The following are the methods we have constructed in our source code which will be explained one by one:

In the 1st window named as 'Window1', there are 4 methods being used which are **clearTextField()**, **checkDictionary()**, **listToLowercase()**, **convertMapToArray()**, and **convert1dTo2d()**.

The 1st method **clearTextField()** is a void method. This method is used to set all the JTextFields to empty after an exception occurs.

The 2nd method **checkDictionary()** returns a boolean value and it has 2 parameters which are HashSet and ArrayList with String as the parameterized type. 2 conditions are being applied where if the dictionary in the HashSet does not contain all words in the ArrayList, it will return true, and false otherwise.

The 3rd method **listToLowercase()** takes in 1 parameter which is an ArrayList with String as the parameterized type. This method is used to set the ArrayList element to lowercase by using ListIterator and while loop to iterate through the list.

The 4th method **convertMapToArray()** returns an array of String and takes in 2 parameters which are an array of Map that contains String and Term as the parameterized type which contains all the solutions to be passed to a String array, and an array of String which will be used to hold all solutions. The method works by converting the map into a String, then all

special characters such as [,], {, }, space, 'A', 'B', 'C', 'D', 'E', 'F', and '=' will be removed by using the `replaceAll()` method. After that, the String will be constructed to have comma-separated values which contains words for all solutions. The comma is then split and inserted into an array of String. Lastly, the method will return the list of words for all solutions as the array of String.

The last method **`convert1dTo2d()`** has 2 parameters which are `listOfSolutions` as a one-dimensional String array and `noSolutions` as an integer. The method is used to convert a one-dimensional String array into two-dimensional String array by inserting all the elements into the two-dimensional array.

In the 2nd window named as 'Window2', there are 3 methods which are **`chooseSolution()`**, **`splitWordIntoChar()`**, and **`setCharToTextField()`**.

The 1st method **`chooseSolution(int index)`** returns an array of String and takes in 1 parameter for an index as an integer. The method is used to select the solutions that user want to show by specifying the index. For example, given that there are 10 solutions for a query and the user wants to see the 6th solution, the index should be specified to number 6 and it will return a list of words for the specified solution in an array of String.

The 2nd method **`splitWordIntoChar()`** has no argument. Basically, the method is used to split a one-dimensional String array that contains a list of words for the specified solution into characters and store it into a two-dimensional char array which will be used to populate the JTextFields.

The last method **`setCharToTextField(JTextField[] jtf, char [][] arr)`** has 2 parameters which are a one-dimensional JTextField array and a two-dimensional character array. This method is used to populate all JTextFields with the characters that have been split earlier in order to display the solution to the crossword puzzle.

3.4 Error Validation

This system has several error validation (exception handling) to help minimize and catch errors done by the user when using the system. This system uses 2 types of error validation which are a real-time error validation using **DocumentFilter** and a normal error validation using **JOptionPane** as message prompts. The reason why a real-time validation is being used will be explained in the “**Discussion**” section.

3.4.1 Real-Time Error Validation

i) Input validation that is real-time for user input.

The system will automatically restrict any special characters such as number and only allow 7 letters into the JTextField. It uses the DocumentFilter function to insert, replace, or remove any elements that have been modified in the Document. One of the method in AbstractDocument class is used in order to set the DocumentFilter. The validation is implemented in an inner class of Window1 called ‘**filterLetters**’ that extends DocumentFilter.

This class has 2 main methods which are **insertString()** and **replace()**. The purpose of the methods is to perform a real-time validation on the JTextFields. The **insertString()** method is used to restrict users from entering more than 7 characters in the JTextFields, delete the character if it is not a letter, and insert the characters into the JTextFields if it is a letter. The **replace()** method is used to remove a highlighted or specified region of text and to insert a copied text into the JTextFields.

3.4.2 Normal Error Validation

The system will prompt the user about every error action that they perform. The message box that is alerted and prompted uses the JOptionPane function that is in Java. This is so that users can be notified with the errors that they have performed. The error validations are listed as below:

- i) Validate if users have inputted text into all JTextFields.** A message will be prompted to users if users leave any JTextFields blank.
- ii) Validate if users have entered less than 7 characters into all JTextFields.** A message will be prompted if users have only entered 1 to 6 characters (JTextField is limited to 7 characters only.)
- iii) Validate if users have inputted duplicate words into all JTextFields.** A message will be prompted if there are duplicate words found in any of the JTextFields.
- iv) Validate if the input of users exist in the embedded dictionary.** A message will be prompted to users if the any input of the users does not exist in the dictionary. The particular word that does not exist will be shown to users so that the user would know which word to correct.
- v) Validate if there are no solutions to solve for the crossword puzzle.** A message will be prompted to users if the system is not able to find any possible solutions for the crossword puzzle. Users will then need to correct their input of words in order to come out with a proper solution.

3.5 Knowledge Base

The PROLOG knowledge base that we have constructed is written in a .pl file and it consists of **6 facts** and **2 rules**. The facts are always updated based on user input (from Java).

3.5.1 Facts

The words are split and each of the characters are separated by a comma. Each of the characters of a word will be matched against a certain word. Below are the 6 facts of the knowledge base:

```
words([l,e,v,e,r,e,d],levered).  
words([l,e,v,e,l,e,r],leveler).  
words([l,e,v,e,l,e,d],leveled).  
words([m,e,r,i,n,o,s],merinos).  
words([m,e,r,i,t,e,d],merited).  
words([m,e,i,o,s,e,s],meioses).
```

3.5.2 Rules

The first rule **notEqual(A,B,C,D,E,F)** is used to prevent duplicate values from being generated by the SWI-PROLOG engine. Below is the first rule of the knowledge base:

```
notEqual(A,B,C,D,E,F) :- (A \= B, A \= C, A \= D, A \= E, A \= F), (B \= C, B \= D, B  
\= E, B \= F), (C \= D, C \= E, C \= F), (D \= E, D \= F), (E \= F).
```

The second rule **solver(A,B,C,D,E,F)** has a few constraints such as:

1. The 2nd character of H1 must correspond to the 2nd character of V1, the 4th character of H1 must correspond to the 2nd character of V2, and the 6th character of H1 must correspond to the 2nd character of V3.
2. The 2nd character of H2 must correspond to the 4th character of V1, the 4th character of H2 must correspond to the 4th character of V2, and the 6th character of H2 must correspond to the 4th character of V3.
3. The 2nd character of H3 must correspond to the 6th character of V1, the 4th character of H3 must correspond to the 6th character of V2, and the 6th character of H3 must correspond to the 6th character of V3.

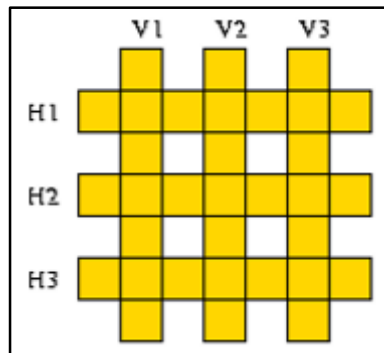


Figure X. Each character of a word is represented by horizontal or vertical box.

The characters to be matched are always located at the 2nd, 4th, and 6th position. An anonymous variable is used to denote a distinct variable and it can be regarded as a variable whose values we are not interested in. Another reason of using an anonymous variable instead of a normal variable is to avoid the singleton variable warning. The rules will match variables corresponding to each character of the word and also ensure all the words are not the same.

Below is the second rule of the knowledge base:

solver(A,B,C,D,E,F) :-

**words([_,X1Y1,_,X1Y2,_,X1Y3,_],A), words([_,X2Y1,_,X2Y2,_,X2Y3,_],B),
words([_,X3Y1,_,X3Y2,_,X3Y3,_],C), words([_,X1Y1,_,X2Y1,_,X3Y1,_],D),
words([_,X1Y2,_,X2Y2,_,X3Y2,_],E), words([_,X1Y3,_,X2Y3,_,X3Y3,_],F),
notEqual(A,B,C,D,E,F).**

3.6 Algorithm Analysis

The purpose of the analysis is to find out the time taken for an algorithm solve a particular problem. There are 3 measurements involved in order to ensure the algorithms we have constructed are able to work properly, which are: **effectiveness, correctness, and termination**. Effectiveness refers to an easily understandable code and the logical steps are well organized. Correctness refers to the output that is as expected and correct. Termination refers to the steps of execution that contains an ending and there are no looping problems or out of memory exceptions. All of our algorithms have fulfilled these three measurements.

There are several data structures and algorithms that have been used in order to optimize the code performance (execution time) such as ArrayList, HashSet, Array, String, Loops, and Java 8 Streams. The algorithm complexity is calculated by using big-O notation. The complexity presented in this report only covers the best case of the algorithms.

A description of a few big-O functions that are used in our algorithms are as follows:

$O(1)$ = constant time

$O(n)$ = linear time

$O(n^2)$ = quadratic time

Below are the explanation for the complexity of data structures and methods used in our program:

1. ArrayList

- a. `clear()` has a **$O(n)$** complexity which is much faster than `removeAll()` method that has **$O(n^2)$** because `clear()` does not have additional method calls like `contains` and `remove`.
- b. `get()` has **$O(1)$** complexity which runs in constant time. This method does not have any significant effect to the algorithm performance since it is solely used for getting some elements from a small ArrayList.

2. HashSet

- a. `clear()` method is used to set all elements into null and reuse it again instead of creating a new HashSet object. This will simplify the design pattern and also to prevent confusion because of too many objects being created. In terms of the optimization, there is no significant difference of both ways since we are reading a small amount of data. It has **$O(n)$** complexity.
- b. Both `add()` and `contains()` methods have **$O(1)$** complexity and the `HashSet.contains()` method is much faster than `ArrayList.contains()` which has **$O(n)$** complexity.

