# BAB 2
# LANDASAN TEORI

## 2.1 Landasan Teori

Landasan Teori merupakan definisi, konsep yang telah disusun secara sistematis dan dasar yang kuat dalam sebuah penelitian. Landasan teori yang digunakan dalam penyusunan Pembangunan Aplikasi *Geofencing* berbasis *mobile* meliputi: Aplikasi Mobile, Pengertian Aplikasi, Pengertian Mobile, Pengertian Aplikasi Mobile, Data dan Informasi, GPS, Location Based Service (LBS), Latitude dan Longitude, Geofencing, Google Maps, Google *Maps* API, Basis Data, *Unified Modeling Language* (UML), Android, Android dan GPS, HTML5 Canvas, PHP, Phonegap, Perangkat Lunak Pendukung, MySQL, Web Server dan Web Service.

## 2.2 Aplikasi Mobile

Aplikasi *mobile* saat ini sangat dibutuhkan karena alat-alat telekomunikasi yang tersebar di seluruh dunia membutuhkan aplikasi-aplikasi yang dapat memepermudah pekerjaan penggunanya dimanapun dan kapanpun terutama dalam hal informasi. Informasi sendiri merupakan hasil dari proses analisis, manipulasi dan presentasi data untuk mendukung proses pengambilan keputusan.

## 2.3 Pengertian Aplikasi

Aplikasi adalah satu unit perangkat lunak yang dibuat untuk melayani kebutuhan akan beberapa aktivitas. Jika ingin mengembangkan program aplikasi sendiri, maka untuk menulis program aplikasi tersebut, dibutuhkan suatu bahasa pemrograman, yaitu *language software*, yang dapat berbentuk *assembler*, *compiler* ataupun *interpreter*. Jadi *language software* merupakan bahasanya dan program yang ditulis merupakan program aplikasinya. *Language software* berfungsi agar dapat menulis program dengan bahasa yang lebih mudah, dan akan menterjemahkannya ke dalam bahasa mesin supaya bisa dimengerti oleh komputer. Bila hendak mengembangkan suatu program aplikasi untuk

memecahkan permasalahan yang besar dan rumit, maka supaya program aplikasi tersebut dapat berhasil dengan baik, maka dibutuhkan prosedur dan perencanaan yang baik dalam mengembangkannya.

Sekarang, banyak sekali program-program aplikasi yang tersedia dalam bentuk paket-paket program. Ini adalah program-program aplikasi yang sudah ditulis oleh orang lain atau perusahaan-perusahaan perangkat lunak. Beberapa perusahaan perangkat lunak telah memproduksi paket-paket perangkat lunak yang mempunyai reputasi internasional. Program-program paket tersebut dapat diandalkan, dapat memenhi kebutuhan pemakai, dirancang dengan baik, relatif bebas dari kesalahan-kesalahan, *user friendly* (mudah digunakan), mempunyai dokumentasi manual yang memadai, mampu dikembangkan untuk kebutuhan mendatang, dan didukung perkembangannya. Akan tetapi, bila permasalahannya bersifat khusus dan unik, sehingga tidak ada paket-paket program yang sesuai untuk digunakan, maka dengan terpaksa harus mengembangkan program aplikasi itu sendiri. supaya program aplikasi tersebut dapat berhasil dengan baik, maka dibutuhkan prosedur dan perencanaan yang baik dalam mengembangkannya.

## 2.4 Pengertian Mobile

*Mobile* diartikan sebagai perpindahan yang mudah dari satu tempat ke tempat yang lain, misalnya telepon *mobile* berarti bahwa terminal telepon yang dapat berpindah dengan mudah dari satu tempat ke tempat lain tanpa terjadi pemutusan atau terputusnya komunikasi.

### 2.4.1 Pengertian Aplikasi Mobile

Aplikasi *mobile* berasal dari kata *application* dan *mobile*. *Application* yang artinya penerapan, lamaran, penggunaan. Secara istilah aplikasi adalah program siap pakai yang direka untuk melaksanakan suatu fungsi bagi pengguna atau aplikasi yang lain dan dapat digunakan oleh sasaran yang dituju sedangkan *mobile* dapat di artikan sebagai perpindahan dari suatu tempat ke tempat yang lain.

Maka aplikasi *mobile* dapat di artikan sebuah program aplikasi yang dapat dijalankan atau digunakan walaupun pengguna berpindah – pindah dari satu tempat ke tempat yang lain serta mempunyai ukuran yang kecil. Aplikasi *mobile*

ini dapat di akses melalui perangkat nirkabel, pager, PDA, telepon seluler, *smartphone*, dan perangkat sejenisnya.

## 2.5 Data dan Informasi

Berikut adalah pengertian tentang data dan informasi:

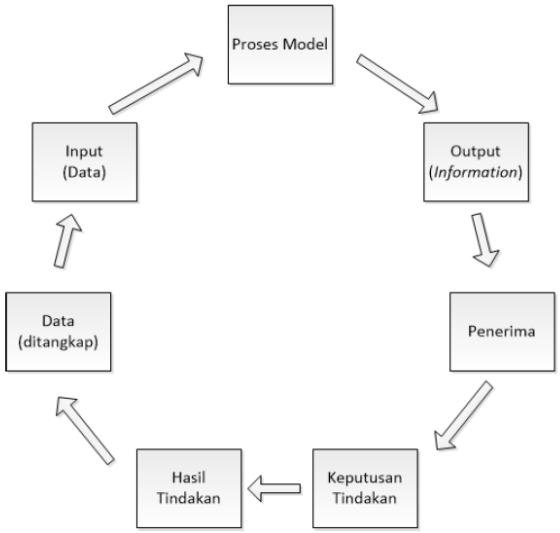
### 2.5.1 Data

Data merupakan sesuatu yang tidak mempunyai arti bagi penggunanya dan masih memerlukan adanya suatu pengolahan terhadap data tersebut. Data bisa berbentuk suatu kejadian, benda, aktifitas, gambar, huruf dan sebagainya. Data adalah bahan mentah untuk menjadi suatu informasi.

### 2.5.2 Informasi

Informasi merupakan data yang telah di proses atau diolah sehingga memiliki nilai tertentu bagi penggunanya. Keberadaan data sangatlah penting karena data merupakan bahan utama untuk didapatkannya suatu informasi yang dibutuhkan.

