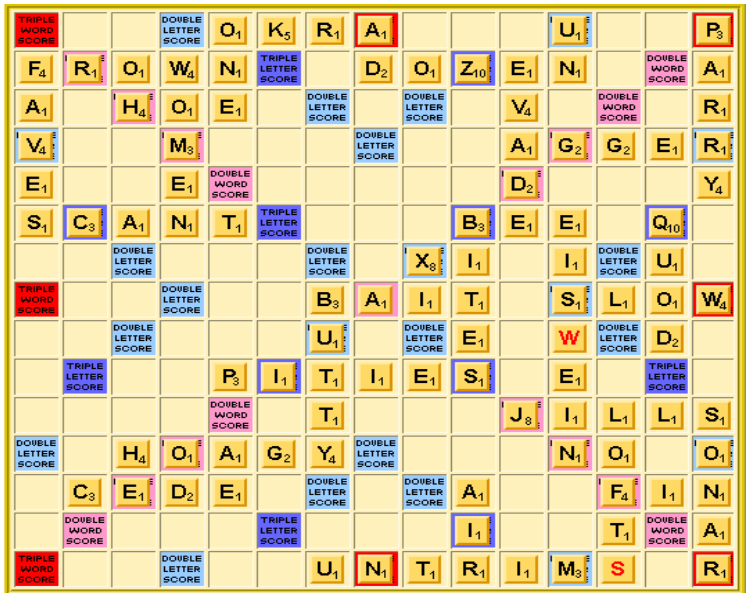
BAB 3

ANASLIS DAN PERANCANGAN

3.1. **Analisis Masalah**

Scrabble adalah permainan papan dan permainan menyusun kata yang dimainkan 2 atau 4 orang yang mengumpulkan poin berdasarkan nilai kata yang dibentuk dari keping huruf di atas papan permainan berkotak-kotak (15 kolom dan 15 baris). Biji permainan berupa keping berbentuk bujur sangkar yang bertuliskan huruf pada salah satu sisi. Pemain mengambil hingga sebanyak tujuh buah keping huruf dari kantong, dan berusaha menyusun kata secara mendatar atau menurun seperti teka-teki silang. Kata-kata yang dibuat harus merupakan kata yang diizinkan untuk dimainkan berdasarkan kamus standar sesuai dengan bahasa yang dimainkan. Pemain yang mengumpulkan total poin tertinggi dinyatakan sebagai pemenang.



Gambar 3.1 contoh penyelesaian *scrabble*

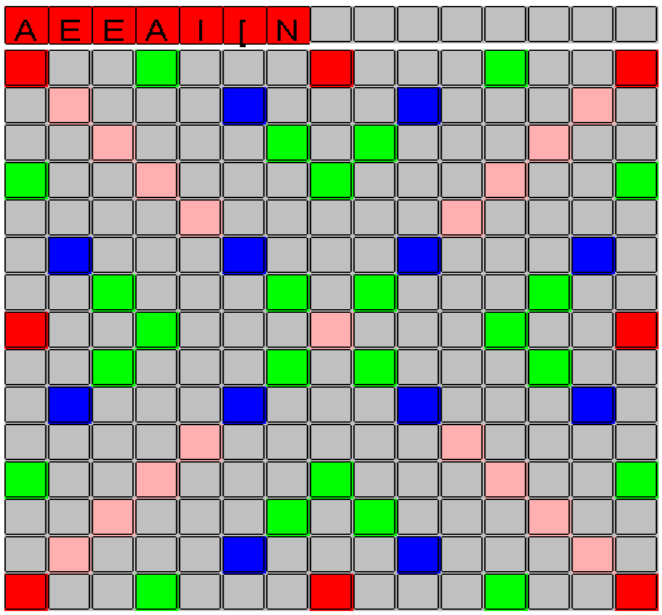
Oleh karena itu di butuhkan teknik untuk membangkitkan soal yang bersifat unik dengan system random yang akan menghasilkan soal yang akan saling terkait

dari soal satu dengan soal lainnya dan mencari solusi dalam penyelesaikan jawaban pada permainan scrabble dengan beberapa syarat di dalamnya dengan menggunakan prinsip dasarnya, algoritma genetika mencari segala kemungkinan solusi dengan menerapkan ruang keadaan dalam bentuk pencarian solusi hingga menghasilkan solusi yang layak. Kelayakan solusi pada permainan scrabble didefinisikan dengan suatu nilai akan dinyatakan layak apabila terdapat huruf yang tersusun.

3.2. **Analisis Game**

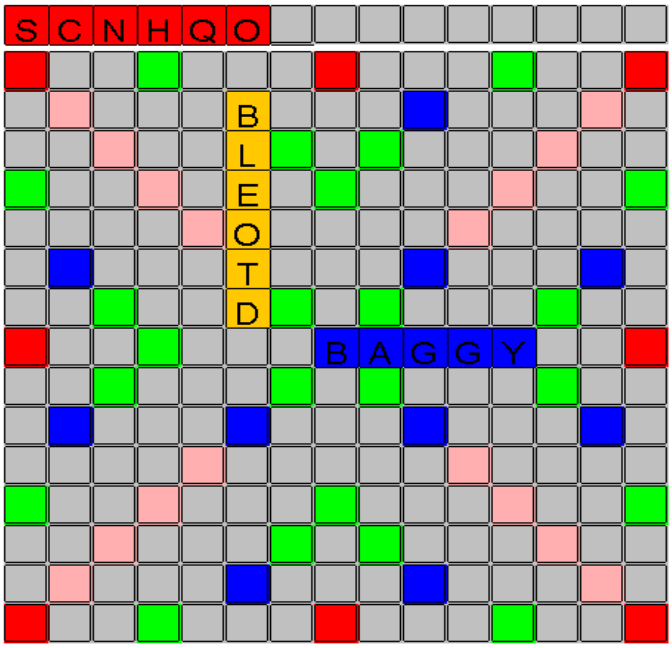
Analisis *game* yang akan diteliti pada *game scrabble* adalah dengan membangkitkan soal permainan serta menyelesaikan jawaban. Pada dasarnya permainan *scrabble* ini disetiap soal maupun jawaban akan saling terkait satu sama lainnya dan mempunyai nilai pasti disetiap jawabannya. Oleh karena itu, algoritrma genetika akan diterapkan pada kasus *scrabble* ini.

Berikut penjelasan alur proses pada game *srabble* dengan kasus permainan *srabble* :



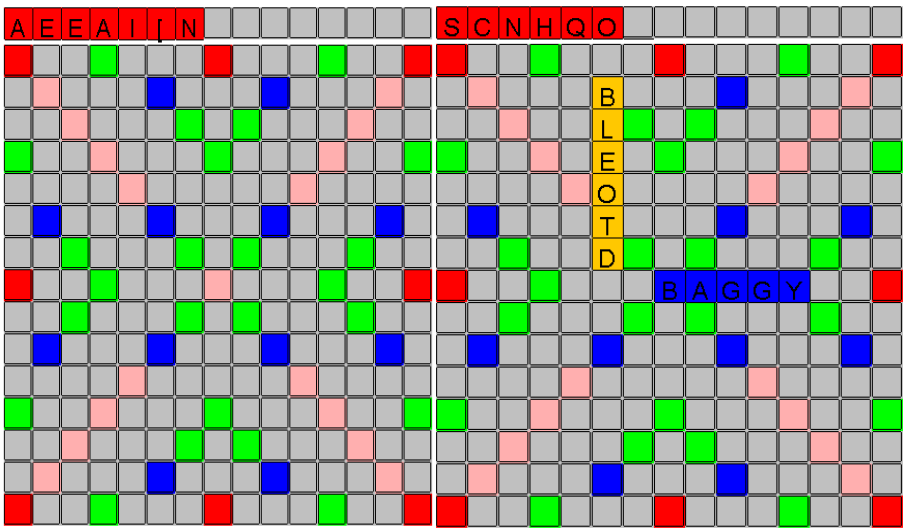
Gambar 3.2 Inisialisasi kotak permainan

Gambar 3.2 menjelaskan tentang keadaan awal permainan sebelum menentukan nilai acuan pada lajur. Dimana kotak permainan disusun berdasarkan kotak warna yang ada nilainya.



Gambar 3.3 Membangkitkan jawaban

Gambar 3.3 menjelaskan tentang pembangkit nilai jawaban. Pengisian ini bertujuan untuk membangkitkan nilai soal pada kotak blok. Selain bertujuan mencari nilai soal proses ini juga bertujuan untuk menentukan jawabannya.



Gambar 3.4 Akumulasi jawaban dan cetak soal

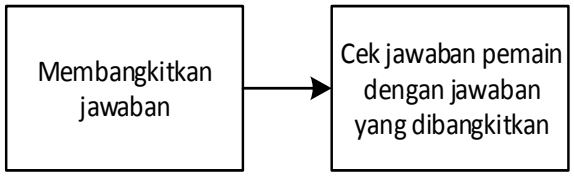
Gambar 3.4 menjelaskan tentang akumulasi nilai jawaban dan cetak nilai soal untuk permainan ini. Untuk mencetak soal, kotak yang berisikan nilai jawaban dalam satu lajur akan diakumulasikan dari total nilai jawaban.

