BAB 2
LANDASAN TEORI

2.1 Permainan (*Game*)

Game atau permainan merupakan aktivitas rekreasi dengan tujuan bersenang – senang mengisi waktu luang atau berolahraga ringan. Permainan biasanya dilakukan sendiri atau bersama – sama (kelompok), cara bermain (*gameplay*), budaya. Permainan dalam hal ini, merujuk pada pengertian kelincahan intelektual (*Intellectual Playability Game*) yang juga bisa diartikan sebagai arena keputusan dan aksi pemainnya. Permainan bertujuan untuk menghibur, biasanya permainan banyak disukai oleh anak-anak hingga orang dewasa. Berdasarkan perkembangannya pun permainan dibedakan menjadi dua yaitu, permainan tradisional dan permainan modern. Dari perbedaan dua permainan tersebut dapat dilihat sebagai berikut[5] :

Tabel 2.1 Perbedaan Permainan Modern Dan Permainan Tradisional

Permainan Modern	**Permainan Tradisional**
a. Tidak Memerlukan biaya yang mahal.	a. Memerlukan biaya yang mahal.
b. Meningkatkan kekompakan anak.	b. Bermain secara individual (karena perminan tergantung skill masing – masing anak semakin sering memainkan semkain hebat).
c. Kerjasama antar anak yang kuat	c. Permainan kerjasamanya kurang, lebih ber-ego dan gengsi.

2.1.1 Pengertian Game

Dalam kamus bahasa Indonesia “Game” adalah permainan. Permainan merupakan bagian dari bermain dan bermain juga bagian dari permainan keduanya saling berhubungan. Permainan adalah kegiatan yang kompleks yang didalamnya terdapat peraturan, *play* dan budaya. Sebuah permainan adalah sebuah sistem dimana pemain terlibat dalam konflik buatan, disini pemain berinteraksi dengan sistem dan konflik dalam permainan merupakan rekayasa atau buatan, dalam permainan terdapat peraturan yang bertujuan untuk membatasi perilaku pemain dan menentukan permainan. Game bertujuan untuk menghibur, biasanya game banyak disukai oleh anak – anak hingga orang dewasa. Games sebenarnya penting untuk perkembangan otak, untuk meningkatkan konsentrasi dan melatih untuk memecahkan masalah dengan tepat dan cepat karena dalam game terdapat berbagai konflik atau masalah yang menuntut kita untuk menyelesaikannya dengan cepat dan tepat. Tetapi game juga bisa merugikan karena apabila kita sudah kecanduan game kita akan lupa waktu dan akan mengganggu kegiatan atau aktifitas yang sedang kita lakukan[5].

2.1.2 Klasifikasi Game

Game berdasarkan *genre*-nya terbagi menjadi 10 jenis,yaitu *Board Game, Arcade, Action, Strategy, Shooting, Fighting, Racing, Simulation, Real Time Strategy, Role Playing Game, Masive Multiplayer Online* [6].

1. Board Game

   Boardgame atau dapat kita istilahkan sebagai “permainan papan”. Pada jenis *game* ini, pemain diberikan sebuah tampilan yang berisi tentang masalah untuk diselesaikan. Contoh dari *game* ini antara lain : *Pipedream*, Catur, permainan kartu dan sebagainya.

2. Arcade

   Game bertipe *arcade* merupakan *game* yang menguji kecepatan tangan pemainnya. Pada permainan bertipe *arcade*, semakin tinggi level

permainan, permainan akan berjalan semakin cepat dan sulit. Contoh dari *game* bertipe *arcade* adalah *zuma*, *pacman, Arcanoid* dan sebagainya.

3. Action

   Berbeda dengan *game* bertipe *arcade*, *game* bertipe *action* menjadikan pemain mengendalikan karakter utama dalam *game* tersebut melakukan beberapa kegiatan seperti melompat, menembak dan sebagainya. Contoh dari *game* bertipe *action* adalah *supermario*, petualangan *paddlepop* dan sebagainya.

4. Shooting

   Game bertipe *shooting* sebagian besar menggunakan *mouse* sebagai alat pengendalinya. Pada *game* ini pemain seolah-olah berperan sebagai penembak atau pemain mengendalikan seorang penembak. Contoh *game* bertipe *shooting* adalah *DuckHunt, CounterStrike* dan sebagainya.

5. Fighting

   Game bertipe *fighting* pada dasarnya sama dengan *game* bertipe *action*, hanya saja *game* bertipe *fighting* pemain mengendalikan sebuah karakter untuk berkelahi dengan karakter lain sampai salah satu karakter kalah. Contoh *game* bertipe *fighting* adalah *streetfighter*.

6. Racing

   Game bertipe *racing* pada dasarnya adalah sebuah permainan menggerakkan kamera. Pemain diberikan sebuah kendaraan atau sejenisnya untu menempuh rute tertentu. Contoh dari *game* bertipe *racing* adalah *NeedForSpeed*.

7. Simulation

   Game bertipe simulasi adalah sebuah *game* yang mensimulasikan suatu kegiatan yang sesungguhnya. Contoh dari *game* bertipe simulasi adalah *tycoon*, simulator pesawat, *burgerempire* dan sebagainya.

8. Real Time Strategy

   Game bertipe RTS memposisikan pemain sebagai seorang pemimpin yang mengatur sesuatu (bisa berupa pasukan, koloni, kerajaan

dan sebagainya). Contoh dari *game* bertipe RTS antara lain *warcommander*, *empireearth* dan sebagainya.