Informasi juga diartikan sebagai hasil proses, manipulasi, dan penataan dari sekelompok data yang memiliki sebuah nilai pengetahuan bagi penggunanya. Sumber dari suatu informasi adalah data. Data yang diolah melalui suatu model menjadi informasi, penerima kemudian menerima informasi tersebut, membuat suatu keputusan dan melakukan tindakan, yang berarti menghasilkan suatu tindakan yang lain yang akan membuat sejumlah data kembali. Data tersebut akan ditangkap sabagai input, diproses kembali lewat suatu model dan seterusnya membentuk suatu siklus. Siklus informasi ini dapat digambarkan sebagai berikut :




**Gambar 2.1 Siklus Informasi**

## 2.6 Data Portal Indonesia

**data.go.id** adalah portal resmi data terbuka Indonesia sebagai wujud operasionalisasi inisiatif Satu Data. Portal ini berisi data kementerian, lembaga pemerintahan, pemerintahan daerah, dan semua instansi lain yang terkait yang menghasilkan data yang berhubungan dengan Indonesia.Satu Data adalah sebuah inisiatif Pemerintah Indonesia untuk meningkatkan interoperabilitas dan pemanfaatan data pemerintah. Pemanfaatan data pemerintah tidak terbatas pada penggunaan internal antar instansi, tetapi juga sebagai bentuk pemenuhan kebutuhan data publik bagi masyarakat. [3]



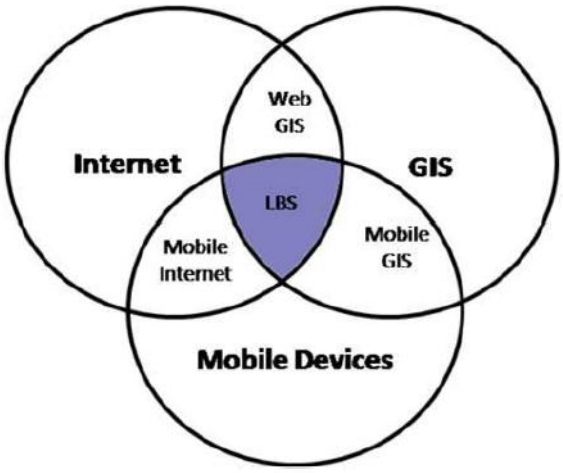

**Gambar 2.2 Data Indonesia Dalam Satu Portal**

## 2.7 GPS

*Global Positioning System* (GPS) merupakan sistem navigasi yang menggunakan satelit dalam penggunaannya. GPS terdiri dari tiga bagian yaitu sistem kontrol, satelit dan pengguna. Sistem kontrol adalah bagian yang mengontrol pergerakan satelit – satelit yang ada dan saling berinteraksi satu sama lain kemudian pengguna adalah alat navigasi yang digunakan seperti perangkat mobile yang kini sudah memiliki fitur GPS didalamnya. GPS biasanya digunakan untuk menunjukan suatu lokasi yang berada di permukaan bumi dengan tingkat akurasi yang cukup baik yaitu kurang dari 10 meter selama tidak ada benda padat yang dapat menghambat sinyal untuk mendapatkan lokasi pengguna.

## 2.8 Location Based Service (LBS)

*Location Based Service* (LBS) atau layanan berbasis lokasi adalah layanan informasi yang dapat diakses melalui *mobile device* dengan menggunakan *mobile network*, yang dilengkapi dengan kemampuan untuk memanfaatkan lokasi dari *mobile device* tersebut. LBS memungkinkan untuk melakukan komunikasi dan interaksi dua arah. LBS ini dapat digambarkan sebagai suatu layanan yang berada pada pertemuan tiga teknologi yaitu *Geographic Information System* (GIS), Internet *Service* dan *mobile device*, seperti pada gambar 2.3 : [4]




**Gambar 2.3 Teknologi Location Based Service**

### 2.8.1 Metode Penentu Lokasi

LBS merupakan layanan yang mampu bereaksi aktif pada suatu perubahan posisi sehingga dapat mendeteksi letak objek dan memberikan layanan sesuai dengan letak objek yang telah diketahui tersebut. Pada teknologi LBS berbasis jaringan seluler, penentuan posisi pada sebuah peralatan komunikasi ditentukan berdasarkan posisi relatif terhadap lokasi BTS (*Base Transceiver Station*). Dalam menentukan posisi dari sebuah perangkat mobile yang sedang aktif, secara umum terdapat tiga tingkat metode yang digunakan saat ini, diantaranya adalah sebagai berikut :

1. Metode *Basic Positioning* berbasis pada *Cell Identification* (Cell ID)

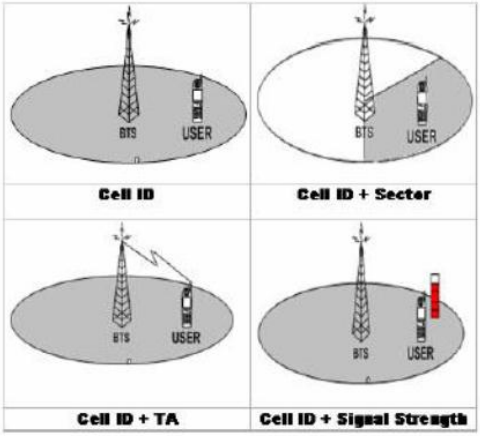
   Penentuan posisi didasarkan pada daerah geografis yang tercakup oleh sebuah *Cell* berhubungan dengan daerah cakupan dari sinyal radio. Ketika sebuah *handphone* terhubung secara aktif dengan sebuah *base station*, berarti *handphone* tersebut diasumsikan berada dalam *cell* dari *base station* tersebut. Untuk mengukur jarak dan arah *handset* dari *base station* tidak dapat diketahi dengan pasti, oleh karena itu untuk lebih meningkatkan lagi akurasi hasil pencarian pada metode ini seringkali dikombinasikan dengan metode lain seperti:

a. *Timing Advance* (TA)

Dengan menggunakan TA ini metode Cell ID akan ditambahkan sebuah fungsionalitas untuk menghitung *Round Trip Time* (RTT), yaitu waktu transmisi sebuah *frame* (dari *base station* ke *handphone*) dan waktu sebuah *frame* (dari *handphone* ke *station*). Dengan tambahan ini jarak antara *handphone* dan *base station* dapat ditentukan dengan keakuratan 50 m.

b. *Network Measurement Report* (NMR)

Berdasarkan pada besar kecilnya sinyal (*Receiver Signal Strength*) yang diterima *handphone* yang ada di suatu "*sector cell*", maka posisi itu dapat ditentukan dengan lebih akurat.