1. Proses penyelesaian jawaban pada kotak kosong.

Pada proses ini, penyelesaian jawaban dengan algoritma genetika dengan sistem teracak dalam pencarian nilai. Proses pencarian nilai dilakukan dari huruf A hingga Z, dan dengan melakukan tahapan pengecekan nilai yaitu boleh terdapat huruf yang sama pada satu lajur serta total nilai jawaban pada satu lajur yang akan dibandingkan dengan nilai soal.

2. Proses penyelesaian jawaban pada kotak terisi nilai.

Pada proses ini, penyelesaian jawaban dengan algoritma genetika hampir sama dengan proses penyelesaian jawaban pada kotak tidak terisi. Hanya saja pada proses pengecekan nilai, nilai jawaban yang telah diisi oleh pemain akan dicek ulang dengan nilai jawaban pada algoritma genetika.



Gambar 3.5 Proses penyelesaian jawaban pada kotak terisi huruf

3.3. **Analisis Masukan**

Analisis masukan yang dilakukan dalam penelitian ini adalah menganalisis inputan *scrabble* dalam algoritma genetika pada *game scrabble*. Masukan tersebut berupa nilai (huruf), akan tetapi masukan diberi batasan nilai yaitu dari huruf A hingga Z, berikut merupakan tabel analisis masukan pada *game scrabble*.

Tabel 3.1 huruf masukan

Nilai acuan lajur	**Nilai kombinasi**
aa	Aa
aeb	Bae
taekc	Caket
tidal	Dalit
euheuw	Euweuh
nanarmif	Firmanna

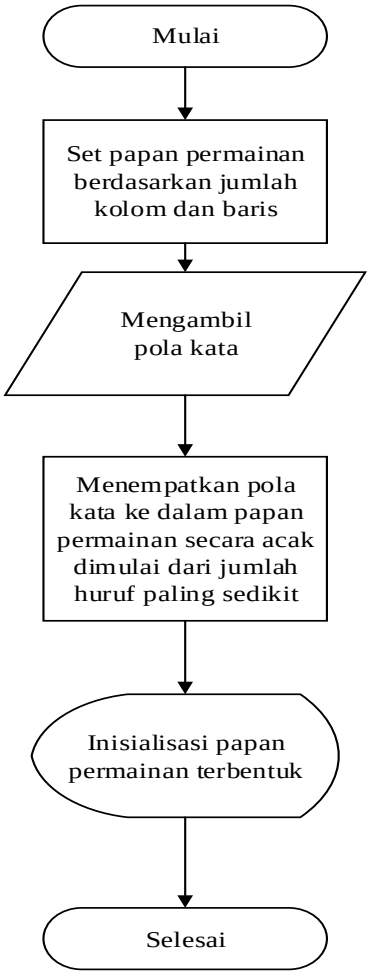
Tabel 3.2 huruf masukan A sampai Z

Nilai acuan lajur	Nilai kombinasi
Yaa	Aya
Saba	Basa
lacana	Calana
Tede	Deet
ahden	Endah
asnailitsaf	Fasilitasna
oaragg	Gagaro
eutanh	Haneut
ngateunu	Ingeutan
entanj	Janten
okawn	Kawon
mial	Lami
adsakm	Maksad
andakny	Nyandak
ngeprom	Ompreng
apmeur	Pareum
narhapiq	Qiprahna
Arag	Raga
osatb	Sobat
urbukt	Tubruk
eumlu	Uleum
euleuliyanv	Vileuleuyan
gariw	Wargi
	X
suway	Yuswa

Nilai acuan lajur	Nilai kombinasi
	Z

3.3.1. Inisialisasi Papan Permainan

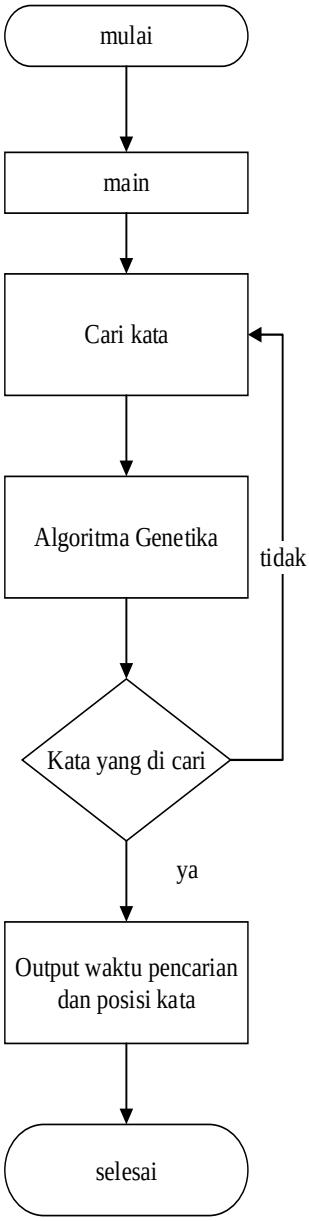
Papan permainan disusun secara acak disusun secara horizontal, vertikal dan dapat ditulis dalam posisi terbalik maupun tidak berdasarkan pola kata yang terdapat pada file exstensi .txt. Papan permainan diacak secara random dari alpabet A-Z dengan jumlah total abjad yaitu 26 huruf. Papan permainan tersebut diacak berdasarkan pola kata pada papan permainan yang kemudian ditempatkan secara berurutan dimulai dengan kata yang mempunyai huruf paling sedikit yaitu 2 huruf sampai dengan jumlah huruf terbanyak yaitu 7 huruf. Berikut adalah gambar 3.7 flowchart inisialisasi papan permainan :



Gambar 3.6 inisialisasi papan permainan

3.4. **Analisis Algoritma**

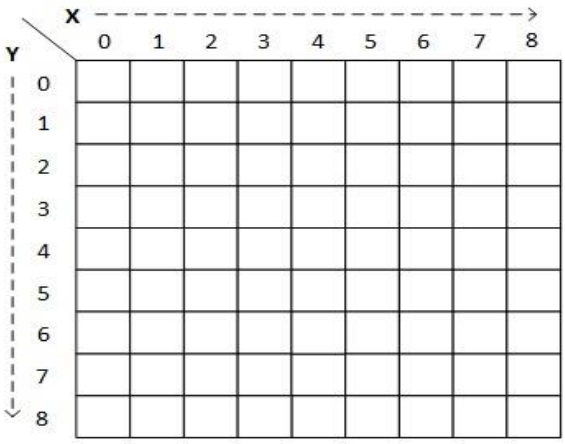
Analisis algoritma yang akan diteliti pada game *scrabble* ini adalah untuk menganalisis cara kerja algoritma genetika terhadap kasus permainan *scrabble*, bagaimana cara kerja algoritma genetika dalam proses pencarian kata pada game *scrabble*. Bagan pencarian secara utuh dapat di lihat pada gambar 3.7 di bawah ini.



Gambar 3.7 pencarian kata

Sebelum meneliti cara kerja algoritma genetika pada *game scrabble* ini, akan dibahas terlebih dahulu bagaimana cara menginisialisasikan kotak terlebih dahulu yang akan menghasilkan kotak kosong (kotak jawaban) dan kotak blok (kotak nilai soal / pembatas).

Dalam proses inisialisasi kotak pada *scrabble*, di setiap kotak *scrabble* akan diberikan nilai indeks yang akan digambarkan dengan titik koordinat sumbu (X, Y). Pada sumbu X menyatakan posisi kotak secara horizontal, dan sumbu Y menyatakan posisi kotak secara vertikal. Titik koordinat sumbu X, dan Y akan di gambarkan sebagai berikut :



Gambar 3.8 Titik koordinat sumbu (X, Y)

Pada Gambar 3.8 menjelaskan keadaan awal kotak sebelum terjadinya inisialisasi kotak permainan yang akan menentukan nilai acuan lajur dan kotak permainan. Dimana kotak permainan disusun berdasarkan kotak blok yang berwarna abu dan kotak kosong yang berwarna warni seperti yang terjadi pada proses gambar 3.3 Dalam menentukan kotak blok dan kosong dalam permainan *srabble* ini, maka alur proses yang akan terjadi adalah sebagai berikut :

