BAB II
DASAR TEORI

2.1 REKAYASA PERANGKAT LUNAK

2.1.1 Pengertian Rekayasa Perangkat Lunak

Rekayasa perangkat lunak adalah sebuah ilmu yang mencakup segala hal secara luas yang berhubungan dengan proses pengembangan perangkat lunak mulai dari tahap perancangan hingga tahapan implementasi sehingga alur dari perangkat lunak dapat berlangsung secara efisien dan tepat. Maka, rekayasa perangkat lunak adalah sebuah dasar utama dari setiap pengembangan perangkat lunak.

2.1.2 Metode Rekayasa Perangkat Lunak

Metode rekayasa perangkat lunak merupakan pendekatan-pendekatan terstruktur terhadap notasi, model, aturan, saran perancangan sistem dan panduan proses. Komponen-komponen metode rekayasa perangkat lunak adalah :

a. Deskripsi model sistem : merupakan representasi dari notasi untuk mendefiniskan sebuah model sistem.
b. Aturan : merupakan batasan yang selalu berlaku bagi pemodelan sistem.
c. Rekomendasi : merupakan saran yang membentuk praktek perancangan yang baik.
d. Panduan proses : merupakan penjelasan kegiatan untuk mengembangkan model sistem dan organisasi.

Pada Rekayasa Perangkat Lunak terdapat *Case, Case* (*Computer-Aided Software Engineering*) merupakan berbagai macam program yang digunakan untuk medukung semua kegiatan perangkat lunak seperti analisis persyaratan, pemodelan sistem, *debugging*, dan *testing*.

2.1.3 *UNIFIED MODELING LANGUANGE* (UML)

2.1.3.1 Pengertian UML

Defenisi dari UML adalah suatu himpunan struktur dan teknik yang digunakan untuk pemodelan dan desain program berorientasi obyek. UML adalah suatu proses

untuk mengembangkan sistem OOP (*Object Oriented Programming)* dan sekelompok *tool* untuk mendukung pengembangan sistem OOP tersebut. UML merupakan dasar bagi desain pemrograman berorientasi obyek.

Perancangan perangkat lunak (program-program aplikasi), merupakan salah satu pekerjaan yang menuntut keahlian dan keterampilan manusia, yaitu kemampuan dalam hal analisis dan perancangan, kemampuan teknis pemrograman, serta kemampuan pengelolaan.

Pada umumnya untuk merancang *software* (perangkat lunak) berawal dari upaya pemahaman permasalahan secara detail (*analysis*), perancangan perangkat lunak yang akan dikerjakan (*design*), kemudian dilanjutkan dengan implementasi (penerapan penulisan hasil perancangan dengan bahasa pemrograman tertentu), dan diakhiri dengan pengujian (*testing*), sehingga perangkat lunak benar-benar sesuai dengan kebutuhan dan harapan pengguna.

2.1.3.2 Diagram UML

UML terdiri dari pengelompokan diagram-diagram sistem menurut aspek atau sudut pandang tertentu. Diagram yang dimaksud adalah diagram yang menggambarkan permasalahan maupun solusi dari *Object Oriented* yang digambarkan dalam bentuk model.

UML terdiri atas beberapa diagram, yaitu [2] :

1. Diagram *Use Case*

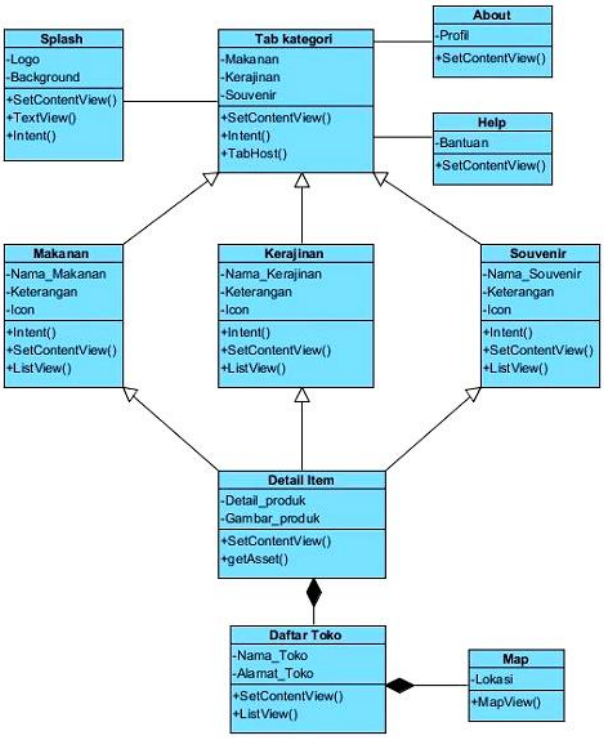
   Diagram *Use Case* menggambarkan sekelompok *use case* dan aktor yang disertai dengan hubungan diantaranya. Aktifitas yang dilakukan oleh suatu sistem dari sudut pandang pengamatan luar di mana pada diagram ini lebih membahas apa yang dilakukan sistem, bukan bagaimana melakukannya.

