CHAPTER V
IMPLEMENTATION AND TESTING

5.1 Implementation

5.1.1 GADDAG Pattern

On this project using the Java programming language. When the program start, first time program will read dictionary to store on the data structure. While program read each word on dictionary, each word will form into GADDAG pattern. On the GADDAG algorithm, a word will be formed into some form that has a pattern (Pattern GADDAG can be seen in Figure 3). Here is the code to generate the pattern.

```
public String[] generate(String str2){

  String str = str2.toUpperCase();
  String[] result = new String[str.length()];
  String depan = ">" ,belakang = "";
  int prefix = 1;

  for(int i=0;i<str.length();i++){
    belakang = str.substring(prefix);
    depan = str.charAt(prefix-1) + depan;

    result[i] = depan + belakang;

    System.out.println(result[i]);
      trie.addWord(result[i]);
      System.out.println("====");

    prefix++;
  }
  return result;
}
```

Figure 12: Code to establish a pattern of the GADDAG

5.1.2 Insert Trie

After find the GADDAG pattern of a word then each pattern will be insert into the data trie structure. Look figure below for the insert word to trie data structure.

```
public void addWord(String newWord){
    NodeTrie tmpRoot = root;
    boolean isEOW = false;
    for (int i=0;i<newWord.length() ;i++ ) {
        NodeTrie returnNodeTrie = getChild(newWord.charAt(i),tmpRoot);
        if( i == (newWord.length()-1) || (newWord.charAt(i+1) == '>' && (newWord.length()-2) == i))
            isEOW = true;
        else
            isEOW = false;
        if(newWord.charAt(i) == '>' && isEOW)
            isEOW = false;
        if(returnNodeTrie==null){
            System.out.println("Tambah Kaki "+newWord.charAt(i)+isEOW+" di "+tmpRoot+", turun ke tingkat berikutnya ");
            tmpRoot = tmpRoot.addChild(newWord.charAt(i),isEOW,i+1);
        }else{
            tmpRoot = returnNodeTrie;
            System.out.println(returnNodeTrie +" Ditemukan di "+tmpRoot+", turun ke tingkat berikutnya ");
            if(!tmpRoot.getArti())
                tmpRoot.setArti(isEOW);
        }
    }
}
```

Figure 13: Code for Store Word into Trie Data Structure.

```
W>ORD
W Item Tidak Ditemukan.
Tambah Kaki Wfalse di [7ce1ef11,*,false,0], turun ke tingkat berikutnya
> Item Tidak Ditemukan.
Tambah Kaki >false di [4ba6b85d,W,false,1], turun ke tingkat berikutnya
O Item Tidak Ditemukan.
Tambah Kaki Ofalse di [2e920878,>,false,2], turun ke tingkat berikutnya
R Item Tidak Ditemukan.
Tambah Kaki Rfalse di [756a162a,O,false,3], turun ke tingkat berikutnya
D Item Tidak Ditemukan.
Tambah Kaki Dtrue di [7d55b9f,R,false,4], turun ke tingkat berikutnya
====
OW>RD
O Item Tidak Ditemukan.
Tambah Kaki Ofalse di [7ce1ef11,*,false,0], turun ke tingkat berikutnya
W Item Tidak Ditemukan.
Tambah Kaki Wfalse di [487227bd,O,false,1], turun ke tingkat berikutnya
> Item Tidak Ditemukan.
Tambah Kaki >false di [3e8f0e73,W,false,2], turun ke tingkat berikutnya
R Item Tidak Ditemukan.
Tambah Kaki Rfalse di [21fdc01b,>,false,3], turun ke tingkat berikutnya
D Item Tidak Ditemukan.
Tambah Kaki Dtrue di [1a4f3e0c,R,false,4], turun ke tingkat berikutnya
====
ROW>D
R Item Tidak Ditemukan.
Tambah Kaki Rfalse di [7ce1ef11,*,false,0], turun ke tingkat berikutnya
O Item Tidak Ditemukan.
Tambah Kaki Ofalse di [48729352,R,false,1], turun ke tingkat berikutnya
W Item Tidak Ditemukan.
Tambah Kaki Wfalse di [5a2611a6,O,false,2], turun ke tingkat berikutnya
> Item Tidak Ditemukan.
Tambah Kaki >false di [7950d786,W,false,3], turun ke tingkat berikutnya
D Item Tidak Ditemukan.
Tambah Kaki Dtrue di [2a75dca3,>,false,4], turun ke tingkat berikutnya
====
DROW>
D Item Tidak Ditemukan.
Tambah Kaki Dfalse di [7ce1ef11,*,false,0], turun ke tingkat berikutnya
R Item Tidak Ditemukan.
Tambah Kaki Rfalse di [1fe8671c,D,false,1], turun ke tingkat berikutnya
```

