BAB 2

LANDASAN TEORI

2.1 Objek Penelitian

Penelitian dilakukan agar memperkuat alasan dibangunnya aplikasi pengenalan hukum tajwid pada masyarakat. Berdasarkan hasil wawancara dengan ustadz V.Faizal Adha selaku Mudir Ma'had Al Imarat bahwa terdapat beberapa kendala dalam mempelajari ilmu tajwid dimana penghafalan sejumlah kaidah – kaidah dalam membaca al-Qur'an, serta mempraktekkan atau mengaplikasikannya ketika membaca al-Qur'an, serta kendala dalam mempelajari ilmu tajwid berdasarkan hasil wawancara dengan ustadz V.Faizal Adha selaku Mudir Ma'had Al Imarat untuk siswa MTS dan SLTP biasanya sudah mulai acuh terhadap pelajaran ini karena lebih fokus terhadap pelajaran yang akan diujikan untuk kenaikan kelas atau kelulusan. Siswa MTS dan SLTP biasanya lebih mudah menghapalnya karena mudah memahami, tapi kurang bisa melafalkan karena penguasaan makhrajul khurufnya kurang. Dari hasil penelitian di Ma'had Al Imarat, penulis mendapatkan gambaran umum, bahwa kemampuan mempelajari ilmu tajwid belum sesuai dengan kaidah pembacaan ayat suci al-Qur'an terutama dalam mempraktikan (membaca) yang benar sesuai dengan tuntutan ilmu tajwid. Hal ini dikarenakan oleh beberapa faktor yang mempengaruhinya.

1.2 Pembelajaran

Pembelajaran adalah adalah proses interaksi peserta didik dengan pendidik dan sumber belajar pada suatu lingkungan belajar. Dengan kata lain, pembelajaran adalah proses untuk membantu peserta didik agar dapat belajar dengan baik. Dalam konteks pendidikan, guru mengajar agar peserta didik dapat belajar dan menguasai isi pelajaran hingga mencapai sesuatu objektif yang ditentukan (aspek kognitif), juga dapat memengaruhi perubahan sikap (aspek afektif), serta keterampilan (aspek psikomotor) seorang peserta didik,

namun proses pengajaran ini memberi kesan hanya sebagai pekerjaan satu pihak, yaitu pekerjaan pengajar saja.

Menurut M. Ngalim Purwanto, dalam bukunya Psikologi Pendidikan yang mempengaruhi belajar, yaitu :

2. Faktor yang ada pada organisme itu sendiri kita sebut faktor individu, antara lain faktor kematangan/pertumbuhan, faktor kecerdasan, faktor latihan, faktor motivasi, dan faktor pribadi.
3. Faktor yang ada di luar individu yang kita sebut faktor sosial, antara lain faktor keluarga/keadaan rumah tangga, guru dan cara mengajarnya, lingkungan dan kesempatan yang tersedia dan motivasi social [2].

3.2 Pengertian Media

Kata media berasal dari bahasa latin dan merupakan bentuk jamak dari kata medium yang secara harfiah berarti perantara atau pengantar. Media adalah segala sesatu yang dapat digunakan untuk menyalurkan pesan dari pengirim ke penerima [4]. Berikut beberapa unsur dari media yaitu:

1. Orang
2. Bahan atau material
3. Alat
4. Teknik
5. Lingkungan

3.2.1 Media Pembelajaran

Media pembelajaran adalah bahan, alat atau teknik yang digunakan dalam kegiatan mengajar dengan maksud agar proses interaksi komunikasi edukasi dapat berlangsung secara tepat guna dan berdaya guna [5]. Media pembelajaran berfungsi sebagai alat penunjang dalam membantu proses pembelajaran sehingga dapat menstimulasi pikiran, perhatian dan minat dalam mempelajari ilmu pengetahuan serta dapat membantu tercapainya tujuan

pembelajaran. Secara umum manfaat dari media pembelajaran adalah sebagai berikut :

1. Memperjelas penyajian agar tidak terlalu bersifat verbalistis.
2. Mengatasi keterbatasan ruang,waktu dan daya indra.
3. Mengatasi sikap pasif pembelajar.
4. Mengatasi kesulitan dalam pembelajaran.

3.3 Pengertian Edukasi

Secara nasional, pendidikan merupakan sarana yang dapat mempersatukan setiap warga negara menjadi suatu bangsa. melalui pendidikan, setiap peserta didik difasilitasi, dibimbing dan dibina untuk menjadi warganegara yang menyadari dan merealisasikan hak dan kewajibannya. Pendidikan juga merupakan alat yang ampuh untuk menjadikan setiap peserta didik dapat duduk sama rendah dan berdiri sama tinggi.

Berikut ini adalah pengertian dan definisi pendidikan:

1. Undang - Undang No. 20 tahun 2003 tentang Sistem Pendidikan Nasional, Pasal 1:

   Pendidikan adalah usaha sadar dan terencana untuk mewujudkan suasana belajar dan proses pembelajaran agar peserta didik secara aktif mengembangkan potensi dirinya untuk memiliki kekuatan spiritual keagamaan, pengendalian diri, kepribadian, kecerdasan, akhlak mulia serta ketrampilan yang diperlukan dirinya, masyarakat, bangsa, dan negara.
2. Prof. Dr. Prayitno, M.Sc., Ed.

   Pendidikan adalah upaya memuliakan kemanusiaan manusia, tujuan pendidikan sepenuhnya mengacu kepada seluruh komponen harkat dan martabat manusia (HMM).
3. John Dewey

   Pendidikan merupakan alat pelanjuta kehidupan sosial (social continuity of life).

4. Doni Koesoema A

   Pendidikan adalah sebuah konsep yang sangat abstrak, yang tidak dapat emmiliki tujuan - tujuan pendidikan dalam dirinya sendiri.

5. Sir Francis Beacon

   Pendidikan bukanlah tujuan akhir untuk mencetak generasi ilmuan yang mengenal ilmu dengan cara meniru sistem tradisional yang telah ada.

6. Profesor Gurrey

   Pendidikan adalah untuk menggali dan mengambil segala seni dan pengetahuan yang amat luas, pendidikan ialah proses mengeluarkan manusia yang mempunyai kebudayaan dan pengetahuan yang luas.

7. W.O.I Smith

   Pendidikan adalah suatu proses menyampaikan segala anasir- anasir kehidupan suatu masyarakat kepada generasi mudanya bagi melanjutkan warisan anasir-anasir itu agar dapat dipertinggikan dan disampaikan pula kepada generasi lain.

8. Prof. Dr. Imam Barnadib

   pendidikan adalah usaha sadar dan sistematis untuk mencapai taraf hidup atau kemajuan yang lebih baik [6].

3.4 Aplikasi Sejenis

Pengenalan mengenai edukasi hukum tajwid memang bukanlah yang pertama kalinya. Edukasi pengenalan hukum tajwid sebelumnya telah banyak yang membahas, akan tetapi kebanyakan dari pembelajaran dilakukan disekolah – sekolah umum, di TQA dan TPA dan Madrasah Diniyah. Perlu dilakukan tinjaun ulang untuk aplikkasi sejenis sebagai salah satu dari penerapan metode penelitian yang akan dilakukan. Diantaranya adalah mengidentifikasi kesenjangan (*identify gaps*) menghindari pembuatan ulang (*reinventing the wheel*), mengidentifikasi metode yang pernah dilakukan, meneruskan penelitian sebelumnya, serta mengetahui orang lain yang spesialisasi dan area

penelitiannya sama di bidang ini. Oleh karena itu kajian mengenai pengenalan hukum tajwid hanya bersifat menambah pada penelitian yang sudah ada.

Adapun hasil penelitian yang ada, memberikan kontribusi besar dalam referensi mengenai edukasi pengenalan hukum tajwid diantaranya :

Pertama, Skirpsi yang berjudul " *Implementasi E-Learning Program Aplikasi Ilmu Tajwid pada Sistem Oprasi Android Versi 2.2"*yang dibangun oleh Ahmad Rifa'I dari Sekolah Tinggi Manajemen Informatika dan Komputer,Yogyakarta. Membahas tentang penerapan ilmu tajwid untuk kalangan umum, aplikasi edukasi ini menerapkan bacaan tentang huruf hijaiyah dengan hukum tajwid : nun sukun dan tanwin, mim sukun, nun tasydid dan mim tasydid, idghaam, lam ta'rif, tarqiiq dan tafkhiim, qalqalah, mad, serta latihan soal. Aplikasi ini membahas tentang penerapan aplikasi ilmu tajwid pada pelajar maupun pengajar, terutama masyarakat muslim untuk mempermudah mempelajari ilmu tajwid dalam aplikasi mobile [7].

Kedua, Skripsi yang berjudul *"Aplikasi Belajar Membaca dan Mengucapkan Huruf Hijaiyah dengan Tajwid Berbasis Android"* yang dibangun oleh Mistari, Rangga Asmara, Rizky Yuniar Hakkun dari Politeknik Elektronika Negri Surabaya, Surabaya. Membahas tentang penerapan ilmu tajwid untuk kalangan umum, aplikasi edukasi ini menerapkan bacaan tentang huruf hijaiyah dengan hukum tajwid : nun sukun dan tanwin, mim sukun, nun tasydid dan mim tasydid, idghaam, lam ta'rif, tarqiiq dan tafkhiim, qalqalah, mad, serta latihan soal [8].

Ketiga, Skripsi yang berjudul *"Pembangunan Game Edukasi Tajwid gembira untuk anak muslim"* yang dibangun oleh Wedya Kristi Setiawan dari Universitas Komputer Indonesia, Bandung. Membahas tentang pengenalan hukum tajwid kepada anak TQA dan TPA, aplikasi game ini menampilkan permainan diantaranya teknik huruf hijaiyah, teknik tajwid gembira, *Puzzle* tajwid cepat dan tepat, quiz nada tajwid. Metode yang digunakan adalah AI

dimana diterapkan pada permainan *Puzzle* untuk mencocokkan kepingan dalam pembelajaran ilmu tajwid [9].

