BAB II

TINJAUAN PUSTAKA

II.1. Penelitian Terdahulu

Penelitian yang dilakukan oleh Fari Ardilla Adrianto, dkk (2015) yang berjudul **"Penerapan Pewarnaan Graf sebagai Metode untuk Mencari Solusi Permainan Sudoku"** menghasilkan sebuah aplikasi yang menerapkan teknik pewarnaan graf sebagai salah satu teknik untuk menyelesaikan permainan Sudoku setelah mentransformasi matriks Sudoku menjadi suatu *Graf*.

Penelitian lain yang dilakukan oleh Misbakhul Mustofin, dkk (2013), dengan judul **"Aplikasi Permainan Sudoku Huruf Hijaiyah Menggunakan Algoritma Backtracking dan Multiplicative CRNG Sebagai Pembangkit dan Penyelesai Permainan"** menghasilkan sebuah aplikasi yang menerapkan algoritma *Backtracking* (Runut-Balik) dalam menyelesaikan permasalahan yang memiliki banyak kemungkinan karena algoritma ini tidak memeriksa semua kemungkinan yang ada. Algoritma ini hanya mempertimbangkan kemungkinan yang mengarah kepada solusi, sehingga proses pencarian menjadi jauh lebih cepat. Algoritma *Multiplicative CRNG* adalah algoritma pembangkit bilangan acak yang baik karena tidak membangkitkan bilangan yang sama secara berturut-turut. Pembangkit dan penyelesai permainan Sudoku Hijaiyah merupakan permasalahan yang dapat diselesaikan dengan baik menggunakan perpaduan antara algoritma *Backtracking* dan algoritma *Multiplicative CRNG*.

Juga terdapat penelitian lain yang dilakukan oleh Valdo Septiansen Widjaja, dkk (2013), dengan judul **"Implementasi Algoritma Backtracking dengan Optimasi Menggunakan Teknik Hidden Single pada Penyelesaian Permainan Sudoku"** menghasilkan sebuah aplikasi yang menerapkan algoritma backtracking dalam penyelesaian *puzzle* Sudoku. Hanya saja, algoritma ini masih membutuhkan waktu yang cukup lama dalam melakukan komputasi penyelesaian *puzzle* Sudoku. Dalam penelitian ini, dibangun sebuah aplikasi untuk mengoptimalkan algoritma backtracking dalam menyelesaikan *puzzle* Sudoku menggunakan sebuah teknik optimasi yang disebut teknik *hidden single*. Aplikasi ini telah berhasil mengimplementasikan teknik *hidden single* ke dalam algoritma backtracking yang digunakan. Hasil uji coba penelitian menunjukkan bahwa dengan menggunakan optimasi teknik *hidden single*, algoritma backtracking mampu mengoptimalkan waktu dan kinerja komputasi pada penyelesaian *puzzle* Sudoku.

Berdasarkan hasil penelitian yang terdahulu, maka peneliti membuat sebuah aplikasi untuk menganalisa penyelesaian masalah *game* sudoku berbasis android dengan algoritma *Brute Force*.

II.2. Perancangan

Dikutip dari jurnal yang dibuat oleh Sandro (2013), perancangan adalah analisis sistem, persiapan untuk merancangan dan implementasi agar dapat menyelesaikan apa yang harus diselesaikan serta mengkonfigurasikan komponen - komponen perangkat lunak ke perangkat keras. Sedangkan menurut Adi Nugroho

(2004), menyatakan bahwa Perancangan adalah strategi untuk memecahkan masalah dan mengembangkan solusi terbaik bagi permasalahan itu.

II.3. Aplikasi

Dikutip dari jurnal yang dibuat oleh Sandro (2013), aplikasi adalah penggunaan atau penerapan suatu konsep yang menjadi pokok pembahasan. Aplikasi dapat diartikan juga sebagai program komputer yang dibuat untuk menolong manusia dalam melaksanakan tugas tertentu.