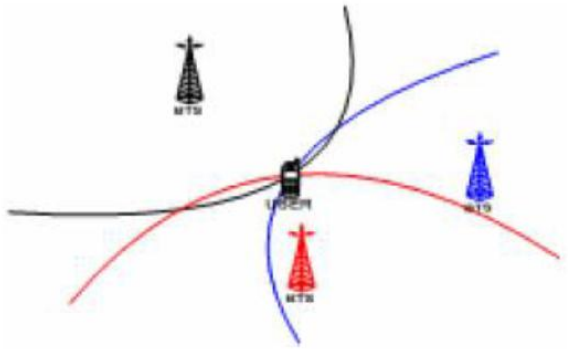



**Gambar 2.4 *Network Measurement Report***

**2.** Metode *Enhanced Posittioning*

Metode ini menggunakan pendekatan *Observe Time Difference* (OTD) dengan jaringan GSM yang sering digunakan seperti *Enhanced*-OTD (E-OTD). E-OTD adalah metode pencarian posisi yang berdasarkan pada waktu. Untuk menentukan posisi *relative*, sebuah handphone harus aktif terhadap tiga *base station* dan perlu ditentukan terlebih dahulu jarak *handphone* terhadap masing- masing *base station* berdasarkan waktu yang ditempuh oleh sebuah sinyal dari handphone ke masing – masing *base station*. Untuk menentukan posisi dari handphone yang sedang aktif

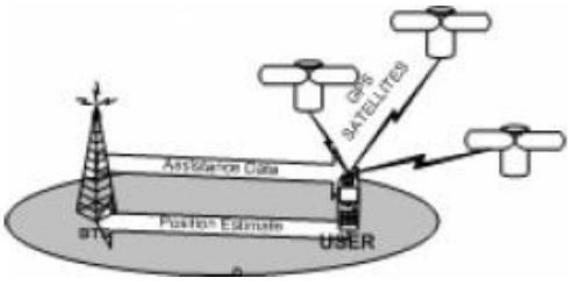
digunakan rumus matematika untuk triangulasi. Dengan metode ini akurasinya akan meningkat hingga ketilitian kurang dari 50 m.



**Gambar 2.5 *Enhanced Positioning***

3. Metode *Advance Positioning*

   Metode ini menggunakan *teknologi Assisted-Global Positioning System* (A-GPS). A-GPS ini merupakan metode yang berbasis pada waktu. Pada metode ini, akan dilakukan pengukuran waktu tiba dari sebuah sinyal yang dikirim dari tiga buah satelit GPS. Hal ini berarti handset harus memiliki fasilitas untuk mengakses GPS. A-GPS juga menghasilkan akurasi secara *vertical* dan estimasi jarak yang baik. Akurasinya pun kurang dari 10 m.




**Gambar 2.6 *Advance Positioning***

### 2.8.2 Komponen *Location Based Service*

Dalam menggunakan *Location Based Service* atau layanan berbasis lokasi terdapat beberapa elemen yang diperlukan, antara lain :

1. *Mobile Devices*

   Sebuah alat yang digunakan untuk meminta informasi yang dibutuhkan seperti perangkat *mobile smartphone* yang memiliki fasilitas navigasi.

2. *Communication Network*

   Jaringan selular yang mengirimkan data pengguna dan permintaan layanan.

3. *Positioning Component*

   Untuk dapat mengolah layanan harus menentukan lokasi pengguna. Posisi pengguna ini dapat diperoleh menggunakan jaringan komunikasi atau dengan menggunakan *Global Positioning System* (GPS).

4. *Service and Application Provider*

   Penyedia layanan pengguna selular yang bertanggung jawab untuk memproses suatu layanan.

5. *Data and Content Provider*

   Penyedia layanan informasi data yang dapat diminta oleh pengguna.

## 2.9 Latitude dan Longitude

Latitude dan longitude adalah sebuah koordinat yang ada di bumi dan digunakan untuk menentukan kedudukan kita di atas bumi.[5]

### 2.9.1 Latitude

Latitude adalah garis yang melintang dari kutub utara dan kutub selatan. Titik 0 adalah sudut ekuator, tanda + menunjukan arah ke atas menuju kutub utara, sedangkan tanda minus di koordinat Latitude menuju ke kutub selatan. Titik yang dipakai dari 0 ke 90 derajat arah kutub utara, dan 0 ke -90 derajat ke kutub selatan.

### 2.9.2 Longitude

Longitude adalah garis lintang. Angka dari sudut bundar bumi horisontal. Titik diawali dari 0 ke 180 derajat, dan - ke arah sebaliknya. Titik 0 dimulai dari garis negara Inggris. Mengarah ke Indonesia akan menjadi angka positif. Kebalikannya koordinat Longitude minus adalah arah kebalikan.

## 2.10 Geofencing

Geofence adalah parameter virtual untuk wilayah geografis di dunia nyata. Parameter virtual yang ditetapkan ini bisa menjadi statis ataupun dinamis. Salah satu contoh khas dari geofence dinamis adalah radius sekitar toko atau titik lokasi. geofence statis dapat di atur dari batas yang telah ditetapkan, seperti zona kehadiran di sekolah atau batas lingkungan. Kostum digital geofencing juga dapat digunakan, jika perangkat yang menyediakan layanan berbasis lokasi (LBS) pengguna masuk atau keluar geofencing, perangkat akan menerima pesan notifikasi. Notifikasi ini mungkin berisi informasi tentang lokasi dari perangkat. Notifikasi Geo-fence mungkin dikirimkan ke ponsel sebagai contoh meningkatkan setiap bentuk lain dari tindakan.

Kita bisa menggambarkan geo fencing sebagai LBS proaktif. Dalam banyak kasus sistem proaktif jauh lebih nyaman dari yang reaktif, di mana pengguna harus secara eksplisit meminta data untuk lokasi. Ada berbagai jenis lokasi Peristiwa untuk memperbaiki posisi GPS yang dapat diuji. Misalnya, apakah pengguna di dekat dengan tempat tujuan (POI) atau ke pengguna lain.

Geofence sekarang ini menjadi salah satu wajah baru di Layanan Berbasis Lokasi. Sekarang pengembangan asli untuk LBS sendiri lebih atau kurang selesai. Kita bisa melihat beberapa standar umum dan aplikasi (berpikir tentang berbagai implementasi layanan Tempat-tempat seperti Foursquare, Twitter places, Google place, Facebook place, dll) Semua dari mereka memudahkan Anda dalam berbagi lokasi info atau mendapatkan beberapa hal yang berada di dekatnya. Dan geo-fence membuka pintu untuk beberapa personalisasi. Anda bisa mendapatkan beberapa kustom disesuaikan dengan data ditempat yang benar, terutama ketika kedua aliran data dan tempat (area) untuk menerima itu dihasilkan secara dinamis. Pada kedua, geo-fence akan terintegrasi dalam waktu terdekat dengan sensor dan aplikasi M2M.[6]

## 2.11 Google *Maps*

Google *Maps* adalah sebuah layanan gratis dari Google tentang berbagai macam teknologi pemetaan. Google *Maps* menampilkan peta dunia secara digital sehingga lebih memudahkan ketika ingin mengetahui suatu lokasi tertentu. Google *Maps* menyediakan beberapa keunggulan seperti peta jalan, menunjukan rute untuk suatu perjalanan yang diinginkan baik itu melalui jalur mobil sampai pejalan kaki. Google maps ini dapat di akses di http://maps.google.com/.