Dari tiga aplikasi yang ada, telah banyak aplikasi edukasi pengenalan hukum tajwid berbasis android, dimana dari beberapa jurnal skripsi yang terkait memiliki pengertian dan alur program yang berbeda. Beberapa Jurnal skripsi yang terkait menerapkan aplikasi dalam bentuk game, serta fitur yang disediakan hampir sama dengan jurnal-jurnal skripsi yang lain. Dengan demikian dapat disimpulkan bahwa masih sedikit aplikasi *belajar tajwid* yang ada menampilkan pelatihan yang lebih interaktif, aplikasi yang di bangun kali ini menampilkan pelatihan tajwid, dimana user memilih letak hukum tajwid dengan menekan potongan ayat suci al-Qur'an, jika jawaban benar maka akan keluar jawaban atau penjelasan hukum tajwid tersebut, serta menampilkan *score*(hasil) jawaban yang dilakukan oleh user ketika selesai menjawab pelatihan, sehingga user mengetahui sudah sejauh mana pengetahui hukum tajwid yang dipelajarinya dan user sebagai pengguna akan lebih interaktif dalam menggunakan aplikasi edukasi tersebut, Serta adanya *Web* aplikasi sebagai *backend* penyimpan data dari semua data yang ada di aplikasi Belajar Tajwid. Fungsionalitas dari *Web backend* aplikasi Belajar Tajwid adalah bisa menambah, mengedit, dan menghapus data, serta menanggulangi terjadinya kesalahan dalam menginputkan data ke aplikasi *Frontend.*

3.5 Perangkat Lunak Aplikasi

3.5.1 Pengertian Perangkat Lunak Aplikasi

Perangkat lunak aplikasi adalah suatu subkelas perangkat lunak komputer yang memanfaatkan kemampuan komputer langsung untuk melakukan suatu tugas yang diinginkan pengguna. Biasanya dibandingkan dengan perangkat lunak sistem yang mengintegrasikan berbagai kemampuan komputer, tapi tidak secara langsung menerapkan kemampuan tersebut untuk mengerjakan suatu tugas yang menguntungkan pengguna. Contoh utama perangkat lunak aplikasi adalah pengolah kata, lembar kerja, dan pemutar

media. Beberapa aplikasi yang digabung bersama menjadi suatu paket kadang disebut sebagai suatu paket atau suite aplikasi (*application suite*). Contohnya adalah *Microsoft Office* dan *OpenOffice.org*, yang menggabungkan suatu aplikasi pengolah kata, lembar kerja, serta beberapa aplikasi lainnya. Aplikasi-aplikasi dalam suatu paket biasanya memiliki antarmuka pengguna yang memiliki kesamaan sehingga memudahkan pengguna untuk mempelajari dan menggunakan setiap aplikasi. Sering kali, aplikasi ini memiliki kemampuan untuk saling berinteraksi satu sama lain sehingga menguntungkan pengguna. Contohnya, suatu lembar kerja dapat dibenamkan dalam suatu dokumen pengolah kata walaupun dibuat pada aplikasi lembar kerja yang terpisah.

3.5.2 Aplikasi Mobile

Aplikasi *mobile* adalah sebuah aplikasi yang memungkinkan Anda melakukan mobilitas dengan menggunakan perlengkapan seperti PDA, telepon seluler atau *handphone*. Dengan menggunakan aplikasi *mobile*, Anda dapat dengan mudah melakukan berbagai macam aktifitas mulai dari hiburan, berjualan, belajar, mengerjakan pekerjaan kantor, *browsing* dan lain sebagainya. Pemanfaatan aplikasi *mobile* untuk hiburan paling banyak digemari oleh hamper 70% pengguna telepon seluler, karena dengan memanfaatkan adanya fitur *game*, *music player*, sampai *video player* membuat kita menjadi semakin mudah menikmati hiburan kapan saja dan dimanapun.

3.6 Pengertian UML

Menurut (Adi Nugroho : 2005). "*Unified Modeling Language* (UML) adalah alat bantu analisis serta perancangan perangkat lunak berbasis objek". Menurut (Joomla dari http://soetrasoft.com : 2007). *"Unified Modeling Language* (UML) merupakan standard modeling language yang terdiri dari kumpulan-kumpulan diagram, dikembangkan untuk membantu para pengembang sistem dan *software* agar bisa menyelesaikan tugas-tugas seperti: Spesifikasi, Visualisasi, Desain Arsitektur, Konstruksi, Simulasi dan testing serta Dokumentasi".

Berdasarkan beberapa pendapat yang dikemukakan diatas dapat ditarik kesimpulan bahwa "*Unified Modeling Language* (UML) adalah sebuah bahasa yang berdasarkan grafik atau gambar untuk menvisualisasikan, menspesifikasikan, membangun dan pendokumentasian dari sebuah sistem pengembangan perangkat lunak berbasis OO (*Object Oriented*)". *UML* merupakan suatu kumpulan teknik terbaik yang telah terbukti sukses dalam memodelkan system yang besar dan kompleks. *UML* tidak hanya digunakan dalam proses pemodelan perangkat lunak, namun hampir dalam semua bidang yang membutuhkan pemodelan. Bagian-bagian utama dari *UML* adalah *View*, *Diagram*, *model element*, dan *general mechanism*.

3.6.1 *View*

View digunakan untuk melihat sistem yang dimodelkan dari beberapa aspek yang berbeda. View bukan melihat grafik, tapi merupakan suatu abstraksi yang berisi sejumlah Diagram. Beberapa jenis View dalam UML antara lain: Use case View, Logical View, Component View, concurrency View, dan Deployment View.

1. Use Case View

Use case View mendeskripsikan fungsionalitas sistem yang seharusnya dilakukan sesuai yang diinginkan *external actors*. *Actor* yang berinteraksi dengan sistem dapat berupa *user* atau sistem lainnya.*View* ini digambarkan dalam *Use case Diagram*s dan kadang-kadang dengan *Activity Diagram*s. *View* ini digunakan terutama untuk pelanggan, perancang (*designer),* pengembang (*developer*), dan penguji sistem (*tester*).

2. Logical View

Logical View mendeskripsikan bagaimana fungsionalitas dari sistem, struktur statis (*Class, object*,dan relationship) dan kolaborasi dinamis yang terjadi ketika *object* mengirim pesan ke *object* lain dalam suatu fungsi tertentu. *View* ini digambarkan dalam *Class Diagram*s untuk struktur statis dan dalam *State*, *Sequence*, *Collaboration*, dan *Activity*

Diagram untuk model dinamisnya. *View* ini digunakan untuk perancang (*designer*) dan pengembang (*developer*).

3. Component View

Component View mendeskripsikan implementasi dan ketergantungan modul. Komponen yang merupakan tipe lainnya dari *code module* diperlihatkan dengan struktur dan ketergantungannya juga alokasi sumber daya komponen dan informasi administrative lainnya. *View* ini digambarkan dalam *Component View* dan digunakan untuk pengembang (developer).

4. Concurency View

Concurrency *View* membagi sistem ke dalam proses dan prosesor. *View* ini digambarkan dalam *Diagram* dinamis (*State, Sequence*, *Collaboration*, dan *Activity Diagram*s) dan *Diagram* implementasi (*Component* dan *Deployment Diagrams*) serta digunakan untuk pengembang (developer), pengintegrasi (integrator), dan penguji (tester).

5. Deployment View

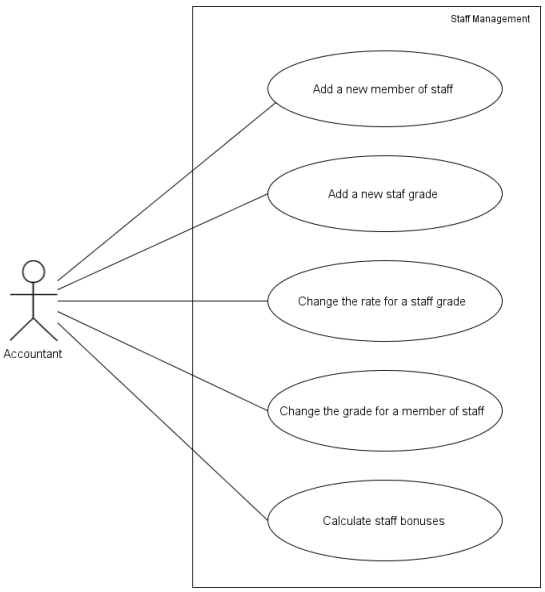
Deployment View mendeskripsikan fisik dari sistem seperti komputer dan perangkat (nodes) dan bagaimana hubungannya dengan lainnya.*View* ini digambarkan dalam *Deployment Diagram*s dan digunakan untuk pengembang (developer), pengintegrasi *(integrator),* dan penguji *(tester).*

6. Diagram

Diagram berbentuk grafik yang menunjukkan simbol elemen model yang disusun untuk mengilustrasikan bagian atau aspek tertentu dari sistem. Sebuah *Diagram* merupakan bagian dari suatu *View* tertentu dan ketika digambarkan biasanya dialokasikan untuk *View* tertentu. Adapun jenis *Diagram* antara lain :

a) Use case Diagram

Use case adalah abstraksi dari interaksi antara system dan actor. *Use case* bekerja dengan cara mendeskripsikan tipe interaksi antara user sebuah system dengan sistemnya sendiri melalui sebuah cerita bagaimana sebuah system dipakai. *Use case* merupakan konstruksi untuk mendeskripsikan bagaimana system akan terlihat di mata user. Sedangkan *Use case Diagram* memfasilitasi komunikasi diantara analis dan pengguna serta antara analis dan client.

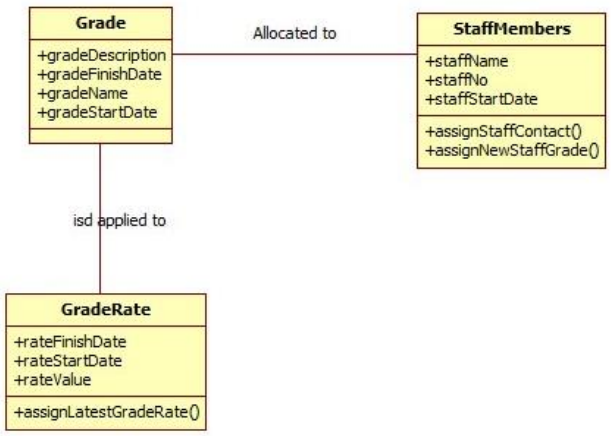


Gambar 2.1 Contoh Use Case Diagram [10]

b) Class Diagram

Class Diagram menjelaskan dalam visualisasi struktur kelas-kelas dari suatu sistem dan merupakan tipe diagram yang paling banyak dipakai. *Class Diagram* memperlihatakan hubungan antar kelas dan penjelasan detail tiap-tiap kelas dalam model desain dari suatu sistem. Selama proses analisis, *class diagram* memperlihatkan aturan-aturan dan tanggung jawab entitas yang menentukan prilaku sistem. Selama

tahap desain, *class diagram* berperan dalam menagkap struktur dari semua kelas yang membentuk arsitektur sistem yang dibuat.