   Diagram *use case* ini menjelaskan dan menerangkan kebutuhan (*requirement*) yang diinginkan/dikehendaki pengguna, serta sangat berguna dalam menentukan struktur organisasi dan model dari pada sebuah sistem. Diagram ini dekat kaitannya dengan *scenario*, di mana pengguna melakukan interaksi dengan sistem, seperti pada Gambar 2.1.

Gambar 2.1 Contoh Diagram *Use Case* [2]

2. Diagram *Class*

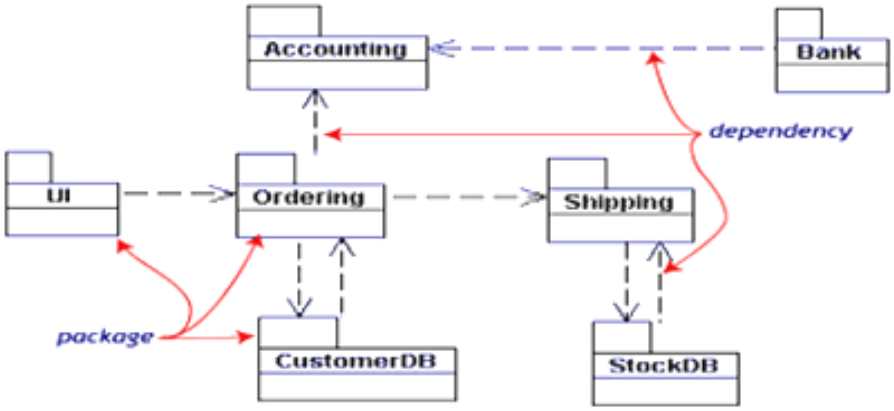
Diagram *Class* memberikan pandangan secara luas dari suatu sistem dengan menunjukan kelas-kelas dan hubungan kelas tersebut. Memperlihatkan struktur statis dari kelas *actual* didalam sistem. Diagram *Class* bersifat statis; menggambarkan hubungan apa yang terjadi, bukan apa yang terjadi jika kelas-kelas berhubungan, seperti pada Gambar 2.2.



Gambar 2.2 Contoh Diagram *Class* [3]

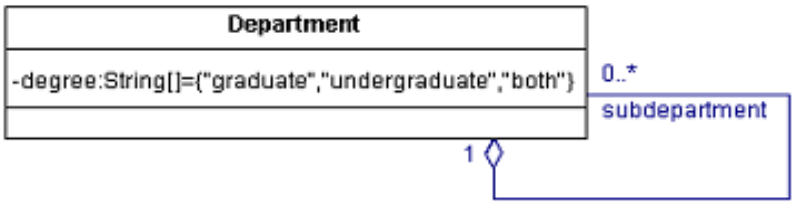
3. Diagram *Package* dan *Object*

Pada diagram Class, untuk mengatur pengorganisasian yang kompleks, dapat dilakukan dengan mengelompokkan kelas-kelas berupa *package* (paket-paket). *Package* adalah kumpulan elemen-elemen logika UML. Pada Gambar 2.3, contoh mengenai model bisnis dengan pengelompokan kelas-kelas dalam bentuk paket-paket.



Gambar 2.3 Contoh Diagram *Package* [2]

Ada jenis khusus dari diagram *Class* yaitu diagram *Object*, merupakan varian dari kelas diagram yang memperlihatkan lebih detail banyaknya obyek yang mengintantiasi (*instances*) kelas. Kegunaannya untuk penjelasan dengan relasi yang sulit, khususnya relasi rekursif, seperti pada Gambar 2.4, menunjukkan bahwa 'department' dapat mengandung banyak 'department' yang lain.



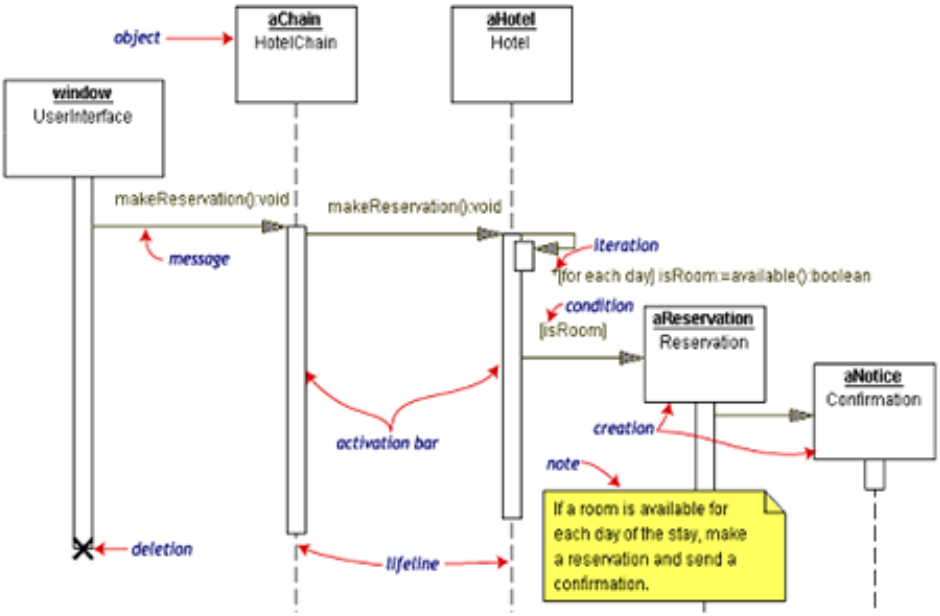
Gambar 2.4 Contoh Diagram *Object* [2]