## 2.12 Google *Maps* API

Google *Maps* API adalah sebuah layanan yang memungkinkan Google *Maps* agar dapat diintegrasikan ke dalam aplikasi yang memang membutuhkan fitur dari Google *Maps* ini. Google *Maps* API adalah sebuah *library Javascript*. Dengan menggunakan Google *Maps* API ini dapat memudahkan pengembang dalam membuat aplikasi yang membutuhkan teknologi peta di dalamnya dan tidak perlu memikirkan cara membuatnya secara manual karena dengan menggunakan Google *Maps* API ini sudah semua yang dibutuhkan terkait teknologi peta digital sudah cukup lengkap dan dapat dimanfaatkan untuk aplikasi yang membutuhkannya.

Saat ini Google *Maps* API telah mengalami perkembangan, hingga saat ini Google Maps API telah mencapai versi 3. Namun, Google *Maps* API versi 3 ini masih hanya bisa digunakan oleh pemrograman berbasis web dengan menggunakan *Javascript*. Untuk penggunaan Google *Maps* API pada perangkat *mobile* seperti misalnya android Google Maps API yang bisa digunakan saat ini telah mencapai versi 2. Perkembangan pada Google Maps Versi 2 ini adalah dengan memiliki fitur – fitur yang lengkap dari versi sebelumnya salah satunya adalah dapat menampilkan peta dengan sudut yang diingkan dan berupa 3D *view*. Selain itu pada Google *Maps* API terdapat beberapa pilihan model peta yang ditampilkan seperti *Roadmap*, *Satellite*, *Terrain* dan *Hybrid*.

## 2.13 *Unified Modeling Language* (UML)

UML singkatan dari *Unified Modeling Language* yang berarti bahasa pemodelan standard. Michael Jesse Chonoles [8] mengatakan sebagai bahasa, berarti UML memiliki sintak dan semantik. Ketika kita membuat model menggunakan konsep UML ada aturan-aturan yang harus diikuti. Bagaimana elemen pada model-model yang kita buat berhubungan satu dengan yang lainnya harus mengikuti standar yang ada. UML bukan hanya sekedar diagram, tetapi juga menceritakan konteksnya.

UML diaplikasikan untuk maksud tertentu, biasanya antara lain untuk :

1. Merancang perangkat lunak
2. Sarana komunikasi antara perangkat lunak dengan proses bisnis.
3. Menjabarkan sistem secara rinci untuk analisa dan mencari apa yang diperlukan sistem.
4. Mendokumentasi sistem yang ada, proses-proses dan organisasinya.

### 2.13.1 Diagram-Diagram UML

Beberapa literatur menyebutkan bahwa UML menyediakan sembilan jenis diagram, yang lain menyebutkan delapan karena ada beberapa diagram yang digabung, misalnya diagram komunikasi, diagram urutan dan diagram pewaktuan digabung menjadi diagram interaksi. Namun demikian model-model itu dapat dikelompokan berdasarkan sifatnya yaitu statis atau dinamis. Jenis diagram itu antara lain :

1. Diagram Kelas. Bersifat statis. Diagram ini memperlihatkan himpunan kelas-kelas, antarmuka-antarmuka, kolaborasi-kolaborasi, serta relasi-relasi. Diagram ini umum dijumpai pada pemodelan sistem berorientasi objek. Meskipun bersifat statis, sering pula diagram kelas memuat kelas-kelas aktif.
2. Diagram Paket (*Package Diagram*). Bersifat statis. Diagram ini memperlihatkan kumpulan kelas-kelas, merupakan bagian dari diagram komponen.
3. Diagram Use-Case. Bersifat statis. Diagram ini memperlihatkan himpunan use-case dan aktor-aktor (suatu jenis khusus dari kelas).

Diagram ini terutama sangat penting untuk mengordinasikan dan memodelkan perilaku suatu sistem yang dibutuhkan serta diharapkan pengguna.

4. Diagram interaksi dan sequence (urutan). Bersifat dinamis. Diagram urutan adalah diagram interaksi yang menekankan pada pengiriman pesan dalam suatu waktu tertentu.
5. Diagram Komunikasi (*Communication Diagram*). Bersifat dinamis. Diagram sebagai pengganti diagramkolaborasi UML 1.4 yang menekankan organisasi struktural dari objek-objek yang menerima serta mengirim pesan.
6. Diagram Statechart (*Statechart Diagram*). Bersifat dinamis. Diagram status memperlihatkan keadaan-keadaan pada sistem, memuat status (state), transisi, kejadian serta aktifitas. Diagram ini terutama penting untuk memperlihatkan sifat dinamis dari antarmuka (interface), kelas, kolaborasi dan terutama penting pada pemodelan sistem-sistem yang reaktif.
7. Diagram Aktivitas (*Activity Diagram*). Bersifat dinamis. Diagram aktivitas adalah tipe khusus dari diagram status yang memperlihatkan aliran dari suatu aktivitas ke aktivitas lainnya dalam suatu sistem. Diagram ini terutama penting dalam pemodelan fungsi-fungsi suatu sistem dan memberi tekanan pada aliran kendali antar objek.
8. Diagram komponen (*Component Diagram*). Bersifat statis. Diagram komponen ini memperlihatkan organisasi serta kebergantungan sistem/perangkat lunak pada komponen-komponen yang telah ada sebelumnya. Diagram ini berhubungan dengan diagram kelas dimana komponen secara tipikal dipetakan kedalam satu atau lebih kelas-kelas, antaramuka-antarmuka serta kolaborasi-kolaborasi.
9. Diagram Deployment (*Deployment Diagram*). Bersifat statis. Diagram ini memperlihatkan konfigurasi saat aplikasi dijalankan (*run-time*). Memuat simpul-simpul beserta komponen-komponen yang ada di dalamnya. Diagram deployment berhubungan erat dengan diagram

komponen dimana diagram ini memuat satu atau lebih komponen-komponen. Diagram ini sangat berguna saat aplikasi kita berlaku sebagai aplikasi yang dijalankan pada banyak mesin (*distributed computing*).

Kesembilan diagram ini tidak mutlak harus digunakan dalam pengembangan perangkat lunak, semuanya dibuat sesuai dengan kebutuhan. Pada UML dimungkinkan kita menggunakan diagram-diagram lain (misalnya *data flow diagram*, *Entity Relationship diagram* dan sebagainya) [7].