Gambar 2.2 Contoh Class Diagram [10]

c) Behavior Diagram

Behavior diagram dapat dikelompokan menjadi tiga diagram, yaitu :

a. Statechart Diagram

Statechart Diagram berfungsi untuk memodelkan prilaku dinamis satu kelas satu objek.

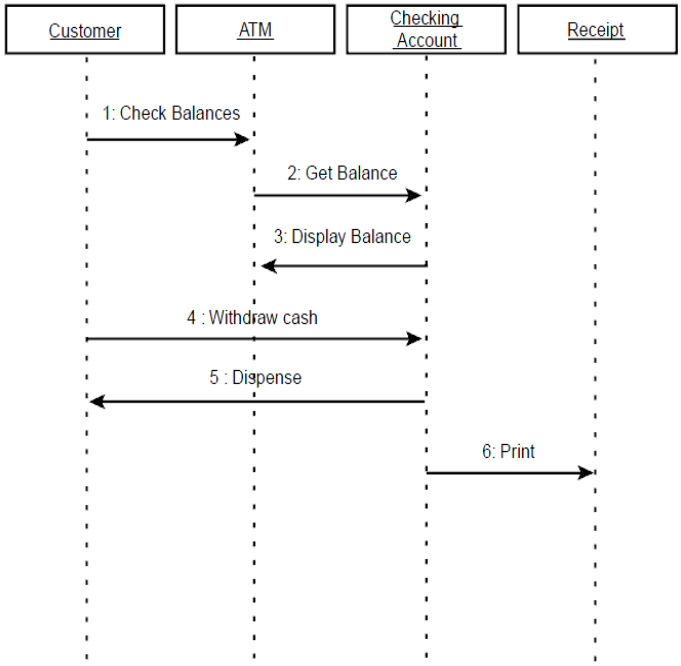
b. Activity Diagram

Activity Diagram memodelkan alur kerja (work flow) sebuah proses bisnis dan urutan aktifitas dalam suatu proses.

c. *Interaction Diagram*

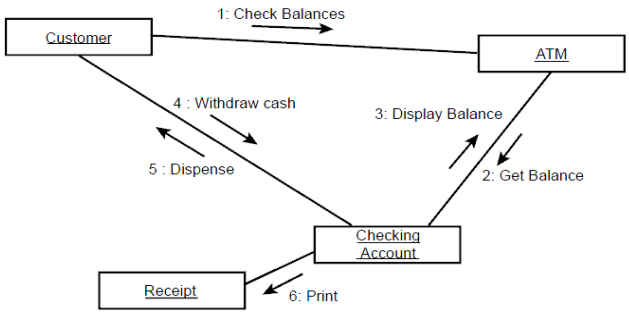
Interaction Diagram dibagi menjadi dua model diagram yaitu :

1. *Sequence Diagram* menjelaskan interaksi objek yang disusun dalam suatu urutan waktu. Diagram ini secara khusus bersosialisasi dengan *use case*. Sequence diagram, memperlihatkan tahap demi tahap apa yang seharusnya terjadi untuk menghasilkan sesuatu dalam *use case*.



Gambar 2.3 Sequence Diagram [11]

7. *Collaboration Diagram* melihat pada interaksi dan hubungan terstruktur antar objek. Tipe diagram ini menekankan pada hubungan (*relationship*) antar objek, sedangkan *sequence diagram* menekankan pada urutan kejadian. Dalam *collaboration diagram* terdapat beberapa objek, *link*, dan *message*.



Gambar 2.4 Collaboration Diagram [11]

d) ***Implementatiton Diagram***

Implementation diagram dibagi menjadi dua diagram, yaitu :

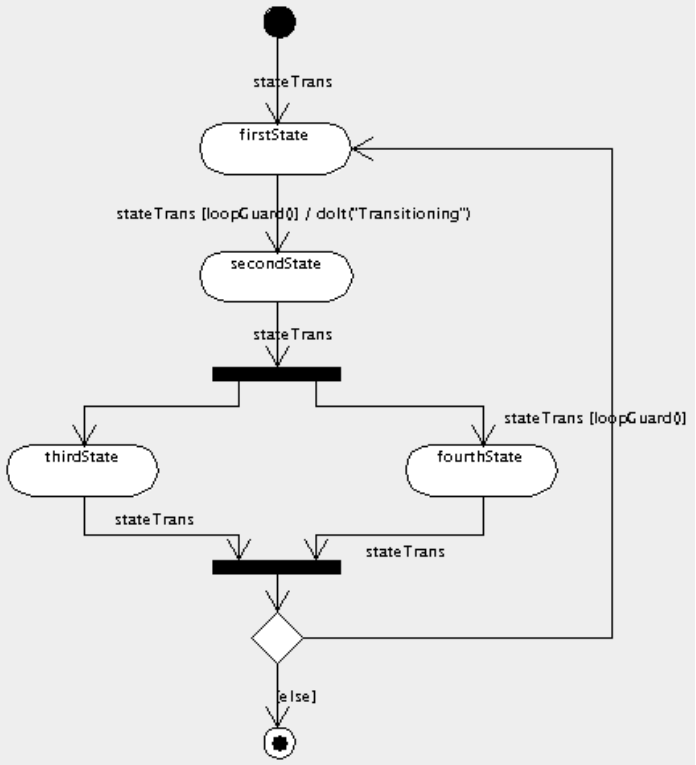
1. *Component Diagram* menggambarkan alokasi semua kelas dan objek kedalam komponen-komponen dalam desain fisik sistem *software*. Diagram ini memperlihatkan pengaturan dan kebergantungan antara komponen-komponen *software*, seperti *source code*, *binary code*, dan komponen tereksekusi (*execute components*)
2. *Deployment Diagram* memperlihatkan pemetaan *software* kepada *hardware*. Dimana akan berjalan (di *server/multitier*, *standalone* atau lainnya), dan menggambarkan model koneksi dan kemampuan jaringan dan hal lainya yang bersifat fisik.

e) ***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 *Use case* atau interaksi.

Adapun tujuan penggunaan UML, yaitu :

a. Memberikan bahasa pemodelan yang bebas dari berbagai bahas pemrograman dan proses rekayasa.
b. Menyatukan praktek-praktek terbaik yang terdapat dalam pemodelan.
c. Memberikan model yang siap pakai, bahasa pemodelan visual yang ekspresif untuk mengembangkan dan saling menukar model dengan mudah dan dimengerti secara umum.
d. *UML* bisa juga berfungsi sebagai sebuah (blue print) cetak biru karena sangat lengkap dan detail. Dengan cetak biru ini maka akan bias diketahui informasi secara detail tentang coding program atau bahkan membaca program dan menginterpretasikan kembali ke dalam bentuk *Diagram (reserve enginering).*



Gambar 2.5 Contoh Activity Diagram [10]

3.7 Pemodelan Berorintasi Objek

Analisis dan desain berorientasi objek adalah cara baru dalam memikirkan suatu masalah dengan menggunkan model yang dibuat menurut konsep sekitar dunia nyata. Dasar pembuatan adalah objek, yang merupakan kombinasi antara struktur data dan perilaku dalam suatu entitas. Model berorientasi objek bermanfaat untuk memahami masalah, komunikasi dengan ahli aplikasi, pemodelan suatu organisasi, meyiapkan dokumentasi serta perancangan program dan basis data. Pertama-tama suatu model analisis dibuat untuk menggambarkan aspek dasar dari domain aplikasi, dimana model tersebut berisi objek yang terdapat dalam domain aplikasi termasuk deskripsi dari keterangan objek dan perilakunya.

Model data berorientasi objek dikatakan dapat memberi fleksibilitas yang lebih, kemudahan mengubah program, dan digunakan luas dalam teknik

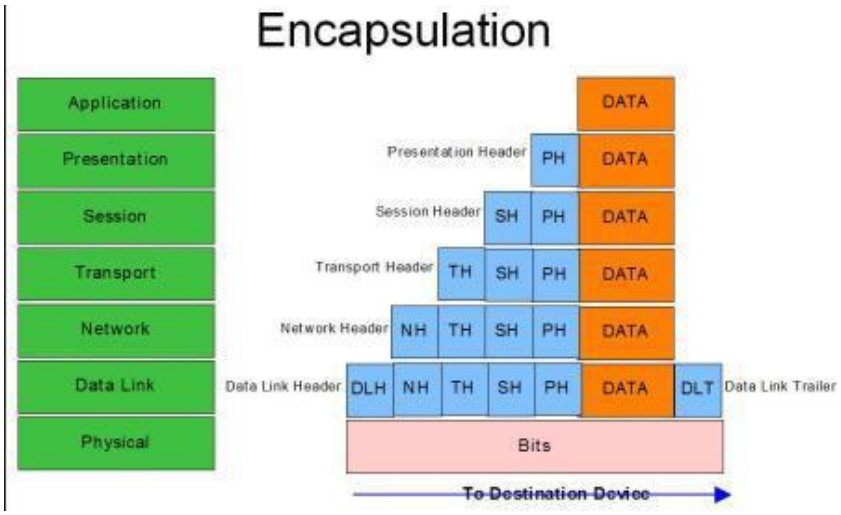
piranti lunak skala besar. Lebih jauh lagi, pendukung OOP (*Object Oriented Programing*) mengklaim bahwa OOP lebih mudah dipelajari bagi pemula dibanding dengan pendekatan sebelumnya, dan pendekatan OOP lebih mudah dikembangkan dan dirawat. Dengan menggunakan OOP maka dalam melakukan pemecahan suatu masalah kita tidak melihat bagaimana cara menyelesaikan suatu masalah tersebut (terstruktur) tetapi objek-objek apa yang dapat melakukan pemecahan masalah tersebut.

3.7.1 Karakteristik Metodologi Berorientasi Objek

Metodologi pengembangan sistem berorientasi objek mempunyai tiga karateristik utama , yaitu:

1. *Encapsulation*

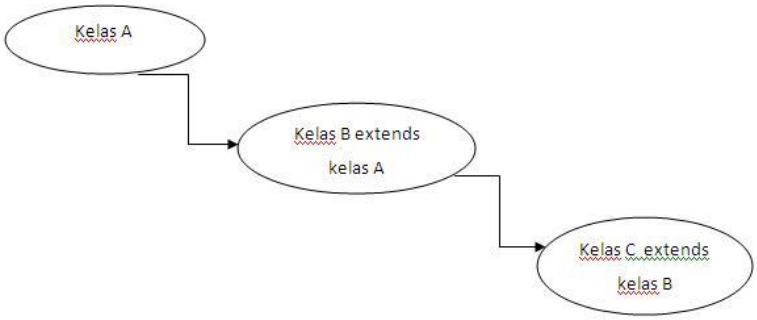
   Encapsulation (pengkapsulan) merupakan dasar untuk pembatasan ruang lingkup program terhadap data yang diproses. Data dan prosedur atau fungsi dikemas dalam bersama-sama dalam suatu objek, sehingga prosedur atau fungsi lain dari luar tidak dapat mengaksesnya. Data terlindung dari prosedur atau objek lain kecuali prosedur yang berada dalam objek itu sendiri.



