BAB 2
LANDASAN TEORI

2.1 Sejarah TOEFL

Tes TOEFL adalah merek dagang terdaftar dari Educational Testing Service (ETS) dan diberikan di seluruh dunia. Tes ini pertama kali diberikan pada tahun 1964 dan sejak itu telah diambil oleh lebih dari 23 juta siswa. Tes awalnya dikembangkan di Pusat Linguistik Terapan dipimpin oleh ahli bahasa, Dr Charles A. Ferguson.

Komite TOEFL Penguji terdiri dari 12 spesialis dalam linguistik, pengujian bahasa, pengajaran atau penelitian. Tanggung jawab utamanya adalah untuk memberi nasehat tentang isi tes TOEFL. Komite ini membantu memastikan tes adalah ukuran yang valid kemampuan berbahasa Inggris mencerminkan tren saat ini dan metodologi.

TOEFL adalah singkatan dari Test Of English as a Foreign language. Toefl merupakan sebuah test yang digunakan untuk mengetahui kemampuan seseorang dalam berbahsa inggris. Test toefl berawal dari sebuah proyek ferguson yang digunakana untuk mengukur kemampuan bahasa pegawai pemerintah dan mahasiswa dan sejak tahun 1960-an, tes TOEFL telah dikelola oleh Educational Testing Service (ETS), sebuah organisasi standar-pengujian internasional.Test toefl terdiri dari test membaca, menulis , mendengarkan dan berbicara. Dulu test toefl hanya berbasis test kertas dan test komputer, namun sejak tahun 2005 test berbasis internet telah banyak menggantikan 2 test terdahulu tersebut.Waktu yang diberlakukan untuk tes TOEFL secara keseluruhan kurang lebih 150 menit, untuk Paper and Pencil Based TOEFL, dan sekitar 240 menit untuk Computer Based TOEFL. Waktu untuk Computer Based TOEFL ini relatif lama karena sudah termasuk prosedur tutorial.

.

2.1.1 Definisi TOEFL

Kemampuan berbahasa asing pada zaman globalisasi ini merupakan hal yang sangat penting. Segala bidang kehidupan saat ini menuntut adanya penguasaan bahasa asing, termasuk sektor pendidikan. Test of English As A Foreign Language (TOEFL) merupakan tes untuk menguji kemampuan bahasa Inggris pada negara-negara yang tidak menggunakan bahasa Inggris sebagai bahasa sehari-hari. TOEFL merupakan tolak ukur yang dipercaya dalam menyatakan kemampuan berbahasa Inggris seseorang.

Jika dulu TOEFL hanya digunakan pada saat pelamaran beasiswa belajar ke luar negeri, maka saat ini TOEFL juga digunakan sebagai salah satu syarat untuk melengkapi kelulusan atau pun syarat minimal untuk dapat mengikuti jenjang pendidikan tertentu. Dalam mencapai skor TOEFL minimal yang telah ditetapkan, maka tidak semua mahasiswa atau calon mahasiswa mampu mencapainya. Apalagi bahasa Inggris bukan merupakan bahasa yang akrab digunakan dalam budaya bangsa Indonesia. Hal ini akhirnya menimbulkan kendala bagi para mahasiswa untuk dapat melanjutkan studinya atau dalam mendapatkan gelar mereka masing-masing. Seringkali mahasiswa tertunda kelulusan atau kelanjutan studinya hanya karena belum melulusi TOEFL.

TOEFL dan bahasa Inggris merupakan hal yang asing di Indonesia. Maka untuk dapat melulusi TOEFL dengan menghafal atau belajar grammar mati-matian juga bukan merupakan strategi yang tepat. Sehingga pengenalan TOEFL dan tips dan trik untuk melulusi TOEFL lah yang sesungguhnya diperlukan.

2.1.2 Contoh Soal TOEFL

Ini adalah contoh soal listening comprehension sebagai berikut :

Anda akan mendengar percakapan dari dua orang berikut [2]:

1. (*man*) : What's the matter? You don't look too good.
   (*woman*) : I was frightened by a loud noise.
   (*narrator*) : **WHAT DOES THE WOMAN MEAN ?**
   (A) She was not alllowed to fight.
   (B) Something scared her.
   (C) she made a load noise that frightened some people
   (D) some load neighbors had a fight
2. (*woman*) : Is nicky going to work on the science project with us ?
   (*man*) : She prefer not to work on group project.
   (*narrator*) : **WHAT DOES THE MAN SAY ABOUT NICKY ?**
   (A) She'd rather work alone.
   (B) Group work is her preferences.
   (C) She's working on a project about group preferences.
   (D) She projected that the group wouldn't work.

Ini adalah contoh soal Structur and Written Expression sebagai berikut :

1. Young children _____ playing actives game such as chasing, hide and seek, and tag.
   (A) Enjoy
   (B) Enjoys
   (C) Enjoying
   (D) Does enjoy
2. The final class presentation is the importan test examination.
   A B B D

Ini adalah contoh soal reading comprehension

The rattles with wich a ratlesnake warns of its presence are formed by loosely interlocking hollow rings of hard skin, which make a buzzing sound when its tails is shaken. As a baby, the snake begins to form its rattles from the button at the very tip of its tail. Ther after, each time it sheds its skin, a new ring is formed.