## 2.14 Android

Android merupakan sebuah sistem operasi telepon seluler dan komputer tablet layar sentuh (*touch screen*) berbasis Linux. Android menyediakan *platform* terbuka bagi para pengembang untuk menciptakan aplikasi mereka sendiri untuk digunakan oleh bermacam peranti bergerak. Android merupakan generasi baru platform mobile, platform yang memberikan pengembangan untuk melakukan pengembangan sesuai dengan yang diharapkan. Pengembangan aplikasi Android diperbolehkan untuk mendistribusikan aplikasi mereka di bawah skema lisensi apapun yang mereka inginkan. Pengembang memiliki beberapa pilihan ketika membuat aplikasi yang berbasis android. Sebagian besar pengembang menggunakan eclipse yang tersediasecara bebas untuk merancang dan mengembangkan aplikasi android, karena memiliki android plug-in yang tersedia untuk memfasilitasi pengembangan android. Selain itu, eclipse juga mendapatdukungan langsung dari google untuk menjadi IDE pengembangan aplikasi Android, ini terbukti dengan adanya penambahan plug-in untuk *eclipse* untuk membuat project android dimana source software langsung dari situs resminya google [8].

### 2.14.1 Kelebihan Android

Sudah banyak *platform* untuk perangkat selular saat ini, termasuk didalamnya Symbian, iPhone, Windows Mobile, BlackBerry, Java Mobile Edition, Linux *Mobile* (LiM0), 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 *virtualDalvik*, sehingga kode program portabel antara ARM, X86, dan arsitekturlainnya. Sama halnya dengan dukungan masukan seperti penggunaan *Keyboard*, layar sentuh, *trackball* dan resolusi layar semua dapat disesuaikan dengan program.

### 2.14.2 Arsitektur Android

Secara garis besar, arsitektur android terdiri atas *Applications* dan *Widgets, Applications Frameworks, Libraries, Android Run Time*, dan *Linux Kernel* [8].

a. *Applications dan Widgets* merupakan *layer* (lapis) dimana kita berhubungan dengan aplikasi kita.
b. *Applications Framework* merupakan *Open Development Platform* yang ditawarkan Android untuk dapat ditawarkan Android untuk dapat dikembangkan guna membangun aplikasi. Pengembang memiliki akses penuh menuju *API Framework* seperti yang dilakukan oleh aplikasi kategori inti. Komponen-komponen yang termasuk dalam *Applications Framework* adalah sebagai berikut :
    1. *Views* yang kaya akan dan *extensible* yang dapat digunakan untuk membangun aplikasi, termasuk list, grids, kotak teks, tombol dan bahkan sebuah *embeddable web*.
    2. *Content Provider* yang memungkinkan aplikasi untuk mengakses data (*seperti dari daftar kontak telepon*) atau membagi data tersebut.
    3. *Resource Manager* yang menyediakan akses ke kode non sumber lokal seperti string, gambar, dan tata letak file.
    4. *Notification Manager* yang memungkinan semua aplikasi menampilkan alert yang bias dikustomisasi di dalam status bar.
    5. *Activity Manager* yang mengelola siklus hidup aplikasi dan menyediakan navigasi umum backstack.

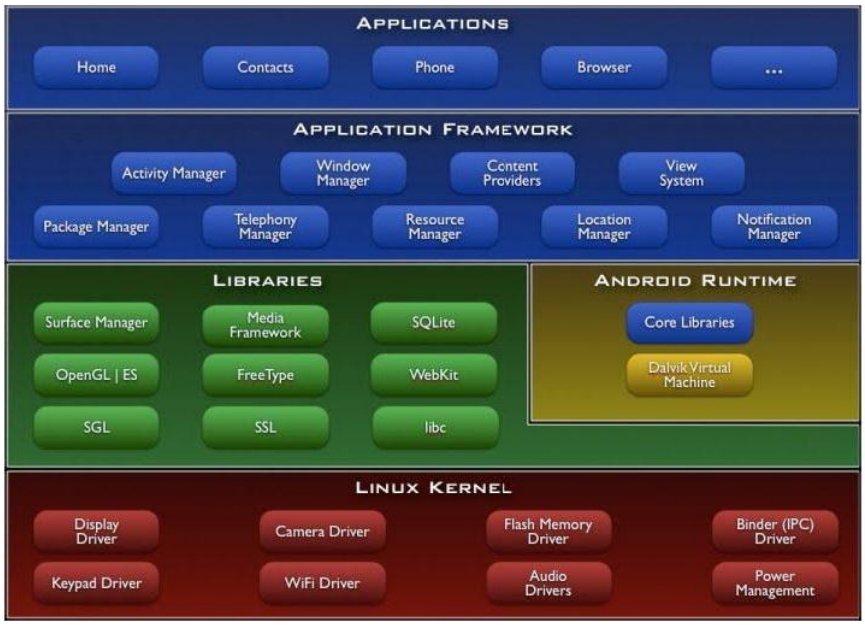
*c. Libraries* merupakan layer dimana fitur-fitur Android berada. Android menyertakan libraries C / C+ + yang digunakan oleh berbagai komponen dari sistem android. Kemampuan ini disediakan kepada *Developer* aplikasi melalui *Framework* aplikasi android.

1. Sistem *C Library* merupakan variasi dari implementasi BSD-berasal pelaksanaan sistem standar *C Library (libe)*, sesuai untuk perangkat *embedded* berbasis Linux.
2. Media *Library* merupakan *library* yang mendukung pemutaran rekaman dan popular banyak format audio dan video, serta file gambar.
3. *Graphic Library* termasuk didalamnya SGL dan OpenGL untuk tampilan 2D dan 3D.
4. *Surface Manager* merupakan *library* untnuk mengelola akses ke subsistem layar.
5. *Lib Webcore* merupakan mesin *web* modern yang *powerfull* yang baik untuk *browser* android.
6. *FreeType* mendukung *bitmap* dan *vector font rendering*.
7. *SQLite* merupakan mesin *database* yang kuat dan ringan, dan penghubung tersedia untuk semua aplikasi.

d. *Android Run Time* merupakan Layer yang membuat aplikasi Android dapat dijalankan, dimana dalam prosesnya menggunakan implementasi Linux. Android terdiri dari satu set *core libraries* yang menyediakan sebagian besar fungsi yang sama dengan yang terdapat pada *core* yang *libraries* bahasa pemrograman Java.