Gambar 2.6 Contoh *Encapsulasi*

2. *Inheritance*

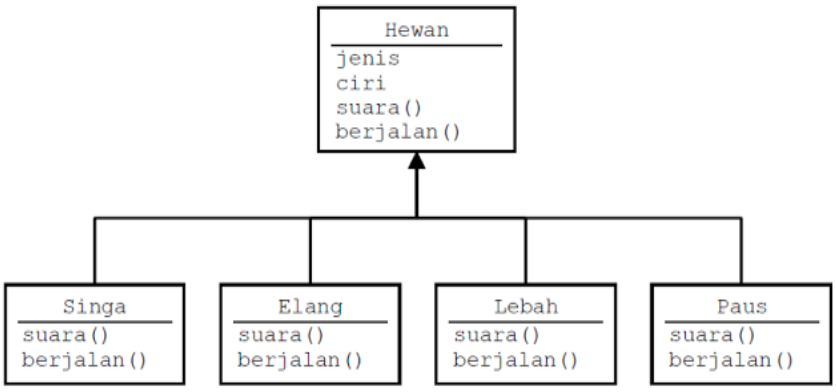
   Inheritance (pewarisan) adalah teknik yang menyatakan bahwa anak dari objek akan mewarisi atribut dan metoda dari induknya langsung. Atribut dan metoda dari objek induk diturunkan kepada anak objek demikian seterusnya. Pendefinisian objek dipergunakan untuk membangun suatu hirarki dari objek turunannya, sehingga tidak perlu membuat atribut dan metode lagi pada anaknya, karena telah mewarisi sifat induknya.



Gambar 2.7 Contoh *Inheritance*

3. *Polymorphism*

   Polymorphism (polimorfisme) yaitu konsep yang menyatakan bahwa sesuatu yang sama dapat mempunyai bentuk dan perilaku berbeda. Polimorfisme mempunyai arti bahwa operasi yang sama mungkin mempunyai perbedaan dalam kelas yang berbeda.



Gambar 2.8 Contoh *Polymorhism*

3.8 Pengertian Json

Json (JavaScript Object Nation) merupakan format yang ringan untuk memasukan data ke dalam sebuah variabel. Sangat mudah dimengerti dan diimplementasikan oleh manusia, dan mudah juga untuk komputer dalam melakukan parsingnya.

JSON merupakan bagian dari bahasa pemrograman JavaScript (Standard ECMA-262 3rd Edition – December 1999). JSON merupakan format teks yang sepenuhnya independen tetapi menggunakan konvensi yang familiar dengan bahasa pemrograman dari keluarga-C, termasuk C, C++, C#, Java, JavaScript, Perl, Python, dan sebagainya. Kelebihan inilah yang membuat JSON menjadi sebuah bahasa data-interchange yang ideal.

Jason dibangun dalam dua struktur :

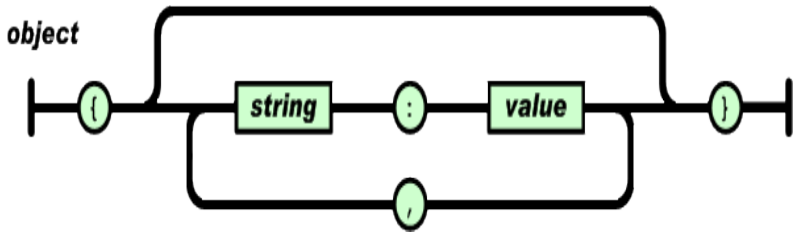
4. Beberapa pasangan dari nama/nilai. Dalam beberapa bahasa perograman biasa disebut dengan istilah object, record, struct, tabel hash, key list atau associative array.
5. Nilai-nilai yang terusun secara ordered list. Biasa disebut dengan array, vector, list atau daftar dalam bahasa pemrograman.

Json adalah struktur data yang universal, dalam artian bisa digunakan dalam berbagai bahasa pemrograman. Hampir semua bahasa pemrograman mendukung penuh JSON dalam berbagai format. Hal ini memungkinkan

format data yang dapat dipertukarkan menggunakan bahasa pemrograman juga menggunakan dasar dari struktur JSON.

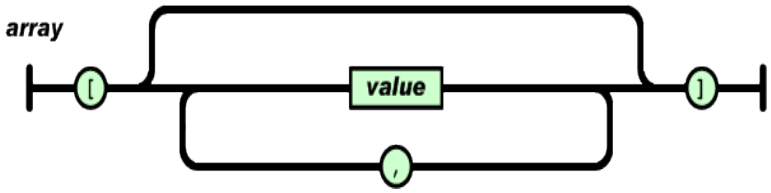
Format data JSON mempunyai aturan sebagai berikut:

Object adalah satu set nama/nilai yang tidak terurut (An object is an unordered set of name/value pairs). Penulisan object dimulai dengan tanda { (left brace) dan diakhiri dengan tanda } (right brace). Setiap nama diikuti oleh tanda : (colon) dan pasangan nama/nilai dipisahkan dengan tanda , (comma).



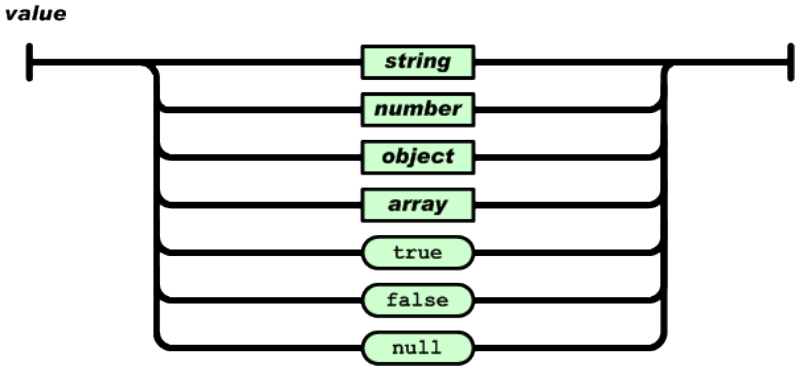
Gambar 2.9 Object dalam JSON

Array adalah sekumpulan nilai yang teratur (*An array is an ordered collection of values*). Penulisan sebuah array dimulai dengan tanda [(*left bracket*) dan diakhiri dengan tanda] (*right bracket*). Nilai dipisahkan menggunakan tanda , (*comma*).



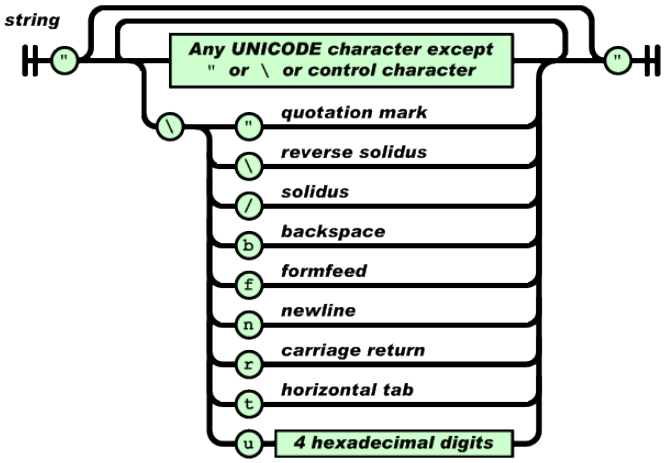
Gambar 2.10 Array dalam JSON

Nilai bisa berupa string dalam tanda kutip, atau number (angka), TRUE atau FALSE atau NULL, sebuah object atau sebuah array. Struktur ini dapat ditulis menggunakan metode bersarang.



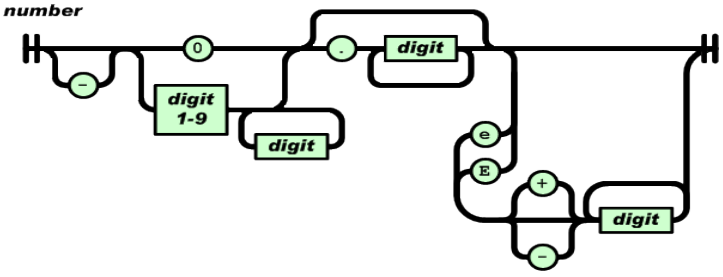
Gambar 2.11 Value atau nilai dalam format JSON

String adalah rangkaian atau urutan karakter unicode yang berada dalam tanda kutip, bisa juga hanya berisi karakter kosong, menggunakan tanda \ (*backslash*) untuk escape. Karakter di representasikan sebagai string tunggal. Tipe data string pada JSON sangat mirip dengan definisi pada bahasa C atau Java.



Gambar 2.12 Tipe data string pada format JSON

Number sangat mirip dengan definisi pada bahasa C atau Java, hanya saja tipe bilangan oktal dan heksadesimal tidak digunakan.



Ruang kosong (*Whitespace*) dapat diselipkan diantara pasangan atau token apa saja. Kecuali ***detail encoding***, yang secara jelas akan berfungsi untuk menjelaskan bahasa yang digunakan.

JSON sangat berguna saat kita sering melakukan pertukaran data pada sebuah aplikasi. Contoh nyata adalah penggunaan tabulasi menggunakan JqGrid, yang merupakan salah satu plugin jQuery untuk menampilkan data dalam bentuk spreadsheet. Plugin ini juga mendukung format JSON, selain format XML [12].