Dikutip dari jurnal yang dibuat oleh Adhytio, dkk (2014), aplikasi merupakan rangkaian kegiatan atau perintah untuk di eksekusi oleh komputer atau suatu perangkat lunak komputer yang memanfaatkan kemampuan komputer langsung untuk melakukan suatu tugas yang diinginkan pengguna.

II.4. *Game*

Game telah menjadi bagian dalam kehidupan manusia modern sekarang ini. Stres sering kali melanda manusia saat terlalu sibuk dengan pekerjaan mereka, saat itulah *game* bisa menjadi alternatif utama untuk menghilangkan rasa stres. *Game* biasanya memberikan hiburan tersendiri bagi orang yang memainkannya. Selain dapat menghibur *game* juga dapat menjadi alternatif untuk menambah wawasan ilmu pengetahuan (Bonifacio Barros, dkk, 2018)

II.4.1. Defenisi *Game*

Game atau permainan adalah sesuatu yang dapat dimainkan dengan aturan tertentu sehingga ada yang menang dan ada yang kalah, biasanya dalam konteks tidak serius dengan tujuan *refreshing*. Bermain *game* sudah dapat dikatakan sebagai *lifestyle* masyarakat dimasa kini (Aji Hunadi, 2012).

Terdapat beberapa pengertian tentang *game* oleh beberapa ahli yaitu: Menurut Agustinus Nilwan dalam bukunya Pemrograman Animasi dan *Game* Profesional terbitan Elex Media Komputindo, *Game* merupakan permainan komputer yang dibuat dengan teknik dan metode animasi. Jika ingin mendalami pengunaan animasi haruslah memahami pembuatan game. Atau jika ingin membuat *game*, maka haruslah memahami teknik dan metode animasi, sebab keduanya saling berkaitan.

1. Menurut Clark C. Abt, *Game* adalah kegiatan yang melibatkan keputusan pemain, berupaya mencapai tujuan dengan dibatasi oleh konteks tertentu (misalnya, dibatasi oleh peraturan).
2. Menurut Bernard Suits *Game* adalah upaya sukarela untuk mengatasi rintangan yang tidak perlu .
3. Menurut Greg Costikyan, *Game* adalah sebentuk karya seni di mana peserta, yang disebut Pemain, membuat keputusan untuk mengelola sumberdaya yang dimilikinya melalui benda di dalam game demi mencapai tujuan. (https://elib.unikom.ac.id/download.php?id=203928)

II.4.2. Klasifikasi *Game*

Klasifikasi *game* dimaksudkan untuk memudahkan pengelompokan jenis **game**. Beberapa klasifikasi *game* adalah seperti berikut :

1. *Game as Game*, *game* yang dimaksud adalah *game* untuk kesenangan atau *fun*.
2. *Game as Media*, tujuan utama dari *Game As Media* adalah untuk menyampaikan pesan tertentu, menyampaikan pesan dari pembuat *game* tersebut.
3. *Game Beyond Game*, bisa disebut juga dengan istilah *gamification*. *Gamification* adalah penerapan konsep atau cara berpikir *game design* ke dalam lingkup *non-game* (Kurniawan Teguh Martono, 2015).

Selain klasifikasi *game*, terdapat jenis *platform* yang digunakan dalam pengembangan atau pengaplikasian *game*. Beberapa jenis *platform game* adalah sebagai berikut :

1. *Arcade Games*, yaitu yang sering disebut ding-dong di Indonesia, biasanya berada di daerah / tempat khusus dan memiliki *box* atau mesin yang memang khusus di *design* untuk jenis *video games* tertentu dan tidak jarang bahkan memiliki fitur yang dapat membuat pemainnya lebih merasa “masuk” dan “menikmati”, seperti pistol, kursi khusus, sensor gerakan, sensor injakkan dan stir mobil.
2. *PC Games* , yaitu *video game* yang dimainkan menggunakan *Personal Computers*.