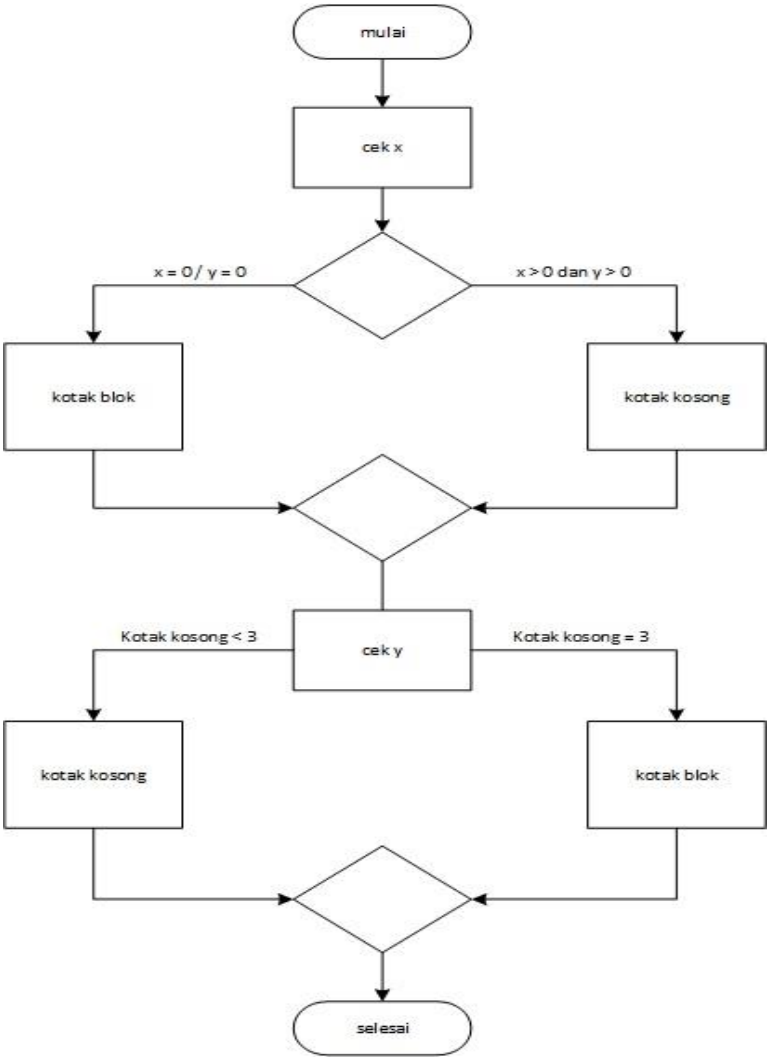
1. Mengecek lajur nilai X.

Apabila nilai indeks pada kotak yang dipilih terdapat nilai 0 pada sumbu X atau Y, maka kotak tersebut akan menjadi kotak blok. Sedangkan apabila nilai indeks pada kotak yang dipilih terdapat nilai > 0 pada sumbu X atau Y, maka kotak tersebut akan menjadi kotak kosong.

2. Mengecek lajur nilai Y.

Nilai indeks pada kotak yang dipilih akan menjadi kotak blok, apabila lajur pada sumbu Y diatasnya terdapat 3 kotak kosong. Sedangkan nilai indeks pada kotak yang dipilih akan menjadi kotak kosong, apabila lajur pada sumbu Y diatasanya kurang dari 3 kotak kosong. Pada pengecekan lajur sumbu Y terdapat batasan, karena pada permainan yang dibuat hanya dibatasi 2 hingga 3 kotak kosong sebagai nilai masukan.

Dari pengecekan tersebut, didapatlah kotak blok dan kosong untuk menentukan lajur permainan seperti pada gambar 3.9 Proses pengecekan nilai pada titik koordinat (X, Y) diatas akan dijelaskan pada alur *flowchart* berikut :

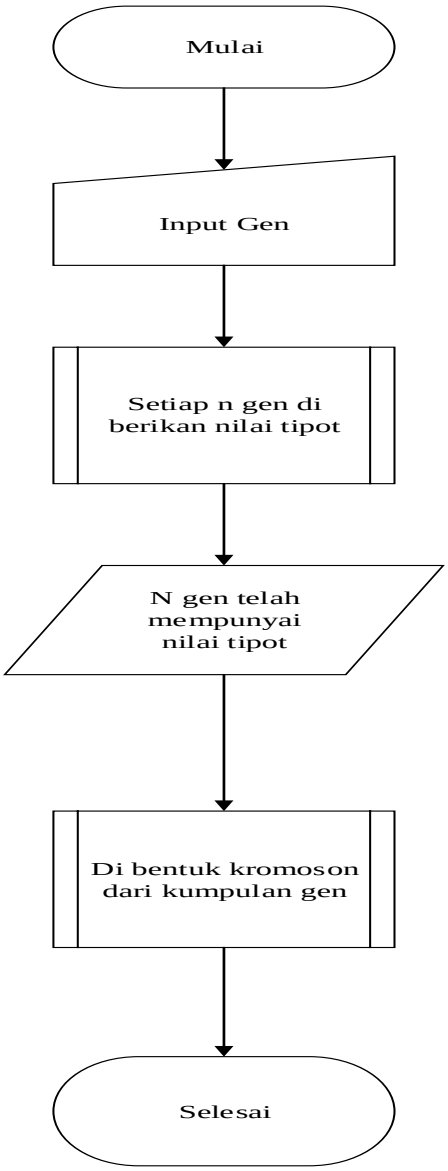


Gambar 3. 9 *Flowchart* pengecekan nilai pada titik koordinat (X, Y)

3.4.1. Analisis Algoritma genetika

3.4.1.1. Pembentukan Kromosom

Kromosom merupakan representasi suatu solusi scrabble yang berisi huruf yang di isikan pada kotak kosong. Panjang kromosom sama dengan jumlah kotak kosong pada scrabble. Pada penelitian ini di batasi dengan minimal 2 huruf dan maksimal 7 huruf. Representasi kromosom yang di gunakan adalah deretan huruf-huruf pada masing-masing gen pada kromosom.



Gambar 3. 10 *Flowchart* algoritma genetika

Repsesentasi Kromosom

Tabel 3.3 Repsesentasi Kromosom

No	Gen	Nilai Tipot
1	AYA	1
2	YEN	1
3	KAWON	1
4	DEET	1
5	ENDAH	0
6	FASILITASNA	0
7	GAGARO	0

Kromosom													
Gen 1		Gen 2		Gen 3		Gen 4		Gen 5		Gen 6		Gen 7	

3.4.1.2. Inisialisasi

Proses inisialisai di lakukan dengan cara memberikan nilai gen-gen awal dengan huruf acak dari A sampai Z. setelah di ketahui panjang kromosomnya dan tentukan nilai populasinya

A. Menempatkan gen pertama ke dalam kotak

x,y	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
1															
2							G								
3							A								
4							G								
5							A								
6							R								
7							O								
8															
9															
10															
11															
12															
13															

14															
15															

Gambar 3.11 inisialisasi populasi

B. Menyimpannya ke dalam array

7,2G	7,3A	7,4G	7,5A	7,6R	7,7O

C. A.Menempatkan gen selanjutnya dengan mencari titik potong

1	Aya	1

X,y	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
1															
2							G								
3							A								
4							G								
5							A								
6							R								
7							O								
8															
9															
10															
11															
12															
13															
14															
15															

Gambar 3.12 inisialisasi populasi

Mencari huruf A

D. Setelah titik potong di temukan memeriksa nilai tipot (tidak boleh bernilai sama)
E. Menempatkan gen sesuai dengan titik potong

X,y	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
1															
2							G								
3							A	Y	A						
4							G								
5							A								
6							R								
7							O								
8															
9															
10															
11															
12															
13															
14															
15															

Gambar 3.13 inisialisasi populasi

F. Menempatkan gen sesuai dengan titik potong

Menghitung nilai gen : $f = \frac{tp}{tk}$

F= $^{1}/_{3}$=0,33

G. Dicari titik potong lain

X,y	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
1															

2							G								
3							A								
4							G								
5							A								
6							R								
7							O								
8															
9															
10															
11															
12															
13															
14															
15															

Gambar 3.14 inisialisasi populasi

Mencari huruf A

X,y	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
1															
2							G								
3							A								
4							G								
5							A	Y	A						

6							R								
7							O								
8															
9															
10															
11															
12															
13															
14															
15															

Gambar 3.15 inisialisasi populasi

Menghitung nilai gen : $f = \frac{tp}{tk}$

F= $^{1}/_{3}$= 0,33

H. Mencari kemungkinan titik potong lain dari karakter selanjutnya setelah huruf "A" dari kata "AYA"

X,y	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
1															
2							G								
3							A								
4							G								
5							A								
6							R								
7							O								

8															
9															
10															
11															
12															
13															
14															
15															

Gambar 3.16 inisialisasi populasi

Mencari huruf Y

I. Memilih titik potong yang akan di gunakan dengan rumus :
Axmax-xmax

Keterangan :

Axmax = nilai x maksimum

Xmax = nilai x dari jawaban

Untuk jawaban alternatif 1 :

= axmax-xmax

=9-7=2

X,y	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
1															

2							G								
3							A	Y	A						
4							G								
5							A								
6							R								
7							O								
8															
9															
10															
11															
12															
13															
14															
15															

Gambar 3.17 inisialisasi populasi

Untuk jawaban alternatif 2 :

= axmax-xmax

=9-7=2

X,y	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
1															
2							G								
3							A								
4							G								

5							A	Y	A						
6							R								
7							O								
8															
9															
10															
11															
12															
13															
14															
15															

Gambar 3.18 inisialisasi populasi

Dari hasil kedua alternatif jawaban di dapat jawaban yang bernilai sama. Oleh karena itu di pilih alternatif 1

X,y	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
1															
2							G								
3							A	Y	A						
4							G								
5							A								
6							R								
7							O								

8															
9															
10															
11															
12															
13															
14															
15															

Gambar 3.19 inisialisasi populasi

Tabel 3.4 inisialisasi populasi

7,2,G	7,3,A	7,4,G	7,5,A	7,6,R	7,7,O

Array jawaban menurun

7,3,A	7,8,Y	7,9,A

Array jawaban menurun

7,3A	

Array titik potong

Dilanjutkan dengan penempatan gen berikutnya kata “YEN”

X,y	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
1							→	→	→						
2						↓	G								
3						↓	A	Y	A						
4						↓	G								
5						↓	A								
6							R								
7							O								
8															
9															
10															
11															
12															
13															
14															
15															

Gambar 3.20 inisialisasi populasi

Mencari huruf Y

Ditemukan di titik potong pada (8,3) namun untuk nilai tipot kaya “YEN” bernilai 1 sehingga tidak bisa di tempatkan.