*e. Linux Kernel* Merupakan layer inti dari sistem operasi Android berada. Berisi file-file sistem yang mengatur sistem *processing, memory, resources, drivers,* dan sistem-sistem operasi android lainnya. Android bukan Linux, akan tetapi dibangun diatas Linux Kernel yaitu versi 2.6 sehingga kehandalannya dapat dipercaya. Untuk inti sistem *service Linux* yang digunakan seperti keamanan, manajemen memori, proses manajemen, *network* dan *driver model*. *Linux* menyediakan *Driver* layar, Kamera,

*Keypad*, *WiFi*, *Flash Memory*, Audio dan IPC (*Interprocess Communication*) untuk mengatur keamanan. Kernel juga bertindak sebagai lapisan abstrak antara *hardware* dan *software stack*-nya. Untuk lebih jelasnya lihat gambar arsitektur Android berikut:




**Gambar 2.7 Lapisan Arsitektur Aplikasi Android [8]**

## 2.15 Android dan GPS

Dewasa ini, teknologi berkembang dengan pesat. Dulu ponsel hanya sekedar digunakan untuk menelpon dan SMS saja. Sekarang ponsel sudah menjelma menjadi kotak kecil ajaib yang serba bisa. Salah satunya adalah ponsel dengan sistem operasi Android. Dengan standarisasi fitur dan hardware yang dimiliki, menjadikan ponsel Android ponsel canggih yang disukai banyak orang. Tidak lagi canggih karena adanya fitur MMS, radio, atau internet berkecepatan tinggi tapi juga karena ditanamkannya fitur teknologi satelit di dalamnya. Ya, perangkat GPS receiver yang dulu besar dan eksklusif, sekarang sudah bisa dimiliki dengan “hanya” membeli sebuah ponsel.

Dengan ponsel berteknologi satelit (GPS), banyak hal yang bisa dilakukan. Ingin melihat di mana posisi user sekarang dalam sebuah peta? Mengambil foto/video yang sudah dilengkapi dengan data koordinat? Ingin tahu jalur olahraga bersepeda yang sudah pernah user lalui? Atau ingin tahu dimana posisi ponsel yang hilang dicuri orang? Semua itu bisa dilakukan dengan ponsel Android yang memiliki fitur GPS. Tidak hanya itu, user juga dapat pergi ke tempat wisata tertentu dengan dipandu gambar dan suara dari sebuah ponsel! Bahkan lebih jauh lagi, GPS dapat digunakan untuk membantu memberikan peringatan dini terhadap terjadinya bencana alam.

Sekarang ini banyak sekali pengembang - pengembang aplikasi untuk sistem operasi Android termasuk aplikasi-aplikasi GPS. Yang menyenangkan aplikasi-aplikasi tersebut jenisnya beragam dan jumlahnya pun banyak. [9]

## 2.16 PHP

PHP (*Hypertext Preprocessor*) merupakan bahasa yang hanya dapat berjalan pada server yang hasilnya dapat ditampilkan pada klien. Dalam mengeksekusi kode PHP pada sisi server (disebut *server side*) berbeda dengan mesin maya Java yang mengeksekusi program pada sisi klien (*client side*). Proses eksekusi kode PHP yang disisipkan pada halaman HTML.

PHP merupakan bahasa standar yang digunakan dalam dunia website, PHP adalah bahasa pemrograman yang berbentuk script yang diletakkan didalam server web. Sekitar tahun 1994, Rasmus Lerdorf telah meletakkan bersama Per-*script* untuk membuat siapa yang telah melihat resumenya terkesan. Kemudian sedikit demi sedikit user mulai menyukai script ini.

## 2.17 JavaScript Object Notation (JSON)

JSON (*JavaScript Object Notation*) adalah format pertukaran data yang ringan, mudah dibaca dan ditulis oleh manusia, serta mudah diterjemahkan dan dibuat (*generate*) oleh komputer. Format ini dibuat berdasarkan bagian dari Bahasa Pemprograman JavaScript, Standar ECMA-262 Edisi ke-3 - Desember 1999. JSON merupakan format teks yang tidak bergantung pada bahasa pemprograman apapun karena menggunakan gaya bahasa yang umum digunakan oleh programmer keluarga C termasuk C, C++, C#, Java, JavaScript, Perl, Python dll. Oleh karena sifat-sifat tersebut, menjadikan JSON ideal sebagai bahasa pertukaran-data [10]

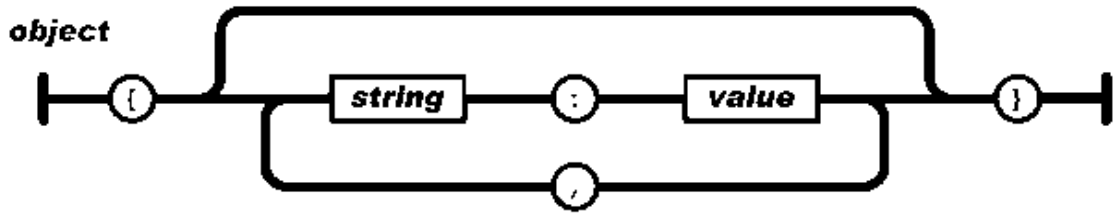
JSON terbuat dari dua struktur:

a. Kumpulan pasangan nama/nilai. Pada beberapa bahasa, hal ini dinyatakan sebagai objek (*object*), rekaman (*record*), struktur (*struct*), kamus (*dictionary*), tabel hash (*hash table*), daftar berkunci (*keyed list*), atau associative array.
b. Daftar nilai terurutkan (*an ordered list of values*). Pada kebanyakan bahasa, hal ini dinyatakan sebagai larik (*array*), vektor (*vector*), daftar (*list*), atau urutan (*sequence*).

Struktur-struktur data ini disebut sebagai struktur data universal. Pada dasarnya, semua bahasa pemprograman moderen mendukung struktur data ini dalam bentuk yang sama maupun berlainan. Hal ini pantas disebut demikian karena format data mudah dipertukarkan dengan bahasa-bahasa pemprograman yang juga berdasarkan pada struktur data ini.

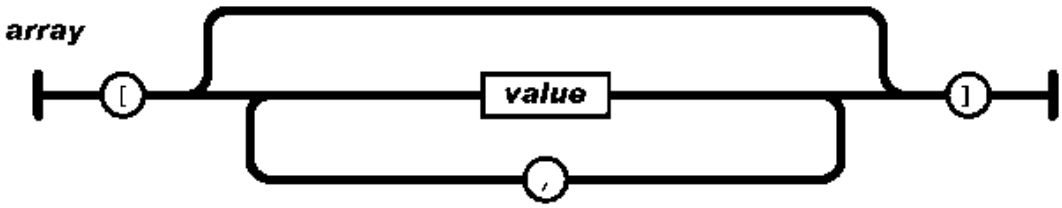
JSON menggunakan bentuk sebagai berikut:

**Objek** adalah sepasang nama/nilai yang tidak terurutkan. Objek dimulai dengan { (kurung kurawal buka) dan diakhiri dengan } (kurung kurawal tutup). Setiap nama diikuti dengan : (titik dua) dan setiap pasangan nama/nilai dipisahkan oleh , (koma).