2.10 Android



Gambar 2.13 Logo *Android*

Android adalah sistem operasi untuk telepon seluler yang berbasis Linux. Android menyediakan *platform* terbuka bagi para pengembang buat menciptakan aplikasi mereka sendiri untuk digunakan oleh bermacam piranti bergerak. Android pertama kali dikembangkan oleh perusahaan bernama Android Inc yang kemudian pada tahun 2005 diakuisisi oleh raksasa Internet Google. Android dibuat dengan basis kernel Linux yang telah dimodifikasi, dan untuk setiap release-nya diberi kode nama berdasarkan nama hidangan makanan. Kemudian untuk mengembangkan Android, dibentuklah Open Handset Alliance, konsorsium dari 34 perusahaan peranti keras, peranti lunak, dan telekomunikasi, termasuk Google, HTC, Intel, Motorola, Qualcomm, T-*Mobile*, dan Nvidia. Keunggulan utama Android adalah gratis dan *open source*, yang membuat smartphone Android dijual lebih murah dibandingkan dengan Blackberry atau iPhone meski fitur (*hardware*) yang ditawarkan Android lebih

baik. Beberapa fitur utama dari Android antara lain WiFi hotspot, Multi-touch, Multitasking, GPS, support java, mendukung banyak jaringan (GSM/EDGE, IDEN, CDMA, EV-DO, UMTS, Bluetooth, Wi-Fi, LTE, and WiMAX) dan juga kemampuan dasar *handphone* pada umumnya. Beberapa pengertian lain dari *Android*, yaitu:

1. Merupakan platform terbuka *(open source)* bagi para programmer untuk membuat aplikasi.
2. Merupakan sistem operasi yang dibeli Google Inc dari Android Inc.
3. Bukan bahasa pemrograman, akan tetapi hanya menyediakan lingkungan hidup atau *run time environment* yang di sebut DVM *(Dalvik Virtual Machine)* yang telah dioptimasi untuk *device* dengan sistem memori yang kecil.

Android adalah sebuah sistem oprasi pada *handphone* yang bersifat terbuka dan berbasis pada sistem oprasi Linux. Antusiasme masyarakat pengguna *smartphone* saat ini semakin besar, karena beberapa *smartphone* menyediakan sistem operasi terbaru yaitu android. Sistem operasi ini disediakan dengan memberikan kemudahan kepada para pengguna yaitu kemudahan dalam pengaksesan konten dan aplikasi di dalamnya serta platform yang disediakan bersifat open source. Seperti dikutip dari Asia Media Journal, temuan yang menjadi highlight adalah fakta bahwa di beberapa negara Asia ternyata smartphone telah berhasil mengalahkan feature phone. Tercatat setidaknya di Singapura, Korea, Hong Kong, dan Taiwan, persentase pengguna smartphone sudah lebih dari 50%. Cina sebagai pasar terbesar di planet ini juga menyusul dengan capaian pengguna smartphone sebesar 42%. Indonesia sendiri masih berada di posisi belakang, di mana hanya 19% pengguna yang sudah menggunakan smartphone, dibandingkan pengguna feature phone sebanyak 81%. Di Indonesia sendiri pada tahun 2013 dilaporkan 58% pengguna smartphone telah menggunakan OS android, diikuti oleh 40% pengguna BlackBerry dan 20% pengguna

iOS. Angka ini diikuti oleh 4% pengguna Symbian dan 3% pengguna Windows Mobile (dan Windows Phone) [3]

2.10.1 Versi Android

Banyak smartphone dan PC Tablet menggunakan sistem operasi dengan versi yang berbeda. Semakin tinggi versi, fiturnya semakin canggih dan banyak. Telepon pertama yang memakai sistem operasi *Android* adalah HTC Dream yang dirilis pada tangggal 22 oktober 2008. Beberapa uraian versi *android* seperti dibawah ini:

a) Android Versi 1.5 (Cupcake)
b) Android Versi 1.6 (Donut)
c) Android Versi 2.0/2.1 (Eclair)
d) Android Versi 2.2 (Froyo: Frozen Yoghurt)
e) Android Versi 2.3 (Gingerbread)
f) Android Versi 3.0 (Honeycomb)
g) Android Versi 4.0 Ice Cream Sandwich
h) Android Versi 4.1 (Jelly Bean), dan
i) Android Versi 4.4 (Kitkat)

1. *Android* versi 1.1

Pada 9 Maret 2009, Google merilis Android versi 1.1. *Android* versi ini dilengkapi dengan pembaruan estetis pada aplikasi, jam alarm, voice search (pencarian suara), pengiriman pesan dengan Gmail, dan pemberitahuan email.

2. *Android* versi 1.5 *(Cupcake)*

Pada pertengahan Mei 2009, Google kembali merilis telepon seluler dengan menggunakan *Android* dan SDK *(Software Development Kit)* dengan versi 1.5 *(Cupcake)*. Terdapat beberapa pembaruan termasuk juga penambahan beberapa fitur dalam seluler versi ini yakni kemampuan merekam dan menonton video dengan modus kamera, mengunggah video ke Youtube dan gambar ke Picasa langsung dari telepon, dukungan *Bluetooth* A2DP,

kemampuan terhubung secara otomatis ke *headset Bluetooth*, animasi layar, dan keyboard pada layar yang dapat disesuaikan dengan sistem.

3. *Android* versi 1.6 *(Donut)*

Donut (versi 1.6) dirilis pada September dengan menampilkan proses pencarian yang lebih baik dibanding sebelumnya, penggunaan baterai indikator dan kontrol applet VPN. Fitur lainnya adalah galeri yang memungkinkan pengguna untuk memilih foto yang akan dihapus pada kamera, camcorder dan galeri yang dintegrasikan pada CDMA / EVDO, 802.1x, VPN, Gestures, dan Text-to-speech engine. Kemampuan dial kontak teknologi text to change speech (tidak tersedia pada semua ponsel).

4. *Android* versi 2.0/2.1 *(Eclair)*

Pada 3 Desember 2009 kembali diluncurkan ponsel *Android* dengan versi 2.0/2.1 *(Eclair)*, perubahan yang dilakukan adalah pengoptimalan *hardware*, peningkatan Google Maps 3.1.2, perubahan UI dengan browser baru dan dukungan HTML 5, daftar kontak yang baru, dukungan flash untuk kamera 3,2 MP, digital Zoom, dan *Bluetooth* 2.1.

Untuk bergerak cepat dalam persaingan perangkat generasi berikut, Google melakukan investasi dengan mengadakan kompetisi aplikasi *Mobile* terbaik (killer apps-aplikasi unggulan). Kompetisi ini berhadiah $25,000 bagi setiap pengembang aplikasi terpilih. Kompetisi diadakan selama dua tahap yang tiap tahapnya dipilih 50 aplikasi terbaik.

Dengan semakin berkembangnya dan semakin bertambahnya jumlah handset Android, semakin banyak pihak ketiga yang berminat untuk menyalurkan aplikasi mereka kepada sistem operasi *Android*. Aplikasi terkenal yang diubah ke dalam sistem operasi *Android* adalah Shazam, Backgrounds, dan *WeatherBug*.

Sistem operasi *Android* dalam situs Internet juga dianggap penting untuk menciptakan aplikasi *Android* asli, contohnya oleh MySpace dan Facebook.

5. *Android* versi 2.2 *(Froyo: Frozen Yoghurt)*

Pada 20 Mei 2010, *Android* versi 2.2 *(Froyo)* diluncurkan. Perubahan-perubahan umumnya terhadap versi-versi sebelumnya antara lain dukungan Adobe Flash 10.1, kecepatan kinerja dan aplikasi 2 sampai 5 kali lebih cepat, intergrasi V8 JavaScript *engine* yang dipakai Google Chrome yang mempercepat kemampuan rendering pada browser, pemasangan aplikasi dalam SD Card, kemampuan WiFi *Hotspot portabel*, dan kemampuan *auto update* dalam aplikasi *Android* Market.

6. *Android* versi 2.3 *(Gingerbread)*

Pada 6 Desember 2010, *Android* versi 2.3 *(Gingerbread)* diluncurkan. Perubahan-perubahan umum yang didapat dari *Android* versi ini antara lain peningkatan kemampuan permainan *(gaming)*, peningkatan fungsi *copy paste*, layar antar muka *(User Interface)* didesain ulang, dukungan format video VP8 dan WebM, efek audio baru *(reverb, equalization, headphone virtualization, dan bass boost)*, dukungan kemampuan *Near Field Communication* (NFC), dan dukungan jumlah kamera yang lebih dari satu.

7. *Android* versi 3.0 *(Honeycomb)*

Android Honeycomb dirancang khusus untuk tablet. *Android* versi ini mendukung ukuran layar yang lebih besar. *User Interface* pada *Honeycomb* juga berbeda karena sudah didesain untuk tablet. *Honeycomb* juga mendukung multi prosesor dan juga akselerasi perangkat keras *(hardware)* untuk grafis. Tablet pertama yang dibuat dengan menjalankan *Honeycomb* adalah Motorola Xoom.

8. *Android* versi 4.0 *(Ice Cream Sandwich)*

Dirilis tahun 2012 ini. *Android Jelly Bean* ini diperuntukkan untuk komputer tablet dan memungkinkan untuk digunakan pada sistem operasi PC atau Komputer. Sehingga rumornya kemunculan *Android Jelly Bean* ini untuk menyaingi rilis terbaru Windows 8 yang juga akan segera dirilis. Karena kita

ketahui bersama perbincangan versi Android sebelumnya yaitu *Android Ice Cream Sandwhich* pun masih hangat di telinga.

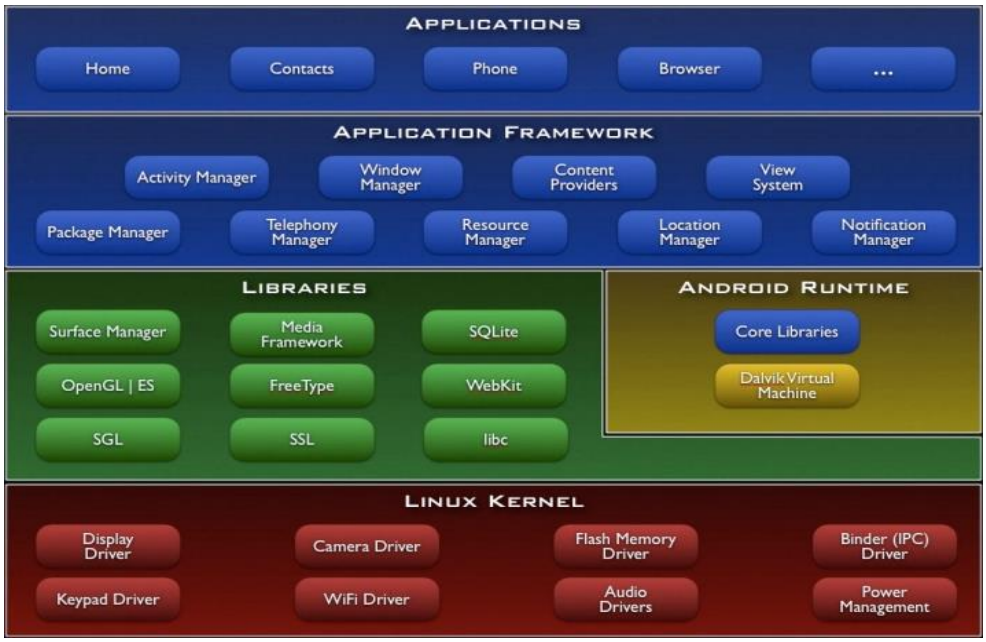
9. *Android* versi 4.4 (*Kitkat*)

Dirilis tahun 2013. *Kitkat* merupakan sebuah produk makanan coklat yang dibuat oleh *nestle.* Menurut Direktur *Android, Jhon Lagerling.* Pemilihan nama *Kitkat* bertujuan untuk melakukan sesuatu yang menyenangkan dan tidak terduga.