4. Diagram *Sequence*

Diagram *Class* dan diagram *Object* merupakan suatu gambaran model statis, namun ada juga yang bersifat dinamis, seperti Diagram *Interaction.* Diagram *Sequence* merupakan salah satu diagram *Interaction* yang menjelaskan bagaimana suatu operasi itu dilakukaan, seperti *message* (pesan) apa yang dikirim dan kapan pelaksanaannya.

Diagram ini memperlihatkan kolaborasi dinamik antara obyek-obyek dengan suatu urutan pesan (*a sequence of message*) antar obyek tersebut. Diagram ini diatur berdasarkan waktu. Obyek-obyek yang berkaitan dengan proses berjalannya operasi diurutkan dari kiri ke kanan berdasarkan waktu terjadinya dalam pesan yang terurut. Pada Gambar 2.5, menjelaskan diagram pembuatan Hotel

Reservation, dimana obyek yang mengawali urutan *message* adalah '*aReservation Window*'.



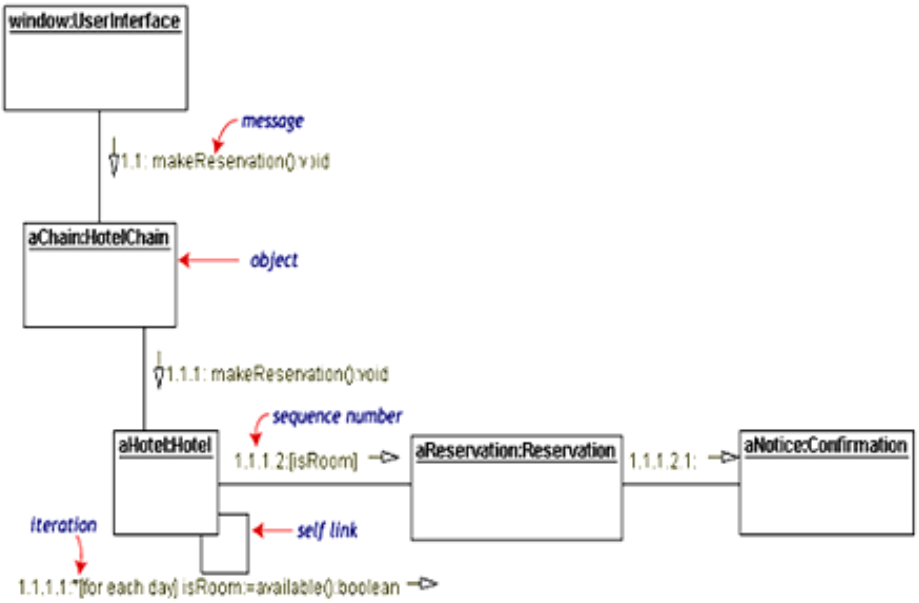
Gambar 2.5 Contoh Diagram *Sequence* [2]

5. Diagram *Collaboration*

Diagram *Collaboration* juga merupakan diagram *interaction*, yang memperlihatkan kolaborasi dinamika antar obyek tanpa memperhatikan aspek waktu. Diagram membawa informasi yang sama dengan diagram *Sequence*, tetapi lebih memusatkan atau memfokuskan pada kegiatan obyek dari waktu pesan itu dikirimkan.

Kotak kegiatan obyek diberi label dengan nama kelas atau obyek (atau keduanya). Nama kelas dibatasi dengan *colons* (titik dua) (:).

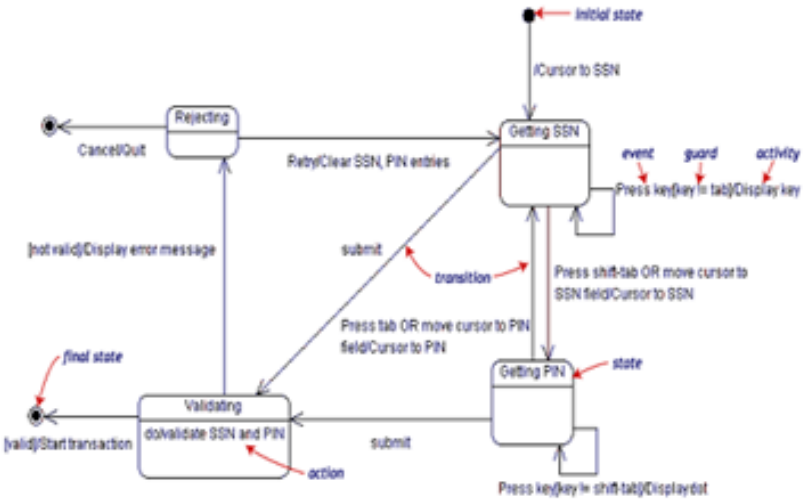
Setiap pesan pada diagram *Collaboration* mempunyai angka yang terurut. Pesan yang tingkatannya tertinggi adalah angka 1. Pesan yang berada pada tingkat yang sama memiliki *prefix* yang sama, namun *suffix* berbeda bergantung pada posisinya; hanya untuk angka 1, 2, dan seterusnya, seperti pada Gambar 2.6.