9. Role Playing Game

   Pada *game* bertipe RPG pemain memerankan sebuah karakter dalam *game*. Berbeda dengan *game* bertipe *action*, pada game RPG hal yang diutamakan adalah cerita dalam *game*. Selain itu di dalam *game* bertipe RPG biasanya terdapat *subgame* dengan tipe lain. Contoh *game* RPG adalah *Zelda*, *Megaman* dan sebagainya.

10. MMO

    MMO (Masive Multiplayer Online) merupakan *game* yang dapat dimainkan secara bersama-sama pada internet *browser*. Hal yang diutamakan dalam sebuah *game* bertipe MMO adalah kebersamaan dengan pemain lain. Contoh *game* bertipe MMO adalah *Ragnarok*.

2.1.2.1 Berdasarkan Umur Pengguna Permainan Game

Berdasarkan umur pengguna permainan *game* dibagi menjadi beberapa jenis, yaitu[8]:

1. EC (*Early Childhood*), permainan *game* ini memiliki konten yang mungkin cocok untuk anak usia 3 ke atas. Dan tidak memiliki kontent yang dianggap tidak pantas dilihat oleh anak dengan usia tersebut.



Gambar 2.1 EC (*Early Childhood*)[7].

2. E (*Everyone*), permainan *game* ini memiliki konten yang mungkin cocok untuk anak dengan usia 6 atau lebih tua. Judul dalam kategori ini mungkin berisi kartun, fantasi atau kekerasan ringan atau menggunakan bahasa ringan (tidak kasar).



Gambar 2.2 E (*Everyone*)[7].

3. E10+ (*Everyone 10 and older*), permainan *game* ini memiliki konten yang mungkin cocok untuk usia 10 atau lebih tua. Judul dalam kategori ini mungkin berisi kartun, fantasi atau kekerasan ringan, bahasa ringan atau dengan tema sugestif minimal.



Gambar 2.3 E10+ (*Everyone 10 and older*)[7].

4. T (*Teen*), permainan *game* ini memiliki konten yang mungkin cocok untuk usia 13 atau lebih tua. Judul dalam kategori *game* ini mungkin berisi kekerasan, tema sugestif, humor kasar, minimal darah, simulasi perjudian, atau menggunakan bahasa kuat yang minim.



Gambar 2.4 T (*Teen*)[7].

5. M (*Mature*), permainan *game* ini memiliki konten yang mungkin cocok untuk orang usia 17 atau lebih tua. Judul dalam kategori ini mungkin berisi kekerasan *intens*, darah dan gores, konten seksual dan bahasa kuat.



Gambar 2.5 M (*Mature*)[7].

6. AO (*Adults Only*), permainan *game* ini memiliki konten yang hanya dan harus dimainkan oleh orang 18 tahun keatas. Judul dalam kategori ini mungkin termasuk adegan kekerasan berkepanjangan intens dan konten seksual grafis dengan kontent yang berisi gambar atau video.



Gambar 2.6 AO (*Adults Only*)[7].

7. RP (*Rating Pending*), Judul terdaftar sebagai RP (Rating Pending) telah disampaikan kepada ESRB dan sedang menunggu penilaian akhir. (Simbol ini hanya akan muncul dalam iklan sebelum merilis *game*.)



Gambar 2.7 RP (*Rating Pending*)[7].

2.2 Teknik Penyelesaian Masalah AI (*Artificial Intelligence*)

Terdapat empat teknik dasar penyelesaian masalah yang terdapat pada bidang *artificial intelegence* diantaranya adalah[8]:

A. *Searching*

Pada teknik searching atau pencarian ini terdiri dari beberapa langkah untuk merealisasikannya. Langkah pertama adalah mendefinisikan ruang masalah untuk suatu masalah yang dihadapi. Langkah kedua adalah mendefinisikan aturan produksi yang digunakan untuk mengubah suatu keadaan ke keadaan lainnya. Langkag terakhir adalah memilih metode pencarian yang tepat sehingga dapat menemukan solusi terbaik.

B. *Reasoning*

Teknik *reasoning* atau penalaran merupakan teknik penyelesaian masalah dengan cara merepresentasikan masalah ke dalam basis pengetahuan

menggunakan *logic* atau bahasa formal (bahasa yang dipahami komputer). Teknik ini melakukan proses penalaran berdasarkan basis pengetahuannya untuk menemukan solusi.

C. *Planning*

Planning adalah suatu metode penyelesaian masalah dengan cara memecah masalah ke dalam sub-sub masalah yang lebih kecil, menyelesaikan sub-sub masalah satu demi satu, kemudian menggabungkan solusi-solusi dari sub-sub masalah tersebut menjadi sebuah solusi lengkap dengan tetap mengingat dan menangani interaksi yang terdapat pada sub-sub masalah tersebut.

D. *Learning*

Pada ketiga teknik sebelumnya, seseorang harus mengetahui aturan yang berlaku untuk system yang akan dibangunnya. Tetapi, pada masalah tertentu terkadang aturan tidak bisa didefinisikan secara benar ataupun lengkan. Hal tersebut mungkin dikarenakan data-data yang didapat tidak lengkap. Melalui teknik yang disebut *learning* ini, secara otomatis aturan yang diharapkan bisa berlaku umum untuk data-data yang belum pasti diketahui dapat ditemukan.