Sesuai dengan namanya, logo *Android Kitkat* adalah robot *Android* yang terbuat dari batangan coklat *Kitkat. Android Kitkat* sendiri diklaim *Google* bakal menjadi nyawa setidaknya 1 miliar resmi, *Google* telah membuat situs web khusus untuk *Android Kitkat* [13].

2.10.2 Anatomi Aplikasi Android

Dalam paket sistem operasi Android tediri dari beberapa unsur seperti tampak pada gambar 2.14 Secara sederhana arsitektur Android merupakan sebuah kernel Linux dan sekumpulan pustaka C / C++ dalam suatu *Framework* yang menyediakan dan mengatur alur proses aplikasi.



Gambar 2.14 Detail Anatomi Android [14]

2.10.2.1 Linux Kernel

Android dibangun di atas kernel Linux 2.6. Namun secara keseluruhan Android bukanlah linux, karena dalam Android tidak terdapat paket standar yang dimiliki oleh linux lainnya. Linux merupakan sistem operasi terbuka yang handal dalam manajemen memori dan proses. Oleh karenanya pada Android hanya terdapat beberapa pelayanan yang diperlukan seperti keamanan, manajemen memori, manajemen proses, jaringan dan *driver*. Kernel linux menyediakan *driver* layar, kamera, *keypad*, *WiFi*, *Flash Memory*, audio, dan IPC (*Interprocess Communication*) untuk mengatur aplikasi dan lubang keamanan.

2.10.2.2 *Libraries*

Android menggunakan beberapa paket pustaka yang terdapat pada C/C++ dengan standar *Berkeley Software Distribution* (BSD) hanya setengah dari yang aslinya untuk tertanam pada kernel Linux. Beberapa pustaka diantaranya:

1. *Media Library* untuk memutar dan merekam berbagai macam format audio dan video.
2. *Surface Manager* untuk mengatur hak akses layer dari berbagai aplikasi.
3. *Graphic Library* termasuk didalamnya *SGL* dan *OpenGL*, untuk tampilan 2D dan 3D.
4. SQLite untuk mengatur relasi database yang digunakan pada aplikasi.
5. *SSl* dan *WebKit* untuk browser dan keamanan *internet*.

Pustaka-pustaka tersebut bukanlah aplikasi yang berjalan sendiri, namun hanya dapat digunakan oleh program yang berada di level atasnya. Sejak versi Android 1.5, pengembang dapat membuat dan menggunakan pustaka sendiri menggunakan *Native Development Toolkit* (NDK).

2.10.2.3 Android *Runtime*

Pada Android tertanam paket pustaka inti yang menyediakan sebagian besar fungsi Android. Inilah yang membedakan Android dibandingkan dengan

sistem operasi lain yang juga mengimplementasikan Linux. Android *Runtime* merupakan mesin virtual yang membuat aplikasi Android menjadi lebih tangguh dengan paket pustaka yang telah ada. Dalam Android Runtime terdapat 2 bagian utama, diantaranya :

1. Pustaka Inti, Android dikembangkan melalui bahasa pemrograman Java, tapi Android *Runtime* bukanlah mesin virtual Java. Pustaka inti Android menyediakan hampir semua fungsi yang terdapat pada pustaka Java serta beberapa pustaka khusus Android.
2. Mesin Virtual Dalvik, Dalvik merupakan sebuah mesin virtual yang dikembangkan oleh Dan Bornstein yang terinspirasi dari nama sebuah perkampungan yang berada di Iceland. Dalvik hanyalah interpreter mesin virtual yang mengeksekusi file dalam format *Dalvik Executable* (*.dex). Dengan format ini Dalvik akan mengoptimalkan efisiensi penyimpanan dan pengalamatan memori pada file yang dieksekusi. Dalvik berjalan di atas kernel Linux 2.6, dengan fungsi dasar seperti *threading* dan manajemen memori yang terbatas.

2.10.2.4 *Aplication Framework*

Kerangka aplikasi menyediakan kelas-kelas yang dapat digunakan untuk mengembangkan aplikasi Android. Selain itu, juga menyediakan abstraksi generik untuk mengakses perangkat, serta mengatur tampilan *User Interface* dan sumber daya aplikasi. Bagian terpenting dalam kerangka aplikasi Android adalah sebagai berikut:

1. *Activity Manager*, berfungsi untuk mengontrol siklus hidup aplikasi dan menjaga keadaan ”*Backstack*“ untuk navigasi penggunaan.
2. *Content Providers*, berfungsi untuk merangkum data yang memungkinkan digunakan oleh aplikasi lainnya, seperti daftar nama.
3. *Resuource Manager*, untuk mengatur sumber daya yang ada dalam program. Serta menyediakan akses sumber daya diluar kode program, seperti karakter, grafik, dan *file layout*.

4. *Location Manager*, berfungsi untuk memberikan informasi detail mengenai lokasi perangkat Android berada.

Notification Manager, mencakup berbagai macam peringatan seperti, pesan masuk, janji, dan lain sebagainya yang akan ditampilkan pada *status bar*.

2.10.2.5 *Application Layer*

Puncak dari diagram arsitektur Android adalah lapisan aplikasi dan *widget*. Lapisan aplikasi merupakan lapisan yang paling tampak pada pengguna ketika menjalankan program. Pengguna hanya akan melihat program ketika digunakan tanpa mengetahui proses yang terjadi dibalik lapisan aplikasi. Lapisan ini berjalan dalam Android *runtime* dengan menggunakan kelas dan service yang tersedia pada *Framework* aplikasi.

Lapisan aplikasi Android sangat berbeda dibandingkan dengan sistem operasi lainnya. Pada Android semua aplikasi, baik aplikasi inti (*native*) maupun aplikasi pihak ketiga berjalan diatas lapisan aplikasi dengan menggunakan pustaka API (*Application Programming Interface*) yang sama.

2.10.3 Komponen Aplikasi Android

Fitur penting Android adalah bahwa satu aplikasi dapat menggunakan elemen dari aplikasi lain (untuk aplikasi yang memungkinkan). Sebagai contoh, sebuah aplikasi memerlukan fitur *scroller* dan aplikasi lain telah mengembangkan fitur *scroller* yang baik dan memungkinkan aplikasi lain menggunakannya. Maka pengembang tidak perlu lagi mengembangkan hal serupa untuk aplikasinya, cukup menggunakan *scroller* yang telah ada.

Agar fitur tersebut dapat bekerja, sistem harus dapat menjalankan aplikasi ketika setiap bagian aplikasi itu dibutuhkan, dan pemanggilan objek java untuk bagian itu. Oleh karenanya Android berbeda dari sistem-sistem lain, Android tidak memiliki satu tampilan utama program seperti fungsi main() pada aplikasi lain. Sebaliknya, aplikasi memiliki komponen penting yang memungkinkan sistem untuk memanggil dan menjalankan ketika dibutuhkan.

2.10.3.1 *Activities*

Activity merupakan bagian yang paling penting dalam sebuah aplikasi, karena *Activity* menyajikan tampilan visual program yang sedang digunakan oleh pengguna. Setiap *Activity* dideklarasikan dalam sebuah kelas yang bertugas untuk menampilkan antarmuka pengguna yang terdiri dari *Views* dan respon terhadap *Event*. Setiap aplikasi memiliki sebuah *activity* atau lebih. Biasanya pasti akan ada *activity* yang pertama kali tampil ketika aplikasi dijalankan.

Perpindahan antara *activity* dengan *activity* lainnya diatur melalui sistem, dengan memanfaatkan *activity stack*. Keadaan suatu *activity* ditentukan oleh posisinya dalam tumpukan *acitivity*, LIFO (*Last In First Out*) dari semua aplikasi yang sedang berjalan. Bila suatu *activity* baru dimulai, *activity* yang sebelumnya digunakan maka akan dipindahkan ketumpukan paling atas. Jika pengguna ingin menggunakan *activity* sebelumnya, cukup menekan tombol *Back*, atau menutup *activity* yang sedang digunakan, maka *activity* yang berada diatas akan aktif kembali. *Memory Manager* Android menggunakan tumpukkan ini untuk menentukan prioritas aplikasi berdasarkan *activity*, memutuskan untuk mengakhiri suatu aplikasi dan mengambil sumber daya dari aplikasi tersebut.

Ketika *activity* diambil dan disimpan dalam tumpukkan *activity* terdapat 4 kemungkinan kondisi transisi yang akan terjadi :

1) ***Active***, setiap *activity* yang berada ditumpukan paling atas, maka dia akan terlihat, terfokus, dan menerima masukkan dari pengguna. Android akan berusaha untuk membuat *activity* aplikasi ini untuk untuk tetap hidup dengan segala cara, bahkan akan menghentikan *activity* yang berada dibawah tumpukkannya jika diperlukan. Ketika *activity* sedang aktif, maka yang lainnya akan dihentikan sementara.
2) ***Paused***, dalam beberapa kasus *activity* akan terlihat tapi tidak terfokus pada kondisi inilah disebut *paused*. Keadaan ini terjadi jika *activity* transparan dan tidak *fullscreen* pada layar. Ketika *activity* dalam keadaan

paused, dia terlihat *active* namun tidak dapat menerima masukkan dari pengguna. Dalam kasus ekstrim, Android akan menghentikan *activity* dalam keadaan *paused* ini, untuk menunjang sumber daya bagi *activity* yang sedang aktif.

3) ***Stopped***, ketika sebuah *activity* tidak terlihat, maka itulah yang disebut *stopped*. *Activity* akan tetap berada dalam memori dengan semua keadaan dan informasi yang ada. Namun akan menjadi kandidat utama untuk dieksekusi oleh sistem ketika membutuhkan sumberdaya lebih. Oleh karenanya ketika suatu *activity* dalam kondisi *stopped* maka perlu disimpan data dan kondisi antarmuka saat itu. Karena ketika *activity* telah keluar atau ditutup, maka dia akan menjadi *inactive*.
4) ***Inactive***, kondisi ketika *activity* telah dihentikan dan sebelum dijalankan. *Inactive activity* telah ditiadakan dari tumpukan *activity* sehingga perlu *restart* ulang agar dapat tampil dan digunakan kembali.