Popular belief hold thats a snake age can be told by counting the rings, but this idea is fallacious. In fact, a snake may lose its old skin as often as four times ayear. Also, rattless tend to wear or break off wich time.

1. A rattlesnake’s rattles are made of
   (a) Skin
   (b) Wood
   (c) Bone
   (d) Muscle
2. How often does a rattlesnake shed it skin ?
   (a) Once every four years
   (b) Once every four month
   (c) Up to four times a year
   (d) Four times more often than other snake

2.2 Android



Gambar 2.2 Logo Android

Android adalah sistem operasi yang berbasis *Linux* untuk telepon seluler seperti telepon pintar (*Smartphone*) dan komputer tablet. Android menyediakan *platform* terbuka bagi para pengembang untuk menciptakan aplikasi mereka sendiri untuk digunakan oleh bermacam peranti begerak. Awalnya, *Google Inc.* membeli *Android Inc.*, pendatang baru yang membuat peranti lunak untuk ponsel. Kemudian untuk mengembangkan *Android*, dibentuklah *Open Handset Alliance*, konsorsium dari 34 perusahaan piranti keras, piranti lunak dan telekomunikasi, termasuk *Google, HTC, Intel, Motorola, Qualcomm, T-Mobile* dan *Nvidia.*[5]

Bebrapa pengertian lain dari *Android*, yaitu :

1. Merupakan *platform* terbuka (*Open Source*) bagi para programmer untuk membuat atau mengembangkan aplikasi.

2. Merupakan sistem operasi yang dibeli *Google Inc* dari *Android Inc*.
3. Bukan bahasa pemrograman, akan tetapi hanya menyediakan lingkungan hidup atau *runtime environment* yang disebut DVM (*Dalvik Virtual Machine*) yang telah dioptimasi untuk *device* dengan sistem memori yang kecil.

2.2.1 Sejarah Android

Pada Juli 2000, *Google* bekerjasama dengan *Android Inc*., perusahaan yang berada di Palo Alto, California, Amerika Serikat. Para pendiri *Android Inc*. bekerja pada *Google,* diantaranya Andy Rubin, Rich Miner, Nick Sears dan Chris White. Saat itu banyak yang menganggap fungsi *Android Inc.* hanyalah sebagai perangkat lunak pada telepon seluler. Sejak saat itu muncul rumor bahwa *Google* hendak memasuki pasar telepon seluler. Di perusahaan *Google,* tim yang dipimpin Rubin bertugas mengembangkan program perangkat seluler yang didukung oleh *Kernel Linux*. Hal ini menunjukan indikasi bahwa *Google* sedang bersiap menghadapi persaingan dalam pasara telepon seluler. Versi *android* terbaru yaitu versi 5.0. (*KitKat*). *Android* juga sudah bergabung dengan beberapa *smart mobile* seperti *LG, Samsung, Sony Ericsson, HTC, Motorola* dan lainnya.

Sekitar September 2007 sebuah studi melaporkan bahwa *Google* mengajukan hak paten aplikasi telepon seluler (akhirnya *Google* mengenalkan *Nexus One,* salah satu jenis telepon pintar GSM yang menggunakan *Android* pada sistem operasinya. Telepon seluler ini diproduksi oleh *HTC Corporotaion* dan tersedia di pasaran pada 5 Januari 2010).

Pada Desember 2008, diumumkan anggota baru yang bergabung dalam program kerja *Android ARM Holdings, Atheros Communications,* diproduksi oleh *Asustek Computer Inc. Garmin Ltd, Softbank, Sony Ericsson Toshiba Corp* dan *Vodafone Group Plc.* Seiring pembentukan *Open Handset Alliance,* OHA mengumumkan produk perdana mereka, *Android,* perangkat bergerak (*Mobile*) yang merupakan modifikasi *kernel Linux* 2.6. sejak *Android* dirilis telah dilakukan berbagai pembaruan berupa perbaikan *bug* dan penambahan fitur baru.

2.2.2 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 tanggal 22 Oktober 2008. Beberapa uraian versi *android* seperti dibawah ini :

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 diintegrasikan pada CDMA / EVDO, 802.1x, VPN, *Gestures*, dan *Text-to-spech 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 mengoptimalkan *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, integrasi dengan 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 versi ini

antara lain peningkatan kemampuan permainan (*gaming*), peningkatan fungsi *copy paste*, layar antarmuka (*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 3.1 (*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 peratam yang dibuat dengan menjalankan *Honeycomb* adalah Motorola Xoom.

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

Diumumkan pada tanggal 19 Oktober 2011, membawa fitur *Honeycomb* untuk *smartphone* dan menambahkan fitur baru termasuk membuka kunci dengan pengenalan wajah, jaringan data pemantauan penggunaan dan kontrol, terpadu kontak jaringan sosial, perangkat tambahan fotografi, mencari email secara *offline* dan berbagi informasi dengan menggunakan NFC.