2.2.1 Algoritma Pencarian (*Searching*)

Metode pencarian disebut penting dalam menyelesaikan permasalahan karena setiap keadaan menggambarkan langkah-langkah untuk menyelesaikan permasalahan. Dalam sebuah permainan metode pencarian akan menentukan langkah apa yang harus dilakukan, dimana setiap langkah menggambarkan kemungkinan posisi dalam sebuah rangkaian deduktif.

Permasalahan pencarian dapat diselesaikan dengan beberapa metode pencarian sebagai berikut [9]:

1. Pencarian Buta (*Blind Search*)

   Blind search atau disebut juga pencarian buta merupakan metode pencarian sederhana yang berusaha mencari kemungkinan penyelesaian,

dan pencarian ini tidak memiliki informasi awal. Ciri-ciri *blind search* adalah sebagai berikut[10]:

a. Membangkitkan simpul berdasarkan urutan.
b. Kalau ada solusi, solusi akan langsung ditemukan.
c. Hanya memiliki informasi tentang *node* yang telah dibuka (*node* selanjutnya tidak diketahui).

Blind search pun di bagi menjadi 3 seperti, BFS (*Breadth-First Search*), DFS (*Depth-First Search*), UCS (*Uniform Cost Search*).

2. Pencarian Terbimbing (*Heuristic Search*)

Heuristic search atau disebut juga pencarian terbimbing merupakan metode pencarian yang memperhatikan nilai *heuristic* (nilai perkiraan), dan pencarian ini akan dapat mencari jarak yang terpendek. *Heuristic* memperkirakan jarak ke *Goal* (tujuan) yang disebut dengan fungsi *heuristic*. Ciri-ciri *heuristic search* adalah sebagai berikut[10]:

a. Mengesampingkan usaha yang dapat memboroskan waktu.
b. Menggunakan fungsi *heuristic* untuk mengevaluasi keadaan dari masalah dan mendapatkan solusi yang diinginkan.
c. Kemungkinan dapat mengorbankan kelengkapan (*completeness*).

Heuristic search pun mempunyai contoh seperti, *Museum Procedure*, *Branch and Bound*, *Dynamic Programming*, *Best First Search*,*Greedy Search*, A* (A-*Star*) *Search*, dan *Hill Climbing Search*.

2.2.2 Algoritma A* (A-*Star*)

Algoritma A* (A-*Star*) adalah algoritma pencarian yang merupakan pengembangan dari algoritma *Best First Search* (BFS). Seperti halnya pada BFS, untuk menemukan solusi, A* juga di bimbing oleh fungsi heuristik, yang menentukan urutan titik mana yang akan dikunjungi terlebih dahulu. Heuristik merupakan penilai yang memberi harga pada tiap *verteks* yang memandu A* mendapatkan solusi yang diinginkan[10].

Dengan fungsi heuristik Algoritma ini membangkitkan *verteks* yang paling mendekati solusi. *Verteks* ini kemudian disimpan suksesornya ke dalam *list* sesuai

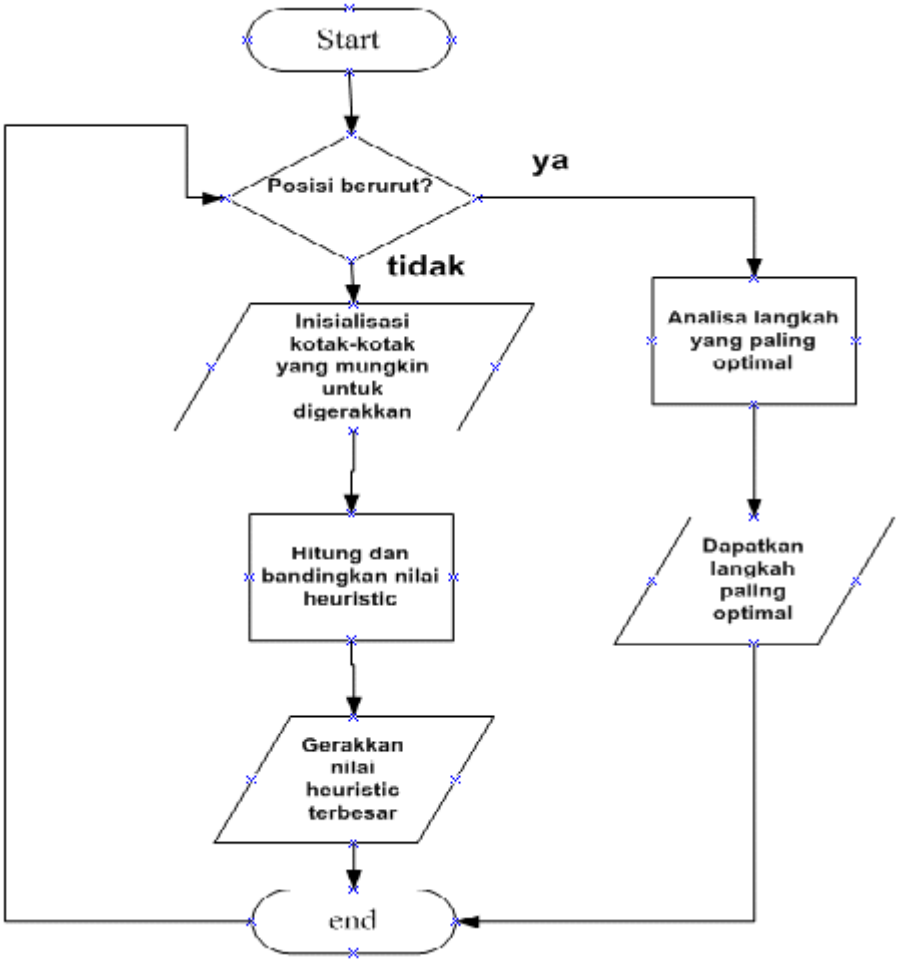
dengan urutan yang paling mendekati solusi terbaik. Kemudian, *verteks* pertama pada *list* diambil, dibangkitkan suksesornya dan kemudian suksesor ini disimpan ke dalam *list* sesuai dengan urutan yang terbaik untuk solusi. *List verteks* ini disebut dengan *verteks* terbuka (*open node*)[10].