Pada huruf “O” di temukan titik pootong pada (7,7)

Menghitung nilai gen : $f = \frac{tp}{tk}$

$$F= {}^{1}/_{5} = 0,2$$

X,y	1	2	3	4	5	6	7	8	9	10	11	12		13	14	15
1																
2							G									
3							A	Y	A							
4							G									
5							A									
6							R									
7				K	A	W	O	N								
8																
9																
10																
11																
12																
13																
14																
15																

Gambar 3.21 inisialisasi populasi

Pencarian selesai dan menghasilkan inisialisasi kromosom :

Tabel 3.5 inisialisasi populasi

7,2,G	7,3,A	7,4,G	7,5,A	7,6,R	7,7,O

Array jawaban menurun

7,3,A	7,8,Y	7,9,A	4,7,K	5,7,A	6,7,W	7,7O	8,7,N

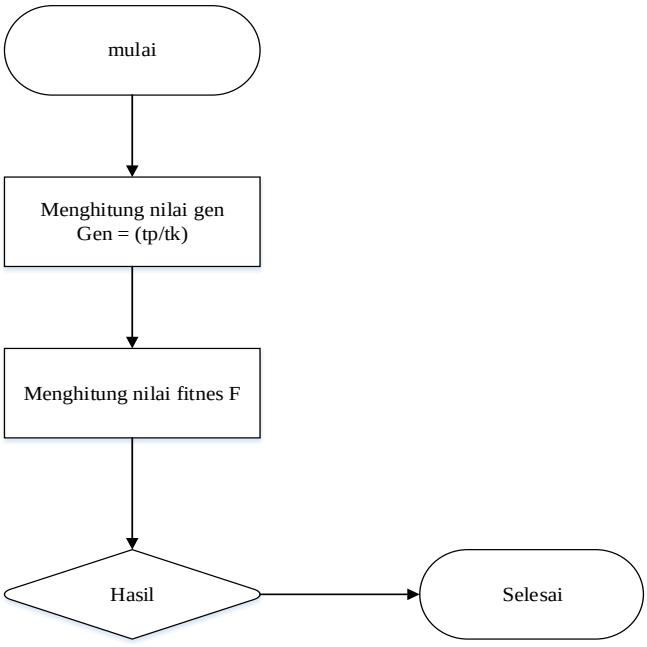
Array jawaban menurun

7,3A	7,7,O

Array titik potong

3.4.1.3. Fungsi Evaluasi :

Flowchart untuk fungsi evaluasi :



Gambar 3. 22 *Flowchart* Fungsi evaluasi

Perhitungan nilai fitnes :

Tabel 3.6 fungsi evaluasi

Gen ke	Gen	Jumlah titik potong (tp)	Jumlah total karakter (tk)	Nilai gen (tp/tk)
1	AYA	1	3	0,33
2	YEN	-	3	-
3	KAWON	1	5	0,2
4	DEET	-	4	-
5	ENDAH	-	5	-
6	FASILITASNA	-	11	-
7	GAGARO	2	6	0,33

F = 0,33 + 0,2 + 0,33

= 0,86

3.4.1.4. Seleksi

Metode seleksi yang digunakan adalah metode Roulette-Wheel yang membuat kromosom-kromosom dengan nilai fitness tinggi memiliki kemungkinan terpilih yang tinggi pula.

Probabilitas keterpilihan setiap kromosom ditentukan dengan cara membagi nilai fitness kromosom tersebut dengan total nilai fitness. Untuk itu, yang terlebih dahulu harus dihitung adalah total nilai fitness. Kemudian dilakukan pembagian masing-masing nilai fitness dengan total fitness tersebut. Nilai tersebut adalah probabilitas setiap kromosom yang berkisar antara nol dan satu.

Kemudian pada proses pemutaran roulette-wheel, teknisnya juga sederhana. Suatu bilangan acak dibangkitkan, dengan jangkauan nilai antara nol dan satu. Nilai acak tersebut kemudian dibandingkan dengan nilai probabilitas masing-masing

kromosom. Pembandingan dilakukan dari kromosom yang terkecil. Kromosom yang memiliki nilai probabilitas yang lebih besar atau sama dengan nilai acak akan menjadi individu yang terpilih.

Dalam tahap ini nilai fitnes yang di hasilkan hanya satu jadi yang akan di ambil dari proses seleksi adalah parent yang mempunyai nilai fitnes 0,86.

X,y	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
1															
2							G								
3							A	Y	A						
4							G								
5							A								
6							R								
7				K	A	W	O	N							
8															
9															
10															
11															
12															
13															
14															
15															

Gambar 3.23 inisialisasi populasi

3.4.1.5. Crossover

Setelah melakukan proses seleksi, maka hasil tersebut akan digunakan dalam proses *crossover*. Metode *crossover* atau pindah silang yang digunakan adalah pindah silang dengan satu titik. Posisi *crossover* dipilih bilangan random dengan batasan dari 1 sampai N (N = panjang kromosom). Pertukaran gen kemudian dilakukan pada kedua kromosom mulai titik gen tertentu sampai akhir kromosom.

a) Jumlah kromosom calon *parrent* yang mengalami *crossover* di tentukan oleh parameter *crossover* (pc).
b) Nilai PC yang di pakai pada penelitian ini adalah 0,5
c) Di bangkitkan bilangan random sebanyak ukuran populasi yaitu 30

Tabel 3.7 inisialisasi crossover

No	Bilangan random R	Terpilih
1	0,45	Ya
2	0,63	Tidak
3	0,71	Tidak
4	0,34	Ya
5	0,15	Ya
6	0,23	Ya
7	0,65	Tidak
8	0,22	Ya
9	0,11	Ya
10	0,23	Ya
11	0,65	Tidak
12	0,84	Tidak
13	0,94	Tidak
14	0,1	Ya
15	0,02	Ya

16	0,35	Ya
17	0,77	Tidak
18	0,84	Tidak
19	0,14	Ya
20	0,7	Tidak
21	0,69	Tidak
22	0,96	Tidak
23	0,88	Tidak
24	0,77	Tidak
25	0,63	Tidak
26	0,44	Ya
27	0,32	Ya
28	0,25	Ya
29	0,15	Ya
30	0,1	Ya

a) r1 < PC (0,45<0,5) maka kromosom 1 terpilih sebagai *parent.*
b) Nilai r2 adalah 0,63, karena r2 > PC maka kromosom 2 tidak terpilih sebagai *parent.*

3.4.1.6. Mutasi

Mutasi di lakukan satu titik, yaitu dengan mengganti nilai gen suatu kromosom dengan suatu nilai yang di pilih secara acak dari titik-titik yang ada.

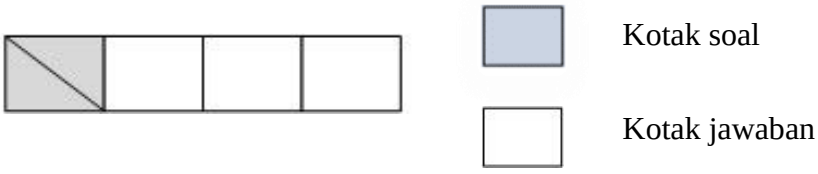
Tabel 3.8 mutasi gen dan nilai tipot

No	Gen	Nilai Tipot
1	AYA	1
2	YEN	1
3	KAWON	1
4	DEET	1
5	ENDAH	0
6	FASILITASNA	0
7	GAGARO	0

Tabel 3.9 mutasi gen dan nilai tipot

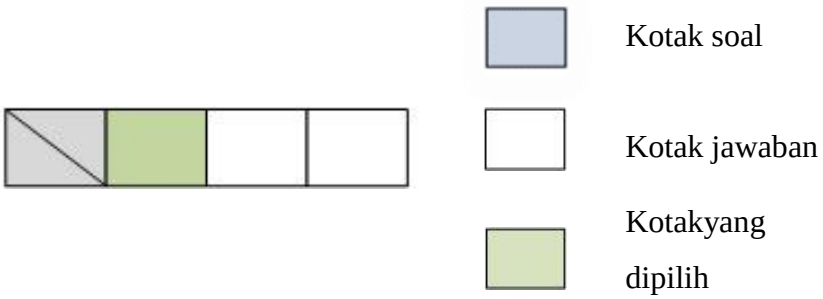
No	Gen	Nilai Tipot
1	AYA	0
2	YEN	0
3	KAWON	0
4	DEET	0
5	ENDAH	1
6	FASILITASNA	1
7	GAGARO	1

Analisis algoritma genetika pada game *scrabble* merupakan penjelasan alur atau cara kerja algoritma genetika saat terjadi pembuatan huruf di dalam game scrabble yang sudah terinisialisasi kotak terlebih dahulu. Algoritma genetika akan bekerja saat pencarian huruf. Penjelasan tersebut akan digambarkan pada kotak sebagai berikut.