Figure 14: Log of Insert Into Trie

The example is if program want insert “WORD” to the data structure look on figure 14. For easily imagine the trie structure look at the memory adders from log. Program will search child of root where valued ”W”. If found the root change into child which valued “W”. After that “O” if not found program will create new node valued “O” as a child then root become that child and soon.

5.1. Search Word

There are 2 main procedure in the GADDAG algorithm there is Gen and Goon. Function procedure the Gen is put up each letter on the rack to check on the procedure Goon. Goon used to check letter to data structure is a letter can be formed a word. These 2 procedure is run recursively. Below are the log of searching word where “OR” on the board and “WD” rack from player.

```
 ANCHOR SQUARE 0
FrameCoba : gaddag.Gen(0,'',WD,[7950d786,O,false,1],[X:6,Y:7,yParent:7,xParent:7,H:true]);
Rack ke 0 level 0 Papan 7,6 Dipasang huruf W POS:0
1 Huruf [2a75dca3,W,false,2] Bisa dipasang di 7,6 Karena ada di [7950d786,O,false,1]
checkProsesingWord = false
Rack ke 0 level 2 Papan 7,5 Dipasang huruf D POS:-1
3 Huruf D Tidak Bisa dipasang di 7,5 Karena null di [2a75dca3,W,false,2]
2 RECORD :
 Cek huruf di depan parent
3 Huruf [1fe8671c,>,false,3] Bisa dipasang di 7,5 Karena ada di [2a75dca3,W,false,2]
Posisi jadi 7,8
checkProsesingWord = false
Mencari R di [1fe8671c,>,false,3]
5 Huruf [52c96fa1,R,false,4] Bisa dipasang di 7,8 Karena ada di [1fe8671c,>,false,3]
checkProsesingWord = false
Rack ke 0 level 6 Papan 7,9 Dipasang huruf R POS:2
7 Huruf R Tidak Bisa dipasang di 7,9 Karena null di [52c96fa1,R,false,4]
Rack ke 1 level 6 Papan 7,9 Dipasang huruf D POS:2
7 Huruf [1626ac0,D,true,5] Bisa dipasang di 7,9 Karena ada di [52c96fa1,R,false,4]
Menemukan Arti pada [1626ac0,D,true,5]
=cekJalurPerHuruf= word: W>RD
AnchorSquare : [X:6,Y:7,yParent:7,xParent:7,H:true]
Y : 7|X : 6
Papan 8,6 dan 6,6 kosong maka hasil true
Y : 7|X : 8
Papan 8,8 dan 6,8 kosong maka hasil true
Y : 7|X : 9
Papan 8,9 dan 6,9 kosong maka hasil true
=EndcekJalurPerHuruf=true
CEK JALUR : 2,7,9,4,[X:6,Y:7,yParent:7,xParent:7,H:true],Papan 7,5 Kosong
Papan 7,10 Kosong
=getBonusValue=W>RD
PAPAN : 7,6=>W:4 BONUS :
PAPAN : 7,8=>R:1 BONUS :
PAPAN : 7,9=>D:2 BONUS :
PAPAN : 7,7=>O:1 BONUS : DW
=END OF getBonusValue=
W>RD Memiliki Arti [Y:7,X:6] - W>RD = 16
 Hasil Cek Jalur : true
checkProsesingWord = false
Rack ke 0 level 8 Papan 7,10 Dipasang huruf R POS:3
```

Figure 15: Log of Searching Word

5.2 Testing

Testing will do in several condition there are add behind of word, add in front of word and both. The first experiment is adding a letter behind the word. Examples on the board are the letters G and O, then the letter in hand are N and E. The following are examples of conditions the board on figure 16.

