CHAPTER V
IMPLEMENTATION AND TESTING

5.1. Implementation

At first, the program will ask some input from the user. The inputs are number of words that will be placed in genetic algorithm, number of individual in a population, and chance of mutation. Number of individual and chance of mutation will be used in the algorithm. User can change the number of individual in population and mutation chance so that the user can experiments the algorithms with ease.

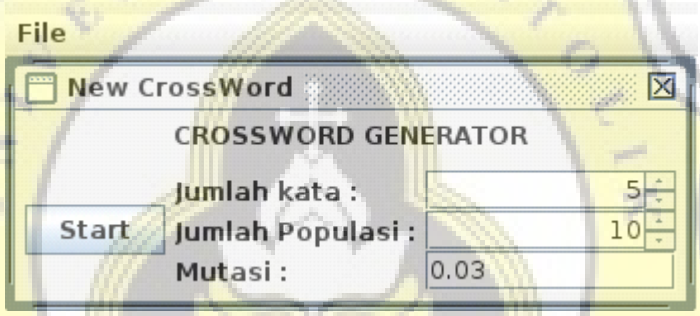

Figure 11. Interface for Input

After user fill or change the field the program will store the data in an object named Request. Then for the next step, program will ask user to fill the table with words that wanted to be placed in the crossword in column named Words and the clue of the words in column named Clue. Number of rows shows in the table depends on how many number of words the user ask in the previous step.

Figure 12. Interface Table Input

After filling the tables the program will store the values in 2-dimensional array. Besides 2-dimensional array the data will be saved in file named data.txt.

Table 1. Two Dimensional Array

[x][y]	[x][0]	[x][1]
[0][y]	MOBIL	car
[1][y]	RUMAH	house
[2][y]	PERAHU	boat
[3][y]	MATAHARI	sun
[4][y]	LEMARI	cupboard

After store the words, the program will start the genetic algorithm process. The process will start at creating a population. In initialization process an individual created by merging 2 arrays. The first array consists of shuffled number from 0 until number of words. And for the second array consist of true and false value. Number of the value is as much as number of words. The value true in the second array mean horizontal and false

mean vertical. This direction will be used to determine how the word will be placed in the board. The merging for an individual is simple. The first gen of an individual will using the first array at index 0 and second array also at index 0. The second gen will be created by first array at index 1 and second array at index 1 too and so on. The process of creating individuals will be repeated until number of individual in a population meet.

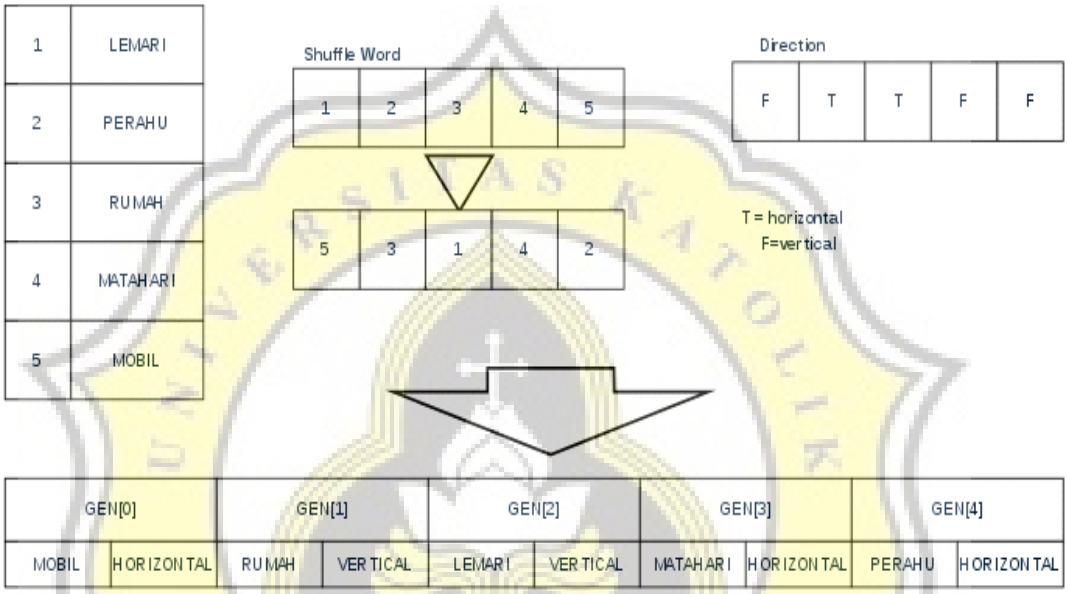

Figure 13. Creating an Individu

In Code 1 shows the code of the initialization process. The code use Collections to shuffle the array of Integer. Since method shuffle in Collections only accept a list of Integer, the code use tool Arrays to convert the array into a list.