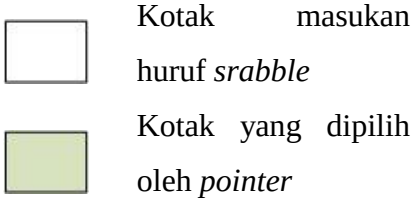
Gambar 3.24 Keadaan awal satu lajur

Gambar 3.24 menjelaskan tentang keadaan awal satu lajur sebelum menentukan huruf soal. Dimana kotak dari satu lajur disusun berdasarkan kotak jawaban dan kotak soal. Setelah tampil satu lajur tersebut, alur kerja algoritma genetika akan dimulai dengan menujuk kotak jawaban pertama menggunakan pointer yang akan dicari nilai jawaban kedalam array.



Gambar 3.25 Memilih kotak jawaban

Gambar 3.25 menjelaskan ketika pointer memilih kotak jawaban. Sebelum menentukan huruf jawaban, kotak jawaban di cek terlebih dahulu oleh sistem apakah kotak jawaban dalam kondisi kosong atau terisi huruf dalam satu lajur. Apabila kotak jawaban dalam kondisi kosong, maka sistem akan mengisi kotak jawaban dengan cara random. Dan apabila kotak jawaban dalam kondisi terisi nilai maka pointer akan berpindah ke kotak jawaban yang lain.



Gambar 3.26 Pencarian huruf masukan di dalam *array*

3.5. **Analisis Kebutuhan Perangkat Lunak**

Perangkat lunak (*software*) merupakan hal terpenting dalam mendukung kinerja sebuah sistem. Perangkat lunak digunakan dalam sebuah sistem untuk memberikan perangkat keras (*hardware*) sebuah perintah agar perangkat keras melakukan apa yang diinginkan oleh pemain. Berikut ini, kebutuhan perangkat lunak untuk menerapkan algoritma *genetika* pada *srabble* pada Tabel 3.10

Tabel 3.10 Spesifikasi perangkat lunak

Spesifikasi Perangkat Lunak
Windows XP, Windows 7, Windows 8
Eclipse

3.5.1. Analisis Kebutuhan Perangkat Keras

Agar aplikasi dapat berjalan dengan baik, maka dibutuhkan perangkat keras (*hardware*) yang sesuai dengan kebutuhan aplikasi.

Tabel 3.11 adalah spesifikasi minimum perangkat keras yang dibutuhkan agar dapat menjalankan aplikasi permainan *srabble* secara optimal.

Tabel 3.11 Spesifikasi perangkat keras

Komputer	Spesifikasi Perangkat Keras
Prosesor	Prosesor 1,6 Ghz
Resolusi Layar	800 x 600 *pixel*
Memori	Memori 128 Mb
Hardisk	*Hardisk* 100 Mb
Keyboard dan *mouse*	Standar
Sound card	Standar

3.5.2. Analisis Kebutuhan Pengguna

Selain dibutuhkannya perangkat lunak dan perangkat keras, *user* atau penguna adalah bagian yang sangat penting untuk dapat menerapkan aplikasi ini. Analisis pengguna diperlukan untuk dapat mengetahui sasaran calon pengguna yang dapat menggunakan aplikasi. Pada penelitian ini pembangunan aplikasi ditujukan untuk pengujian implementasi algoritma sehingga pengguna disini disarankan memiliki karakteristik sebagai berikut:

1. Bisa mengoperasikan komputer dengan sistem operasi *windows.*
2. Mengerti mengenai konsep game *srabble*.

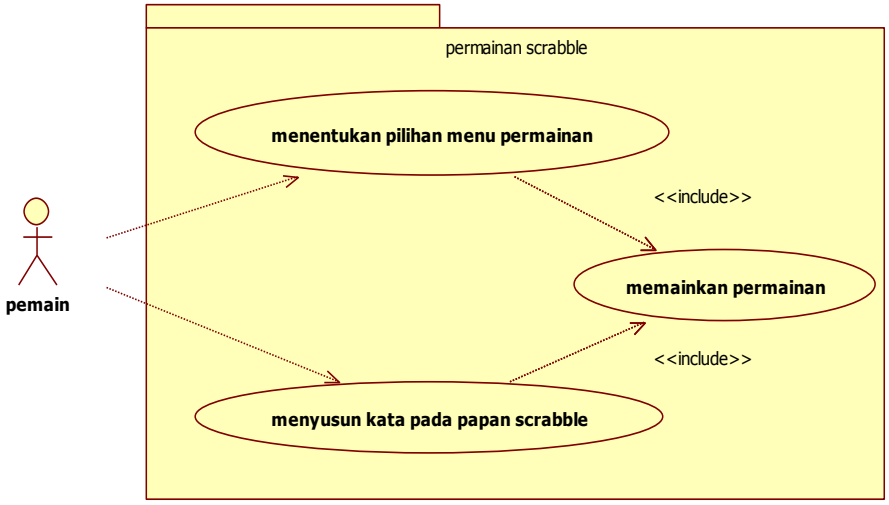
3.5.3. Analisis Kebutuhan Non Fungsional

Analisis kebutuhan non fungsional merupakan analisis yang dibutuhkan untuk menentukan spesifikasi kebutuhan sistem. Spesifikasi yang meliputi elemen atau komponen-komponen yang dibutuhkan untuk sistem yang akan dibangun hingga sistem tersebut diimplementasikan. Analisis kebutuhan ini menentukan spesifikasi masukan yang diperlukan sistem, keluaran yang akan dihasilkan sistem dan proses yang dibutuhkan untuk mengolah masukan sehingga menghasilkan suatu keluaran yang diinginkan.

3.5.4. Analisis Kebutuhan Fungsional

Pemodelan yang digunakan pada *game* yang akan dibuat adalah analisis perancangan dan pengembangan perangkat lunak berbasis objek, dimana alat yang akan digunakan untuk pengembangan perangkat lunak yaitu pemodelan *Unified Modeling Language* (UML).

3.5.4.1. Use case Diagram



Gambar 3.27 Use case diagram

Definisi *use case* berfungsi untuk menjelaskan fungsi dari *use case* yang terdapat pada diagram use case. Definisi *use case* dijelaskan pada Tabel 2.5

3.5.4.2. Definisi Aktor

Definisi aktor berfungsi untuk menjelaskan aktor yang terdapat pada use case diagram. Definisi aktor dapat di lihat pada tabel berikut :

Tabel 3.12 Definisi Aktor

No	Aktor	Keterangan
1.	pemain	Pengguna sistem

3.5.4.3. Definisi Use Case

Definisi use case menjelaskan tentang fungsi use case yang terdapat pada use case diagram. Definisi use case dapa dilihat pada tabel berikut:

Tabel 3.13 Definisi *use case*

No	*Use Case*	Keterangan
1.	Menentukan pilihan menu Permainan	Proses untuk masuk ke menu utama permainan *scrabble*
2.	Memainkan permainan	Proses untuk memulai memainkan permainan
3.	Menyusun kata pada papan *scrabble*	Proses menyusun kata pada permainan *srabble* dengan acak

3.5.4.4. Use Case Scenario

Skenario *use case* merupakan penjelasan mengenai alur kejadian yang di dalamnya menjelaskan urutan interaksi antara aktor dengan *use case* tersebut. Adapun skenario *use case* digambarkan sebagai berikut.