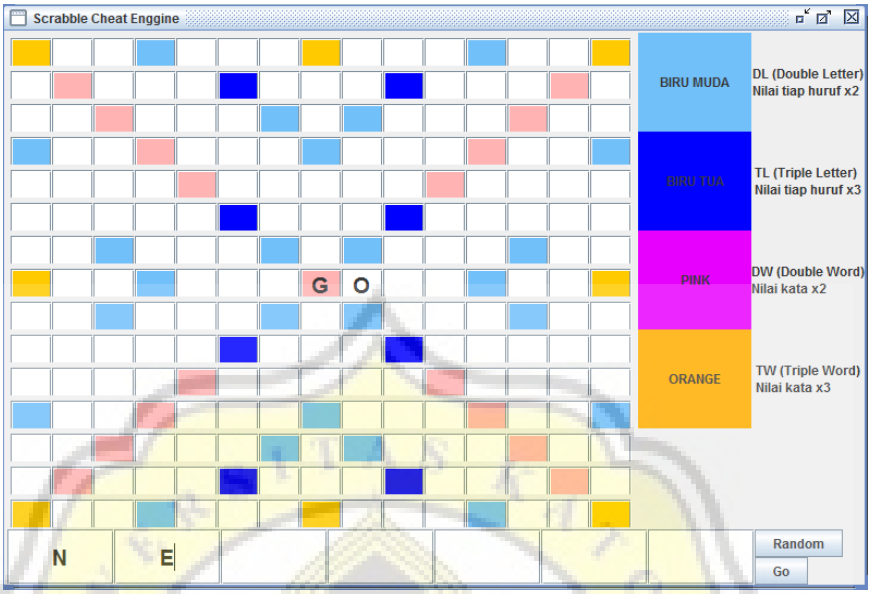

Figure 16: Gone Example

The search process will begin with a word search for the coordinates of the board to be mounted letter (Anchor Square). In each of these coordinates the program will find all the possible words that can be formed. Here is the result of conditions the board.

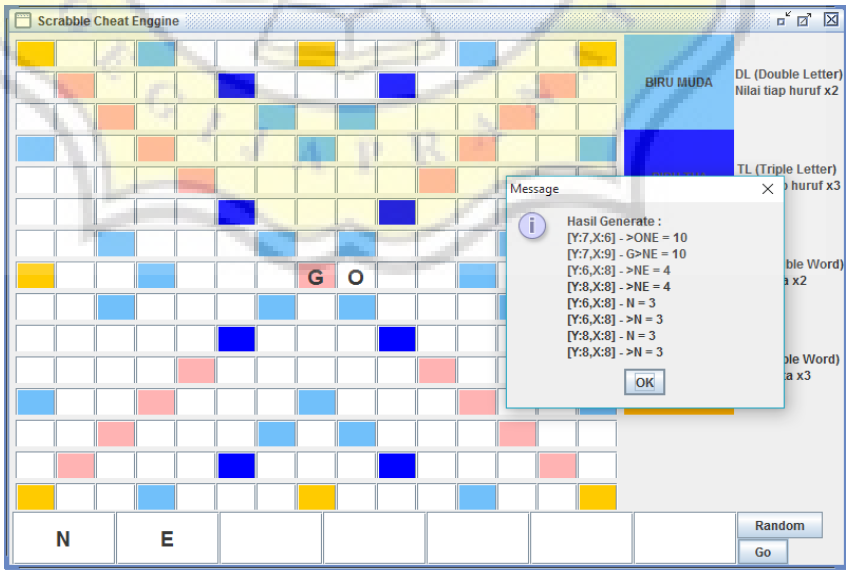

Figure 17: Gone Example 2

After the program is finished searching for all the words, the program will display all words are formed from each coordinate. In the above example the coordinates of the line 7 columns 6 has the highest value. After that program will update tho condition of board, look at figure 18

Figure 18: Result Gone

The second experiment are when put behind of a word on the board. In this example, the board has the letters O and R. Then the letter in hand O and P. Here is a picture of the condition of the board.