Kondisi transisi ini sepenuhnya ditangani oleh manajer memori Android. Android akan memulai menutup aplikasi yang mengandung *activity inactive*, kemudian *stopped activity*, dan dalam kasus luar biasa *paused activity* juga akan ditutup.

2.10.3.2 *Services*

Suatu *service* tidak memiliki tampilan antarmuka, melainkan berjalan di *background* untuk waktu yang tidak terbatas. Komponen *service* diproses tidak terlihat, memperbarui sumber data dan menampilkan notifikasi. *Service* digunakan untuk melakukan pengolahan data yang perlu terus diproses, bahkan ketika *Activity* tidak aktif atau tidak tampak.

2.10.3.3 *Intents*

Intens merupakan sebuah mekanisme untuk menggambarkan tindakan tertentu, seperti memilih foto, menampilkan halaman *web*, dan lain sebagainya. *Intents* tidak selalu dimulai dengan menjalankan aplikasi, namun juga digunakan oleh sistem untuk memberitahukan ke aplikasi bila terjadi suatu hal,

misal pesan masuk. *Intents* dapat eksplisit atau implisit, contohnya jika suatu aplikasi ingin menampilkan *URL*, sistem akan menentukan komponen apa yang dibutuhkan oleh *Intents* tersebut.

2.10.3.4 *Broadcast Receiver*

Broadcast Receivers merupakan komponen yang sebenarnya tidak melakukan apa-apa kecuali menerima dan bereaksi menyampaikan pemberitahuan. Sebagian besar *Broadcast* berasal dari sistem misalnya, Batre sudah hampir habis, informasi zona waktu telah berubah, atau pengguna telah merubah bahasa *default* pada perangkat. Sama halnya dengan *service*, *Broadcast Receivers* tidak menampilkan antarmuka pengguna. Namun, *Broadcast Receivers* dapat menggunakan *Notification Manager* untuk memberitahukan sesuatu kepada pengguna.

2.10.3.5 *Content Providers*

Content Providers digunakan untuk mengelola dan berbagi *database*. Data dapat disimpan dalam file sistem, dalam database *SQLite*, atau dengan cara lain yang pada prinsipnya sama. Dengan adanya *Content Provider* memungkinkan antar aplikasi untuk saling berbagi data. Komponen ini sangat berguna ketika sebuah aplikasi membutuhkan data dari aplikasi lain, sehingga mudah dalam penerapannya.

2.10.4 Tipe Aplikasi Android

Terdapat tiga kategori aplikasi pada Android :

1. ***Foreground Activity***

   Aplikasi yang hanya dapat dijalankan jika tampil pada layar dan tetap efektif walaupun tidak terlihat. Aplikasi dengan tipe ini pasti mempertimbangkan siklus hidup *activity*, sehingga perpindahan antar *activity* dapat berlangsung dengan lancar.

2. ***Background Service***

   Aplikasi yang memiliki interaksi terbatas dengan user, selain dari pengaturan konfigurasi, semua dari prosesnya tidak tidak tampak pada layar. Contohnya aplikasi penyaringan panggilan atau sms *auto respon*.

3. ***Intermittent Activity***

   Aplikasi yang masih membutuhkan beberapa masukkan dari pengguna, namun sebagian sangat efektif jika dijalankan di *background* dan jika diperlukan akan memberi tahu pengguna tentang kondisi tertentu. Contohnya pemutar musik.

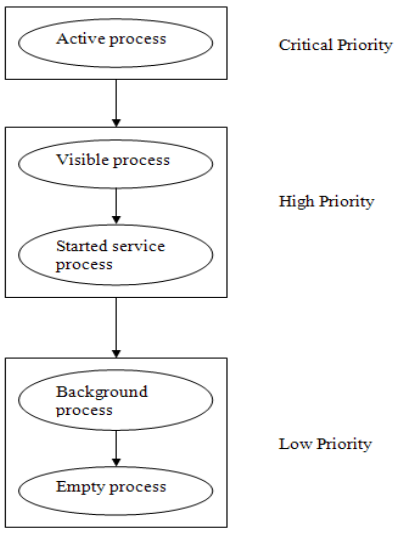
Untuk aplikasi yang kompleks akan sulit untuk menentukan kategori aplikasi tersebut apalagi aplikasi memiliki ciri-ciri dari semua kategori. Oleh karenanya perlu pertimbangan bagaimana aplikasi tersebut digunakan dan menentukan kategori aplikasi yang sesuai.

2.10.5 Siklus Hidup Android

Siklus hidup aplikasi Android dikelola oleh sistem, berdasarkan kebutuhan pengguna, sumberdaya yang tersedia, dan sebagainya. Misalnya Pengguna ingin menjalankan *browser web*, pada akhirnya sistem yang akan menentukan menjalankan aplikasi. Sistem sangat berperan dalam menentukan apakah aplikasi dijalankan, dihentikan sementara, atau dihentikan sama sekali. Jika pengguna ketika itu sedang menjalankan sebuah *Activity*, maka sistem akan memberikan perioritas utama untuk aplikasi yang tersebut. Sebaliknya, jika suatu *Activity* tidak terlihat dan sistem membutuhkan sumber daya yang lebih, maka *Activity* yang prioritas rendah akan ditutup.

Android menjalankan setiap aplikasi dalam proses secara terpisah, yang masing-masing memiliki mesin *virtual* pengolah sendiri, dengan ini melindungi penggunaan memori pada aplikasi. Selain itu juga Android dapat mengontrol aplikasi mana yang layak menjadi prioritas utama. Karenanya Android sangat sensitive dengan siklus hidup aplikasi dan komponen-komponennya. Perlu

adanya penanganan terhadap setiap kondisi agar aplikasi menjadi stabil. Gambar 2.2 menunjukkan prioritas dari aplikasi.



Gambar 2.15 Prioritas Aplikasi Berdasarkan *Activity*

2.10.6 Kelebihan Android

Sudah banyak platform untuk perangkat selular saat ini, termasuk didalamnya Symbian, iPhone, Windows Mobile, BlackBerry, Java Mobile Edition, Linux Mobile (LiM), dan banyak lagi. Namun ada beberapa hal yang menjadi kelebihan Android. Walaupun beberapa fitur-fitur yang ada telah muncul sebelumnya pada *platform* lain, Android adalah yang pertama menggabungkan hal seperti berikut :

1. Keterbukaan, Bebas pengembangan tanpa dikenakan biaya terhadap sistem karena berbasiskan Linux dan *open source*. Pembuat perangkat menyukai hal ini karena dapat membangun *platform* yang sesuai yang diinginkan tanpa harus membayar royality. Sementara pengembang *software* menyukai karena Android dapat digunakan diperangkat manapun dan tanpa terikat oleh *vendor* manapun.
2. Arsitektur komponen dasar Android terinspirasi dari teknologi *internet Mashup*. Bagian dalam sebuah aplikasi dapat digunakan oleh aplikasi

lainnya, bahkan dapat diganti dengan komponen lain yang sesuai dengan aplikasi yang dikembangkan.

3. Banyak dukungan *service*, kemudahan dalam menggunakan berbagai macam layanan pada aplikasi seperti penggunaan layanan pencarian lokasi, *database* SQL, *browser* dan penggunaan peta. Semua itu sudah tertanam pada Android sehingga memudahkan dalam pengembangan aplikasi.
4. Siklus hidup aplikasi diatur secara otomatis, setiap program terjaga antara satu sama lain oleh berbagai lapisan keamanan, sehingga kerja sistem menjadi lebih stabil. Pengguna tak perlu kawatir dalam menggunakan aplikasi pada perangkat yang memorinya terbatas.
5. Dukungan grafis dan suara terbaik, dengan adanya dukungan 2D grafis dan animasi yang diilhami oleh Flash menyatu dalam 3D menggunakan OpenGL memungkinkan membuat aplikasi maupun *game* yang berbeda.
6. *Portabilitas* aplikasi, aplikasi dapat digunakan pada perangkat yang ada saat ini maupun yang akan datang. Semua program ditulis dengan menggunakan bahas pemrograman Java dan dieksekusi oleh mesin *virtual Dalvik*, sehingga kode program *portable* antara ARM, X86, dan arsitektur lainnya. Sama halnya dengan dukungan masukan seperti penggunaan *Keyboard*, layar sentuh, *trackball* dan resolusi layar semua dapat disesuaikan dengan program.

2.11 Tinajuan Perangkat Lunak Pendukung

Perangkat pendukung yang penulis pergunakan adalah bahasa pmrograman Java, Eclipse.

2.11.1 Java

Java adalah bahasa pemrograman yang dapat dijalankan di berbagai komputer termasuk telepon genggam. Bahasa ini awalnya dibuat oleh James Gosling saat masih bergabung di Sun Microsystems saat ini merupakan bagian dari Oracle dan dirilis tahun 1995. Bahasa ini banyak mengadopsi sintaksis

yang terdapat pada C dan C++ namun dengan sintaksis model objek yang lebih sederhana serta dukungan rutin-rutin aras bawah yang minimal. Aplikasi-aplikasi berbasis java umumnya dikompilasi ke dalam p-code *(bytecode)* dan dapat dijalankan pada berbagai Mesin Virtual Java (JVM). Java merupakan bahasa pemrograman yang bersifat umum/non-spesifik *(general purpose)*, dan secara khusus didisain untuk memanfaatkan dependensi implementasi seminimal mungkin.



Gambar 2.16 Logo *Java*

Karena fungsionalitasnya yang memungkinkan aplikasi java mampu berjalan di beberapa platform sistem operasi yang berbeda, java dikenal pula dengan slogannya, "Tulis sekali, jalankan di mana pun". Saat ini java merupakan bahasa pemrograman yang paling populer digunakan, dan secara luas dimanfaatkan dalam pengembangan berbagai jenis perangkat lunak aplikasi ataupun aplikasi berbasis web.

2.11.2 Sejarah perkembangan Java

Bahasa pemrograman Java pertama lahir dari *The Green Project*, yang berjalan selama 18 bulan, dari awal tahun 1991 hingga musim panas 1992. Proyek tersebut belum menggunakan versi yang dinamakan Oak. Proyek ini dimotori oleh Patrick Naughton, Mike Sheridan, James Gosling dan Bill Joy, beserta sembilan pemrogram lainnya dari Sun Microsystems. Salah satu hasil proyek ini adalah maskot Duke yang dibuat oleh Joe Palrang. Pertemuan proyek berlangsung di sebuah gedung perkantoran Sand Hill Road di Menlo Park. Sekitar musim panas 1992 proyek ini ditutup dengan menghasilkan

sebuah program Java Oak pertama, yang ditujukan sebagai pengendali sebuah peralatan dengan teknologi layar sentuh (touch screen), seperti pada PDA sekarang ini. Teknologi baru ini dinamai "*7" *(Star Seven*).