3. Array

- a. `Arrays.fill(array, null)` has **$O(n)$** complexity which is to set all elements in the array to null.

4. String

- a. `toArray()` has **$O(n)$** complexity since it depends on the number of array elements by copying each character and put it into an array.
- b. `replaceAll()` has **$O(n)$** complexity. `replaceAll()` is used because it can replace any Strings by using regex (*In this case, `\\B` is used to indicate non-word boundary that match between two word character for eg. `a,b,a,l,o,n,e`.*) while `replace()` method does not accept regex patterns and it only replaces character sequences such as char or String.
- c. `join()` has **$O(n)$** complexity. `join()` is used rather than `StringBuilder.append()` because of its cleanliness without having to write more lines of code since `join()` accepts char sequences such as character or String. In terms of performance, both are much the same.

5. Loops

- a. Nested for loops have a **$O(m*n)$** complexity. There are 2 nested loops which are being used in the algorithm. The first nested loops is used in `convert1dTo2d()`

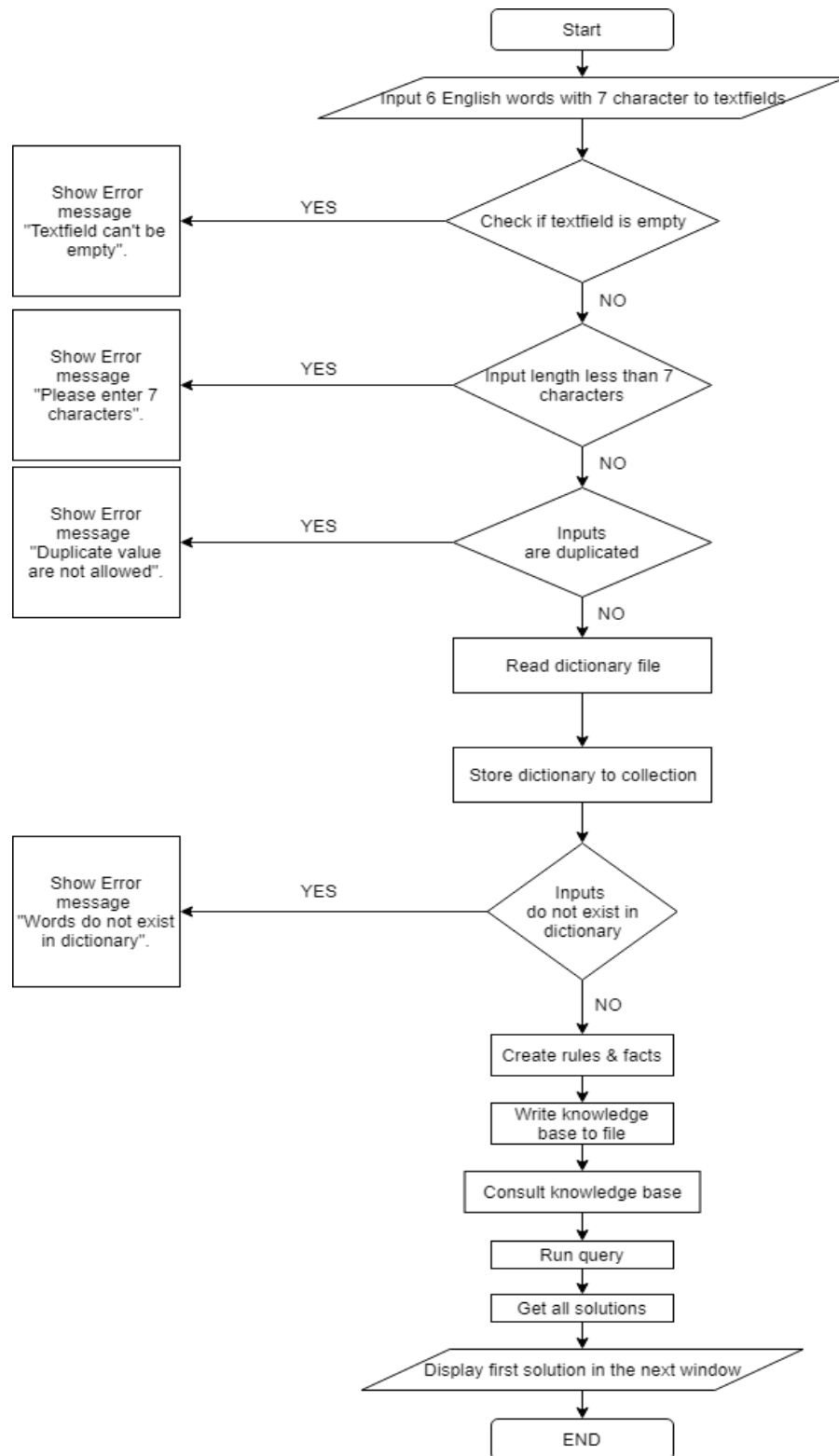
method in order to insert all elements of a one-dimensional String array into a two-dimensional String array to hold all the solutions. The second nested loops are used in `setCharToTextField()` method in order to set characters into all `JTextFields`.

- b. While loops have a **O(n)** complexity and one of its implementation is when reading through a text file that contains 32.909 words line by line and add all the dictionary's words into a `HashSet`. The combination of while loop and `BufferedReader` to read the file are one of the common practices that a developer should follow when reading a large text file.

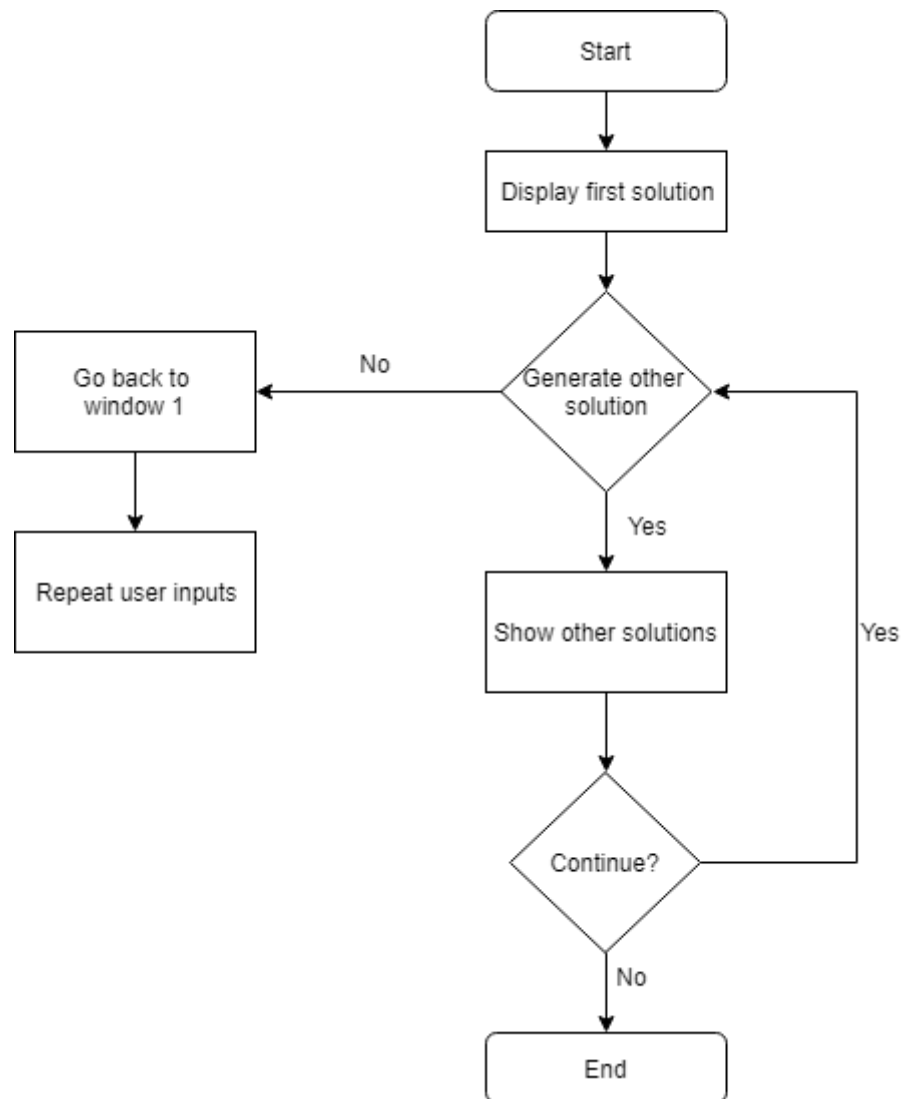
6. Java 8 Streams

- a. Java 8 Streams is a new API published in March 2014 and it has become a new potential use for developers to write cleaner codes. The logic behind streams is very concise and expressive. Streams improves code readability by utilizing a lambda expression to process data declaratively. There are 2 methods are being used which are `filter()` to filter out elements based on a certain condition and `collect()` to combine the elements that have been processed into a collection. These methods are used to check whether the words collected from user inputs exist in the dictionary. Java 8 Stream works best in our scenario where it does not need any micro-optimization or deep performance analysis of the code since it is relatively a small program. However, if the codebase requires extra attention to the micro-optimization, then a simple loop is better than Streams since Streams have another layer of abstraction.
- b. on under the hood which require programmer to do extra work when it comes to error debugging because they have to relate how the code works with Streams.