3. *Console Games*, yaitu *video games* yang dimainkan menggunakan *console* tertentu, seperti *Playstation 2*, *Playstation 3*, *XBOX 360*, dan *Nintendo Wii*.
4. *Handheld games*, yaitu yang dimainkan di *console* khusus *video game* yang dapat dibawa kemana-mana, contoh *Nintendo DS* dan *Sony PSP*.
5. Mobile games, yaitu yang dapat dimainkan atau khusus untuk *mobile phone* atau PDA (Kurniawan Teguh Martono, 2015).

II.5. Kecerdasan Buatan (*Artificial Intelligence*)

Kecerdasan buatan yang biasa disingkat AI (*Artificial Intelligence*) merupakan ilmu tentang bagaimana membangun suatu sistem komputer yang menunjukkan kecerdasan dalam berbagai cara. AI merupakan area penelitian yang dinamis dalam topik riset ilmu komputer. Sampai saat ini, telah banyak penelitian mengenai perkembangan AI di antaranya *neural network*, *evolutionary computing*, *machine learning*, natural language processing, dan *object oriented programming*. Kenyataannya, saat ini banyak *game* yang membangun AI untuk meningkatkan ketertarikan pengguna. Sehingga adanya AI merupakan salah satu faktor penting yang harus ada dalam *game* (Andhik Ampuh Yunanto, 2017).

Kecerdasan komputasional (CI) dan kecerdasan buatan (AI) memiliki tujuan akhir yang sama yakni mendapatkan suatu kecerdasan sistem yang dapat menunjukkan perilaku intelektual seperti perilaku manusia. Perbedaan kecerdasan buatan dan kecerdasan komputasional ialah terletak pada komputasi dimana AI melakukan hard computing sedangkan CI melakukan soft computing. Selain itu, Bezdek yang seorang ilmuwan mengasumsikan bahwa CI merupakan bagian dari

AI. Sehingga beberapa metode tentang AI juga dapat digunakan pada ilmu kecerdasan komputasional (Andhik Ampuh Yunanto, 2017).

Saat ini juga banyak sekali *game* yang memiliki kecerdasan buatan (AI) dan kecerdasan komputasional didalam suatu permainan. Menurut Georgios, *Game* AI merupakan *game* yang mengubah metode, proses, dan algoritma pada kecerdasan tersebut yang akan diaplikasikan ke pembuatan dan pengembangan *game*. Dia menyebutkan terdapat tiga panorama dalam *game* AI yakni perspektif metode (komputer), perspektif pengguna (manusia) dan perspektif interaksi pemain. Dalam *game* AI, Dagsthul juga menyebutkan terdapat sepuluh jenis *game* AI. Hal ini menunjukkan bahwa kecerdasan buatan dan kecerdasan komputasional sering digunakan pada kebanyakan game saling memiliki ketergantungan interaksi terhadap pemain. Sehingga AI memiliki peran yang penting untuk meningkatkan letertarikan pengguna dalam bermain *game* (Andhik Ampuh Yunanto, 2017).

II.6. Permainan Sudoku

Salah satu *game* logika yang kini mulai banyak diminati oleh banyak kalangan adalah “Sudoku”. Sudoku adalah sebuah permainan teka-teki angka sederhana yang telah dikenal seluruh dunia mulai dari anak-anak sampai orang tua. Di Jepang, ribuan *puzzle* mampu diselesaikan setiap harinya. Bahkan di beberapa sekolah sudoku telah menjadi bagian dari kurikulum. Untuk memainkan *puzzle* angka ini pemain tidak perlu berpikir keras, tidak harus mempunyai intelektualitas yang tinggi, dan tidak perlu handal berhitung, karena yang

dibutuhkan hanya kemampuan berpikir secara logika, kesabaran tinggi dan ketajaman akurasi (Jimmy Hadinata, 2011)

Sudoku adalah sebuah permainan teka-teki angka yang berbasis logika. Pada umumnya, sebuah *puzzle* Sudoku terdiri dari 81 kotak yang disusun menjadi 9 baris, 9 kolom, dan 9 sub bagian. Tujuan utama dari permainan ini adalah mengisi seluruh kotak tersebut dengan angka 1 sampai 9. Permainan Sudoku mengharuskan pemainnya untuk berpikir secara logis sesuai dengan aturan-aturan permainan yang ada. Namun demikian, para ilmuwan komputer dan matematikawan berusaha untuk meneliti permainan ini karena tingkat kerumitan dan banyaknya variasi *puzzle* Sudoku yang masing- masing memiliki suatu solusi unik (Valdo Septiansen Widjaja, 2013).