C

```
public void initialization()
{
        populasi = new Individu[jumlahPopulasi];
        Integer[] for_shuffle = new Integer[req.getjmlhkata()];
        boolean[] arah = new boolean[req.getjmlhkata()];
        for(int i=0;i<req.getjmlhkata();i++){
                for_shuffle[i] = i;
        }
        for(int x=0;x<jumlahPopulasi;x++){
                for(int i=0;i<req.getjmlhkata();i++){
                        if(randomNum.nextInt(2)==0){
                                arah[i] = true;
                        }
                        else{
                                arah[i] = false;
                        }
                }
                Collections.shuffle(Arrays.asList(for_shuffle));
                populasi[x] = new Individu(for_shuffle,arah,words);
        }
}
```

Code 1. Initialization

After creating a population the next step is to count the fitness of each individual. Since all individual's fitness has to be count, the counting will be made into a thread. And whenever an individual created, the thread will also be created to find its fitness value.

In the Genetic Algorithm to determine whether an individual is good or not is by looking at their fitness value. So Every time an individual created the program will find the fitness value. To find an individual's fitness the program will try to place each gen (consist of words and direction) in a temporary board. The fitness value of an individual will depend on how many words that successfully placed in the board. More words placed in a board mean higher fitness value.

The first step in counting the fitness is creating a 2-dimensional array of char. This 2-dimensional array will be the board. The size of the board will be 50 x 50. After the board prepared, the program will start to place the words sequentially. The first words will be placed in the middle of the board. Let say the first gen we get have a word "MOBIL" placed

horizontally. Letter M in the word “MOBIL” will be put in the board at position 25,25. And then the program will place the next letter which is O in the position (i,j+1) which is 25,26. And go on until all the letter in that word placed.

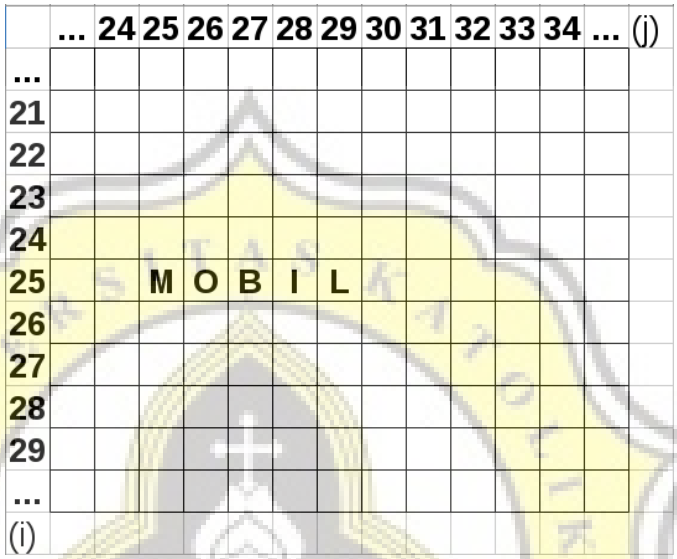

Figure 14. Placing the first word

Besides placing it on the board the word will be saved in array of list. The process will have 2 array of list which are list for horizontal words and list for vertical words. From the case above the program will store word “MOBIL” in horizontal list. See picture 5.5.