Tabel 3.14 Tabel memainkan permainan

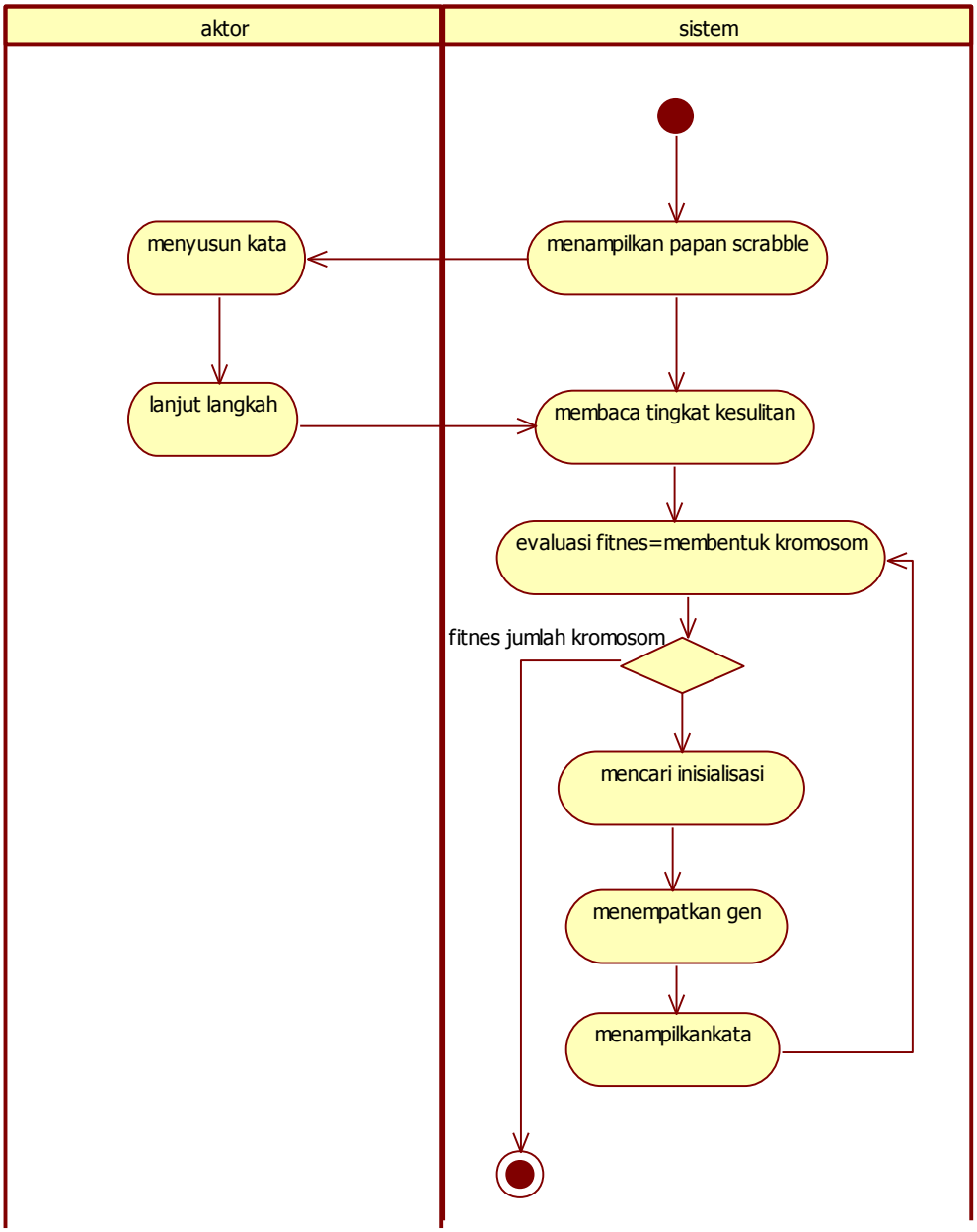
Identifikasi	
Nomor	1
Nama	Memainkan Permainan
Tujuan	Memainkan permainan
Deskripsi	Proses untuk memainkan permainan
Aktor	Pemain
Skenario Utama	
Kondisi Awal	Pemain berada di permainan
Aksi Aktor	Reaksi Sistem
	1. Mengatur papan permainan
	2. *Random* jumlah kolom
	3. *Random* jumlah baris

	4. Mengatur huruf pada papan dengan *random*
	5. Mengecek huruf
	6. Melakukan genetika
	7. Cek jumlah kolom
	8. Jika papan pada kolom tidak kosong maka lakukan pencarian genetika, jika kosong maka huruf input di nyatakan salah
	9. Cek jumlah baris
	10. Jika papan pada baris tidak kosong maka lakukan pencarian genetika, jika kosong maka huruf input di nyatakan salah
	11. Menampilkan nilai komputer
	12. Menampilkan nilai user

3.5.4.5. Activity Diagram

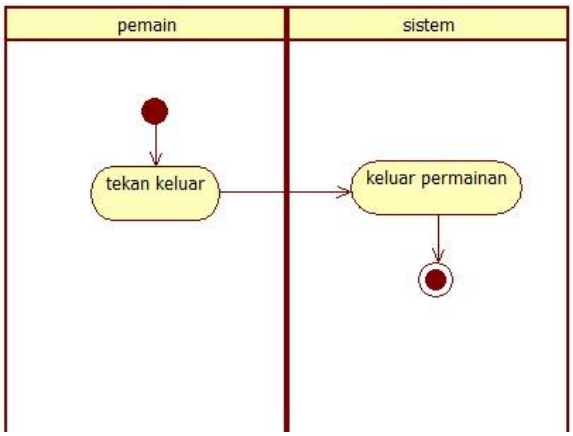
Activity diagram merupakan gambaran alur aktivitas pada permainan yang sedang dirancang, bagaimana masing-masing alur aktivitas berawal, *decision* yang mungkin terjadi, dan bagaimana berakhir. *Activity* diagram dijelaskan melalui gambar 3.28 berikut

1. *Activity* memainkan permainan



Gambar 3.28 *Activity* memainkan permainan

2. *Activity* keluar permainan

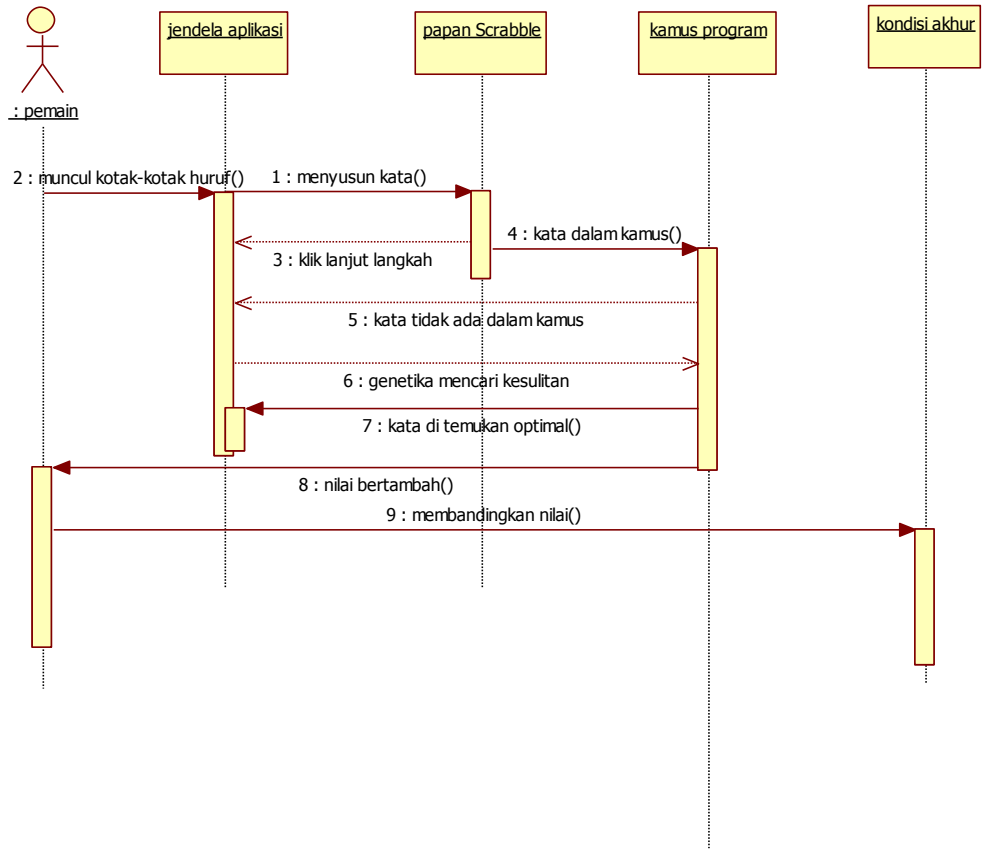


Gambar 3.29 *Activity* keluar permainan

3.5.4.6. *Squence* Diagram

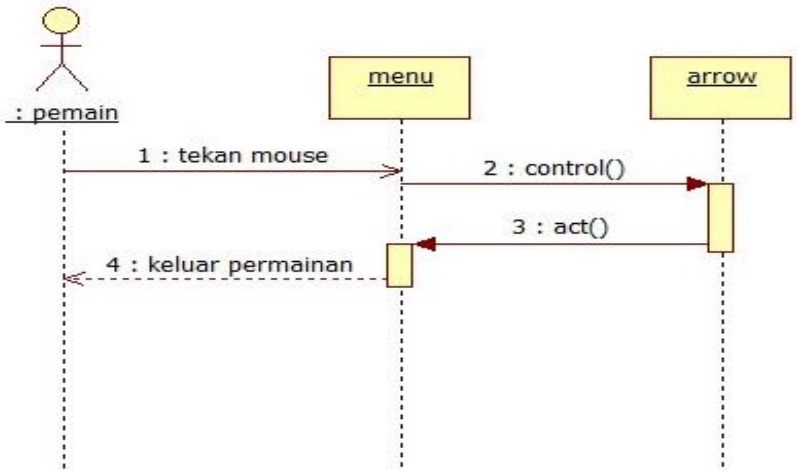
Sequence diagram menggambarkan tindakan objek pada *use case* dengan mendeskripsikan waktu hidup objek dan *message* yang dikirimkan dan diterima antar objek. *Sequence diagram* dapat dilihat pada gambar berikut.