3.7 Flowchart



Flowchart for Window1



Flowchart for Window2

3.6 Pseudocode

For Window1

BEGIN

GET user inputs

IF user inputs = empty

SHOW error message “Textfields can’t be empty”

ELSE IF length of user input < 7 characters

SHOW error message “Please enter 7 characters”

ELSE IF user input contains duplicate values

SHOW error message “Duplicate values are not allowed”

ELSE

READ dictionary

IF words do not exist in dictionary **THEN**

SHOW error message “Words do not exist in dictionary. Please enter a valid english words”

ELSE

INITIALIZE knowledge-base file

SET user input to lowercase

SET rules and facts

WRITE rules and facts to file

CONSULT knowledge-base file

INITIALIZE query

RUN query

IF query has no solution **THEN**

SHOW error message “No solution”

ELSE

GET all solutions

SEND all solutions to the next window

END IF

END IF

END IF

END

For Window2

BEGIN

INITIALIZE counter as 0

CHOOSE first solution

SET words to characters

STORE characters to textfields

SHOW first solution

INCREMENT solutions counter BY 1

IF counter = 0

CHOOSE second solution

SET words into characters

STORE characters to textfields

SHOW second solution

INCREMENT counter BY 1

ELSE IF counter > 0 AND counter < totalSolutions

CHOOSE next solution

SET words into characters

STORE characters to textfields

SHOW the solution

INCREMENT counter BY 1

ELSE

SET counter to 0

CHOOSE the last solution

STORE words into characters

SET characters to textfields

SHOW the last solution

END IF

SET solutions label TO counter + 1

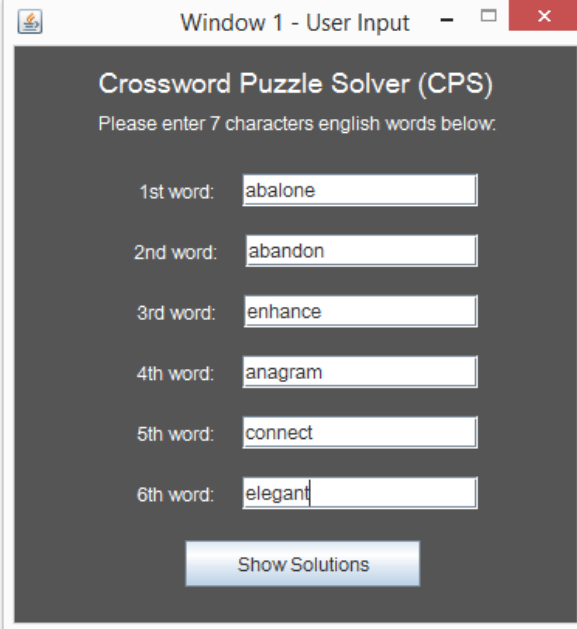
SHOW counter and totalSolutions

END

4.0 RESULTS

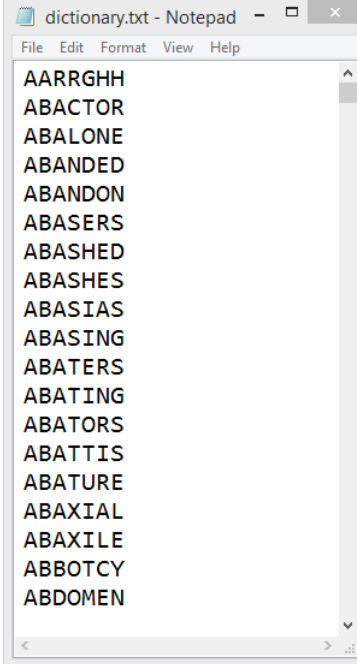
This system that is developed has these features that are listed below:

- i) Taking in input from users (6 English words, 7 letters of each word)



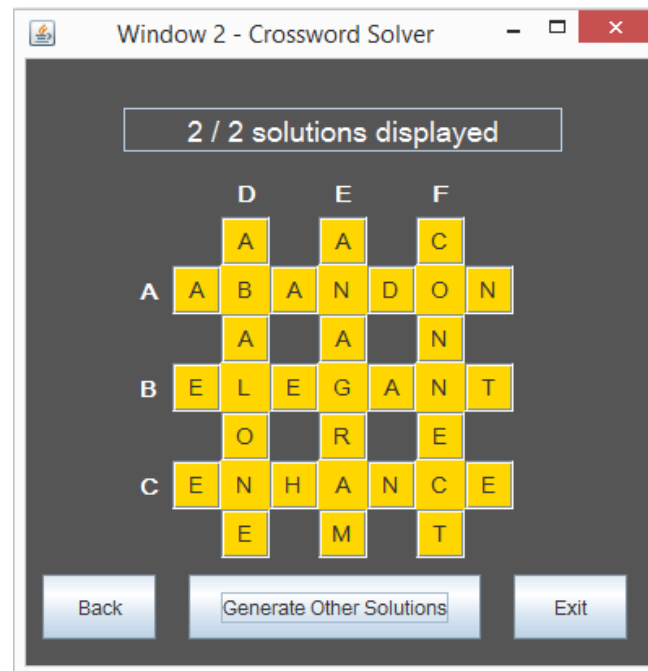
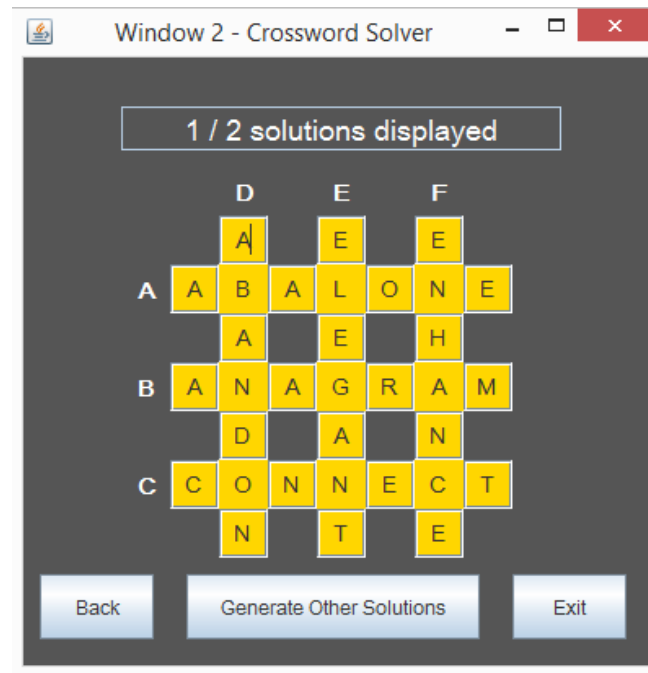
The screenshot shows a window titled "Window 1 - User Input". Inside the window, the title "Crossword Puzzle Solver (CPS)" is displayed at the top. Below the title, a prompt says "Please enter 7 characters english words below:". There are six input fields, each preceded by a label: "1st word:", "2nd word:", "3rd word:", "4th word:", "5th word:", and "6th word:". The input fields contain the words "abalone", "abandon", "enhance", "anagram", "connect", and "elegant" respectively. At the bottom of the window, there is a button labeled "Show Solutions".

- ii) Checking user's input is valid by checking through a given dictionary



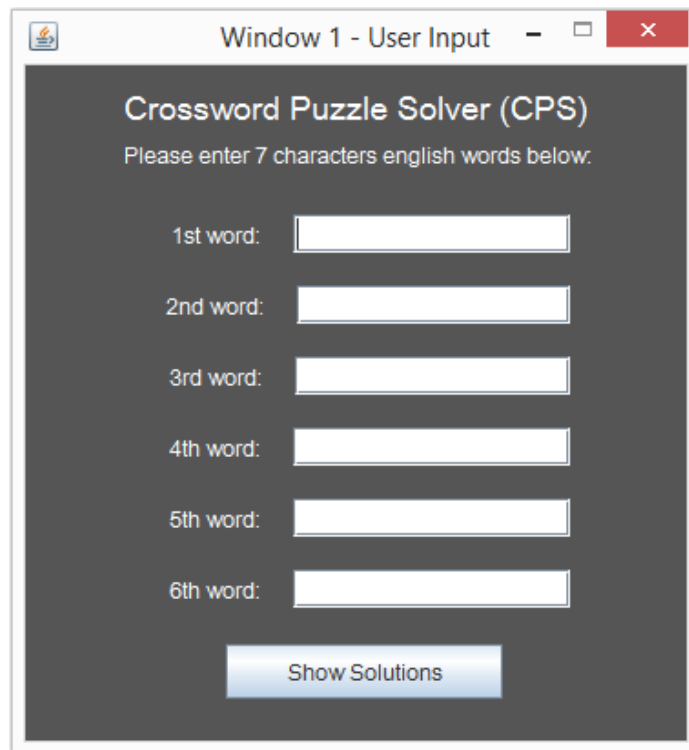
The screenshot shows a Notepad window titled "dictionary.txt - Notepad". The window contains a list of words, all starting with the letter 'A', arranged in alphabetical order. The words are: AARRGHH, ABACTOR, ABALONE, ABANDED, ABANDON, ABASERS, ABASHED, ABASHES, ABASIAS, ABASING, ABATERS, ABATING, ABATORS, ABATTIS, ABATURE, ABAXIAL, ABAXILE, ABBOTCY, and ABDOMEN.

iii) Generating a crossword puzzle based on the words inputted by the user



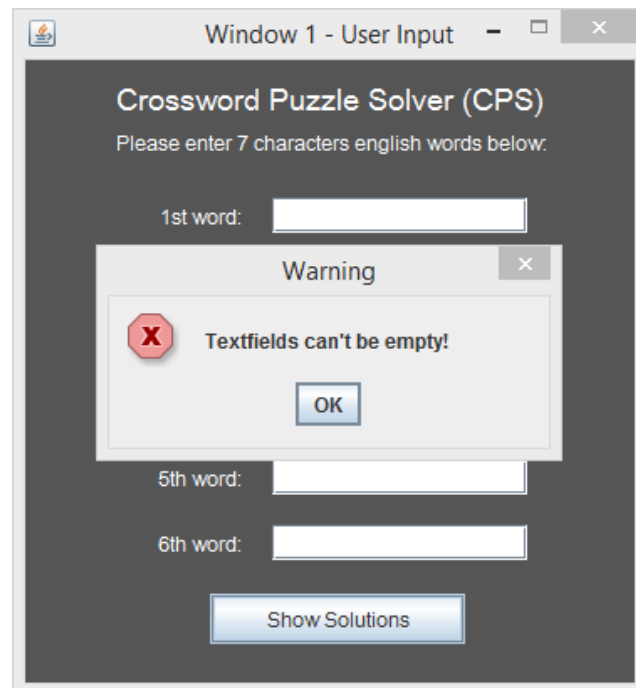
The following diagrams are screenshots of the system that is developed.