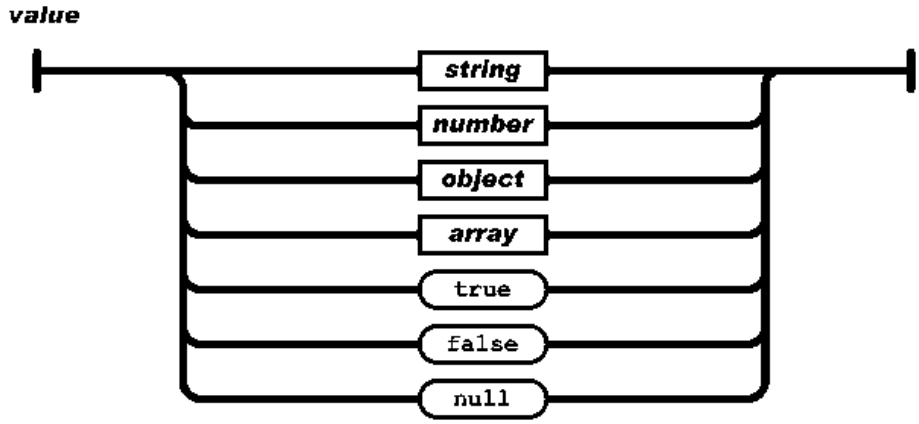

**Gambar 2.8 Objek JSON**

**Larik** adalah kumpulan nilai yang terurutkan. Larik dimulai dengan [ (kurung kotak buka) dan diakhiri dengan ] (kurung kotak tutup). Setiap nilai dipisahkan oleh , (koma).



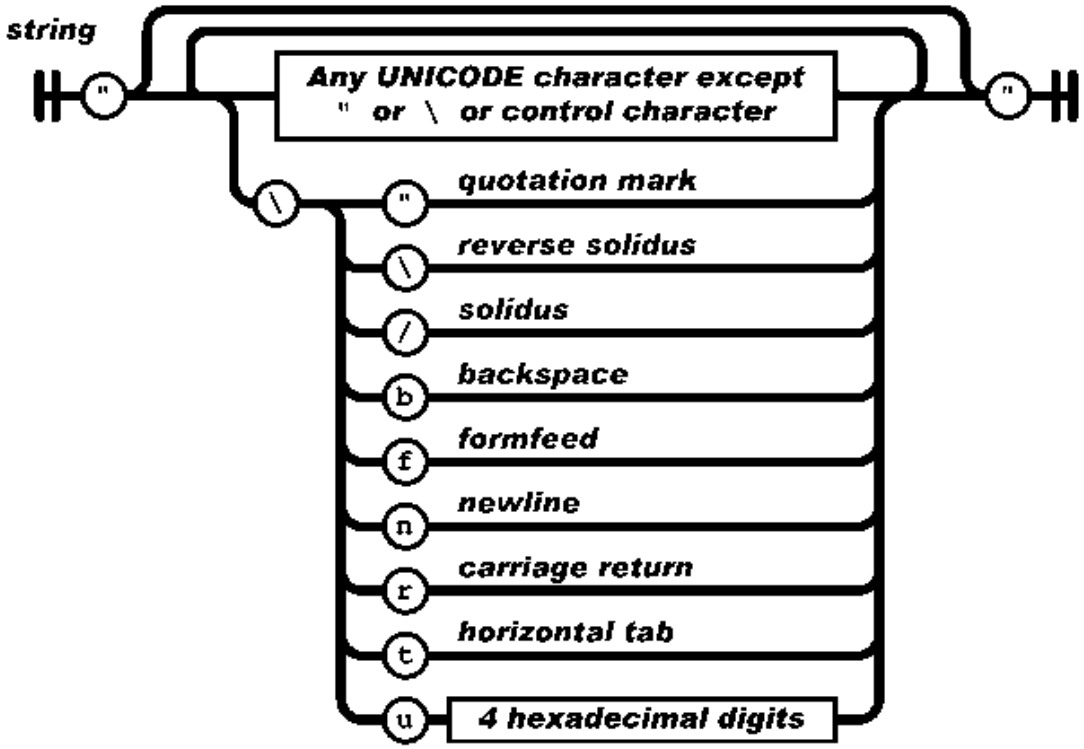

**Gambar 2.9 Array JSON**

**Nilai** (value) dapat berupa sebuah string dalam tanda kutip ganda, atau angka, atau true atau false atau null, atau sebuah objek atau sebuah larik. Struktur-struktur tersebut dapat disusun bertingkat.



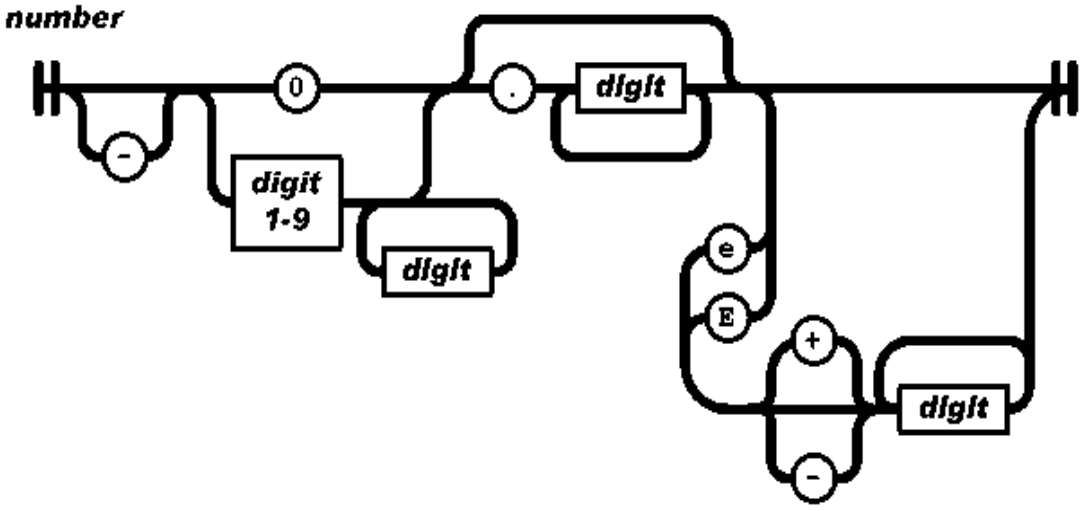

**Gambar 2.10 Value JSON**

**String** adalah kumpulan dari nol atau lebih karakter Unicode, yang dibungkus dengan tanda kutip ganda. Di dalam string dapat digunakan backslash escapes "\" untuk membentuk karakter khusus. Sebuah karakter mewakili karakter tunggal pada string. String sangat mirip dengan string C atau Java.