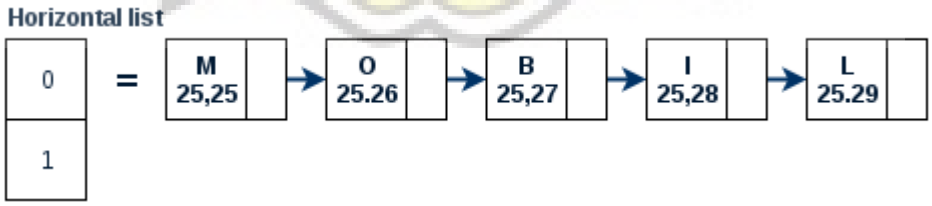

Figure 15. Array of Horizontal List

Because the first word placed successfully, the fitness value will be added by 1. So for now the fitness of placing the first words will be 1. Next, the second gen contains word “RUMAH” placed vertically. Since vertical word can only be ‘checked’ or intersected with horizontal word, the program will read the horizontal list and check each letter in the list which letter that can be intersected. The program found that letter M here can be intersected. Then it will try to placed the word “RUMAH” into the board.

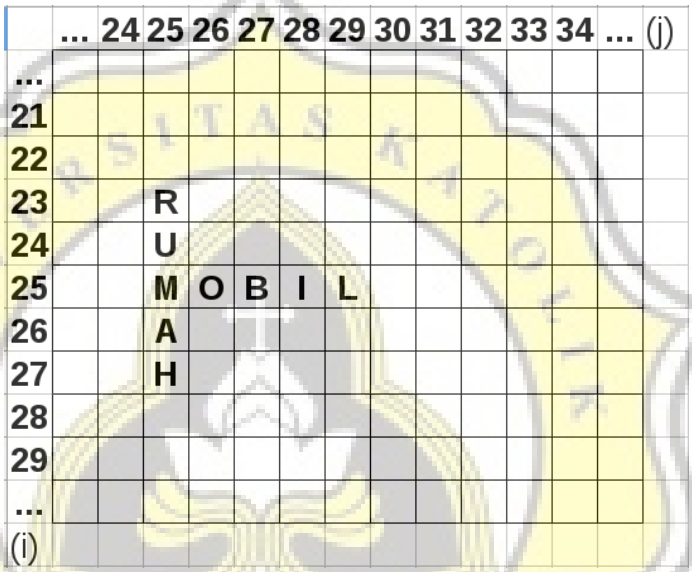

Figure 16. Placing the second word

Since word “RUMAH” successfully placed in the board, the program will insert that word into the vertical list. And give another point to the fitness value. In this case now the individual’s fitness value is 2. So the first step of the program in counting fitness is to check the list. If the word that will be placed is vertical then it will check the horizontal list and vice versa. This process will be repeated until all word placed or when the program finds a word that cannot be placed.

Figure 17. Array of Vertical List

Since the order of the gen is important in this problem. The first gen will be placed first. The second gen will be placed second and so on. So if the second gen can't be placed the program will not continue to place the third gen. For example we already have the word "MOBIL" placed horizontally. The second gen has the word "RUMAH" but instead placed vertically the word placed horizontally. Horizontal word can't be placed in another horizontal word. So, The word "RUMAH" cannot be placed. The program then will stop counting the fitness and will set the fitness value with 1 since the program only able to place one word.

There are some conditions in the program that will determine that the gen cannot be placed. The first condition is if the first word's direction has the same value with the second word's directions, then the second word cannot be placed. Another condition is when the word that will be placed cannot find any letter that can be intersected. The last condition is where the word is coinciding with another word. These conditions will stop the counting process.

After counting all of the individual's fitness value, the program will enter selection process. In selection process, each individual will be checked whether they will be crossed over or not. The checking will be done by generate a random value between 0 until 1. If the random value is less than or equal to the chance, then the individual will be crossed over later on. In this program the default crossover chance are 0.8 or 80%.

Table 2. Crossover Chance

		population		
Individual	**Fitness**	**Random**	**cross chance**	**cross or not**
1	2	0.339	0.8	cross
2	1	0.652	0.8	cross
3	1	0.834	0.8	not
4	3	0.922	0.8	not
5	1	0.827	0.8	not
6	3	0.503	0.8	cross
7	1	0.291	0.8	cross
8	3	0.281	0.8	cross
9	1	0.588	0.8	cross
10	4	0.409	0.8	cross

Example in table 2 there are 10 individuals with its fitness value. Each individual will have a random value. Table 2 shows that random value for individual number 1 is 0.339 and the crossover chance is 0.8. Since the 0.339 is less than 0.8, individual number 1 will be crossover later on. Another example from table 2, individual number 3 have 0.834 for its random number. Since 0.834 is higher than 0.8, individual number 3 won't be crossover. Each Individual that passes the crossover chance will be stored in crossover pool. In the crossover there are 2 individuals needed. In the pool, each individual become parent1 for the crossover. Roulette wheel selection will be used in the pool to determine the second parent.