9. *Android* versi 4.1 (*Jelly Bean*)

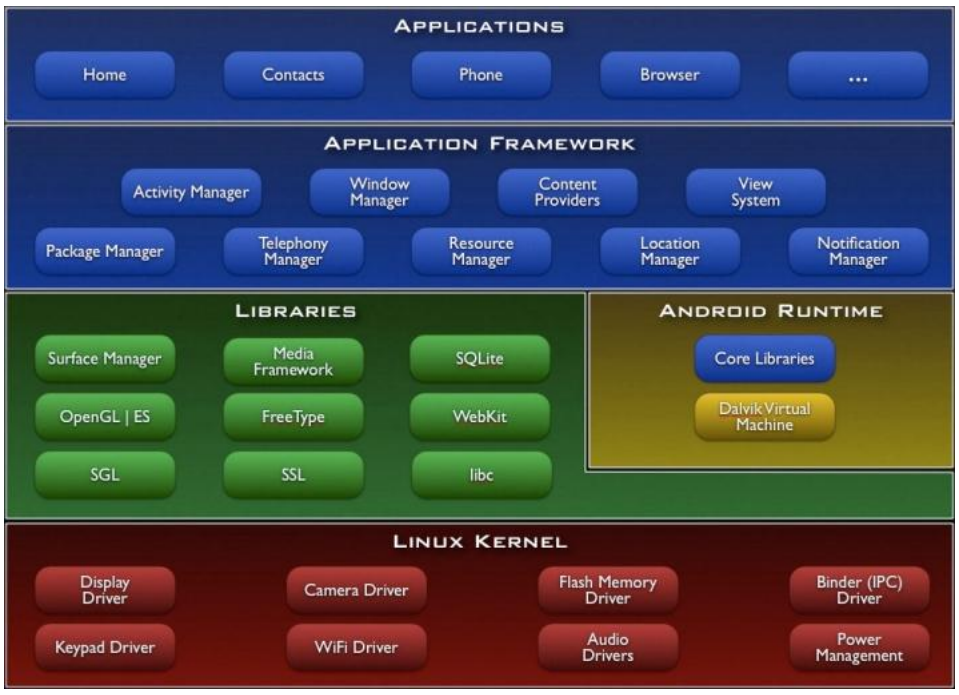
Pada tanggal 27 Juni 2012 Google mengumumkan *Android* versi 4.1 (*Jelly Bean*) dalam konferensi Google I/O berdasarkan kernel Linux 3.0.31. *Jelly Bean* adalah pembaruan penting yang bertujuan untuk meningkatkan fungsi dan kinerja antarmuak pengguna (*User Interface*). Pembaruan ini diwujudkan dalam proyek *Butter*, perbaikan ini termasuk antisipasi sentuh, *triple buffering*, perpanjangan waktu vsync dan peningkatan *frame rate* hingga 60fps untuk menciptakan UI yang lebih halus. Perangkat pertama yang menggunakan sistem operasi ini adalah tablet Nexus 7, yang dirilis pada 13 Juli 2012.

10. *Android* versi 4.4 (KitKat)

Android versi 4.4 (KitKat) direncanakan akan dirilis pada bulan Oktober 2013. Setelah sebelumnya beredar rumor bahwa Android versi berikutnya setelah *Jelly Bean* diperkirakan akan diberi nomor 5.0 dan dinamai Key Lime Pie.

2.2.3 Anatomi Aplikasi *Android*

Dalam paket sistem operasi Android tediri dari beberapa unsur seperti tampak pada gambar 2.1. Secara sederhana arsitektur Android merupakan sebuah kernel Linux dan sekumpulan pustaka C / C++ dalam suatu *Framework* yang menyediakan dan mengatur alur proses aplikasi.



Gambar 2.3 Detail Anatomi Android [9]

2.2.3.1 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.

2.2.3.2 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).

2.2.3.3 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.

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

2.2.3.5 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.

2.2.4 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.

2.2.4.1 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.

2.2.4.2 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.

2.2.4.3 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.

2.2.4.4 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.

2.2.4.5 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.

2.2.5 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***

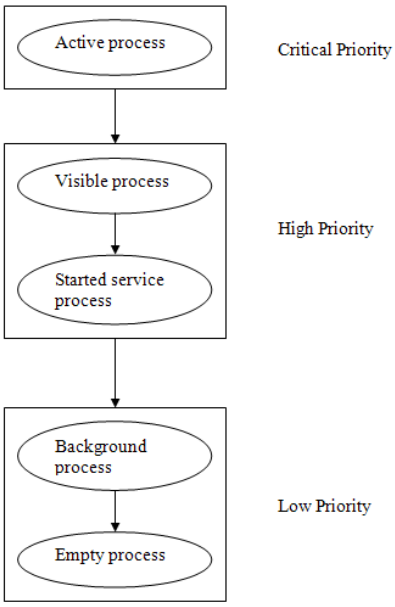
   Aplikasi yang masih membutuhkan beberapa masukkan dari pengguna, namun sebagian sangat efektif jika dijalankan di *background* dan jika diperlukan akan memberi tahu pengguna tentang kondisi tertentu. Contohnya pemutar musik.

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.

2.2.6 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 2.3 Prioritas Aplikasi Berdasarkan *Activity*

2.2.7 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.