- i) The main window that contains 6 JTextFields to receive user input



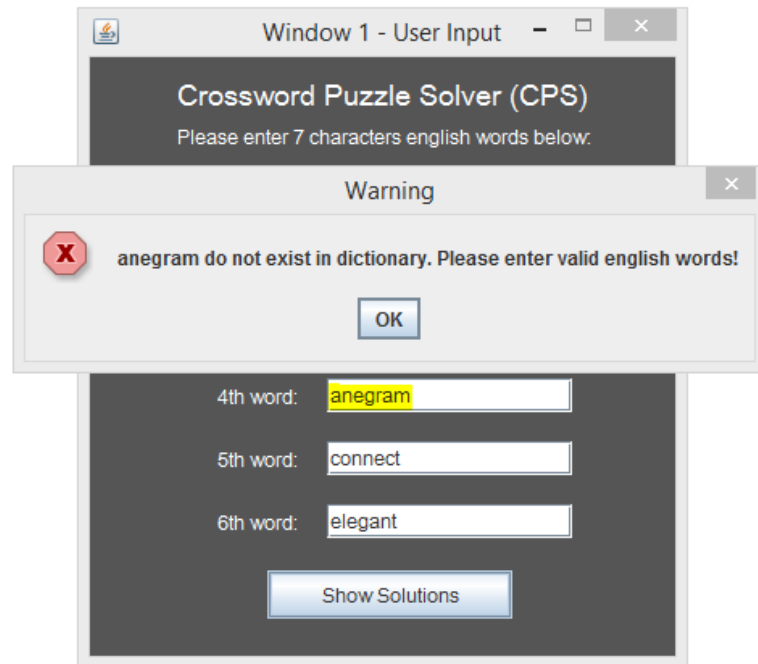
The screenshot shows a Java Swing window titled "Window 1 - User Input". The window has a dark gray background. At the top, it says "Crossword Puzzle Solver (CPS)" in a bold, light blue font. Below that, it says "Please enter 7 characters english words below:" in a smaller, light blue font. There are six text input fields, each preceded by a label: "1st word:", "2nd word:", "3rd word:", "4th word:", "5th word:", and "6th word:". All fields are empty. At the bottom center, there is a blue button with the text "Show Solutions".

- ii) Error validation if "Show Solutions" button is clicked when all JTextField are empty

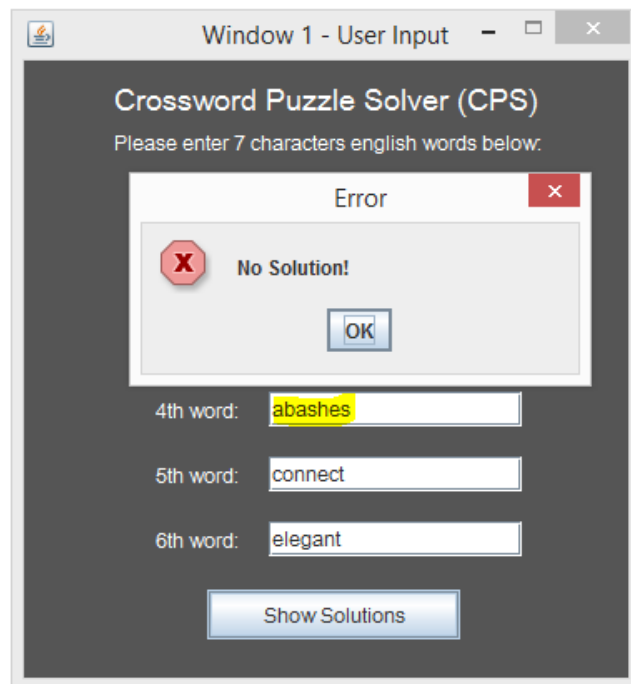


The screenshot shows the same "Window 1 - User Input" window as before, but with an error dialog box overlaid in the center. The dialog box is titled "Warning" and has a red octagonal icon with a white 'X' on the left. The text inside the dialog says "Textfields can't be empty!". There is an "OK" button at the bottom of the dialog. The background window is dimmed, and the "Show Solutions" button is still visible at the bottom.

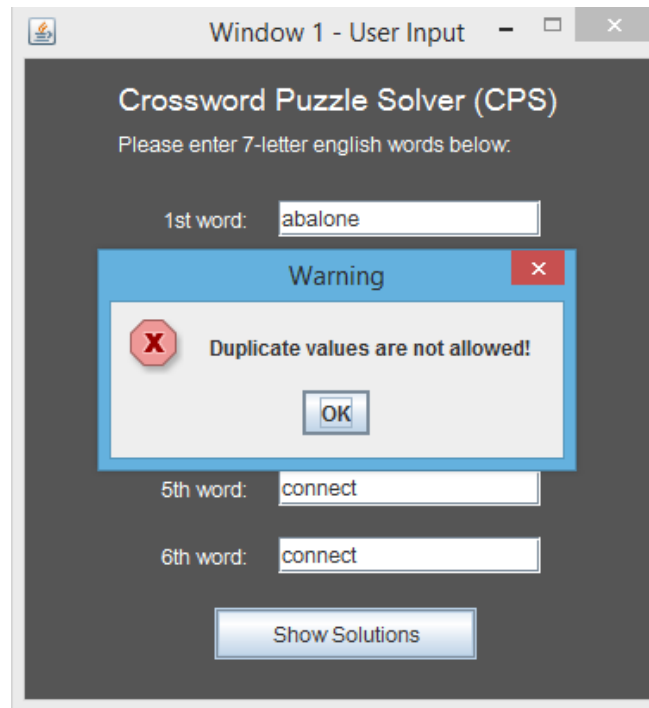
iii) Error validation if “Show Solutions” button is clicked one or more of the words inputted do not exist in the dictionary



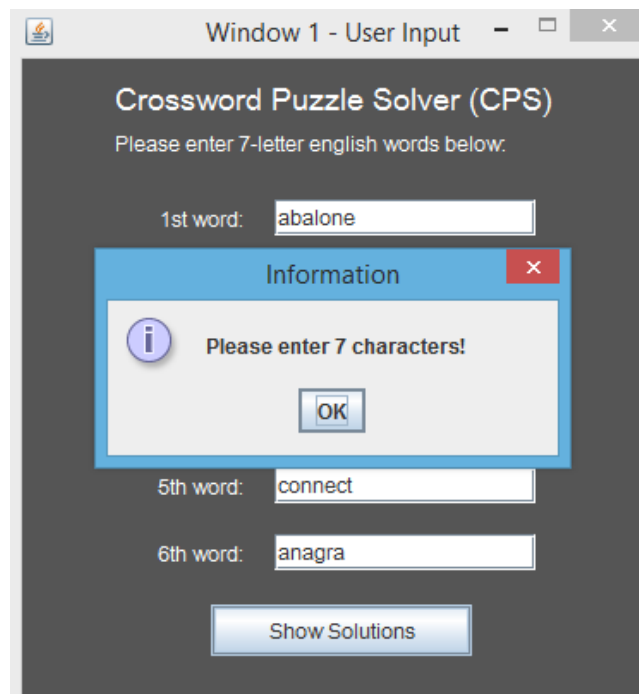
iv) Error validation if “Show Solutions” button is clicked when there is no solution to construct the crossword puzzle for the words inputted by the user



v) Error validation if “Show Solutions” button is clicked if there are any duplicate values found in any of the JTextFields



vi) Error validation if “Show Solutions” button is clicked if the input of users is less than 7 characters. At least 7 characters must be inputted.



5.0 DISCUSSION

There were several challenges that all of us faced when working on the project from the very start to the very end. The very first challenge that we faced was the construction of the knowledge base in PROLOG to solve the crossword puzzle problem. Brainstorming sessions were organised very regularly and we took quite some time actually getting the right knowledge base. This however lead to another challenge as well for the connection for PROLOG to NetBeans IDE that is to be integrated together with Java. Through this was also how we learnt to set up variables and paths, import and connect and code the query through an online source and we all managed to gain something new for us.

The integration of PROLOG and Java was another challenging task and we were trying to figure out how to use suitable data structures such as 2D arrays, ArrayLists, HashSets and HashMaps. It took some time for us to remember how to implement them but we eventually found out the implementation of all of them. The next challenge was the population of words (and each letter of each word) into multiple JTextFields. Because of the overall design of the crossword puzzle, it was quite difficult to generate the entire 42 JTextFields together to form the crossword puzzle.