Pencarian menggunakan algoritma A* mempunyai prinsip yang sama dengan algoritma BFS, hanya saja dengan dua faktor tambahan.

1. Setiap sisi mempunyai "*cost*" yang berbeda-beda, seberapa besar *cost* untuk pergi dari satu simpul ke simpul yang lain.
2. *Cost* dari setiap simpul ke simpul tujuan bisa diperkirakan. Ini membantu pencarian, sehingga lebih kecil kemungkinan kita mencari ke arah yang salah.

Cost untuk setiap simpul tidak harus berupa jarak. *Cost* bisa saja berupa waktu bila kita ingin mencari jalan dengan waktu tercepat untuk dilalui. Sebagai contoh, bila *user* berkendaraan melewati jalan biasa bisa saja merupakan jarak terdekat, tetapi melewati jalan tol biasanya memakan waktu lebih sedikit.

Agar lebih memahami cara kerja Algoritma A*(A Star) pada sistem ini maka ditampilkan diagram alir (*FlowChart)* fungsi *heuristic* seperti gambar 2.8 berikiut.



Gambar 2.8 *FlowChart* A*(A-Star)

Cost antara simpul adalah jaraknya, dan perkiraan *cost* dari suatu simpul ke simpul tujuan adalah penjumlahan jarak dari simpul tersebut ke simpul tujuan.

Rumus fungsi heuristic **:** $f(n) = g(n) + h(n)$ 2.11

dengan:

$f(n)$ = fungsi evaluasi

$g(n)$ = biaya (cost) yang sudah dikeluarkan dari keadaan sampai keadaan n

$h(n)$ = estimasi biaya untuk sampai pada suatu tujuan mulai dari n

Node dengan nilai terendah merupakan solusi terbaik untuk diperiksa pertama kali pada $g(n) + h(n)$. Dengan fungsi *heuristic* yang memenuhi kondisi tersebut, maka pencarian dengan algoritma A* dapat optimal[10]

Metode A* dapat melakukan *backtracking* jika jalur yang ditempuh ternyata salah. Metode A* dapat melakukannya karena menyimpan jejak yang mungkin sebagai jalur yang optimal. Sebagai contohnya, jika kita sedang menuju suatu kota dan sampai pada persimpangan jalan, dan memutuskan untuk belok kiri daripada ke kanan, dan ternyata jika jalan yang dipilih ternyata salah, kita akan kembali ke persimpangan dan mengambil jalan satunya. Itulah yang dilakukan metode A* ini[10].

2.3 Aplikasi Pembangun Perangkat Lunak

Perangkat lunak adalah istilah umum untuk data yang diformat dan disimpan secara digital, termasuk program komputer, dokumentasinya, dan berbagai informasi yang bisa dibaca dan ditulis oleh komputer. Dengan aplikasi perangkat lunak ini terdiri dari UML, Konsep dasar UML, Java, Adobe Photoshop dan Greenfoot.

2.3.1 Unified Modeling Language (UML)

UML(Unified Modeling Language) adalah bahasa spesifikasi standar untuk mendokumentasikan, menspesifikasikan, dan membangun system (Flowler, 2006). Unified Modeling Language (*UML*) adalah himpunan struktur dan teknik untuk pemodelan desain program berorientasi objek (*OOP*) serta aplikasinya. *UML* adalah metodologi untuk mengembangkan sistem *OOP* dan sekelompok perangkat tool untuk mendukung pengembangan sistem tersebut. UML mulai diperkenalkan oleh Object Management Group, sebuah organisasi yang telah mengembangkan model, teknologi, dan standar *OOP* sejak tahun 1980-an. Sekarang UML sudah mulai banyak digunakan oleh para praktisi *OOP* [11]. Logo *UML* dapat dilihat pada gambar 2.9.



Gambar 2.9 Logo *UML*

UML suatu bahasa yang digunakan untuk menentukan, memvisualisasikan, membangun, dan mendokumentasikan suatu sistem informasi. *UML* dikembangkan sebagai suatu alat untuk analisis dan desain berorientasi objek oleh Grady Booch, Jim Rumbaugh, dan *Ivar Jacobson*. Namun demikian *UML* dapat digunakan untuk memahami dan mendokumentasikan setiap sistem informasi. Penggunaan *UML* dalam industri terus meningkat, ini merupakan standar terbuka yang menjadikannya sebagai bahasa pemodelan yang umum dalam industri peranti lunak dan pengembangan sistem. Bagian-bagian utama dari *UML* adalah view, diagram, model element, dan general mechanism [12]. Diagram berbentuk grafik yang menunjukkan simbol elemen model yang disusun untuk mengilustrasikan bagian atau aspek tertentu 23 dari sistem. Sebuah diagram merupakan bagian dari suatu view tertentu dan ketika digambarkan biasanya. Jenis – jenis UML diagram – diagram sebagai berikut:

1. *Usecase* Diagram

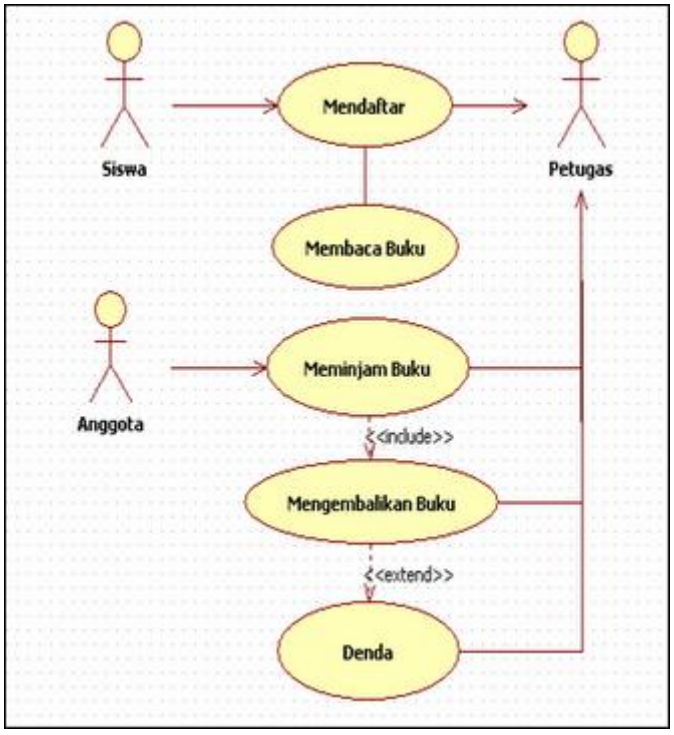
Usecase adalah abstraksi dari interaksi antara system dan actor. *Usecase* bekerja dengan cara mendeskripsikan tipe interaksi antara user sebuah system dengan sistemnya sendiri melalui sebuah cerita bagaimana sebuah system dipakai. *Usecase* merupakan konstruksi untuk mendeskripsikan bagaimana system akan terlihat di mata user. Sedangkan *Usecase* diagram

memfasilitasi komunikasi diantara analis dan pengguna serta antara analis dan client.

Contoh pada *use case diagram* perpustakaa mempunyai Lima fungsionalitas yang terdiri dari :

1. mendaftar artinya siswa harus melakukan pendaftaran sebelum membaca buku di perpustakaan baik menjadi anggota ataupun hanya pengunjung saja.
2. membaca buku artinya siswa yang telah mendaftarkan diri ke petugas perpustakaan boleh membaca buku dalam perpustakaan.
3. meminjam buku artinya jika siswa yang mendaftar ke petugas menjadi anggota perpustakaan maka diperbolehkan meminjam buku perpustakaan.
4. mengembalikan buku artinya anggota yang telah meminjam buku harus mengembalikan lagi ke petugas perpustakaan jika masa pinjaman telah habis.
5. denda artinya anggota yang mengembalikan buku melebihi masa peminjaman maka akan dikenakan denda.

Usecase pada Gambar 2.10 merupakan contoh dari *usecase* diagram sistem perpustakaan



Gambar 2.10 *Usecase* diagram sistem perpustakaan

2. Class Diagram

Class adalah dekripsi kelompok obyek-obyek dengan properti, perilaku (operasi) dan relasi yang sama. Sehingga dengan adanya class diagram dapat memberikan pandangan global atas sebuah system. Hal tersebut tercermin dari *class- class* yang ada dan relasinya satu dengan yang lainnya. Sebuah sistem biasanya mempunyai beberapa class diagram. *Class* diagram sangat membantu alam visualisasi struktur kelas dari suatu system.

Class memiliki tiga area pokok yaitu Nama Kelas, Atribut dan Metode. Atribut dan metode dapat memiliki salah satu sifat berikut :

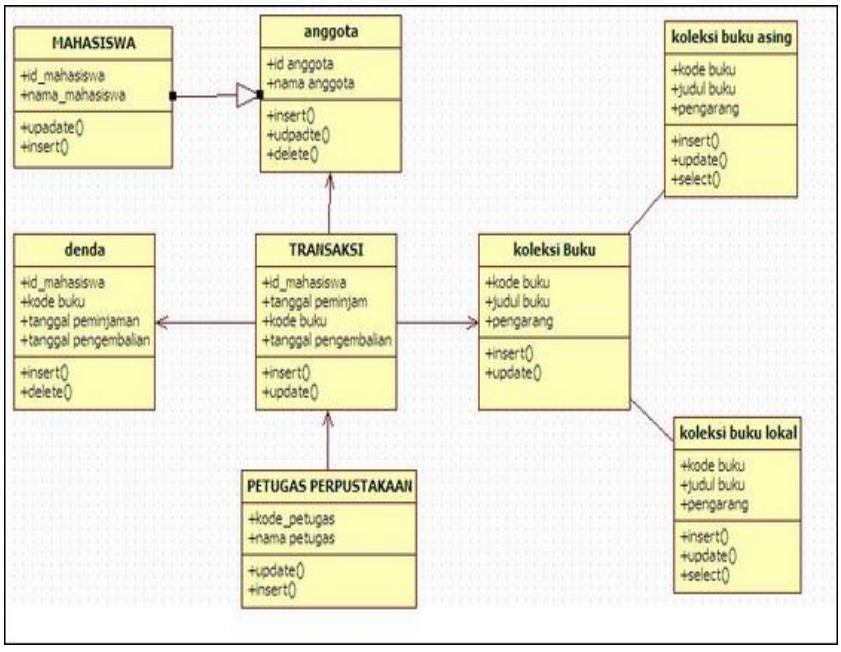
1. *Private* artinya data yang dikendalikan oleh suatu kelas tertentu dan tidak dapat diakses oleh objek-objek yang berbeda kelas. Satu ciri metode *sharing* yang bersifat *private* adalah hanya fungsi-fungsi atau metode – metode dalam kelas/objek yang bersangkutan yang dapat mengakses metode atau variabel serta atribut dalam kelas tanpa batasan. Biasanya tanda (-) digunakan untuk atribut atau metode yang bersifat *private*.
2. *Protected* artinya atribut atau metode yang bersifat *protected* hanya bisa diakses oleh kelas yang bersangkutan serta kelas-kelas yang merupakan turunannya mengikuti hierarki dari generalisasi. Biasanya tanda (#) digunakan untuk atribut atau metode yang bersifat *protected*.
3. *Public* artinya metode atau atribut yang bersifat *public* dapat diakses oleh semua objek dalam aplikasi tertentu tanpa batasan apa pun. Biasanya tanda (+) digunakan untuk atribut atau metode yang bersifat *public*.