2.3 Perancangan Berorientasi Objek

Teknologi objek menganalogikan sistem aplikasi seperti kehidupan nyata yang didominasi oleh objek. Didalam membangun sistem berorientasi objek akan menjadi lebih baik apabila langkah awalnya didahului dengan proses analisis dan perancangan yang berorientasi objek. Tujuannya adalah mempermudah *programmer* didalam mendesain program dalam bentuk objek-objek dan hubungan antar objek tersebut untuk kemudian dimodelkan dalam sistem nyata. Suatu perusahaan *software* yaitu *Rational Software,* telah membentuk konsarium dengan berbagai organisasi untuk meresmikan pemakaian *Unified Modelling Language* (UML) sebagai bahasa standar dalam *Object Oriented Analysist Design* (OOAD).

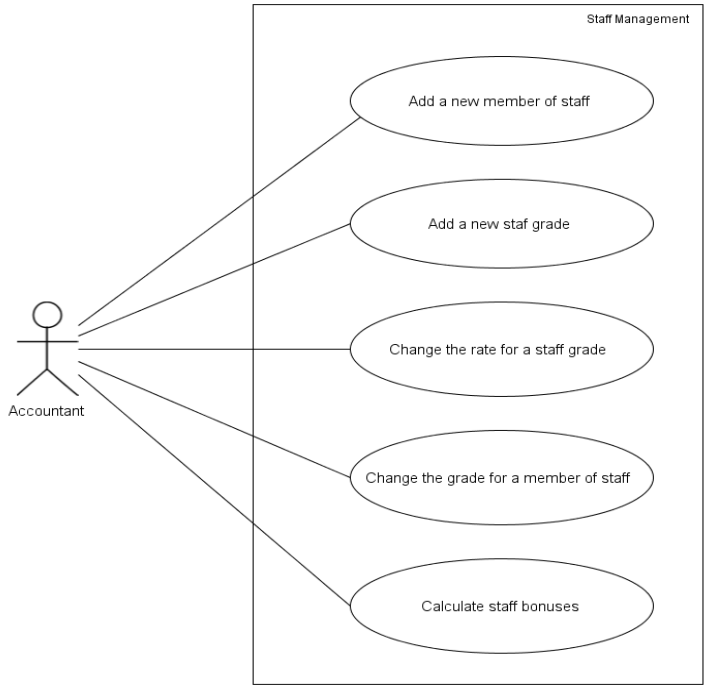
2.3.1 Unified Modelling Language (UML)

UML dalam sebuah bahasa untuk menentukan visualisasi, konstruksi, dan mendokumentasikan artifacts dari sistem *software*, untuk memodelkan bisnis, dan sistem *non-software* lainnya. UML merupakan sistem arsitektur yang bekerja dalam OOAD dengan satu bahasa yang konsisten untuk menentukan, visualisasi, konstruksi dan mendokumentasikan *artifact* yang terdapat dalam sistem. *Artifact* adalah sepotong informasi yang digunakan atau dihasilkan dalam suatu proses rekayasa *software*. *Artifact* dapat berupa model, deskripsi atau *software*.[5]

2.3.1.1 Use Case Diagram

Use Case Diagram menjelakan manfaat sistem jika dilihat menurut pandangan orang yang berada diluar sistem (Aktor). Diagram ini menunjukan fungsionalitas suatu sistem yang berinteraksi dengan dunia luar. *Use Case Diagram* dapat

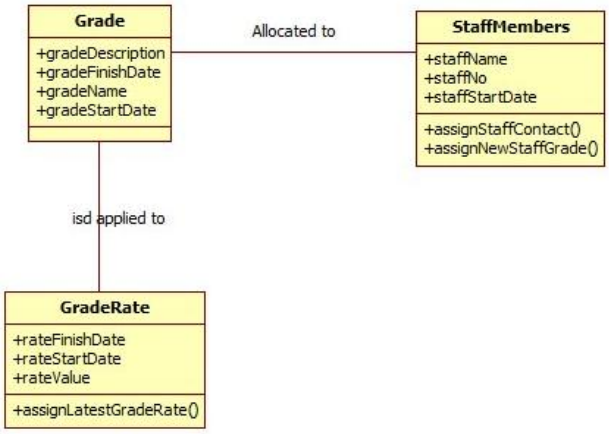
digunakan selama proses analisis untuk menagkap *requitment* sistem dan untuk memahami bagaimana sistem bekerja.



Gambar 2.4 *Use Case Diagram*

2.3.1.2 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 2.5 *Class Diagram*

2.3.1.3 Behavior Diagram

Behavior diagram dapat dikelompokan menjadi tiga diagram, yaitu :