Table 3. Roulette Wheel Selection

individual(i) parent1	**Fitness(f)**	**Σfi**	**Random(0,15)**	**parent2**
1	2	2	13	10
2	1	3	3	2
6	3	6	11	9
7	1	7	11	9
8	3	10	0	1
9	1	11	0	1
10	4	15	2	1
	15			

In roulette wheel selection, the program will count the total fitness of the individuals in the pool. Then generate random number between 0 and the total fitness. In this case the total fitness is 15. Then each individual will have Σfi. This value is the sum of the first individual's fitness until the fitness of itself (see table 3). After that the random value will be compared to the Σfi. If Σfi is same or bigger than the random value, then the program will mate parent1 with that individual. Example like in the table 3, to find the partner of individual 1 we generate a random number(R) between 0 and 15. In the table we get 13. Then we check sequentially value Σfi each individuals. First the program will compare Σfi = 2 and R=13. Since the Σfi value is not greater than or equal to R, the program will move to the next Σfi which is 3. It is not greater than or equal to R, then move to the next until in the individual number 10. It has Σfi = 15, 15 ≥ 13. So the Individual number 10 will become the partner of parent1 for crossover process. This process will be repeated until all of individuals in the pool will have its own partner.

```
public int selection()
{
        int i,pasangan,max,randval;
        pasangan=0;
        i=0;
        max = makeSum();
        pasangan=0;
        int sumsmall=0;
        randval=randomNum.nextInt((max)+1);
        loopcaripasangan:
        while(pasangan<jumlahPopulasi){
                sumsmall=sumsmall+populasitemp[pasangan].getFitness();
                if(sumsmall>=randval){
                        return pasangan;
                }
                pasangan++;
        }
        return -1;
}
```

Code 2. Roulette Wheel Selection code

Code 2 above is the code for the roulette wheel selection process. The method makeSum() is to find the total fitness value of all individuals in the temporary population. The variable sumsmall is a variable that represent Σfi from table 3.

After selection process, now each individual in the pool have its own partner. The program will enter the crossover process. It is like in the previous chapter, the process of crossover is using single point crossover for ordered solution.

Figure 18. Crossover Ilustration.

In the figure 19 there are 2 parents for the crossover which are parent1 and parent2. Each crossover process will produce two new individuals. From the figure 19, the first child's gen [0] and gen [1] taken from parent1. For the rest of the gen will be taken from parent2. The gen

that taken from parent2 will be checked first in the child. Here child1 already has word “MOBIL” placed horizontally and word “RUMAH” placed vertically. Now the rest of the gen should be from parent2. Gen[0] of parent2 is word “MATAHARI” placed false. Since child1 don’t have any word “MATAHARI” yet. Gen[2] of child1 will be gen[0] of parent2. Next gen in parent2 has word “LEMARI” placed horizontally. Child1 doesn’t have it yet so gen[3] will copy that gen from parent2. Next gen of parent2 have word “RUMAH” placed horizontally. Since child1 already has word “RUMAH”, this gen will not copied and move to the next gen. This process will be repeated until the child has all the words in it.

After each crossover process, the child produced from it will be checked whether it will mutate or not. Each individual has 3% chance of mutation by default. User can freely change this mutation chance. For the checking the program will generate a random value between 0 and 1. If the random value is lower than or equal to the chance of mutation which is 0.03(3%) the individual will mutate. Like what have been explained in the previous chapter, the gen of an individual will be recombined again by reshuffle the words.

CHILD 2 before mutation

GEN[0]		GEN[1]		GEN[2]		GEN[3]		GEN[4]	
MATAHARI	FALSE	LEMARI	TRUE	MOBIL	TRUE	RUMAH	FALSE	PERAHU	TRUE

CHILD 2 after mutation

GEN[0]		GEN[1]		GEN[2]		GEN[3]		GEN[4]	
MOBIL	FALSE	RUMAH	TRUE	PERAHU	TRUE	MATAHARI	FALSE	LEMARI	TRUE

Figure 19. Mutation with example

Each time an individual produced the program will count its fitness value. If it have higher fitness value than the least or the smallest fitness

value in current population, it will replace the worst individual(which has the smallest fitness value). But if the worst individual still has higher fitness value than the child, the child will not be placed in the population.