Gambar 2.6 Contoh Diagram *Collaboration* [2]

6. Diagram *StateChart*

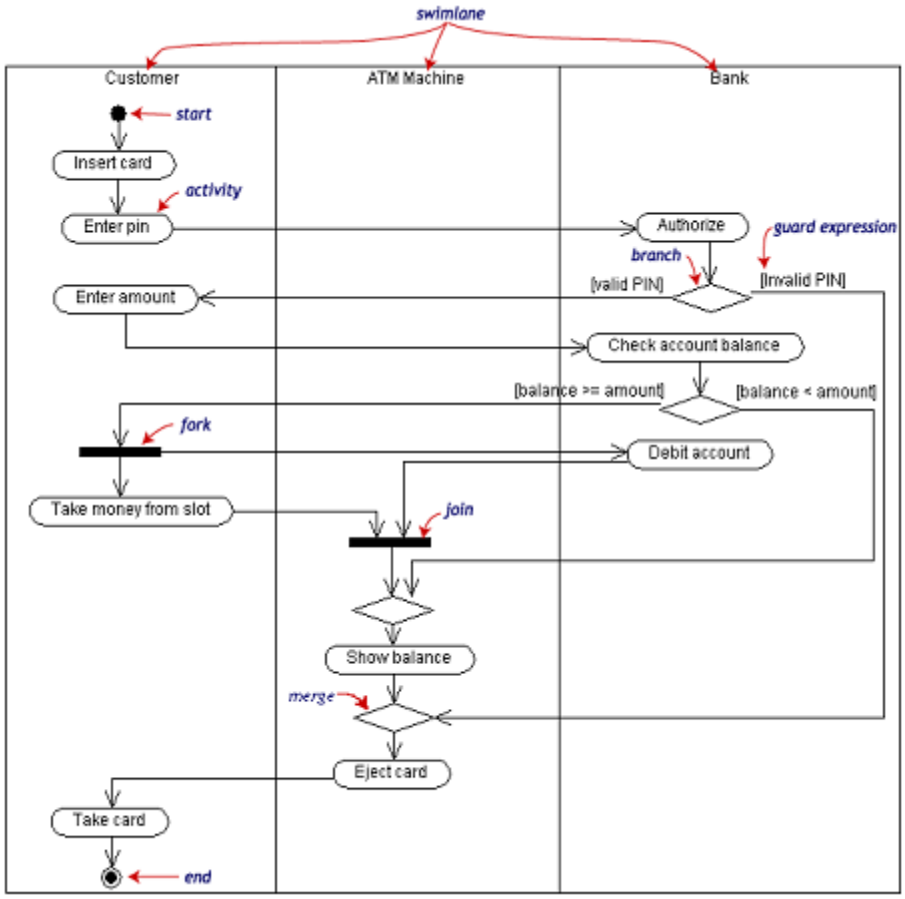
Behaviors dan *state* dimiliki oleh obyek. Keadaan dari suatu obyek bergantung pada kegiatan dan keadaan yang berlaku pada saat itu. Diagram *StateChart* menunjukkan kemungkinan dari keadaan obyek dan proses yang menyebabkan perubahan pada keadaannya. Penjelasan dari contoh yang digunakan yaitu model diagram untuk login, yang merupakan bagian dari *Online Banking System*. *Logging in* terdiri atas masukan *input Social Security Number* dan *Personal* Id *Number* yang berlaku, lalu memutuskan kesahan dari informasi tersebut, seperti pada Gambar 2.7.



Gambar 2.7 Contoh Diagram *StateChart* [2]

7. Diagram *Activity*

Diagram *Activity* memperlihatkan aliran urutan aktifitas. Pada dasarnya diagram *Activity* sering digunakan oleh *flowchart.* Diagram ini berhubungan dengan diagram *Statechart.* Diagram *Statechart* berfokus pada obyek yang dalam suatu proses (atau proses menjadi suatu obyek), diagram *Activity* berfokus pada aktifitas-aktifitas yang terjadi yang terkait dalam suatu proses tunggal. Jadi dengan kata lain, diagram ini menunjukkan bagaimana aktifitas-aktifitas tersebut bergantung satu sama lain, terlihat pada Gambar 2.8.