1. *Statechart Diagram*

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

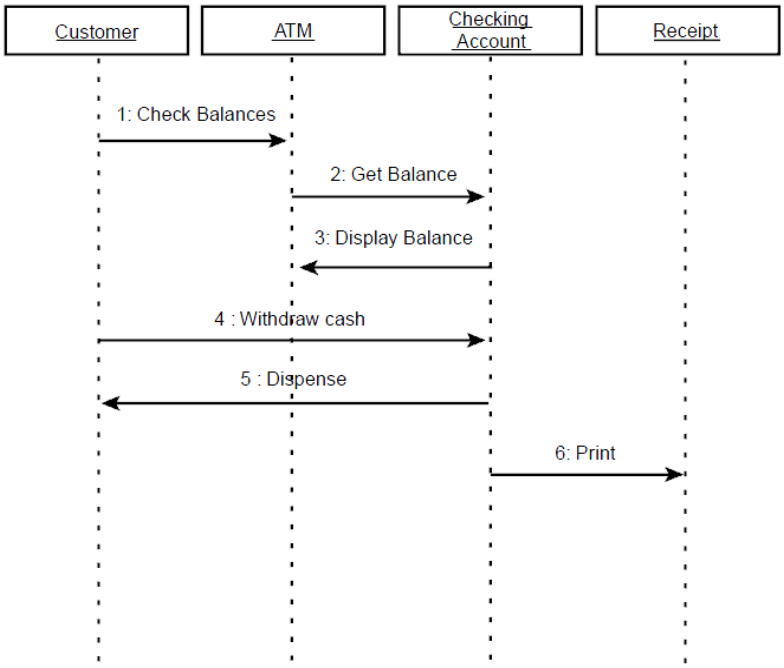
2. *Activity Diagram*

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

3. *Interaction Diagram*

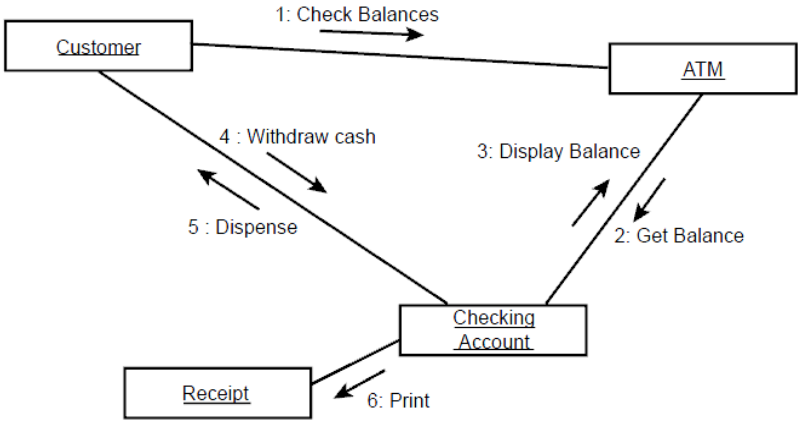
   Interaction Diagram dibagi menjadi dua model diagram yaitu :

   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 2.6 *Sequence Diagram*

2. *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 2.7 *Collaboration Diagram*

2.3.1.4 Implementation 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*)

   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.

2.4 Java

Java adalah bahasa pemrograman yang dapat dijalankan di berbagai komputer termasuk telepon genggam. Dikembangkan oleh Sun Microsystems dan diterbitkan tahun 1995. Java tidak boleh disalahpahami sebagai JavaScript. JavaScript adalah bahasa scripting yang digunakan oleh Web Browser[6]

2.4.1 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 programer 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*).

2.4.2 Versi Awal

Versi awal Java ditahun 1996 sudah merupakan versi *release* sehingga dinamakan Java Versi 1.0. Java versi ini menyertakan banyak paket standar awal yang terus dikembangkan pada versi selanjutnya:

1. Java.lang: Peruntukan kelas elemen-elemen dasar.
2. Java.io: Peruntukan kelas *input* dan *output*, termasuk penggunaan berkas.

3. Java.util: Peruntukan kelas pelengkap seperti kelas struktur data dan kelas-kelas penanggalan.
4. Java.net: Peruntukan kelas TCP/IP, yang memungkinkan berkomunikasi dengan komputer lain menggunakan jaringan TCP/IP.
5. Java.awt: Kelas dasar untuk aplikasi antarmuka dengan pengguna (GUI).
6. Java.applet: Kelas dasar aplikasi antar muka untuk diterapkan pada penjelajah web.

2.4.3 Kelebihan Java

a) *Multiplatform*. Kelebihan utama dari Java ialah dapat dijalankan di beberapa platform / sistem operasi komputer, sesuai dengan prinsip tulis sekali, jalankan di mana saja.
b) OOP (*Object Oriented Programming* - Pemrogram Berorientasi Objek) yang artinya semua aspek yang terdapat di Java adalah Objek. Hal ini sangat memudahkan pemrogram untuk mendesain, membuat, mengembangkan dan mengalokasi kesalahan sebuah program dengan basis Java secara cepat, tepat, mudah dan terorganisir.
c) Perpustakaan Kelas yang Lengkap, Java terkenal dengan kelengkapan library/perpustakaan (kumpulan program program yang disertakan dalam pemrograman java) yang sangat memudahkan dalam penggunaan oleh para pemrogram untuk membangun aplikasinya.
d) Bergaya C++, memiliki sintaks seperti bahasa pemrograman C++ sehingga menarik banyak pemrogram C++ untuk pindah ke Java.
e) Pengumpulan sampah otomatis, memiliki fasilitas pengaturan penggunaan memori sehingga para pemrogram tidak perlu melakukan pengaturan memori secara langsung (seperti halnya dalam bahasa C++ yang dipakai secara luas).