5.1 Strength of the system

i) User interface design

The first strength of our system that can be seen very clearly is the overall user interface design of system. The design of the interface is extremely easy to use, as the user do not actually have to think and analyse first before using the system. It is straightforward and would not cause any confusion. Users can utilize the main full function of the system by just inputting the words and clicking the button to generate the crossword puzzle.

ii) Real time input validation

One of the prominent strengths that can be seen in our system is the real time input validation in our Window1. Window1 is the main window of our system that enables users to input 6 English words with 7 letters each. This feature is applied to all JTextFields to only allow inputs like BackSpace, Delete and letters which can prevent the tedious normal input validation which can be very time consuming. This is to prevent other characters like numbers and special characters from being entered as it will cause errors when generating the crossword puzzle.

iii) Fast execution time

Next is the execution time of our system. Our system offers fast execution due to small complexity of our overall algorithm. Most of our codes are well-optimized and easy to read as we strive to efficiently use the different types of data structures that can have an impact to our program. Several examples are shown below.

- **Reading of a file**

- i) Java 8 Stream

- ii) Line Iterator

- iii) BufferedReader

These 3 methods are used and compared to *read words* from the dictionary.

```
11 public class testJava8Stream {
12
13     public static void main(String[] args) {
14         long startTime = System.nanoTime();
15         Path file = Paths.get("C:/Users/louisjuliendo/Documents/NetBeansProjects/Prolog/dictionary.txt");
16         try
17         {
18             //Java 8: Stream class
19             Stream<String> lines = Files.lines( file, StandardCharsets.UTF_8 );
20
21             for( String line : (Iterable<String>) lines::iterator )
22             {
23                 System.out.println(line);
24             }
25
26         } catch (IOException ioe){
27             ioe.printStackTrace();
28         }
29         long endTime = System.nanoTime();
30         long elapsedTimeInMillis = TimeUnit.MILLISECONDS.convert((endTime - startTime), TimeUnit.NANOSECONDS);
31         System.out.println("Total elapsed time: " + elapsedTimeInMillis + " ms");
32     }
33 }
```

Output - Prolog (run) X

run:
Total elapsed time: 110 ms
BUILD SUCCESSFUL (total time: 0 seconds)

i) Java 8 Stream

This picture above indicates that usage of Java 8 Stream takes a total of **110 milliseconds** to read words from the dictionary.

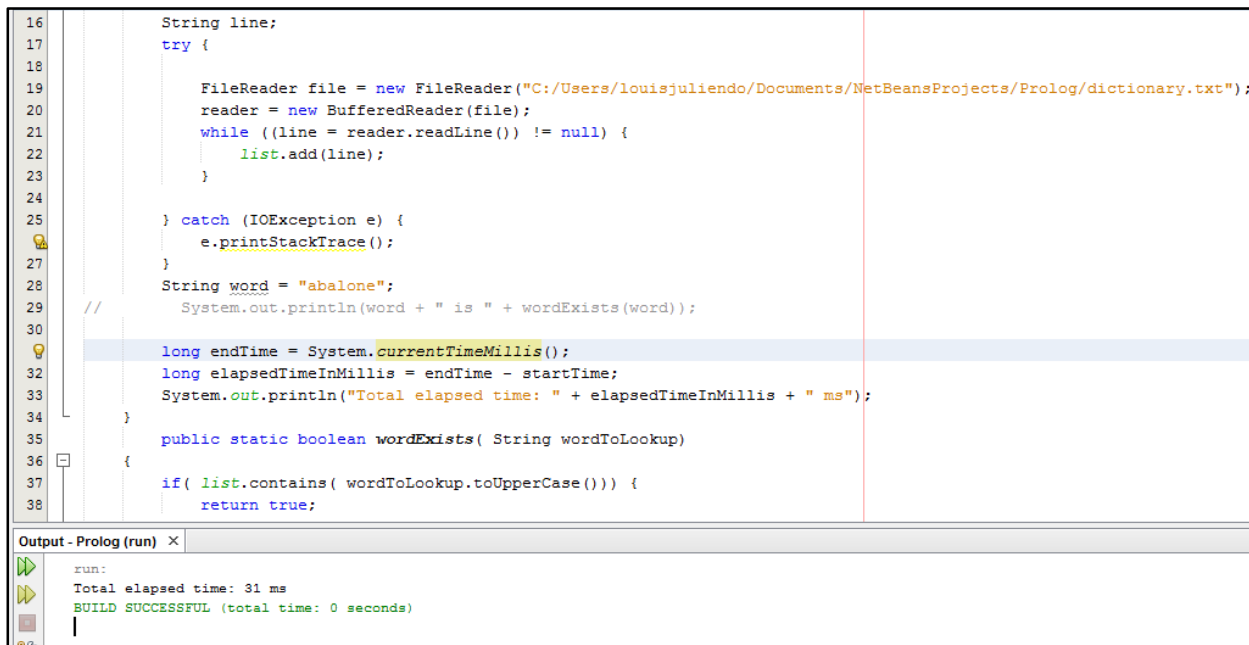
ii) Line Iterator

```
1 import java.io.*;
2 import java.util.concurrent.TimeUnit;
3 import org.apache.commons.io.FileUtils;
4 import org.apache.commons.io.LineIterator ;
5
6 public class testLineIterator {
7     public static void main(String[] args) throws Exception {
8         long startTime = System.nanoTime();
9
10         File file = new File("C:/Users/louisjuliendo/Documents/NetBeansProjects/Prolog/dictionary.txt");
11         LineIterator iterator = FileUtils.lineIterator(file, "UTF-8");
12         try {
13             while (iterator.hasNext()) {
14                 String line = iterator.nextLine();
15             }
16         } finally {
17             LineIterator.closeQuietly(iterator);
18         }
19         long endTime = System.nanoTime();
20         long elapsedTimeInMillis = TimeUnit.MILLISECONDS.convert((endTime - startTime), TimeUnit.NANOSECONDS);
21         System.out.println("Total elapsed time: " + elapsedTimeInMillis + " ms");
22     }
23 }
24
25 }
```

Output - Prolog (run) X

run:
Total elapsed time: 54 ms
BUILD SUCCESSFUL (total time: 0 seconds)

On the other hand, the usage of Line Iterator takes a total of **54 milliseconds** to read words from the dictionary, as shown above.



```
16 String line;
17 try {
18
19     FileReader file = new FileReader("C:/Users/louisjuliendo/Documents/NetBeansProjects/Prolog/dictionary.txt");
20     reader = new BufferedReader(file);
21     while ((line = reader.readLine()) != null) {
22         list.add(line);
23     }
24
25 } catch (IOException e) {
26     e.printStackTrace();
27 }
28 String word = "abalone";
29 // System.out.println(word + " is " + wordExists(word));
30
31 long endTime = System.currentTimeMillis();
32 long elapsedTimeInMillis = endTime - startTime;
33 System.out.println("Total elapsed time: " + elapsedTimeInMillis + " ms");
34
35 public static boolean wordExists( String wordToLookup)
36 {
37     if( list.contains( wordToLookup.toUpperCase())) {
38         return true;
39     }
40 }
```

Output - Prolog (run) x

```
run:
Total elapsed time: 31 ms
BUILD SUCCESSFUL (total time: 0 seconds)
```

iii) BufferedReader

This test has proven that the usage of BufferedReader greatly reduces the time taken of reading and contributes to fast execution of our system. According to the picture above, it takes a total of only **31 milliseconds** to read words from the dictionary.

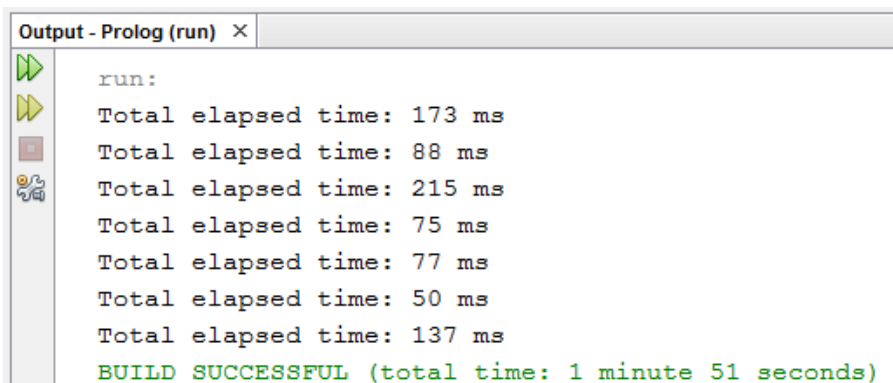
Therefore, for the reading of words in our system, we have implemented the BufferedReader for our program.

- **Program execution time**

Our system execution is considerably fast. The total elapsed time ranges from 50 milliseconds up to 215 milliseconds. The measurement we used is a simple approach using a built-in method to set the **startTime** and **endTime** and then calculate the total elapsed time by subtracting the **endTime** with the **startTime**. The timing begins to elapse when the user clicks the button and it will stop when the button event has executed all the process corresponding to that button. The unit of measurement we have used is milliseconds instead of nanoseconds because it is a small program and does not require much detail regarding the speed performance.