Gambar 2.8 Contoh Diagram Activity [2]

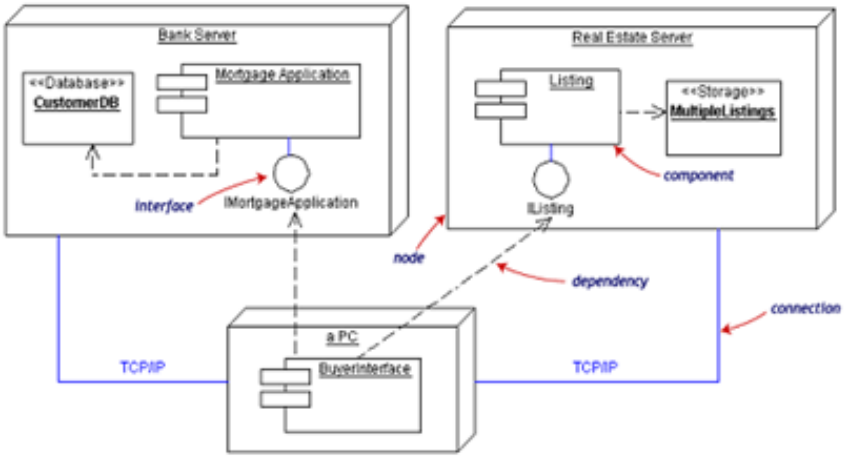
Diagram *Activity* dapat dibagi menjadi beberapa jalur kelompok yang menunjukkan obyek yang mana yang bertanggung jawab untuk suatu aktifitas. Peralihan tunggal (*single transition*) timbul dari setiap adanya *activity* (aktifitas), yang saling menghubungi pada aktifitas berikutnya. Sebuah *transition* (transisi) dapat membuat cabang ke dua atau lebih percabangan *exclusive transition* (transisi eksklusif). *Label Guard Expression* (ada didalam [ ]) yang menerangkan *output*

(keluaran) dari percabangan. Percabangan akan menghasilkan bentuk menyerupai bentuk intan. *Transition* bisa bercabang menjadi beberapa aktifitas *parallel* yang disebut *Fork. Fork* beserta *join* (gabungan dari hasil *output fork*) dalam diagram berbentuk *solid bar* (batang penuh).

8. Diagram *Component dan Deployment*

Component adalah sebuah *code module* (kode-kode modul). Diagram *Component* merupakan fisik sebenarnya dari diagram *Class*. Diagram *Deployment* menerangkan konfigurasi fisik *software* dan *hardware*.

Gambar 2.9 menerangkan hubungan sekitar komponen *software* dan *hardware* yang berperan dalam ruang lingkup *real estate.*



Gambar 2.9 Contoh Diagram *Deployment* [2]

Fisik *hardware* berbentuk seperti *node-node.* Setiap komponen merupakan bagian dari *node*. Pada gambar komponen berbentuk dua kotak tersusun yang terletak di sebelah kiri atas.

2.2 INTERNET

Internet merupakan singkatan dari *Interconnected Network.* Sekitar tahun 1994, internet mulai digunakan di Indonesia. Internet adalah sebuah sistem komunikasi yang menghubungkan jaringan-jaringan komputer di seluruh dunia. Beberapa bentuk jaringan yang berbeda-beda dapat saling bertukar informasi dan data melalui internet menggunakan sebuah protokol yang disebut dengan protokol TCP/IP. Dengan adanya internet mampu membuat pekerjaan menjadi lebih mudah dan efisien. Segala

informasi dapat diperoleh dengan mudah melalui internet dan perbedaan jarak tidak lagi menjadi hambatan dalam melakukan komunikasi. Dengan kata lain, internet memiliki tujuan untuk melayani miliaran pengguna di dunia.

2.3 *VOLUME BASED*

2.3.1 Definisi *Volume based*

Volume based adalah cara perhitungan internet berdasarkan jumlah *volume* data yang dihabiskan, mencakup *download* dan *upload* (*send and receive*) di mana semakin besar data yang digunakan, maka semakin banyak untuk biaya yang dikeluarkan. Perhitungan untuk *volume based* biasanya dikenakan per *Kilo Byte* (KB) untuk setiap penggunaan.

Dengan kata lain *volume based* merupakan perhitungan biaya akses internet yang bergantung pada jumlah data yang diambil dari internet. Pada beberapa penyedia layanan dihitung pula berdasarkan jumlah data yang dikirim dari komputer. *Volume based* digunakan pada akses internet melalui operator telepon seluler berbasis GSM dengan teknologi GPRS.

Koneksi *volume based* sebaiknya digunakan jika sering mengakses informasi internet yang berbasis teks atau bagi pengguna yang lebih sering melakukan aktivitas mengobrol di internet (*chating*). Hal ini dikarenakan informasi berbasis teks ukuran datanya relatif lebih kecil dibandingkan bentuk grafis atau gambar.