1. *Squence diagram* memainkan permainan



Gambar 3.30 *Squence diagram* memainkan permainan

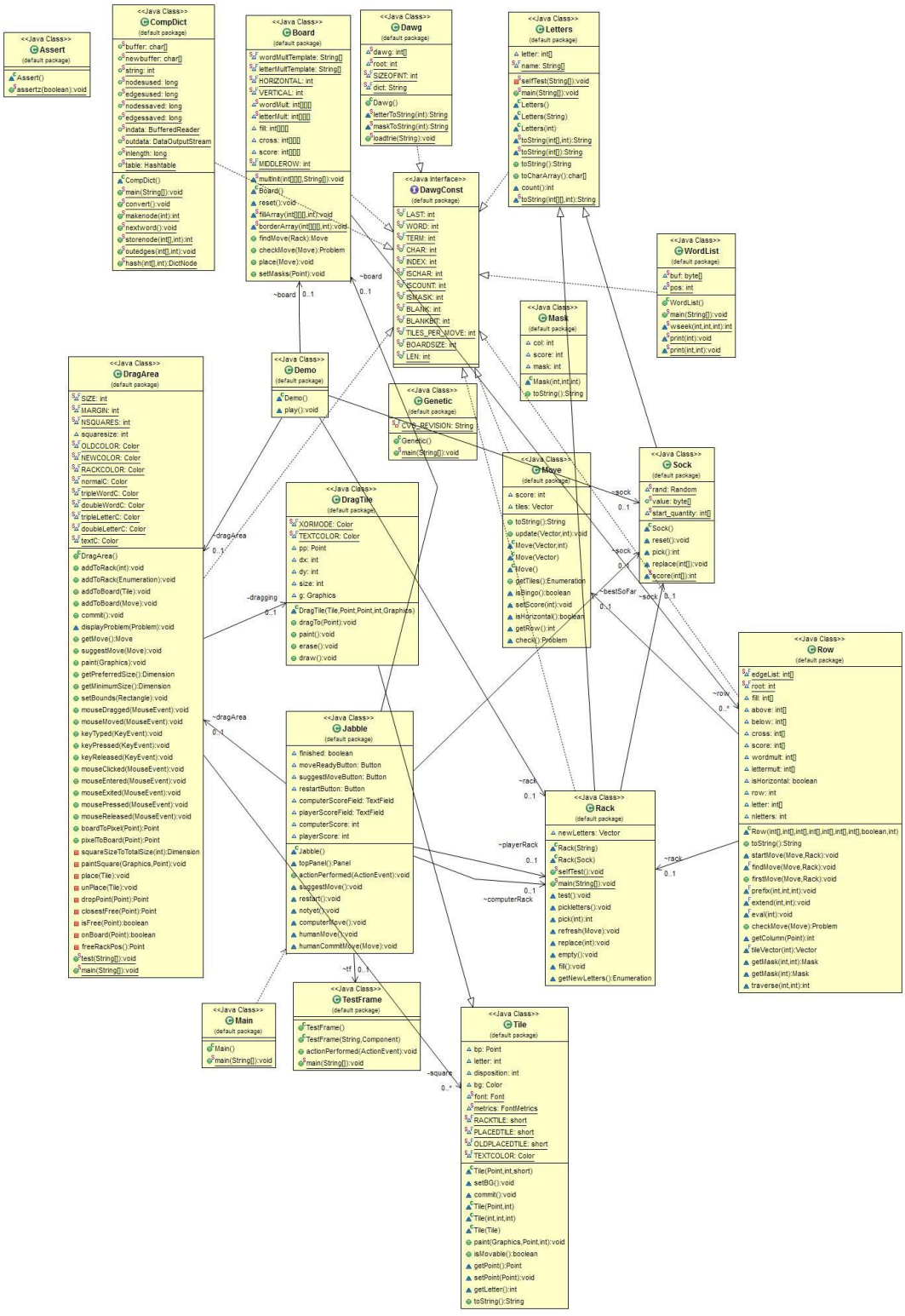
2. *Squence diagram* keluar permainan



Gambar 3.31 *Squence diagram* keluar permainan

3.5.4.7. Class Diagram

Class diagram merupakan gambaran sistem berupa relasi antar kelas-kelas ataupun paket-paketnya. *Class diagram* dijelaskan melalui gambar 3.32 berikut.