Setelah era Star Seven selesai, sebuah anak perusahaan Tv kabel tertarik ditambah beberapa orang dari proyek *The Green Project*. Mereka memusatkan kegiatannya pada sebuah ruangan kantor di 100 Hamilton Avenue, Palo Alto. Perusahaan baru ini bertambah maju, jumlah karyawan meningkat dalam waktu singkat dari 13 menjadi 70 orang. Pada rentang waktu ini juga ditetapkan pemakaian Internet sebagai medium yang menjembatani kerja dan ide di antara mereka. Pada awal tahun 1990-an, Internet masih merupakan rintisan, yang dipakai hanya di kalangan akademisi dan militer. Mereka menjadikan perambah (browser) Mosaic sebagai landasan awal untuk membuat perambah Java pertama yang dinamai Web Runner, terinsipirasi dari film 1980-an, Blade Runner. Pada perkembangan rilis pertama, Web Runner berganti nama menjadi Hot Java. Pada sekitar bulan Maret 1995, untuk pertama kali kode sumber Java versi 1.0a2 dibuka. Kesuksesan mereka diikuti dengan untuk pemberitaan pertama kali pada surat kabar San Jose Mercury News pada tanggal 23 Mei 1995. Sayang terjadi perpecahan di antara mereka suatu hari pada pukul 04.00 di sebuah ruangan hotel Sheraton Palace. Tiga dari pimpinan utama proyek, Eric Schmidt dan George Paolini dari Sun Microsystems bersama Marc Andreessen, membentuk Netscape. Nama Oak, diambil dari pohon oak yang tumbuh di depan jendela ruangan kerja "bapak java", James Gosling. Nama Oak ini tidak dipakai untuk versi release Java karena sebuah perangkat lunak sudah terdaftar dengan merek dagang tersebut, sehingga diambil nama penggantinya menjadi "Java". Nama ini diambil dari kopi murni yang digiling langsung dari biji (kopi tubruk) kesukaan Gosling. Konon kopi ini berasal dari Pulau Jawa. Jadi nama bahasa pemrograman Java tidak lain berasal dari kata Jawa (bahasa Inggris untuk Jawa adalah Java).

2.11.3 Eclipse IDE

Eclipse adalah sebuah komunitas proyek *open source* yang difokuskan pada membangun sebuah platform pengembangan yang dapat diperluas, runtime dan kerangka aplikasi untuk membangun, menyebarkan dan mengelola perangkat lunak di seluruh daur hidup perangkat lunak. Komunitas Eclipse memiliki lebih dari 60 proyek *open source*. Proyek-proyek ini secara konsep terbagi menjadi 7 kategori :

a. *Enterprise Development*
b. *Embedded and Device Development*
c. *Rich Client Platform*
d. *Rich Internet Applications*
e. *Application Frameworks*
f. *Application Lifecycle Management (ALM)*
g. *Service Oriented Architecture (SOA)*

Komunitas Eclipse juga didukung oleh ekosistem besar dan dinamis dari penyedia solusi teknologi informasi utama, start-up inovatif, universitas dan lembaga penelitian, dan individu yang memperluas, mendukung, dan melengkapi *platform* Eclipse.

2.11.3.1 Software Development Kit (SDK)

Software Development Kit (SDK) adalah suatu kit atau library dari bahasa pemrograman untuk pengembangan atau pembangunan suatu perangkat lunak dan biasanya SDK terdiri dari kumpulan tools yang dibutuhkan. Misalnya bahasa pemrograman java, mempunyai SDK yang berisi suatu *library* yang dapat digunakan untuk membuat suatu aplikasi berbasis java.

2.11.3.2 Java Development Kit (JDK)

Java Development Kit (JDK) adalah sekumpulan perangkat lunak yang dapat kamu gunakan untuk mengembangkan perangkat lunak yang berbasis Java, Sedangkan JRE adalah sebuah implementasi dari Java *Virtual Machine* yang benar-benar digunakan untuk menjalankan program java.Biasanya, setiap

JDK berisi satu atau lebih JRE dan berbagai alat pengembangan lain seperti sumber kompiler java, bundling, debuggers, development libraries dan lain sebagainya. Perbedaan JDK dengan SDK *(Software Development Kit)* yaitu JDK adalah sebuah SDK tetapi sebuah SDK tidak harus menjadi sebuah JDK.

2.11.3.3 ADT

Android Development Tools (*ADT*) adalah *plugin* untuk Eclipse *Intergrated Development Environment* (*IDE*) yang dirancang untuk memberikan lingkungan yang terpadu di mana untuk membangun aplikasi *android*. ADT memperluas kemampuan Eclipse untuk membiarkan para *developer* lebih cepat dalam membuat proyek baru Android, membuat aplikasi UI, menambahkan komponen berdasarkan Android *Framework* API, *debug* aplikasi dalam pengunaan Android SDK, dan membuat file APK untuk mendistribusikan aplikasi. Mengembangkan aplikasi di Eclipse dengan ADT sangat dianjurkan dan merupakan cara tercepat untuk memulai membuat aplikasi android, karena banyak kemudahan-kemudahan sebagai tools yang terintegrasi seperti, custom XML editor, dan *debug panel ouput*. Selain itu ADT memberikan dorongan luar biasa dalam mengembangkan aplikasi Android.

2.12 Skala Likert (*Likert Scales*)

Skala likert adalah suatu skala psikometrik yang digunakan dalam kuesioner dan merupakan salah satu teknik yang dapat digunakan dalam evaluasi suatu program atau kebijakan perencanaan. Rensis Likert telah mengembangkan sebuah skala untuk mengukur sikap masyarakat di tahun 1932 yang sekarang terkenal dengan nama skala Likert[16]. Skala Likert ini merupakan skala yang dapat dipergunakan untuk mengukur sikap, pendapat, dan persepsi seseorang atau sekelompok orang mengenai suatu gejala atau fenomena. Dan pada evaluasi, skala likert digunakan untuk (a) Menilai keberhasilan suatu kebijakan atau program (b) Menilai manfaat pelaksanaan

suatu kebijakan atau program (c) Mengetahui kepuasan stakeholder terhadap pelaksanaan suatu kebijakan atau program.

Dalam skala Likert terdapat dua bentuk pernyataan yaitu pernyataan positif yang berfungsi untuk mengukur sikap positif, dan pernyataan negatif yang berfungsi untuk mengukur sikap negatif objek. Skor pernyataan positif dimulai dari 1 untuk sangat tidak setuju (ST), 2 untuk tidak setuju (T), 3 untuk tak memutuskan (N), 4 untuk setuju (S), dan 5 untuk sangat setuju (SS). Skor pernyataan negatif dimulai dari 1 untuk sangat setuju (SS), 2 untuk setuju (S), 3 untuk tak memutuskan (N), 4 untuk tidak setuju (T), dan 5 untuk sangat tidak setuju (ST). Skala Likert digunakan untuk mengukur **kesetujuan dan ketidaksetujuan** seseorang terhadap sesuatu rencana program, pelaksanaan program ataupun tingkat keberhasilan suatu program.

Tabel 2.1 Bobot Penilaian Skala Likert

Arah Pertanyaan	Bobot Penilaian				
	Sangat Setuju (SS)	Setuju (S)	Tak Memutuskan (N)	Tidak Setuju (T)	Sangat Tidak Setuju (ST)
Positif	5	4	3	2	1
Negatif	1	2	3	4	5

Berdasarkan data hasil kuesioner tersebut, dapat dicari persentase masing-masing jawaban dengan menggunakan rumus :

$$Y = \frac{X}{Skor\ ideal}\ x\ 100\%$$

Keterangan :

Y = Nilai persentase yang dicari

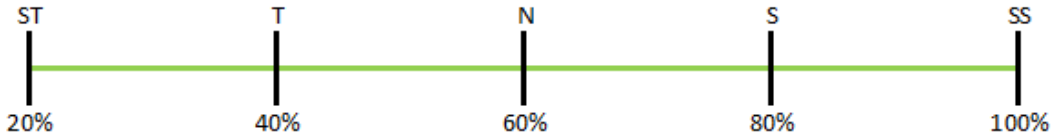
X = Jumlah nilai kategori jawaban dikalikan dengan frekuensi ($\sum$ = N . R)

N = Nilai dari setiap jawaban

R = Frekuensi

Skor ideal = Nilai tertinggi dikalikan dengan jumlah sampel

Setelah diperoleh nilai Y maka interpretasi nilai dapat dilihat sebagai berikut :



Gambar 2.17 Skala Penilaian Kuesioner

Setelah didapatkan jumlah yang pasti dari setiap kuesioner, selanjutnya penulis menarik kesimpulan dari masing-masing kuesioner yang disediakan.

2.12.1 Kelebihan Skala Likert

Beberapa Kelebihan Skala Likert dalam riset berupa survei, antara lain :

1. Mudah dibuat dan diterapkan.
2. Skala Likert lebih mudah membuatnya dibandingkan dengan skala yang lain seperti skala Thurstone.
3. Terdapat kebebasan dalam memasukan pertanyaan- pertanyaan, asalkan sesuai dengan konteks permasalahan yang diteliti.
4. Jawaban suatu item dapat berupa alternatif, sehingga informasi mengenai item tersebut diperjelas.
5. Reliabilitas pengukuran bisa diperoleh dengan jumlah item tersebut diperjelas.

Karena jangka responsi yang lebih besar membuat skala Likert dapat memberikan keterangan yang lebih jelas dan nyata tentang pendapatan atau sikap responden tentang isu yang dipertanyakan.

2.12.2 Kekurangan Skala Likert

Beberapa Kekurangan Skala Likert dalam riset berupa survei, antara lain :

1. Karena ukuran yang digunakan adalah ukuran ordinal, skala Likert hanya dapat mengurutkan individu dalam skala, tetapi tidak dapat membandingkan berapa kali satu individu lebih baik dari individu yang lain.
2. Kadangkala total skor dari individu tidak memberikan arti yang jelas, karena banyak pola respons terhadap beberapa item akan memberikan skor yang sama. Adanya kelemahan di atas sebenarnya dapat dipikirkan sebagai *error* dari respons yang terjadi.