The following shows the method we have implemented and used.

```
long startTime = System.currentTimeMillis();  
long endTime = System.currentTimeMillis();  
long elapsedTimeInMillis = endTime - startTime;  
System.out.println("Total elapsed time: " + elapsedTimeInMillis + " ms");
```



The screenshot shows an IDE output window titled "Output - Prolog (run)". It contains a list of execution results, each preceded by a green double arrow icon. The results are as follows:

Run	Total elapsed time
run:	
1	173 ms
2	88 ms
3	215 ms
4	75 ms
5	77 ms
6	50 ms
7	137 ms

At the bottom of the output window, a green line indicates the build status: "BUILD SUCCESSFUL (total time: 1 minute 51 seconds)".

iv) Dictionary validation

Another strength in our system is the presence of a full dictionary in our system. The dictionary is in a .txt file that contains 32,909 English words. The strength of this function validates every word to check if each word inputted by the users is a valid English word, and also to check if the word is a 7-letter word.

v) Multiple solutions

Another prominent strength in our system is our system is able to display all solutions of a particular crossword puzzle with the given user input. This can benefit users as it gives them more choices to users to solve a particular puzzle with many variations and also to ensure that the logic to solve the crossword is correct.

vi) Better written codes

The overall source code of our program is structured very well. Cleaner codes are constructed and developed through the usage of methods and concepts of object-oriented programming. Also, one-line code such as Java 8 Stream is implemented which can reduce and lessen the repeating chunk of codes.

For example, the populating of JTextFields with characters are done by storing all the JTextFields into an array. It is then populated using two-dimensional array values which results in much cleaner code. This is to avoid the setting of values manually for JTextFields.

List unknownList =

(List)list.stream().filter(x > !dictionary.contains(x)).collect(Collectors.toList());

Above example of Java 8 Stream API methods.

vii) Normal error validation

Besides that, our system also contains error validation. The error validation is present at the very start of running the system, as the system is designed in a way that it catches any errors present in the system. This allows users to help recognize and realize the error that they have done, and to recover from it so that they can proceed in using the system to meet their objectives. For example, if words do not exist in a dictionary, the system will inform the users which particular word does not exist, and will require the user to input another valid words in order to generate a crossword puzzle.

viii) Entertainment use

Lastly, our system is effective and useful for users seeking entertainment. As crossword puzzles are a form of entertainment, this system allows users to solve puzzles much faster, easier and more effectively. This in turn can help reduce anxiety and stress when solving crossword puzzles.

5.2 Limitations and future improvements of the system

The system we developed has indeed achieved its main objectives but there are still many more things to be done as there are certain limitations and some new improvements and features which we wish to add on in the future.

i) Recommendation of words

Our system does not have the ability to recommend similar words for user's choosing. This feature will be beneficial for users when they have inputted invalid words.

ii) Dynamic crossword puzzle

Our system is only restricted to 7-letter words. It would be great to allow users to input any number of letters of words for better flexibility that can solve more puzzles.

iii) Provision of more words in dictionary

Our system only contains a limited dictionary that is listed in a text file. It would be much more effective if there is an increase of vocabulary in our dictionary as it is now only limited to 32,000 words.

iv) Micro-benchmarking

Our system would perform even better if the use of micro-benchmarking and profiling are implemented in the case of the dictionary size becomes larger. Micro-benchmarking is used to measure performance of the system, for example, the execution time of the program which effectively contributes to the productivity of users when utilizing the system.

v) Auto-matching of words

Our system does not have a smart feature that helps users to match words. Having the ability to allow users to enter a wildcard will be good for users with a lower command of English. For example, if a user does not know how to spell the word "abalone", and the user leaves certain letters out like "ab_l_n_", the system should be able to match the missing letters with the ones in the dictionary.

In conclusion, we definitely hope to find other solutions that can improve our system in the future. We hope that more effective improvements can be done such as having a more appealing user interface that can be customized that are catered to more people from different range of age groups so that more people are able to fully utilise our system to achieve their objectives.

6.0 TEAMWORK

Our team consists of 4 people which are as below:

Group Leader:

Louis Juliendo

Group Members:

John Loh Ern-Rong, Chew Cheng Jin, Chai Jun Sheng

There are 2 types of meetings that were conducted which were face-to-face discussions and online discussions. The workload is divided equally through all of the team members to ensure that everyone contributes to the project. Whenever a task is complete, it will be checked by Louis to approve and ensure that there are no errors so that other tasks can be proceeded.

6.1 Planning phase

Before the start of the project, a face-to-face discussion is conducted to discuss the ideas and features that will be added and also to divide tasks for all team members. During this planning phase, we have made many plans for a face-to-face meeting schedules that are set on Fridays and Saturdays, from 10AM until 7PM. Next, we planned on the division of tasks for team members and when the tasks can be completed. Certain platforms are used online during the development of this project such as:

i) **Google Drive** : to upload project codes and report and to update team members with the latest changes

ii) **Facebook** : to communicate with team members

The project officially starts on the **23rd of February 2018**.

6.2 Design phase

Firstly, Cheng Jin started to work on the graphical user interface (GUI) design by following the specified requirements. The GUI is then passed to Louis for further checking to ensure that there are no missing components and to ensure that it follows the specified requirements.

6.3 Implementation phase

Louis started to conduct a detailed study on what are the efficient data structures and methods that should be implemented in order to reduce the code complexity. At the same time, Jun Sheng and John conducted a brainstorming session to discuss and write the PROLOG knowledge base. After the knowledge base has been written, it was passed to Louis for implementation.

Next, Louis creates a few test classes to test on a few methods and data structures such as **testSplitRegex**, **testSplitWordToLetter**, and **testWriteProlog** before starting the actual coding in the main program. After the test classes have been created and tested, Louis started to lead the coding of the program by implementing the data structures and methods that have been written previously in the test classes and he was assisted by John. John provided solutions whenever any error came up and he worked closely together with Louis. Furthermore, John standardized the variable names, add parameterized types, and reformat the code structure.

On the other hand, Cheng Jin and Jun Sheng worked on the error validation and were supervised by Louis. Firstly, Jun Sheng initiated to use the KeyEvent for the input validation but Louis suggested to use DocumentFilter in order to perform real-time validation and also to escape any pasted numeric or special characters as KeyEvent does not have that functionality. Cheng Jin worked closely with Jun Sheng to list out all the necessary errors that need to be validated and coded the error validation. After they have finished with the error validation, it was passed to Louis to implement it to the main program. Lastly, Louis finalized the codes to ensure

that it works correctly and exported the project into a jar file which was later converted into an .exe file using the Launch4J desktop app.

6.4 Documentation phase

In the documentation phase, the report is being written and produced by the end of the project. All team members were assigned parts to work on the documentation which later will be checked by John. Firstly, the introduction and background were written by John. Next, Louis wrote the methodology which includes the pseudocodes, explanation of methods, algorithm analysis and the error validation. Cheng Jin was responsible for the flowchart and the results that contain a brief description and screenshot of our system features and error validation. Jun Sheng was assigned to write for the discussion and conclusion and he was supervised by John.

Louis was also responsible to write for the teamwork as he was the leader of the team who knew the task division among all the team members. After the report has been done, John will proofread and check for any grammatical mistakes and finalize them. Finally, all team members will double-check the report and point out if there are any mistakes that should be changed in order to produce an excellent report.

7.0 CONCLUSION

In conclusion, this entire project has truly benefited all members in the team in many ways. Putting the technical skills aside, time management was something challenging for us since most of us have a different schedules. This has made us more organised in arranging our meetings. Throughout the whole process of completing this project, we have realized how important it is to implement relational languages in software development, that can be useful in solving certain problems. We have also learnt to think differently when coding relationally rather than imperatively most of the time.

We have also learnt the different ways to integrate SWI-PROLOG with an Integrated Development Environment like NetBeans. One of the ways of connecting PROLOG is using the command line, however, we have learnt and chosen another new method which is through a ready made Java library known as JPL. These skills would definitely help in expanding our knowledge in programming which will help in our future careers.

As this was a very challenging project, we have encountered many errors and problems when coding the system. The difficulty level was higher as we needed to implement 2 different languages in our system and integrating them. As a team, we have tried many ways to solve the problems one at a time and all of us have exchanged our ideas to solve every single problem. All in all, we have truly benefited a lot from this project, and this has made us more confident and more certain that this is the starting point for us to achieve greater improvement and greater heights in our career for the years to come.

APPENDIX I

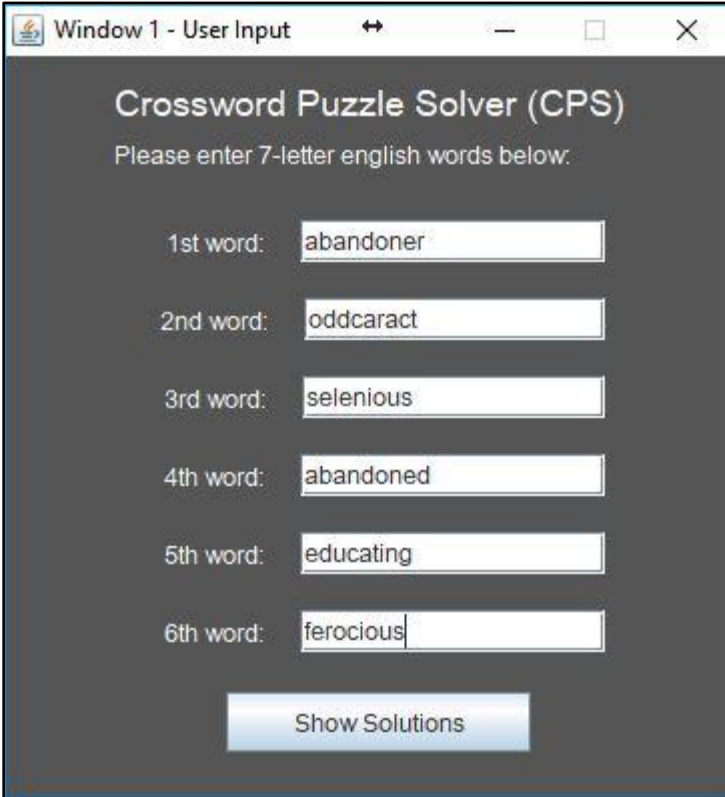
Bonus section

Our Crossword Puzzle Solver System is also able to solve crossword puzzles with 6 different given English words as input with 7, 9 and 11 letters respectively.