Contoh *class* diagram pada sistem informasi perpustakaan memiliki delapan class antara lain:

1. Class mahasiswa memiliki atribut ; id_mahasiswa dan nama_mahasiswa dengan method : update dan insert dan bersifat public.
2. Class anggota memiliki atribut : id anggota dan nama anggota dengan method : insert, update, delete dan bersifat public.
3. Class koleksi buku memiliki atribut : kode buku, judul buku dan pengarang dengan method : insert, update,select. Dan bersifat public.

4. Class denda memiliki atribut : id mahasiswa, kode buku, tgl pinjam, tgl kembali dengan method : insert ,update dan bersifat public.
5. Class transaksi memiliki atribut : atribut : id mahasiswa, kode buku, tgl pinjam, tgl kembali dengan method : insert ,update dan bersifat public.
6. Class koleksi buku memiliki atribut : atribut : id mahasiswa, kode buku, tgl pinjam, tgl kembali dengan method : insert ,update dan bersifat public.
7. Class petugas perpustakaan memiliki atribut : atribut : id mahasiswa, kode buku, tgl pinjam, tgl kembali dengan method : insert ,update dan bersifat public.
8. Class koleksi buku loka memiliki atribut : atribut : id mahasiswa, kode buku, tgl pinjam, tgl kembali dengan method : insert ,update dan bersifat public.

Class diagram pada Gambar 2.11 merupakan contoh class diagram pada sistem informasi perpustakaan.



Gambar 2.11 *Class* Diagram pada sistem informasi perpustakaan.

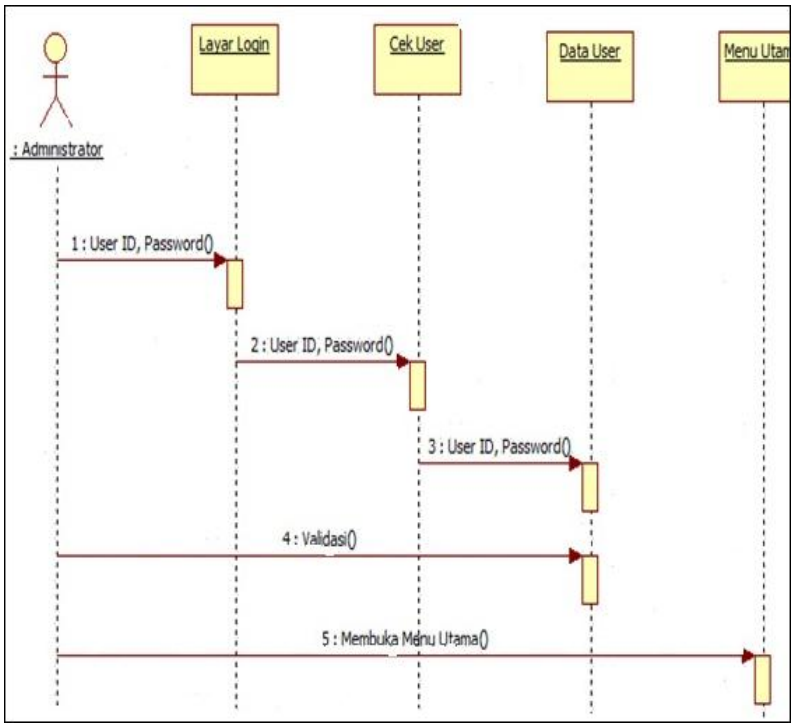
9. *Sequence* Diagram

Sequence Diagram digunakan untuk menggambarkan perilaku pada sebuah scenario. Kegunaannya untuk menunjukkan rangkaian pesan yang dikirim antara object juga interaksi antara object, sesuatu yang terjadi pada titik tertentu dalam eksekusi sistem.

Contoh pada *sequence diagram login* mempunyai interaksi sebagai berikut

a. Administrator (Aktor) melakukan *login*, maka *methodUser ID, Password* () memanggil kelas Layar *Login*.
b. Kelas Layar *Login* memanggil kelas Cek *User* untuk mengecek *User ID* dan *Password*.
c. Kelas Cek *User* memanggil kelas Data *User* untuk memvalidasi *User ID* dan *Password* di *database*.
d. Apabila *User ID* dan *Password* valid maka Administrator beerhasil melakukan *Login*.