II.7. Metode *Brute Force*

Brute Force adalah sebuah pendekatan langsung (*straight forward*) untuk memecahkan suatu masalah, yang biasanya didasarkan pada pernyataan masalah (*problem statement*) dan definisi konsep yang dilibatkan. Pada dasarnya algoritma *Brute Force* adalah alur penyelesaian suatu permasalahan dengan cara berpikir yang sederhana dan tidak membutuhkan suatu pemikiran yang lama. Sebenarnya, algoritma *Brute Force* merupakan algoritma yang muncul karena pada dasarnya alur pikir manusia adalah *Brute Force* (Chichi Rizka Gunawan, dkk, 2018)

Beberapa karakteristik dari algoritma *Brute Force* dapat dijelaskan sebagai berikut : (May Aprina Saragih, 2013)

1. Membutuhkan jumlah langkah yang banyak dalam menyelesaikan suatu permasalahan sehingga jika diterapkan menjadi suatu algoritma program aplikasi akan membutuhkan banyak memori.
2. Digunakan sebagai dasar dalam menemukan suatu solusi yang lebih efektif.
3. Banyak dipilih dalam penyelesaian sebuah permasalahan yang sederhana karena kemudahan cara berpikirnya.

Berikut ini beberapa kelebihan yang dimiliki oleh *Brute Force*, yaitu:

1. Algoritma *Brute Force* dapat digunakan untuk memecahkan hampir sebagian besar masalah.
2. Sederhana dan mudah dimengerti.
3. Menghasilkan algoritma yang layak untuk beberapa masalah penting seperti pencarian, pengurutan, pencocokan string, perkalian matriks.
4. Menghasilkan algoritma baku (standar) untuk tugas-tugas komputasi seperti penjumlahan/perkalian N buah bilangan, menentukan elemen minimum atau maksimum ditabel (Bayu Widia Santoso, 2016).

Berikut ini beberapa kelemahan yang dimiliki oleh *Brute Force*, yaitu:

1. Jarang menghasilkan algoritma yang mangkus/efektif.
2. Lambat sehingga tidak dapat diterima.
3. Tidak sekreatif teknik pemecahan masalah lainnya (Bayu Widia Santoso, 2016).

Secara sistematis, langkah-langkah yang dilakukan Algoritma *Brute Force* pada saat mencocokkan *string* adalah:

1. Algoritma *Brute Force* mulai mencocokkan *pattern* pada awal teks.

2. Dari kiri ke kanan, algoritma ini akan mencocokkan karakter per karakter *pattern* dengan karakter di teks yang bersesuaian, sampai salah satu kondisi berikut dipenuhi.
3. Karakter di *pattern* dan di teks yang dibandingkan tidak cocok (*mismatch*).
4. Semua karakter di *pattern* cocok. Kemudian Algoritma akan memberitahukan penemuan di posisi ini.
5. Algoritma kemudian terus menggeser pattern sebesar satu ke kanan
6. Mengulangi langkah ke-2 sampai pattern berada di ujung teks (http://repository.uin-suska.ac.id/3570/3/BAB%20II.pdf)

II.8. Android

Android adalah sistem operasi (*Operating System*) yang umumnya digunakan pada perangkat dengan navigasi *full touch screen* yang biasa dimiliki oleh *smartphone* dan komputer tablet. Android sudah diambil alih oleh perusahaan *Google Inc.* yang telah membelinya pada tahun 2005 dari *Android Inc. Google* menyediakan *software/tools* yang dikembangkan khusus untuk dijadikan alat pengembang aplikasi android (Angga Aditya Permana, 2018).