2.4.4 Kekurangan Java

1. Tulis sekali, perbaiki di mana saja - Masih ada beberapa hal yang tidak kompatibel antara *platform* satu dengan *platform* lain. Untuk J2SE, misalnya SWT-AWT *bridge* yang sampai sekarang tidak berfungsi pada Mac OS X.

2. Mudah didekompilasi. Dekompilasi adalah proses membalikkan dari kode jadi menjadi kode sumber. Ini dimungkinkan karena kode jadi Java merupakan *bytecode* yang menyimpan banyak atribut bahasa tingkat tinggi, seperti nama-nama kelas, metode, dan tipe data. Hal yang sama juga terjadi pada Microsoft .NET *Platform*. Dengan demikian, algoritma yang digunakan program akan lebih sulit disembunyikan dan mudah dibajak/di-*reverse engineer*.
3. Penggunaan memori yang banyak. Penggunaan memori untuk program berbasis Java jauh lebih besar daripada bahasa tingkat tinggi generasi sebelumnya seperti C/C++ dan Pascal.

2.4.5 Tahap Kompilasi Java

1. Tulis / Ubah. Pemrogram menulis program dan menyimpannya di media dalam bentuk berkas '.java'.
2. Kompilasi. Pengkompilasi membentuk *bytecodes* dari program menjadi bentuk berkas '.*class*'.
3. Muat. Pemuat kelas memuat *bytecodes* ke memori.
4. Verifikasi. Peng-verifikasi memastikan *bytecodes* tidak mengganggu sistem keamanan Java.
5. Jalankan. Penerjemah menerjemahkan bytecodes ke bahasa mesin tidak bisa di pakai.

2.5 Web Service

Web service adalah suatu sistem perangkat lunak yang dirancang untuk mendukung interoperabilitas dan interaksi antar sistem pada suatu jaringan. Web service digunakan sebagai suatu fasilitas yang disediakan oleh suatu web site untuk menyediakan layanan (dalam bentuk informasi) kepada sistem lain, sehingga sistem lain dapat berinteraksi dengan sistem tersebut melalui layanan-layanan (service) yang disediakan oleh suatu sistem yang menyediakan web service. Web service menyimpan data informasi dalam format XML, sehingga data ini dapat diakses oleh sistem lain walaupun berbeda platform, sistem operasi, maupun bahasa compiler. [7]

Web service bertujuan untuk meningkatkan kolaborasi antar pemrogram dan perusahaan, yang memungkinkan sebuah fungsi di dalam Web Service dapat dipinjam oleh aplikasi lain tanpa perlu mengetahui detil pemrograman yang terdapat di dalamnya.

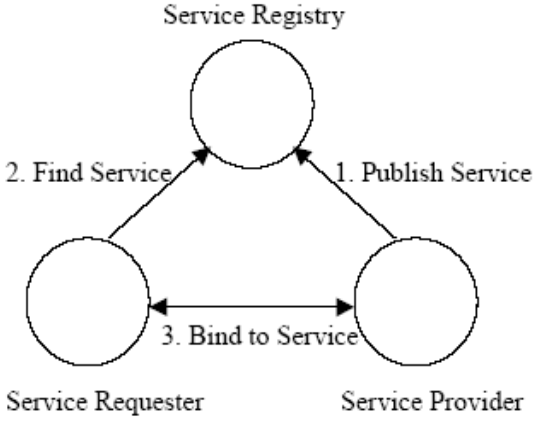
Beberapa alasan mengapa digunakannya web service adalah sebagai berikut:

1. Web service dapat digunakan untuk mentransformasikan satu atau beberapa bisnis logic atau class dan objek yang terpisah dalam satu ruang lingkup yang menjadi satu, sehingga tingkat keamanan dapat ditangani dengan baik.
2. Web service memiliki kemudahan dalam proses deployment-nya, karena tidak memerlukan registrasi khusus ke dalam suatu sistem operasi. Web service cukup di-upload ke web server dan siap diakses oleh pihak-pihak yang telah diberikan otorisasi.
3. Web service berjalan di port 80 yang merupakan protokol standar HTTP, dengan demikian web service tidak memerlukan konfigurasi khusus di sisi firewall.

2.5.1 Arsitekur Web Service

Web service memiliki tiga entitas dalam arsitekturnya, yaitu:

1. Service Requester (peminta layanan)
2. Service Provider (penyedia layanan)
4. Service Registry (daftar layanan)



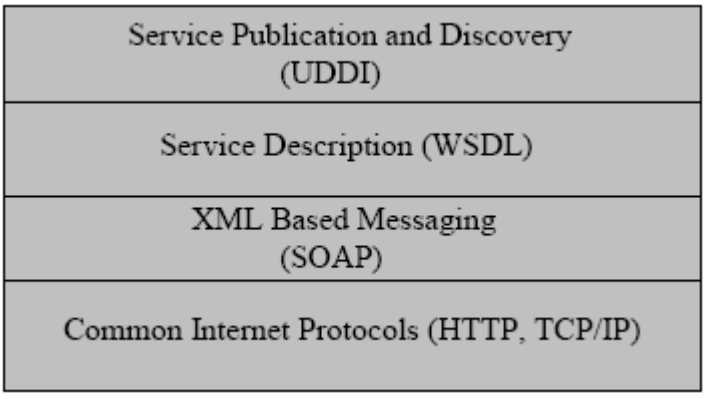
- Service Provider: Berfungsi untuk menyediakan layanan/service dan mengolah sebuah registry agar layanan-layanan tersebut dapat tersedia.
- Service Registry: Berfungsi sebagai lokasi central yang mendeskripsikan semua layanan/service yang telah di-register.
- Service Requestor: Peminta layanan yang mencari dan menemukan layanan yang dibutuhkan serta menggunakan layanan tersebut.