Sequence Diagram pada Gambar 2.12 merupakan contoh dari *sequence* diagram login.



Gambar 2.12 Sequence Diagram login.

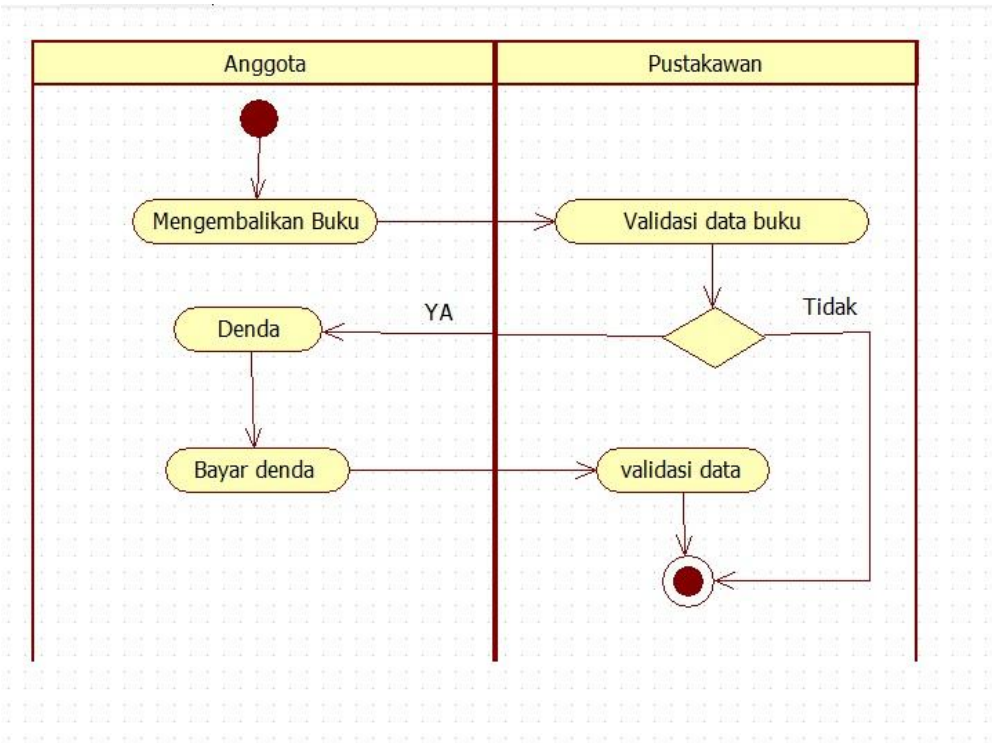
10. Activity Diagram

Activity Diagram menggambarkan rangkaian aliran dari aktivitas, digunakan untuk mendeskripsikan aktifitas yang dibentuk dalam suatu operasi sehingga dapat juga digunakan untuk aktifitas lainnya seperti *Usecase* atau interaksi,

Sebuah aktifitas dapat direalisasikan oleh suatu *use case* atau lebih. Aktifitas menggambarkan proses yang berjalan sementara *use case* menggambarkan bagaimana aktor menggunakan sistem untuk melakukan aktifitas. Contoh pada *activity diagram* pengembalian buku perpustakaan, pada *activity diagram* pengembalian buku perpustakaan mempunyai alir aktifitas sebagai berikut :

a. Anggota perpustakaan mengembalikan buku perpustakaan kepada pustakawan.
b. Pustakawan memvalidasi data buku yang dipinjam oleh anggota perpustakaan.
c. Apabila anggota perpustakaan mengembalikan buku melebihi batas waktu maka anggota perpustakaan akan mendapat denda dengan membayar sejumlah uang.
d. Kemudian pustakawan memvalidasi data anggota perpustakaan.
e. Apabila anggota perpustakaan mengembalikan buku tidak melebihi batas waktu pengembalian maka anggota tidak mendapat denda.

contoh dapat dilihat pada gambar 2.13 merupakan *activity* diagram penegembalian buku perpustakaan.



Gambar 2.13 *Activity* Diagram pengembalian buku perpustakaan.

2.3.1 Java

Sebagai sebuah bahasa pemrograman, Java dapat membuat seluruh bentuk aplikasi, desktop, web dan lainnya, sebagaimana dibuat dengan menggunakan bahasa pemrograman konvensional yang lain. Java adalah bahasa pemrograman yang berorientasi objek (OOP) dan dapat dijalankan pada berbagai *platform* sistem operasi. Perkembangan Java tidak hanya terfokus pada satu sistem operasi, tetapi dikembangkan untuk berbagai sistem operasi dan bersifat *open source*[12].

Sebagai sebuah peralatan pembangun, teknologi Java menyediakan banyak *tools* seperti, compiler, interpreter, penyusun dokumentasi, paket kelas dan sebagainya. Aplikasi dengan teknologi Java secara umum adalah aplikasi serba guna yang dapat dijalankan pada seluruh mesin yang memiliki *Java Runtime Environment* (JRE)[12].

Terdapat dua komponen utama dari *Deployment Environment*. Yang pertama adalah JRE, yang terdapat pada paket J2SDK, mengandung kelas–kelas untuk semua paket teknologi Java yang meliputi kelas dasar dari Java, komponen GUI dan sebagainya. Komponen yang lain terdapat pada Web Browser. Hampir seluruh Web Browser komersial menyediakan *interpreter* dan *runtime environment* dari teknologi Java[12].