After all the individuals in the pool crossed over, the program will perform a check whether the maximum result already achieved. If it is not found yet, the program will start the selection process again but using the latest population.

5.2. Testing

There are two kind of input in the program. The first input is for the user to experiment with the algorithm and the second input are the words that will be processed by the algorithm. figure 20 is showing the log from the first input.

```
[donny@localhost programProjectV3]$ java TestDrive
jumlah kata: 5
jumlah populasi: 10
mutasi: 0.03
```

Figure 20. Log program after input GA

Figure 21 show the data that will be processed with the genetic algorithm. The data showed in figure 21 is in a file with txt format. After the input of words and its clue the program will run the main process.

```
1 MOBIL;CAR;
2 LEMARI;CUPBOARD;
3 RUMAH;HOUSE;
4 MATAHARI;SUN;
5 PERAHU;BOAT;
```

Figure 21. File data.txt after input words and clue received

In figure 22 below shows 10 individuals and its fitness value in a population. The image also show log of one generation. In the genetic algorithm the maximum result sometime can come out in the first generation. But sometime can also come out in hundred of generations.

```
|RUMAH|turun|MOBIL|datar|MATAHARI|turun|PERAHU|datar|LEMARI|datar|

|MOBIL|turun|MATAHARI|turun|LEMARI|turun|RUMAH|datar|PERAHU|datar|

|RUMAH|turun|MOBIL|turun|PERAHU|datar|LEMARI|datar|MATAHARI|datar|

|LEMARI|datar|MOBIL|turun|RUMAH|turun|PERAHU|turun|MATAHARI|datar|

|LEMARI|datar|RUMAH|datar|MATAHARI|datar|PERAHU|turun|MOBIL|datar|

|LEMARI|turun|RUMAH|datar|MATAHARI|turun|MOBIL|turun|PERAHU|turun|

|MOBIL|datar|LEMARI|datar|PERAHU|datar|MATAHARI|turun|RUMAH|datar|

|LEMARI|datar|PERAHU|turun|RUMAH|datar|MOBIL|datar|MATAHARI|turun|

|MATAHARI|datar|LEMARI|datar|MOBIL|turun|PERAHU|datar|RUMAH|datar|

|MATAHARI|turun|LEMARI|datar|RUMAH|datar|MOBIL|datar|PERAHU|turun|

[0] = 5
[1] = 1
[2] = 1
[3] = 5
[4] = 1
[5] = 3
[6] = 1
[7] = 3
[8] = 1
[9] = 4
```

Figure 22. Log Program shows individu in population and fitness value

Once the best solution is found, the program will get that solution. If the best solution did not found but the generation reached its limit, the program will get the best individual from the last generation's population.

```
#|M|#|#|#|#|
#|O|#|#|#|#|
#|B|#|#|#|#|
#|I|#|#|#|#|
#|L|E|M|A|R|I|
#|#|#|A|#|#|#|
#|#|#|T|#|#|#|
P|E|R|A|H|U|#|
#|#|#|H|#|#|#|
R|U|M|A|H|#|#|
#|#|#|R|#|#|#|
#|#|#|I|#|#|#|
```

Figure 23. Simplified Board

After the program get the result from the genetic algorithm (whether it is has maximum value or not), the program will print the temporary board that already simplified (see figure 23.) with all of the list(see figure 24).

```
0
L(4,1)-->E(4,2)-->M(4,3)-->A(4,4)-->R(4,5)-->I(4,6)-->
1
P(7,0)-->E(7,1)-->R(7,2)-->A(7,3)-->H(7,4)-->U(7,5)-->
2
R(9,0)-->U(9,1)-->M(9,2)-->A(9,3)-->H(9,4)-->
list menurun total : 2
0
M(4,3)-->A(5,3)-->T(6,3)-->A(7,3)-->H(8,3)-->A(9,3)-->R(10,3)-->I(11,3)-->
1
M(0,1)-->O(1,1)-->B(2,1)-->I(3,1)-->L(4,1)-->
```

Figure 24. Result's vertical and horizontal list