The overall workaround of the bonus part of this system is to allow users to input 6 different English words, 7, 9 or 11 letter words of user's choice. However, the number of words are only capped at 6 words. With this, more words are now covered to be able to solve more crossword puzzles

Below are some of the examples shown in the bonus section. Examples given will be 6 user input of valid English words with:

i) 9 letters

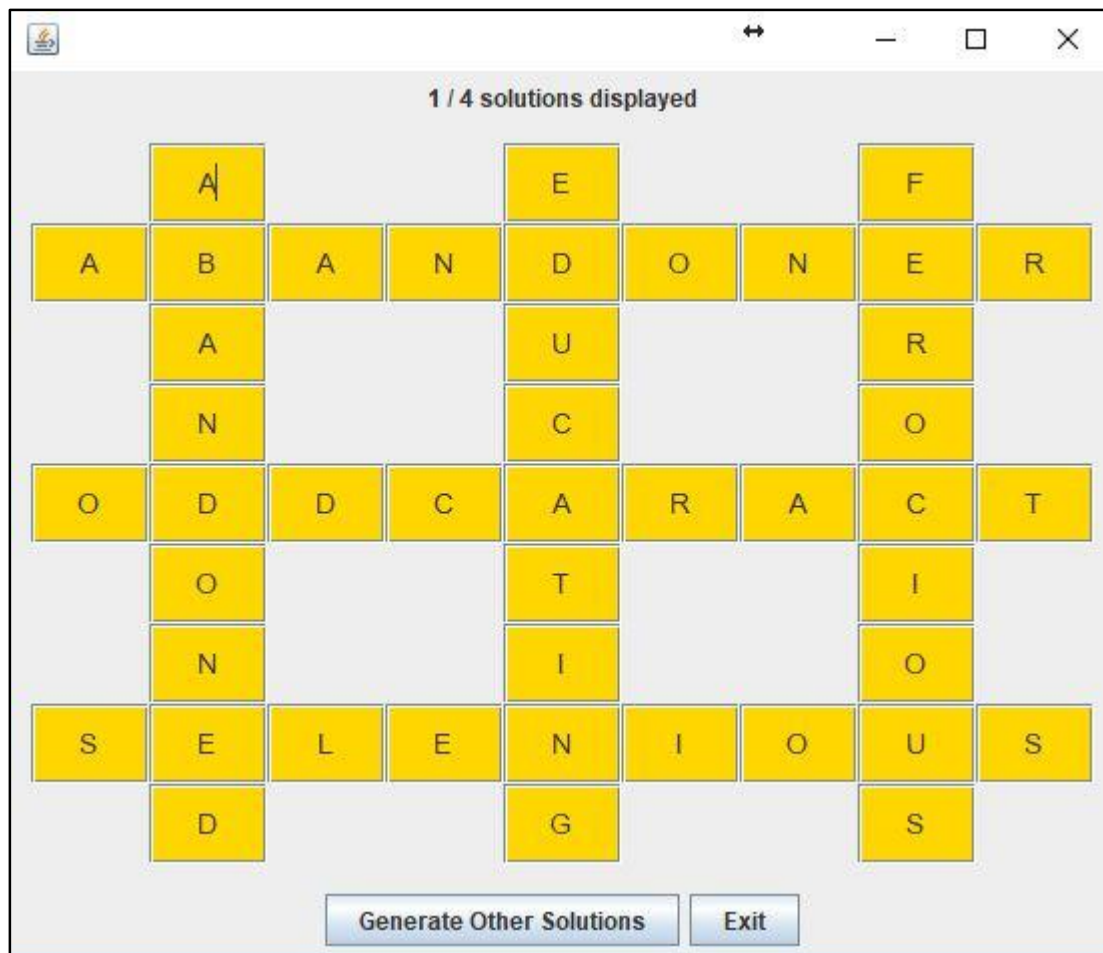


The screenshot shows a window titled "Window 1 - User Input" with a standard Windows title bar. The main content area has a dark gray background and is titled "Crossword Puzzle Solver (CPS)". Below the title, it says "Please enter 7-letter english words below:". There are six text input fields, each preceded by a label: "1st word:", "2nd word:", "3rd word:", "4th word:", "5th word:", and "6th word:". The inputs are: "abandoner", "oddcaract", "selenious", "abandoned", "educating", and "ferocious". At the bottom, there is a blue button labeled "Show Solutions".

Word Index	Input
1st word:	abandoner
2nd word:	oddcaract
3rd word:	selenious
4th word:	abandoned
5th word:	educating
6th word:	ferocious

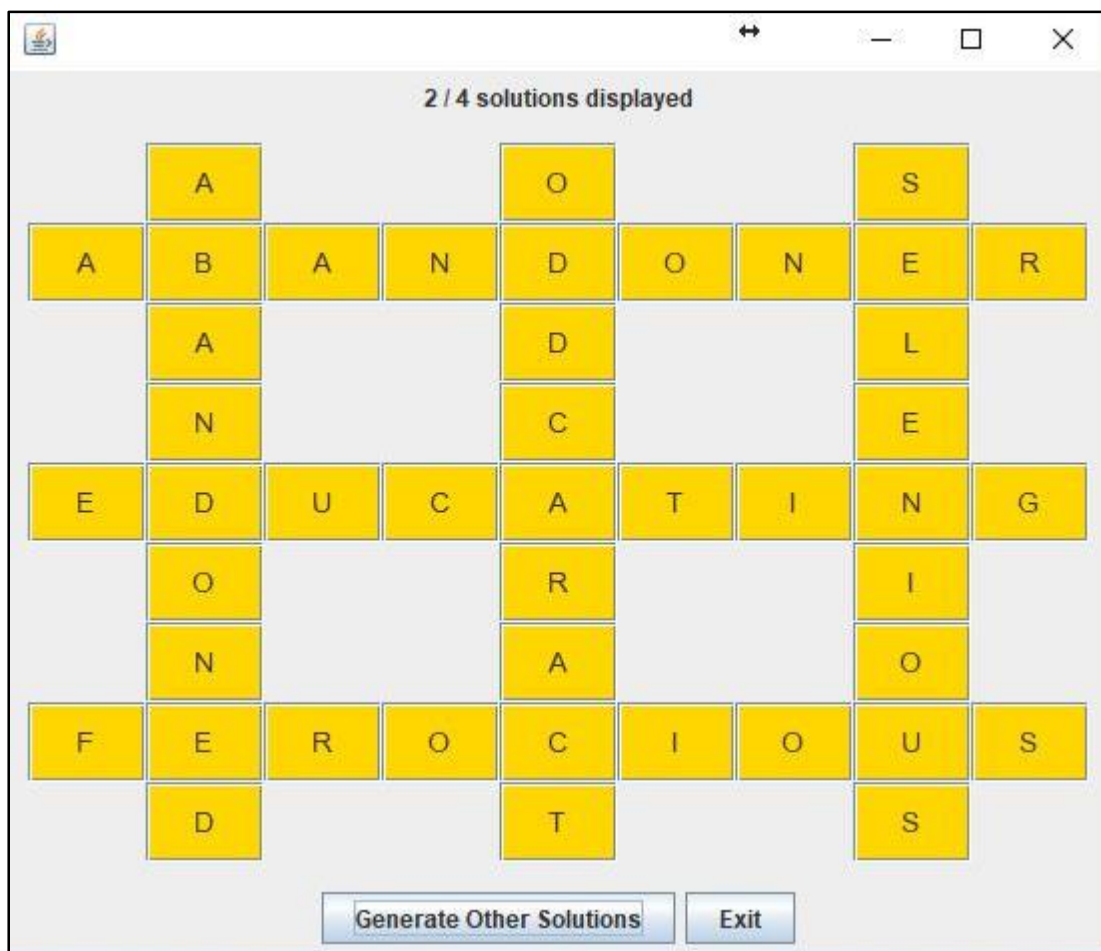
Show Solutions

The picture example above shows the 6 input words that are used to solve the crossword puzzle.

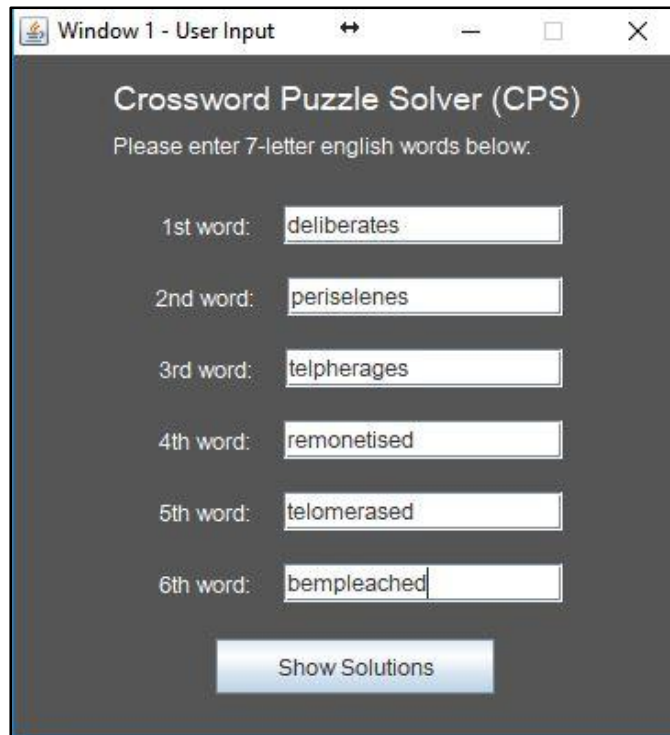


Upon clicking the “Show Solutions” button, all solutions will be generated and shown in the window. Based on the above example, there is a total of 4 solutions being generated. User can click on “Generate Other Solutions”, to display the 2nd solution, right up till the 4th.