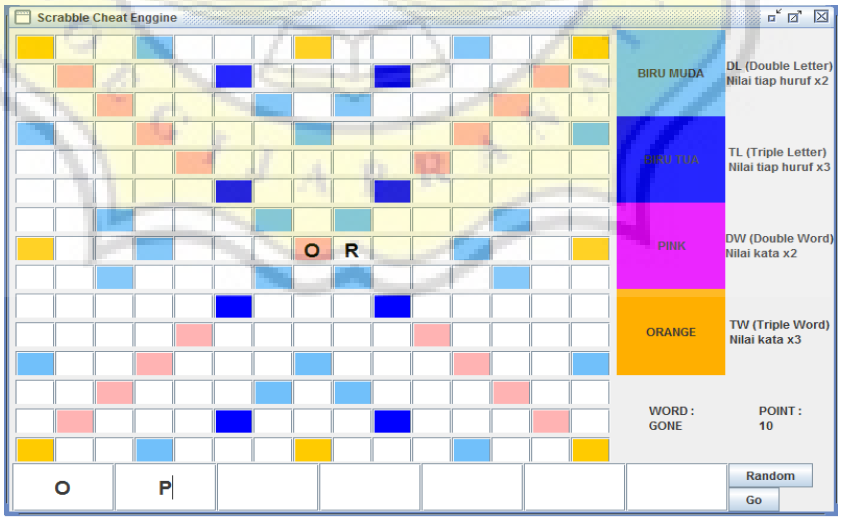

Figure 19: Poor Example

Just like the previous process, the program will first find the coordinates that might be to set a letter. On the boards above conditions produce results as shown below.

Figure 20: Result Poor Example

On the third experiment, GADDAG algorithm can also pair the letter in front of and behind the words on the board.

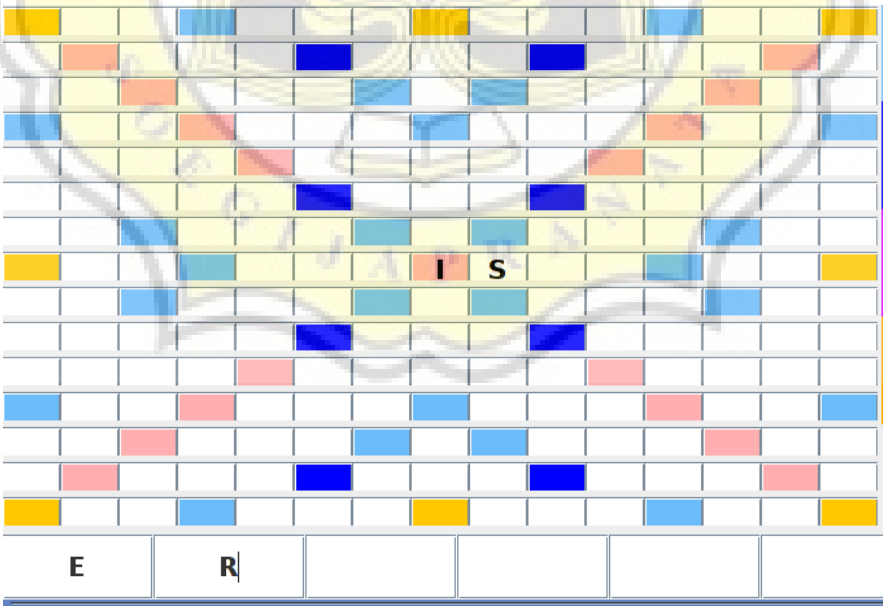

Figure 21: Rise Example

Below is the result for figure 22 condition

Figure 22: Result Rise Example

The last experiment is cross the word with other word. On the condition of the board over the letter which is owned by the I and L look at figure 23.