2.3.2.1 Karakteristik Java

Berdasarkan *white paper* resmi dari SUN, Java memiliki karakteristik berikut[13]:

1. Sederhana

   Bahasa pemrograman Java menggunakan sintaks mirip dengan C++ namun sintaks pada Java telah banyak diperbaiki terutama menghilangkan penggunaan *pointer* yang rumit dan *multiple inheritance*. Java juga menggunakan *automatic memory allocation* dan *memory garbage collection.*

2. Berorientasi objek *(Object Oriented)*

   Java mengunakan pemrograman berorientasi objek yang membuat program dapat dibuat secara modular dan dapat dipergunakan kembali. Pemrograman berorientasi objek memodelkan dunia nyata kedalam objek dan melakukan interaksi antar objek-objek tersebut.

3. Dapat didistribusi dengan mudah

   Java dibuat untuk membuat aplikasi terdistribusi secara mudah dengan adanya *libraries networking* yang terintegrasi pada Java.

4. *Interpreter*

   Program Java dijalankan menggunakan interpreter yaitu *Java Virtual Machine* (JVM). Hal ini menyebabkan *source code* Java yang telah dikompilasi menjadi Java *bytecodes* dapat dijalankan pada *platform* yang berbeda-beda.

5. *Robust*

   Java mempuyai reliabilitas yang tinggi. *Compiler* pada Java mempunyai kemampuan mendeteksi *error* secara lebih teliti dibandingkan bahasa pemrograman lain. Java mempunyai *runtime-Exception handling* untuk membantu mengatasi *error* pada pemrograman.

6. Aman

   Sebagai bahasa pemrograman untuk aplikasi internet dan terdistribusi, Java memiliki beberapa mekanisme keamanan untuk menjaga aplikasi tidak digunakan untuk merusak sistem komputer yang menjalankan aplikasi tersebut.

7. *Architecture Neutral*

   Program Java merupakan *platform independent*. Program cukup mempunyai satu buah versi yang dapat dijalankan pada *platform* yang berbeda dengan *Java Virtual Machine.*

8. *Portabel*

   Source code maupun program Java dapat dengan mudah dibawa ke *platform* yang berbeda-beda tanpa harus dikompilasi ulang.

9. *Performance*

   Performance pada Java sering dikatakan kurang tinggi. Namun *performance* Java dapat ditingkatkan menggunakan kompilasi Java lain seperti buatan *Inprise, Microsoft* ataupun *Symantec* yang menggunakan *Just In Time Compilers* (JIT).

10. *Multithreaded*

    Java mempunyai kemampuan untuk membuat suatu program yang dapat melakukan beberapa pekerjaan secara sekaligus dan simultan.

11. Dinamis

    Java didesain untuk dapat dijalankan pada lingkungan yang dinamis. Perubahan pada suatu class dengan menambahkan *properties* ataupun *method* dapat dilakukan tanpa menggangu program yang menggunakan *class* tersebut.

2.3.2 Adobe Photoshop

Adobe Photoshop adalah *software* pengolah foto yang sangat populer dan telah banyak para pengguna program ini yang mengakui kehebatan dari Adobe Photoshop. Dengan Photoshop kita dapat memamnipulasi sebuah foto agar terlihat lebih menarik dan mengesankan, dengan fasilitas-fasilitas yang ada pada Photoshop, kita juga dapat menggabung foto serta memberikan efek-efek khusus untuk mempercantik tampilan foto.

Tampilan lembar kerja Photoshop terdiri dari *Menu Bar, Panel Control* dan *Panel Tools*. *Menu Bar* adalah baris paling atas yang berisi perintah yang disediakan oleh sebuah program yang mendukung kinerja dari program tersebut. *Panel Control* adalah bagian yang berisi sekumpulan tombol atau pilihan yang letaknya berada tepat di bawah *Menu Bar* yang digunakan untuk mengatur sifat dan kinerja dari piranti yang terpilih di *Panel Tools*. *Panel Tools* adalah sebuah kumpulan piranti yang digunakan untuk mengolah gambar, seperti melakukan seleksi area, memotong bidang gambar, maupun menghapus bidang gambar [13].

2.3.3 Greenfoot

Tutorial ini adalah pengenalan tentang objek(object) dan dunia(world) di Greenfoot.Greenfoot adalah perangkat lunak yang didesain untuk pemula agar dapat terbiasa dengan Pemrograman Berorientasi Objek(Object-Oriented Programming), yang mendukung pengembangan aplikasi bergambar dengan memakai bahasa pemrograman Java™. Greenfoot didesain dan telah diimplentasikan di Universitas Kent, Inggris dan Universitas Deakin, Melbourne - AustraliaInformasi lebih lanjut mengenai Greenfoot bisa dilihat di www.greenfoot.org.

Tutorial ini ditujukan kepada mereka yang ingin mengenal lebih jauh tentang Greenfoot, singkat dan fokus kepada fungsionalitas dan cara kerja Greenfoot itu sendiri. Tutorial ini tidak diperuntukkan bagi mereka yang ingin mempelajari Java, kami menganggap pembaca sudah mengetahui tentang Java sebelumnya atau didampingi oleh pembimbing saat menjalankan tutorial

ini.Tutorial ini menggunakan skenario yang sudah disediakan sebelumnya dan selanjutnya mendiskusikan bagaimana cara memakai atau memodifikasi skenario tersebut.Pembaca yang ingin membuat skenario mereka sendiri, atau ingin membuat perubahan secara signifikan dari skenario yang sudah ada dapat membaca [14].