The below picture shows an example of the display of the 2nd solution.



ii) 11 letters

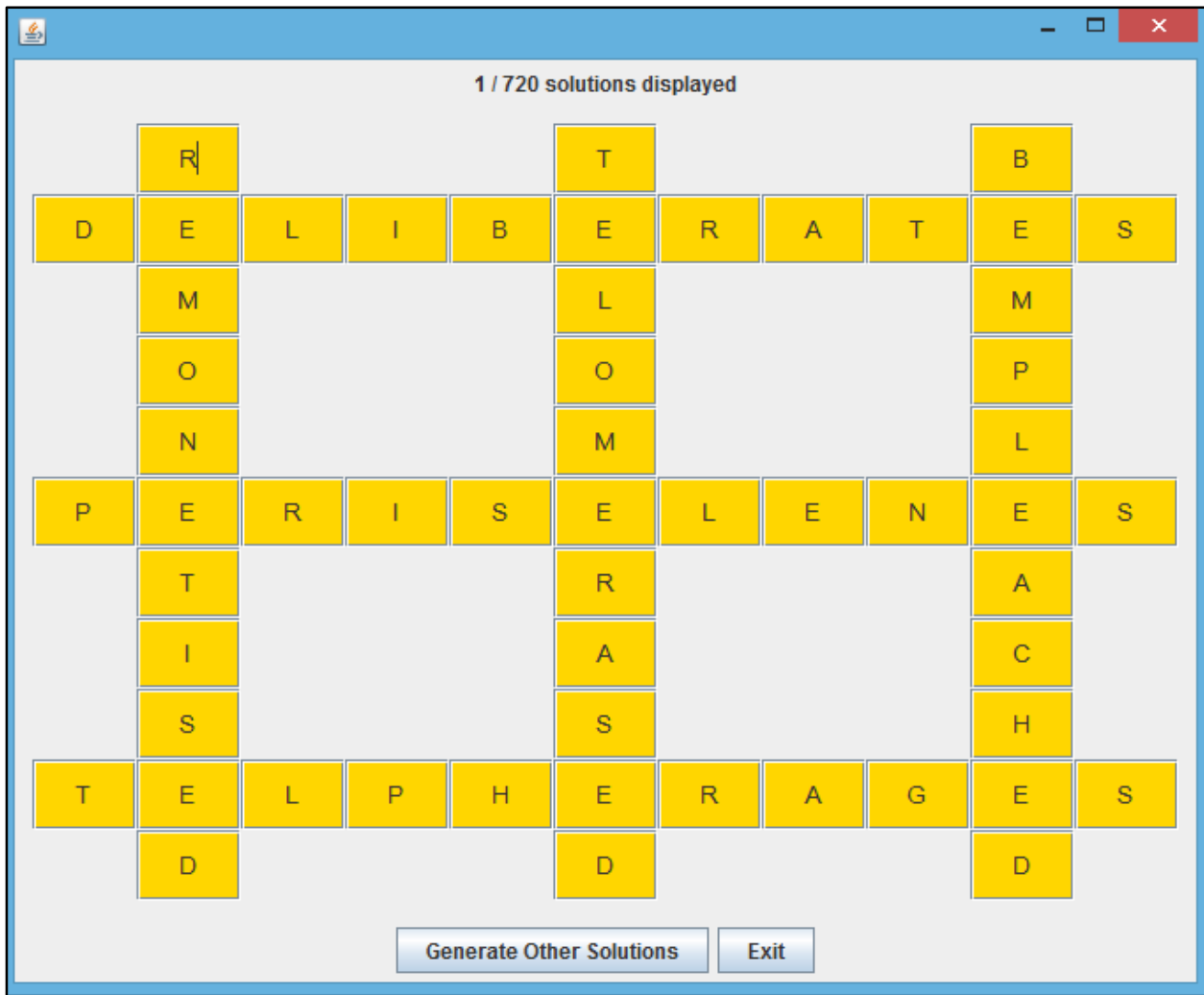


The image shows a software window titled "Window 1 - User Input" with standard window controls (maximize, minimize, close). The main content area has a dark background and is titled "Crossword Puzzle Solver (CPS)". Below the title, it says "Please enter 7-letter english words below:". There are six input fields, each preceded by a label: "1st word:", "2nd word:", "3rd word:", "4th word:", "5th word:", and "6th word:". The words entered in the fields are "deliberates", "periselenes", "telpherages", "remonetised", "telomerased", and "bempleached" respectively. A "Show Solutions" button is located at the bottom center of the window.

Word Index	Word
1st word:	deliberates
2nd word:	periselenes
3rd word:	telpherages
4th word:	remonetised
5th word:	telomerased
6th word:	bempleached

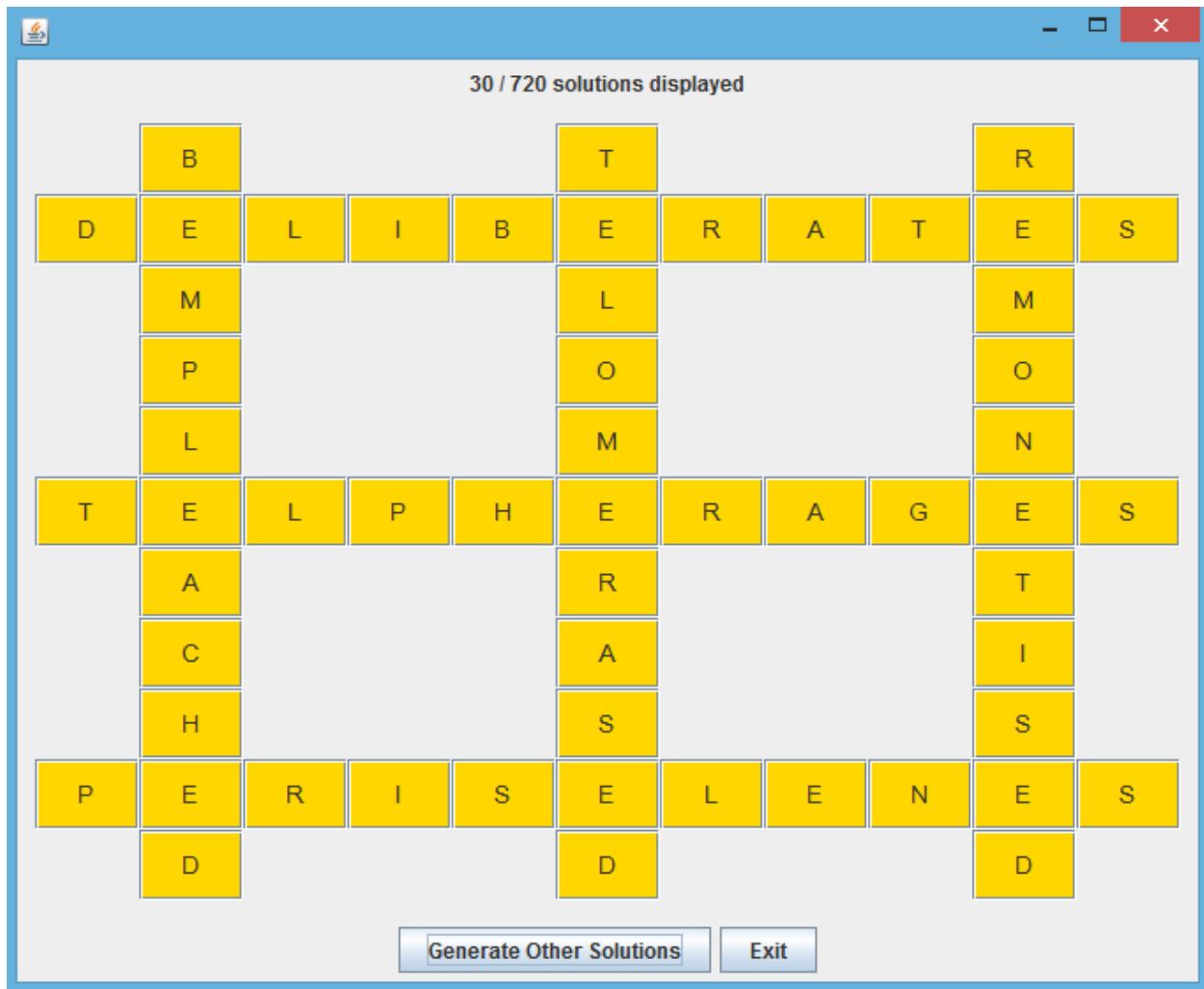
Show Solutions

The picture example above shows the 6 input words that are used to solve the crossword puzzle.



Upon clicking the “Show Solutions” button, all solutions will be generated and shown in the window. Based on the above example, there is a total of 720 solutions being generated. User can click on “Generate Other Solutions”, to display the 2nd solution, right up till the 720th.

The picture below shows an example of the display of the 30th solution.



In conclusion, this bonus section of our system can be referred to as a dynamic crossword solver that can generate 7, 9 and 11 letters of 6 English. This is so that the end user can be able to solve more crossword puzzle solutions which provide more flexibility.