Pada tanggal 23 september 2008, sistem operasi *Android* versi 1.0 resmi diluncurkan. Sekitar sebulan berikutnya, pada tanggal 22 Oktober 2008, *smartphone* pertama yang menjalankan Android 1.0 itu, yaitu HTC Dream, diluncurkan ke pasar. Pada tanggal 9 februari, Android versi 1.1 diluncurkan untuk memperbaiki bug dari versi sebelumnya dan menambah fitur yang tersedia. Setelah versi 1.1, rilis *Android* berikutnya menggunakan nama makanan manis

dengan urutan *alfabetis*, dimulai dengan 1.5 *Cupcake* yang diluncurkan pada tanggal 30 April 2009. Rilis-rilis *Android* selanjutnya, yaitu *Donut*, *clair*, *Froyo*, dan *Gingerbread* semua dibuat untuk smartphone. Namun, *Apple* meluncurkan *iPad* pada tahun 2010 dan meningkatkan ketertarikan masyarakat luas kepada *computer tablet*. Beberapa pengembang *Android* mencoba mengembangkan *tablet Android* untuk menyaingi *iPad*, seperti *Samsung Galaxy Tab* yang menggunakan Gingerbread yang dikustomisasi. *Google* dan OHA pun bergerak dengan melakukan pengembangan *Android* versi baru yang lebih optimal untuk tablet. Pada tanggal 22 februari 2011, android Honeycomb diluncurkan ke pasar dan pada tanggal 24 februari 2011, tablet pertama yang menggunakan *honeycomb*, yaitu *Motorola Xoom*, diluncurkan ke pasar. Pada tanggal 19 Oktober 2011, *Android* meluncurkan *Ice Cream Sandwich* versi ini dapat bekerja secara optimal baik di *smartphone* maupun di *tablet*. Rilis *android* berikutnya, yaitu *Jelly Bean*, bertujuan untuk semakin meningkatkan apa yang sudah tersedia di *Ice Cream Sandwich*, dengan memperbaiki *bug-bug* dn menambahkan fitur-fitur. Pada tanggal 3 september 2013, diumumkan versi *Android* selanjutnya adalah *Android* 4.4 Kit Kat. Android sudah mendapatkan izin dari *Nastle* dan *Hershey* selaku pemilik dagang dagang Kit Kat. Sebelum pengumuman ini, banyak yang berspekulasi bahwa versi *Android* berikutnya akan diberi nomor 5.0 dengan nama *Key Lime Pie* brikut adalah tabel untuk semua sistem operasi *Android* yang sudah diluncurkan sampai sekarang. Saat buku ini ditulis, sistem operasi *Android* yang terbaru adalah Android 6.0 Marshmallow (Satyaputra dan Aritonang, 2014).

Tabel II.1. Versi Android

Versi	Nama	Rilis	Catatan
1.0	*Android 1.0*	23 September 2008	Android pertama hanya untuk *smartphone*
1.1	*Android 1.1*	9 Februari 2008	
1.5	*Cupcake*	30 April 2009	Mulai pakai kode nama
1.6	*Donut*	15 September 2009	
2.0 - 2.1	*Éclair*	26 Oktober 2009 (2.0) 12 Januari 2010 (2.1)	
2.2	*Froyo*	20 Mei 2010	
2.3	*Gingerbread*	6 Desember 2010	
3.0-3.2	*Honeycomb*	22 Februari 2011 (3.0) 10 Mei 2011 (3.1) 15 Juli 2011 (3.2)	Hanya untuk tablet
4.0	*ICS (Ice Cream Sandwich)*	19 Oktober 2011	*Smartphone* dan tablet
4.1-4.3	*Jelly Bean*	9 Juli 2012 (4.1) 13 November 2012 (4.2) 24 Juli 2013 (4.3)	*Update* untuk memperbaiki dan menembah *fitur* pada *ICS*
4.4	*Kit kat*	3 September 2013	
5.0	*Lollipop*	12 November 2014 (5.0) 9 Maret 2015 (5.1)	
6.0	*Marshmallow*	5 Oktober 2015	Terdapat *daze mode*, *Do Not Disturb mode*, mendukung *USB tipe C*, mndukung pembacaan *fingerprint*.