2.3.2 Perhitungan *Volume based*

Perhitungan *volume based*, misalnya jika pengguna mendaftar paket internet dengan menggunakan kuota 500 MB (*Mega Byte)*. Ketika pengguna mengunduh file sebanyak 5 MB, maka kuotanya akan berkurang 5 MB dan sisa kuota yang dapat digunakan menjadi 495 MB. Jika pengguna sudah menggunakan semua kuota atau sisa kuota sudah mencapai 0 MB, maka pengguna tidak dapat lagi menggunakan koneksi internet atau pengguna tetap dapat mengakses internet, namun dikenakan biaya tambahan yang diambil dari pulsa yang tersedia, bukan dari biaya khusus untuk internetan.

2.4 KOMPRESI DATA

Kompresi data merupakan proses mereduksi ukuran suatu data untuk menghasilkan representasi digital yang padat (*compact)* namun tetap dapat mewakili kuantitas informasi yang terkandung pada data tersebut. Tujuan dari kompresi data adalah untuk mengurangi data berlebihan (*redundancy data)* sehingga ukuran data menjadi lebih kecil dan lebih ringan dalam proses transmisi. Contoh, seperti pada *handphone,* visual pada *handphone* sudah melalui proses kompresi, terlihat pada tampilan yang berbeda dengan PC, misalnya seperti tampilan halaman *website*. Pada beberapa literatur, istilah kompresi sering disebut juga *source coding,* kompresi data, kompresi *bandwitdth,* dan *signal compression.*

2.5 PORT

Port merupakan sebuah proses di mana sebuah server dapat memberikan sebuah layanan kepada klien sehingga klien dapat mengakses sebuah layanan yang ada dalam server. Contoh, jika klien ingin menggunakan layanan untuk membuka halaman *website,* maka port yang digunakan adalah port 80.

Terdapat 3 jenis port *software* yaitu: [4]

1) *Well-known ports.* Nomor *well-known port* adalah dari 0 sampai 1023.
2) *Registered ports.* Nomor *registered ports* adalah dari 1024 sampai 49151.
3) *Dynamic/Private ports.* Nomor *dynamic* (sering disebut dengan nama *Private ports*) adalah dari 49152 sampai 65535.

Port dapat dikenali dengan angka 16-bit (dua *byte*) yang disebut dengan *Port Number* dan diklasifikasikan dengan jenis *protocol transport* apa yang digunakan ke dalam Port TCP (*Transmission Control Protocol*) dan port UDP (Pengguna *Datagram Protocol*). Port memiliki angka 16-bit, total maksimum jumlah port untuk setiap *protocol transport* yang digunakan adalah 65536 buah. Namun, hanya nomor port 0 sampai 1024 yang disediakan untuk umum. Port 80 (HTTP), yaitu port untuk membuka sebuah halaman *website*, port 20 (FTP) untuk melakukan *upload* maupun *download* file, port 110 (POP3) untuk menerima e-mail. Port 25 untuk melakukan mengirim e-mail. [4]

2.6 ANDROID OS (*Operating System*)

2.6.1 Pengertian Android OS

Android merupakan sebuah sistem operasi *mobile* yang telah di modifikasi dari Linux atau dengan kata lain yang telah mengadopsi sistem operasi Linux. Pertama sekali, *operating system* ini dikembangkan oleh perusahaan Android.Inc. Nama sistem operasi Android inilah yang berasal dari perusahaan Android.Inc tersebut.

Pada tahun 2005, Google membeli Android dan mengambil alih proses pengembangannya sekaligus *team developer* Android. Hal ini dilakukan Google sebagai bagian dari strategi untuk memasuki pasar *mobile*. Google menginginkan Android untuk menjadi sistem operasi *Open Source* dan gratis, di mana mayoritas *code* Android dirilis di bawah lisensi *Open Source Apache* yang berarti setiap orang bebas untuk menggunakan dan mengunduh *source code* Android secara penuh.

Para *vendor* bebas untuk mengubah sekaligus membuat penyesuaian sesuai dengan kebutuhan untuk Android dan perusahaan dapat secara bebas untuk membuat perbedaan dari produk *vendor* lainnya. Model pengembangan yang sederhana membuat Android sangat atraktif dan hal tersebutlah yang membuat para *vendor* tertarik untuk mencoba sistem operasi Android, antara lain *vendor* tersebut adalah Samsung.

Keuntungan utama dari sistem operasi Android adalah *developer* hanya tertuju pada aplikasi saja, aplikasi tersebut bisa berjalan pada beberapa perangkat yang berbeda selama masih di *support* oleh Android (*developer* tidak perlu mempertimbangkan kebutuhan jenis perangkatnya, karena produsen dapat menambahkan *extension*-nya sendiri ke dalam Android sesuai kebutuhan produk mereka).

Pada umumnya sistem operasi Android memiliki fitur yang dimiliki oleh *smartphone* seperti aplikasi yang melimpah, e-mail, fitur *online* seperti *browser* dan lainnya. Ponsel ini sesuai bagi pengguna yang ingin mengakses layanan internet tanpa batas. Pengguna aplikasi Google seperti Gmail ataupun Google *Maps* dapat mengaksesnya dengan cepat melalui ponsel dengan sistem operasi Android.