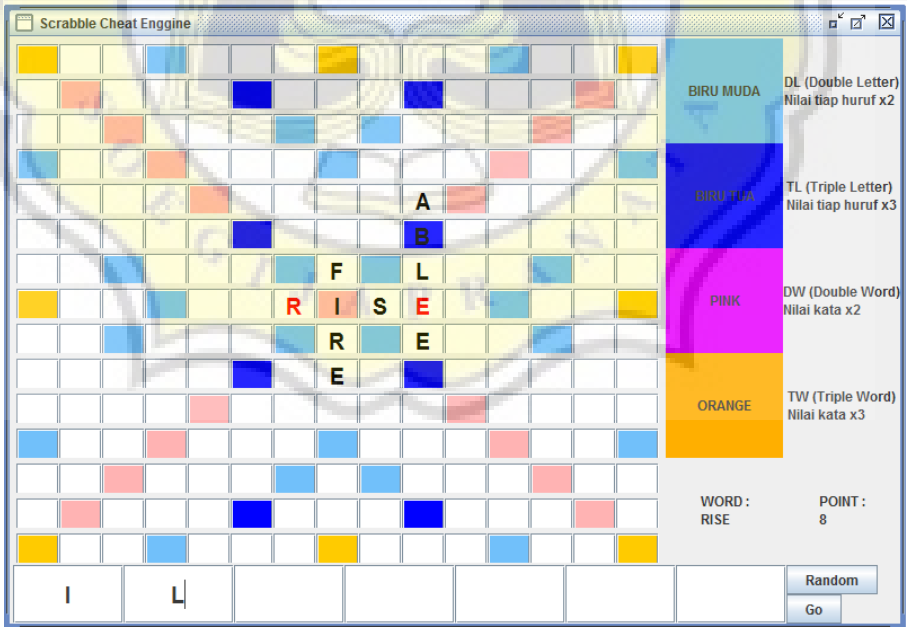

Figure 23: Cross Example

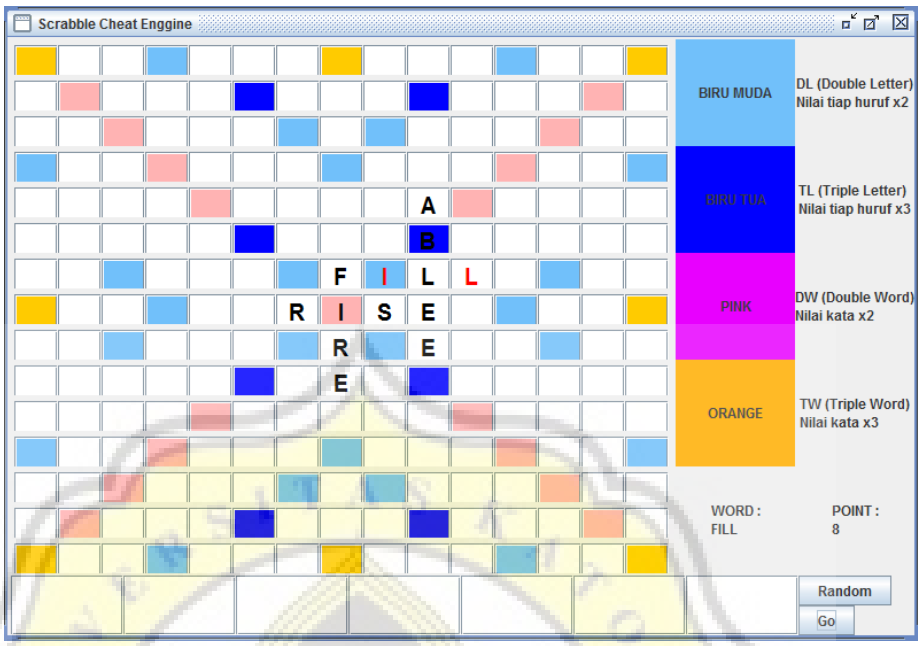

Figure 24: Result Cross Example

In the above conditions the program generates word "FILL" which is in line with the word "RISE". This result is correct because the letter I beside with letter S and form a word of "IS".