(Sumber : Satyaputra dan Aritonang; 2014)

II.9. UML (*Unified Modeling Language*)

Menurut Windu Gata, Grace (2013:4), *Unified Modeling Language* (UML) adalah bahasa spesifikasi standar yang dipergunakan untuk mendokumentasikan, menspesifikasikan dan membangun perangkat lunak. UML merupakan metodologi dalam mengembangkan sistem berorientasi objek dan juga merupakan alat untuk mendukung pengembangan sistem (Ade Hendini : 2016).

Alat bantu yang digunakan dalam perancangan berorientasi objek berbasiskan UML dijelaskan berikut ini.

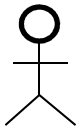
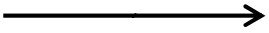
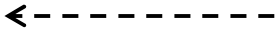
II.8.1. *Use Case Diagram*

Use Case Diagram merupakan pemodelan untuk menggambarkan kelakuan (*behavior*) sistem yang akan dibuat. *Use Case* mendeskripsikan sebuah interaksi antara satu atau lebih aktor dengan sistem yang akan dibuat. Dapat dikatakan *Use Case* digunakan untuk mengetahui fungsi apa saja yang ada didalam sistem informasi dan siapa saja yang berhak menggunakan fungsi-fungsi tersebut.

Simbol-simbol yang digunakan dalam *Use Case Diagram* dapat dilihat pada tabel II.2, yaitu :

Tabel II.2. *Use Case Diagram*

Gambar	Keterangan
	*Use Case* menggambarkan fungsionalitas yang disediakan sistem sebagai unit-unit yang bertukar pesan antara unit dengan aktor, biasanya dinyatakan dengan menggunakan kata kerja.

	Aktor adalah *abstraction* dari orang atau sistem yang lain yang mengaktifkan fungsi dari target sistem. Untuk mengidentifikasikan aktor, harus ditentukan pembagian tenaga kerjadan tugas-tugas yang berkaitan dengan peran pada konteks target sistem. Orang atau sistem bisa muncul dalam beberapa peran. Perlu dicatat bahwa aktor berinteraksi dengan *use case*, tetapi tidak memiliki *control* terhadap *use case*.
	Asosiasi antara aktor dan *use case*, digambarkan dengan garis tanpa panah yang mengindikasikan siapa atau apa yang meminta interaksi secara langsung dan bukannya mengindikasikan aliran data.
	Asosiasi antara aktor dan *use case* yang menggunakan panah terbuka untuk mengindikasikan bila aktor berinteraksi secara pasif dengan sistem.
 <<include>>	*Include*, merupakan di dalam *use case* lain (*required*) atau pemanggilan *use case* oleh *use case* lain, contohnya adalah pemanggilan sebuah fungsi program.
 <<extend>>	*Extend*, merupakan perluasan dari *use case* lain jika kondisi atau syarat terpenuhi.

(Sumber : Ade Hendini ; 2016)

II.8.2. *Class Diagram*

Merupakan hubungan antara kelas dan penjelasan detail tiap-tiap kelas di dalam model desain dari suatu sistem, juga memperlihatkan aturan-aturan dan tanggung jawab entitas yang menentukan perilaku sistem.

Class Diagram juga menunjukkan atribut-atribut dan operasi-operasi dari sebuah kelas dan *constraint* yang berhubungan dengan objek yang dikoneksikan. *Class Diagram* secara khas meliputi Kelas (*Class*), *Relasi Assosiations*, *Generalitation* dan *Aggregation*, atribut (*Attributes*), Operasi (*Operation/Method*)

dan *Visibility*, tingkat akses objek eksternal kepada suatu operasi atau atribut. Hubungan antara kelas mempunyai keterangan yang disebut dengan *multiplicity* atau *cardinality*, dapat dilihat pada tabel II.3 :