2.6.2 Versi Android

Android telah dikembangkan dan di-*update* beberapa kali sejak rilis pertamanya. Tabel 2.1 memperlihatkan versi Android semenjak pertama kali dirilis.

Tabel 2.1 Versi Android [5]

Versi Android	Tanggal Rilis	Nama Kode
Beta	5 November 2007	-
1.0	23 September 2008	-
1.1	9 Februari 2009	-
1.5	30 April 2009	Cupcake
1.6	15 September 2009	Donut
2.0 / 2.1	26 Oktober 2009	Eclair
2.2	20 Mei 2010	Froyo
2.3	6 Desember 2010	Gingerbread
3.0	22 Februari 2011	Honeycomb
4.0.1	19 Oktober 2011	Ice Cream Sandwich
	Sekitar pertengahan 2012	Jelly Bean
	Sekitar 2013	Key Lime Pie

Adapun penjelesan beberapa sistem operasi Android sebagai berikut : [6]

1) Cupcake - Android 1.5

Pada versi ini, tampilan Android secara visual lebih bagus dengan beragam efek animasi. Fasilitas multimedia dikembangkan lagi dengan menambahkan fungsi video, fungsi *Bluetooth* A2DP (*Advanced Audio Distribution Profile)* dan AVRCP (*Audio/Video Remote Control Channel)*, fasilitas *upload* foto-video ke YouTube dan Picasa, dan tambahan sejumlah *widget* baru. Dengan adanya versi ini, pengguna bisa merekam dan menyaksikan video lewat layar Android.

2) Donut - Android 1.6

Versi ini banyak memberikan fungsi baru dalam hal konektivitas, seperti dukungan CDMA/EVDO (*Evolution Data Optimized)*, Wi-Fi (*Wireless Fidelity)*

standar 802.1x dan VPN (*Virtual Private Network)*. Pengembangan lain ada pada Android Market, *voice search* (*text speech engine)*, dukungan resolusi layar WVGA (*Wide Video Graphic Array)*, *framework* untuk *gesture*, navigasi Google. Pada seri inilah, integrasi kamera, perekam video, dan galeri dibentuk.

3) Eclair - Android 2.1

Eclair menerima banyak pembaruan, baik pada optimasi *hardware* ataupun di sisi *software*. Fitur baru yang ditawarka antara lain dukungan Microsoft Exchange, HTML (*HyperText Markup Languange)* 5, fungsi *digital zoom,* dan *keyboard* virtual baru. Tampilan *interface* lebih segar *Live Wallpaper*. Dukungan resolusi dan layar yang lebih besar juga disediakan, selain dukungan *flash built-in,* Google Maps yang lebih baik, Bluetooth 2.1, dan *browser* baru.

4) Froyo - Android 2.2

Froyo (Frozen Yoghurt) menjanjikan kinerja *hardware* yang lebih cepat, fungsi instalasi aplikasi ke memori *eksternal*, dukungan Adobe Flash 10.1, fitur Wi-Fi untuk membuat ponsel menjadi *hotspot* (*tethering*), perubahan pengguna *interface*, dan juga *voice dialling* serta bagi-pakai kontak via Bluetooth. Dukungan lain yang ada adalah dukungan resolusi layar lebih tinggi, dukungan *password* alfanumerik, dan pengembangan Android Marketplace.

5) Gingerbread – Android 2.3

Inilah seri Android (stabil) terakhir. Banyak hal baru ditawarkan, di mana yang paling mencolok adalah *interface*. Gingerbread mulai memberi dukungan fungsi telepon VoIP (*Voice Over Internet Protocol)*, *Near Field Communication*, fungsi *copy-paste* yang lebih baik, *keyboard multi-touch* gaya baru, serta dukungan sensor baru (*gyroscope and* barometer) dan format video-audio baru (WebM/VP8 dan AAC (*Advanced Audio Coding*)). Fasilitas *download manager* dan dukungan lebih dari satu kamera juga tersedia.

6) Honeycomb – Android 3.0

Honeycomb atau versi 3.0, pada versi ini diberi banyak kelengkapan dan fitur untuk mendukung kinerja Android ketika dijalankan di perangkat tablet. Salah satu fungsi yang ditawarkan, di mana fitur ini belum ada terdapat pada seri-

seri sebelumnya, yaitu fungsi *video chat*. Honeycomb sudah mendukung *accelerator hardware* guna mendukung grafis 3D. Aplikasi menarik lain yang ada di Honeycomb yakni Google Body. Aplikasi ini seperti Google Maps, namun untuk tubuh manusia dan pengguna bisa melakukan diagnosa cepat mengenai kondisi tubuh jika ada rasa sakit pada bagian tubuh tertentu. Pengguna bisa mencari referensi medis dan memberikan gambaran mengenai keluhan sakitnya saat berkonsultasi ke dokter.