**Gambar 2.11 String JSON**

**Angka** adalah sangat mirip dengan angka di C atau Java, kecuali format oktal dan heksadesimal tidak digunakan.




**Gambar 2.12 Number JSON**

Spasi kosong (whitespace) dapat disisipkan di antara pasangan tanda-tanda tersebut, kecuali beberapa detil encoding yang secara lengkap dipaparkan oleh bahasa pemprograman yang bersangkutan.

## 2.18 Java

Java menurut definisi dari Sun adalah nama untuk sekumpulan teknologi untuk membuat dan menjalankan perangkat lunak pada komputer standalone ataupun pada lingkungan jaringan. Java dikembangkan pada bulan Agustus 1991, dengan nama semula Oak. Pada Januari 1995, karena nama Oak dianggap kurang komersial, maka diganti menjadi Java. Pada Desember 1998, Sun memperkenalkan nama "Java 2" (J2) sebagai generasi kedua dari java platform. Konvensi nama baru ini diterapkan untuk semua edisi Java yaitu Standard Edition (J2SE), Enterprise Edition (J2EE), dan Micro Edition (J2ME) [11].

Ada tiga platform Java yang masing-masing diarahkan untuk tujuan tertentu dan untuk lingkungan komputasi yang berbeda-beda:

a. Standard Edition (J2SE) : J2SE merupakan inti dari bahasa pemrograman Java. J2SE didesain untuk jalan pada komputer desktop dan komputer workstations.

b. Enterprise Edition (J2EE): Dengan built-in mendukung untuk servlets. JSP, dan XML, edisi ini ditujukan untuk aplikasi berbasis server.

c. Micro Edition (J2ME) : Didesain untuk piranti dengan memori terbatas, layar display terbatas dan power pemrosesan yang juga terbatas.

### 2.18.1 Kelebihan Java

Adapun beberapa kelebihan dari bahasa pemrograman Java adalah sebagai berikut [11] :

**1. Sederhana dan Ampuh**

Java dirancang untuk mudah dipelajari, terutama bagi programmer yang telah mengenal C/C++ akan mudah sekali untuk berpindah ke Java. Pemakai dapat belajar membuat program dengan Java secara cepat jika telah memahami konsep dasar pemrograman berorientasi objek. Java tidak memiliki hal-hal yang mengejutkan dan aneh. Java memberi anda kemampuan untuk menuangkan semua ide, karena bahasa pemrograman ini bukan merupakan scripting language (bahasa naskah) yang menghilangkan kemampuan kita untuk berinovasi, tetapi dengan cara berorientasi objek yang mudah dan jelas.

**2. Aman**

Java dirancang sebagai bahasa pemrograman yang handal dan aman. Aplikasi-aplikasi yang dibangun dengan bahasa Java sangat handal dengan manajemen memori yang bagus. Aplikasi Java juga dikenal sangat secure, yaitu kasus-kasus seperti buffer everflow yang umumnya menjadi lubang keamanan aplikasi-aplikasi berbasis C/C++ tidak terjadi di Java, karena pengaturan securitynya yang bagus.

**3. Berorientasi Objek**

Paradigma pemrograman berorientasi objek merupakan paradigma pemrograman masa depan. Java merupakan bahasa pemrograman berorientasi objek. Java bukan turunan langsung dari bahasa pemrograman manapun, juga sama sekali tidak kompetibel dengan

semuanya. Java memiliki keseimbangan, Java secara cepat jika telah memahami konsep dasar pemrograman berorientasi objek. Java tidak memiliki hal-hal yang mengejutkan dan aneh. Java memberi anda kemampuan untuk menuangkan semua ide, karena bahasa pemrograman menyediakan mekanisme peng-class-an sederhana, dengan model antar muka dinamik yang intuitif hanya jika diperlukan.

4. **Kokoh**

   Java membatasi anda dari beberapa hal kunci supaya anda dapat menemukan kesalahan lebih cepat saat mengembangkan program. Java langsung memeriksa program saat anda menuliskannya, dan sekali lagi ketika program di jalankan. Karena Java adalah bahasa yang sangat ketat dalam hal tipe data dan deklarasi, banyak kesalahan umum terjadi saat kompilasi. Hal ini akan lebih menghemat waktu jika dibandingkan dengan keharusan menjalankan program terlebih dahulu dan memeriksa semua bagian program untuk melihat ketidakcocokan dinamis selama program berjalan. Ini adalah contoh di mana Java lebih luwes dan kokoh dari beberapa bahasa lain, tetapi dengan imbalan yang layak untuk kelebihan itu.

5. **Interaktif**

   Java memiliki beberapa kemampuan yang memungkinkan program melakukan beberapa hal pada saat bersamaan, tanpa harus kesulitan menangani proses yang akan terjadi selanjutnya. Jalinan program-program Java yang mudah digunakan memungkinkan kita untuk memikirkan pembuatan perilaku khusus, tanpa harus mengintegrasikan perilaku tersebut dengan model pemrograman global yang mengatur perulangan kejadian.

6. **Netral terhadap Berbagai Arsitektur**

   Java telah mengambil beberapa keputusan yang sulit dalam pembuatan bahasa Java dan bagaimana program dijalankan, jadi anda dapat sepenuhnya percaya “tulis sekali, jalan di mana saja, kapan saja, selamanya”.

**7. Terintrepentasi dan Berkinerja Tinggi**

Java dilengkapi keajaiban lintas platform yang luar biasa dengan kompilasi ke dalam representasi langsung yang disebut kode-byte Java (Java byte-code), yang dapat diterjemahkan oleh sistem manapun yang memilki program Java didalamnya. Java, bagaimanapun dirancang untuk tetap berkinerja baik pada CPU yang tidak terlalu kuat. Walaupun Java merupakan bahasa terinterpretasi, kodekode Java telah dirancang dengan hati-hati sehingga mudah diterjemahkan ke dalam bahasa asli suatu mesin untuk menghasilkan kinerja yang tinggi.

Sistem program Java yang melakukan optimasi tepat waktu tersebut tidak kehilangan keuntungan dari program yang netral terhadap platform. “lintas platform berkinerja tinggi” bukan sekedar omong-kosong. Dalam aplikasi Java (*.class) merupakan Java bytecode yang berjalan di atas jvm (Java Virtual Machine), yang kemudian jvm-lah yang akan menginterpresentasikan kode-kode tersebut ke kode native atau kode mesin dari arsitektur yang bersangkutan. Hal sangat menarik karena urusan arsitektur mesin bukan jadi masalah bagi programmer tapi menjadi urusan kompiler pada bahasa pemrograman Java.