Tabel II.3. *Multiplicity Class Diagram*

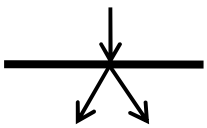
Multiplicity	Penjelasan
1	Satu dan hanya satu
0..*	Boleh tidak ada atau 1 atau lebih
1..*	1 atau lebih
0..1	Boleh tidak ada, maksimal 1
n..n	Batasan antara. Contoh 2..4 mempunyai arti minimal 2 maksimum 4

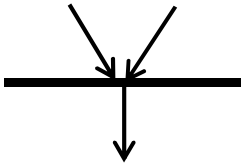
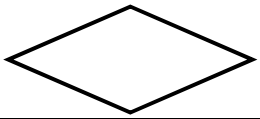
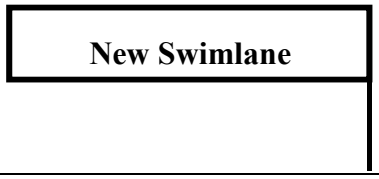
(Sumber : Ade Hendini ; 2016)

II.8.3. *Activity Diagram*

Activity Diagram menggambarkan *workflow* (aliran kerja) atau aktivitas dari sebuah sistem atau proses bisnis. Simbol-simbol yang digunakan dalam *activity diagram* dapat dilihat pada tabel II.4.

Tabel II.4. *Activity Diagram*

Gambar	Keterangan
	*Start point*, diletakkan pada pojok kiri atas dan merupakan awal aktifitas.
	*End point*,akhir aktifitas.
	*Activites*, menggambarkan suatu proses/kegiatan bisnis.
	*Fork* (Percabangan), digunakan untuk menunjukkan kegiatan yang dilakukan secara *pararel* atau untuk menggabungkan dua kegiatan *pararel* menjadi satu.

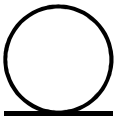
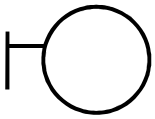
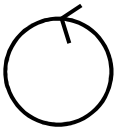
	*Join* (Penggabungan) atau rake, digunakan untuk menunjukkan adanya dekomposisi.
	*Decision Point*, menggambarkan pilihan untuk pengambilan keputusan, *true, false.*
	*Swimlane*, pembagian *activity* diagram untuk menunjukkan siapa melakukan apa.

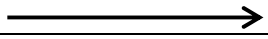
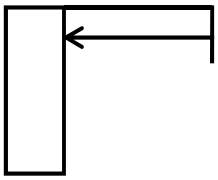
(Sumber : Ade Hendini ; 2016)

II.8.4. *Sequence Diagram*

Sequence Diagram menggambarkan kelakuan objek pada *Use Case* dengan mendeskripsikan waktu hidup objek dan pesan yang dikirimkan dan diterima antara objek. Simbol-simbol yang digunakan dalam *Sequence Diagram,* dapat dilihat pada tabel II.5. berikut.

Tabel II.5. *Sequence Diagram*

Gambar	Keterangan
	*Entity Class*, merupakan bagian dari sistem yang berisi kumpulan kelas berupa entitas-entitas yang membentuk gambaran awal sistem dan menjadi landasan untuk menyusun basis data.
	*Boundary Class*, berisi kumpulan kelas yang menjadi *interface* atau interaksi antara satu atau lebih aktor dengan sistem, seperti tampilan *formentry* dan *form* cetak.
	*Control Class*, suatu objek yang berisi logika aplikasi yang tidak memiliki tanggung jawab kepada entitas, contohnya adalah kalkulasi dan aturan bisnis yang melibatkan berbagai objek.

	*Message*, simbol mengirim pesan antar *class*.
	*Recursive*, menggambarkan pengiriman pesan yang dikirim untuk dirinya sendiri.
	*Activation, activation* mewakili sebuah eksekusi operasi dari objek, panjang kotak ini berbanding lurus dengan durasi aktivitas sebuah operasi.
	Lifeline, garis titik-titik yang berhubungan dengan objek, sepanjang *lifeline* terdapat *activation.*

(Sumber : Ade Hendini ; 2016)