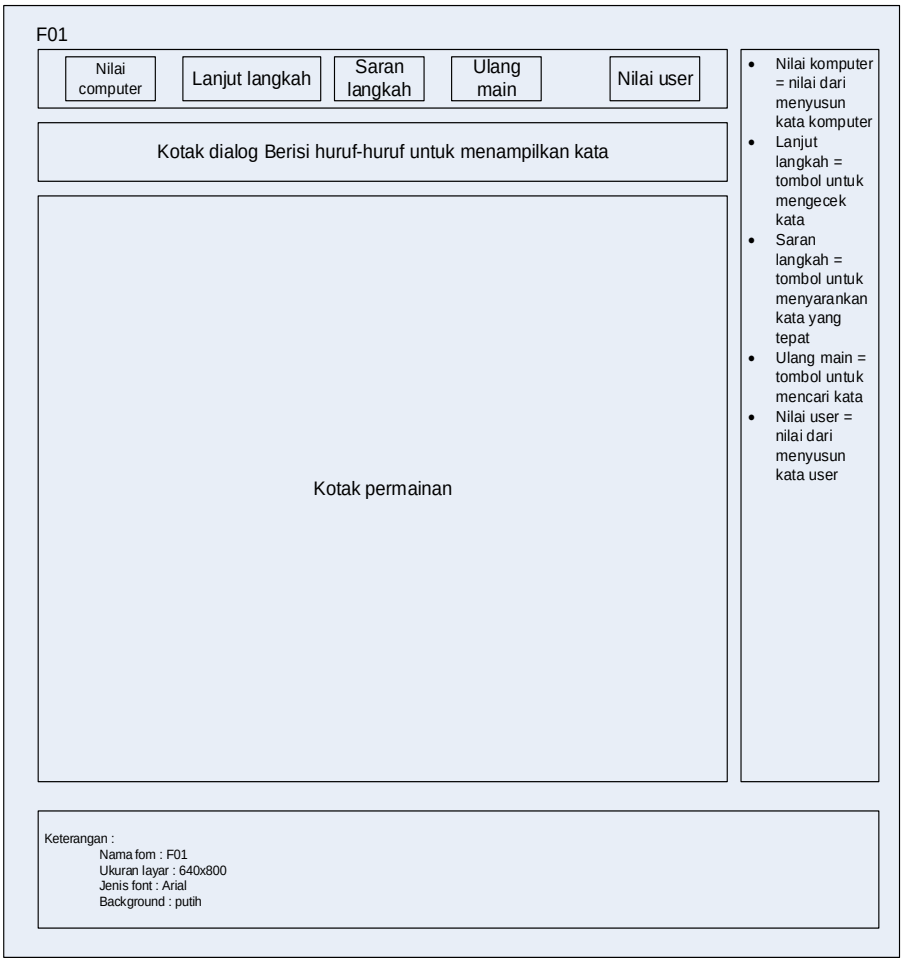

Assert
Assert()
assertz(boolean):void
CompDict
buffer: char[]
newbuffer: char[]
string: int
nodesused: long
edgesused: long
nodessaved: long
edgessaved: long
indata: BufferedReader
outdata: DataOutputStream
inlength: long
table: Hashtable
CompDict()
main(String[]):void
convert():void
makenode(int):int
nextword():void
storenode(int[],int):int
outedges(int[],int):void
hash(int[],int):DictNode
Board
wordMultTemplate: String[]
letterMultTemplate: String[]
HORIZONTAL: int
VERTICAL: int
wordMult: int[][]
letterMult: int[][]
fill: int[][]
cross: int[][]
score: int[][]
MIDDLEROW: int
multinit(int[][],String[]):void
Board()
reset():void
fillArray(int[][],int):void
borderArray(int[][],int):void
findMove(Rack):Move
checkMove(Move):Problem
place(Move):void
setMasks(Point):void
Dawg
dawg: int[]
root: int
SIZEOFINT: int
dict: String
Dawg()
letterToString(int):String
maskToString(int):String
loadtrie(String):void
Letters
letter: int[]
name: String[]
selfTest(String[]):void
main(String[]):void
Letters()
Letters(String)
Letters(int)
toString(int[],int):String
toString(int[]):String
toString():String
toCharArray():char[]
count():int
toString(int[][],int):String
DawgConst
LAST: int
WORD: int
TERM: int
CHAR: int
INDEX: int
ISCHAR: int
ISCOUNT: int
ISMASK: int
BLANK: int
BLANKBIT: int
TILES_PER_MOVE: int
BOARDSIZE: int
LEN: int
WordList
buf: byte[]
pos: int
WordList()
main(String[]):void
wseek(int,int,int):int
print(int):void
print(int,int):void
Mask
col: int
score: int
mask: int
Mask(int,int,int)
toString():String
Demo
Demo()
play():void
Genetic
CVS-REVISION: String
Genetic()
main(String[]):void
DragArea
SIZE: int
MARGIN: int
NSQUARES: int
squaresize: int
OLDCOLOR: Color
NEWCOLOR: Color
RACKCOLOR: Color
normalC: Color
tripleWordC: Color
doubleWordC: Color
tripleLetterC: Color
doubleLetterC: Color
textC: Color
DragArea()
addToRack(int):void
addToRack(Enumeration):void
addToBoard(Tile):void
addToBoard(Move):void
commit():void
displayProblem(Problem):void
getMove():Move
suggestMove(Move):void
paint(Graphics):void
getPreferredSize():Dimension
getMinimumSize():Dimension
setBounds(Rectangle):void
mouseDragged(MouseEvent):void
mouseMoved(MouseEvent):void
keyTyped(KeyEvent):void
keyPressed(KeyEvent):void
keyReleased(KeyEvent):void
mouseClicked(MouseEvent):void
mouseEntered(MouseEvent):void
mouseExited(MouseEvent):void
mousePressed(MouseEvent):void
mouseReleased(MouseEvent):void
boardToPixel(Point):Point
pixelToBoard(Point):Point
squareSizeToTotalSize(int):Dimension
paintSquare(Graphics,Point):void
place(Tile):void
unPlace(Tile):void
dropPoint(Point):Point
closestFree(Point):Point
isFree(Point):boolean
onBoard(Point):boolean
freeRackPos():Point
test(String[]):void
main(String[]):void
DragTile
XORMODE: Color
TEXTCOLOR: Color
pp: Point
dx: int
dy: int
size: int
g: Graphics
DragTile(Tile,Point,Point,int,Graphics)
dragTo(Point):void
paint():void
erase():void
draw():void
Move
score: int
tiles: Vector
toString():String
update(Vector,int):void
Move(Vector,int)
Move(Vector)
Move()
getTiles():Enumeration
isBingo():boolean
setScore(int):void
isHorizontal():boolean
getRow():int
check():Problem
Sock
rand: Random
value: byte[]
start_quantity: int[]
Sock()
reset():void
pick():int
replace(int[]):void
score(int[]):int
Row
edgeList: int[]
root: int
fill: int[]
above: int[]
below: int[]
cross: int[]
score: int[]
wordmult: int[]
lettermult: int[]
isHorizontal: boolean
row: int
letter: int[]
nletters: int
Row(int[],int[],int[],int[],int[],int[],boolean,int)
toString():String
startMove(Move,Rack):void
findMove(Move,Rack):void
firstMove(Move,Rack):void
prefix(int,int,int):void
extend(int,int):void
eval(int):void
checkMove(Move):Problem
getColumn(Point):int
tileVector(int):Vector
getMask(int,int):Mask
getMask(int):Mask
traverse(int,int):int
Jabble
finished: boolean
moveReadyButton: Button
suggestMoveButton: Button
restartButton: Button
computerScoreField: TextField
playerScoreField: TextField
computerScore: int
playerScore: int
Jabble()
topPanel():Panel
actionPerformed(ActionEvent):void
suggestMove():void
restart():void
notyet():void
computerMove():void
humanMove():void
humanCommitMove(Move):void
Rack
newLetters: Vector
Rack(String)
Rack(Sock)
selfTest():void
main(String[]):void
test():void
pickletters():void
pick(int):int
refresh(Move):void
replace(int):void
empty():void
fill():void
getNewLetters():Enumeration
Main
Main()
main(String[]):void
TestFrame
TestFrame()
TestFrame(String,Component)
actionPerformed(ActionEvent):void
main(String[]):void
Tile
bp: Point
letter: int
disposition: int
bg: Color
font: Font
metrics: FontMetrics
RACKTILE: short
PLACEDTILE: short
OLDPLACEDTILE: short
TEXTCOLOR: Color
Tile(Point,int,short)
setBG():void
commit():void
Tile(Point,int)
Tile(int,int,int)
Tile(Tile)
paint(Graphics,Point,int):void
isMovable():boolean
getPoint():Point
setPoint(Point):void
getLetter():int
toString():String
~board
0..1
~dragArea
~dragging
~sock
~bestSoFar
~row
~rack
~playerRack
~computerRack
~tf
-square
0..*

Gambar 3.32 *Class diagram* permainan

3.5.4.8. Perancangan Antarmuka

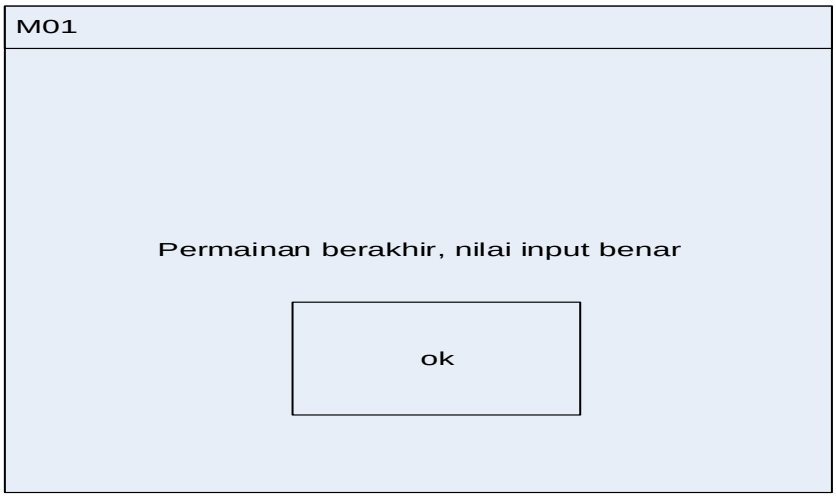
Perancangan antarmuka bertujuan untuk memberikan gambaran tentang aplikasi yang akan dibangun, sehingga akan mempermudah dalam mengimplementasikan aplikasi serta akan memudahkan pembuatan aplikasi. Antarmuka permainan merupakan halaman yang muncul pertama kali pada saat memainkan game. Berikut perancangan antarmuka permainan :

1. Perancangan antarmuka permainan.



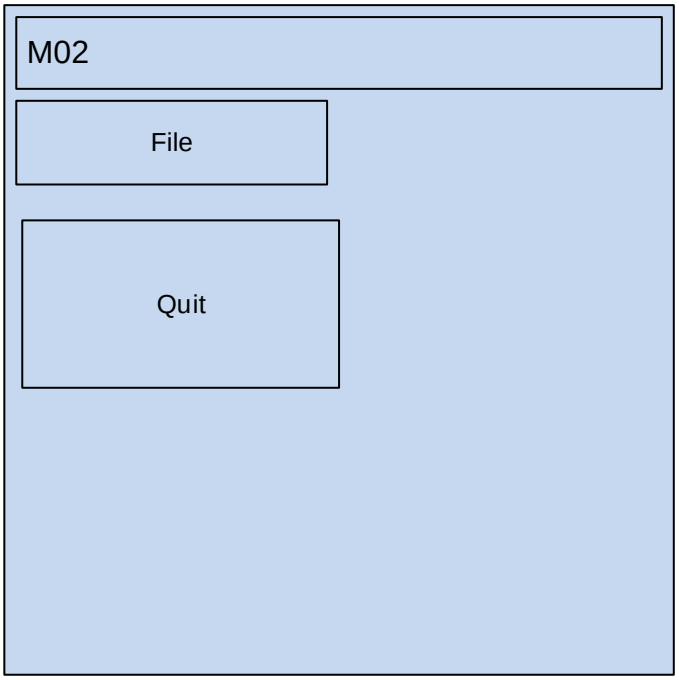
Gambar 3.33 Antarmuka permainan

2. Pesan ketika permaianan terselesaikan.



Gambar 3.34 Pesan ketika permaianan terselesaikan

3. Pesan keluar permainan.



Gambar 3.35 Pesan keluar permainan

3.5.4.9. Jaringan Semantik

Untuk mengetahui keterhubungan dari setiap tampilan yang dibuat dapat menggunakan jaringan semantik.

Gambar 3.36 adalah jaringan semantik pada *game srabble*.