2.5.2 Operasi – Operasi Web Service

Secara umum, web service memiliki tiga operasi yang terlibat di dalamnya, yaitu:

1. Publish/Unpublish: Menerbitkan/menghapus layanan ke dalam atau dari registry.
2. Find: Service requestor mencari dan menemukan layanan yang dibutuhkan.
3. Bind: Service requestor setelah menemukan layanan yang dicarinya, kemudian melakukan binding ke service provider untuk melakukan interaksi dan mengakses layanan/service yang disediakan oleh service provider.

2.5.3 Komponen – Komponen Web Service



Web service secara keseluruhan memiliki empat layer komponen seperti pada gambar di atas, yaitu:

1. Layer 1: Protokol internet standar seperti HTTP, TCP/IP

2. Layer 2: Simple Object Access Protocol (SOAP), merupakan protokol akses objek berbasis XML yang digunakan untuk proses pertukaran data/informasi antar layanan.
3. Layer 3: Web Service Definition Language (WSDL), merupakan suatu standar bahasa dalam format XML yang berfungsi untuk mendeskripsikan seluruh layanan yang tersedia.

2.6 *Database* (Basis Data)

Beberapa definisi basis data sebagai berikut :

1. *Database* adalah sekumpulan data *store* (bias dalam jumlah yang besar) yang tersimpan dalam magnetic disk, optical disk, magnetic drum atau media penyimpanan .[8]
2. *Database* adalah sekumpulan program-program aplikasi umum yang bersifat "batch" yang mengeksekusi dan memproses data secara umum (seperti pencarian,peremajaan,penambahan, dan penghapusan terhadap data)
3. Database terdiri dari data yang akan digunakan atau diperuntukan terhadap banyak 'user' (baik menggunakan teknik pemrosesan yang bersifat batch atau on-line) akan menggunakan data tersebut sesuai dengan tudas dan fungsinya, dan 'user' lain dapat juga menggunakan data tersebut dalam waktu yang bersamaan.

2.6.1 MySQl

MySQL adalah sebuah program *database server* yang mampu menerima dan mengirimkan datanya dengan sangat cepat, *multi user* serta menggunakan perintah standar SQL (*Structured Query Language*).

MySQL dapat juga berperan sebagai *client* sehingga sering disebut database *client/server*, yang *open source* dengan kemampuan dapat beroperasi di OS (*operating system*) manapun .

Contoh :

1. DDL (*Data Definition language*)

a. *Create* : mysql > CREARE DATABASE nama_*database*;
b. *Drop* : mysql > DROP DATABASE nama_*database*;
c. *Alter* : mysql > ALTER nama_tabel alter_spesifikasi kondisi;

2. DML (*Data Manipulation Language*)
   a. *INSERT* : mysql > INSERT INTO nama_tabel VALUES (isi_field);
   b. SELECT :mysql> SELECT * from nama_tabel ;
   c. *UPDATE* : mysql > UPDATE nama_tabel SET isi_field=’data_baru’;
   d. *DELETE* : *DELETE* nama_tabel *WHERE* kondisi;

2.6.2 SQL (Structure Query Language)

SQL adalah sebuah bahasa permintaan *database* yang terstruktur. Bahasa SQL dibuat sebagai bahasa yang dapat merelasikan beberapa tabel dalam *database* maupun merelasikan antar *database*. Bahasa SQL ditulis langsung dalam sebuah program *databases* sehingga pengguna dapat melihat langsung permintaan yang diinginkan sekaligus melihat hasilnya.

SQL dibagi menjadi 2 bentuk query, yaitu [9]

1. DDL (*Data Definition Language*)

DDL adalah sebuah metode *query* sql yang berguna untuk mendefinisikan data pada sebuah *database*, adapun *query* yang dimiliki adalah

a. CREATE : Digunakan untuk melakukan pembuatan tabel dan *database*
b. DROP : Digunakan untuk melakukan penghapusan tabel maupun *database*
c. ALTER : Digunakan untuk melakukan perubahan struktur tabel yang telah dibuat, baik menambah *field* (*add*), mengganti nama *field* (*change*) ataupun menamakannya kembali (*rename*) serta menghapus (*drop*)

2. **DML (*Data Manipulation Language*)**

   DML adalah sebuah metode *query* yang dapat digunakan apabila DDL telah terjadi, sehingga fungsi dari *query* ini adalah untuk melaukan pemanipulasian *database* yang telah ada atau telah dibuat sebelumnya. Adapun *query* yang termasuk di dalamnya adalah

   a. INSERT : Digunakan untuk melakukan penginputan pada tabel di *database* .
   b. UPDATE : Digunakan untuk melakukan pengubahan atau peremajaan terhadap data yang ada pada tabel
   c. DELETE : Digunakan untuk melakukan penghapusan data pada tabel. Penghapusan ini dapat dilakukan secara sekaligus (seluruh isi tabel) maupun hanya beberapa *Recordset*

2.7 Eclipse

Eclipse adalah sebuah IDE (*Integrated Development Environment*) untuk mengembangkan perangkat lunak dan dapat dijalankan di semua platform (*platform-independent*). Berikut ini adalah sifat dari Eclipse:

1. *Multi-platform*: Target sistem operasi Eclipse adalah Microsoft Windows, Linux, Solaris, AIX, HP-UX dan Mac OS X.
2. *Multi-language*: Eclipse dikembangkan dengan bahasa pemrograman Java, akan tetapi Eclipse mendukung pengembangan aplikasi berbasis bahasa pemrograman lainnya, seperti C/C++, Cobol, Python, Perl, PHP, dan lain sebagainya.
3. *Multi-role*: Selain sebagai IDE untuk pengembangan aplikasi, Eclipse pun bisa digunakan untuk aktivitas dalam siklus pengembangan perangkat lunak, seperti dokumentasi, test perangkat lunak, pengembangan web, dan lain sebagainya.

Eclipse pada saat ini merupakan salah satu IDE favorit dikarenakan gratis dan *open source*, yang berarti setiap orang boleh melihat kode pemrograman

perangkat lunak ini. Selain itu, kelebihan dari Eclipse yang membuatnya popular adalah kemampuannya untuk dapat dikembangkan oleh pengguna dengan komponen yang dinamakan *plug-in*.

2.7.1 Sejarah Eclipse

Eclipse awalnya dikembangkan oleh IBM untuk menggantikan perangkat lunak IBM *Visual Age for Java* 4.0. Produk ini diluncurkan oleh IBM pada tahun 2001. Pada tahun 2003 platform Eclipse tumbuh dengan cepat sebagai aplikasi *open source* dan *commercial extensions*, sehingga menjadi pengembang yang terkenal. Pada tahun 2004 Eclipse *Foundation* yang merupakan sebuah organisasi nonprofit dengan keprofesionalannya serta *independent* dapat mengambil alih penuh platform Eclipse. Eclipse adalah pengembang java terkemuka, karena unik dan arsitektur yang dapat diperluas serta dapat digunakan untuk berbagai macam bahasa pemrograman.

2.7.2 Arsitektur Eclipse

Sejak versi 3.0, Eclipse pada dasarnya merupakan sebuah *kernel*, yang mengangkat *plug-in*. Apa yang dapat digunakan di dalam Eclipse sebenarnya adalah fungsi dari *plug-in* yang sudah diinstal. Ini merupakan basis dari Eclipse yang dinamakan *Rich Client Platform* (RCP). Berikut ini adalah komponen yang membentuk RCP:

1. *Core platform*
2. OSGi
3. SWT (*Standard Widget Toolkit*)
4. JFace
5. Eclipse *Workbench*

Secara standar Eclipse selalu dilengkapi dengan JDT (*Java Development Tools*), *plug-in* yang membuat Eclipse kompatibel untuk mengembangkan program Java, dan PDE (*Plug-in Development Environment*) untuk mengembangkan *plug-in* baru. Eclipse beserta *plug-in*-nya diimplementasikan dalam bahasa pemrograman Java.

Konsep Eclipse adalah IDE yang terbuka (*open*), mudah diperluas (*extensible*) untuk apa saja, dan tidak untuk sesuatu yang spesifik. Jadi, Eclipse tidak saja untuk mengembangkan program Java, akan tetapi dapat digunakan untuk berbagai macam keperluan, cukup dengan menginstal *plug-in* yang dibutuhkan. Apabila ingin mengembangkan program C/C++ terdapat *plug-in* CDT (C/C++ *Development Tools*). Selain itu, pengembangan secara visual bukan hal yang tidak mungkin oleh Eclipse, *plug-in* UML2 tersedia untuk membuat diagram UML. Dengan menggunakan PDE setiap orang bisa membuat *plug-in* sesuai dengan keinginannya.

2.7.3 Perkembangan Eclipse

Sejak tahun 2006, Eclipse Foundation mengkoordinasikan peluncuran Eclipse secara rutin dan simultan yang dikenal dengan nama *Simultaneous Release*. Setiap versi peluncuran terdiri dari Eclipse Platform dan juga sejumlah proyek yang terlibat dalam proyek Eclipse. Tujuan dari sistem ini adalah untuk menyediakan distribusi Eclipse dengan fitur-fitur dan versi yang terstandarisasi. Hal ini juga dimaksudkan untuk mempermudah deployment dan maintenance untuk sistem enterprise. Adapun versi Eclipse yang telah diluncurkan yaitu :

1. Eclipse 3.0
2. Eclipse 3.1
3. Callisto
4. Europa
5. Ganymede
6. Galileo
7. Helios
8. Indigo
9. Juno
10. Kepler

2.8 Software Develpment 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.

2.9 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.

2.10 Android Development Tools (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.