2.6.3 Fitur-Fitur Android

Android tersedia secara *open source* bagi manufaktur perangkat keras untuk memodifikasinya sesuai kebutuhan. Meskipun konfigurasi perangkat Android tidak sama antara satu perangkat dengan perangkat lainnya, namun Android sendiri mendukung fitur-fitur berikut ini :

a) *Storage*

Mendukung SQL (*Structured Query Languange)* Lite. SQL Lite adalah sebuah *database relational lite* yang ringan yang digunakan untuk penyimpanan data.

b) *Connectivity*

Mendukung koneksi GSM/EDGE, IDEN (*Integrated Dispatch Enhanced Network*), CDMA, EVDO, UMTS, *Bluetooth, Wifi*, LTE (*Long Term Evolution)*, dan Wimax (*Worldwide Interoperability for Microwave Access*).

c) *Messaging*

Mendukung SMS (*Short Message Service)* dan MMS (*Multimedia Message Service).*

d) *Web Browser*

Web browser yang digunakan adalah *browser* berbasis *open source* Webkit, dengan *engine* Chrome V8 JavaScript.

e) *Media Support*

Dukungan media meliputi *file* media bertipe seperti, H.263, H.264 (dalam bentuk 3GP atau MP4 *container*), MPEG-4 SP (*Moving Picture Expert Group)*, AMR (*Adaptive Multi Rate)*, AMR-WB (3GP *container*), AAC (*Advanced*

Audio Coding), HE-AAC (dalam bentuk MP4 atau 3GP *container*), MP3 (MPEG-1 Layer-3), MIDI (*Musical Instrument Digital Interface*), Ogg Vorbis, WAV (*Waveform Audio Format*), JPEG (*Joint Photographic Experts Group*), PNG, GIF (*Graphics Interchange Format*), dan BMP (*BitMap)*.

f) Dukungan *Hardware*

Dukungan *hardware* terdapat, sensor *accelerometer*, *Camera,* Kompas Digital, sensor *proximity*, dan GPS (*Global Positoning System).*

g) *Multi Touch*

Mendukung layar dengan dukungan *multi touch.*

h) *Multitasking*

Kemampuan untuk melaksanakan tugas secara bersamaan.

i) *Flash*

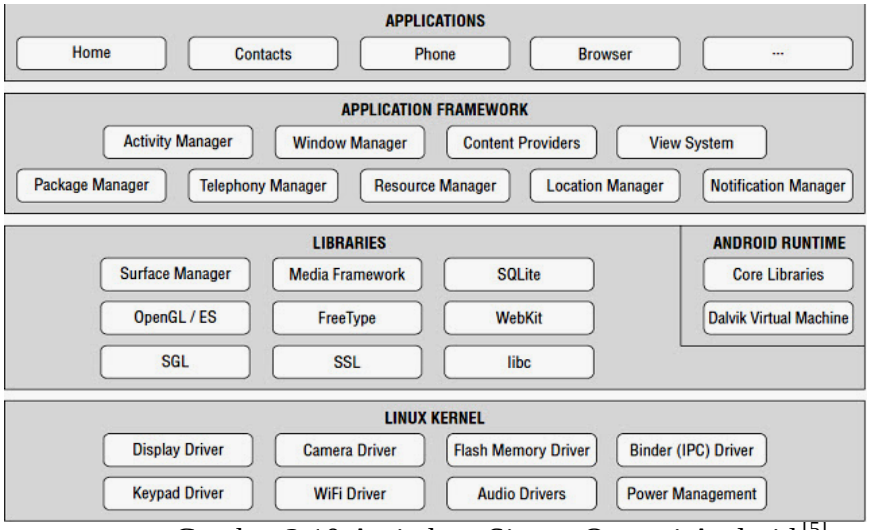
Mendukung animasi *flash.*Untuk Android 2.3 mendukung Flash 10.1.

j) *Tethering*

Mendukung berbagi koneksi internet.

2.6.4 Arsitektur Android

Adapun arsitektur Android dibuat agar dapat memahami lebih mudah bagaimana sistem operasi Android bekerja, berikut ini bagan tingkatan-tingkatan sistem operasi Android.



Gambar 2.10 Arsitektur Sistem Operasi Android [5]

Secara garis besar sistem operasi Android terbagi dalam lima tingkatan yaitu sebagai berikut :

a) Linux Kernel

Merupakan kernel dasar Android. Tingkat ini berisi semua *driver* perangkat tingkat rendah untuk komponen-komponen *hardware* perangkat Android.

b) *Libraries*

Pada *libraries* ini terdapat semua kode program yang menyediakan layanan-layanan utama sistem operasi Android. Sebagai contoh *library* SQLite yang menyediakan dukungan *database* sehingga aplikasi Android dapat menggunakannnya untuk menyimpan data. *Library* WebKit yang menyediakan fungsi-fungsi *browsing web* dan lainnya.

c) Android *Runtime*

Android *Runtime* ini kedudukannya setingkat dengan *libraries*, Android *Runtime* menyediakan kumpulan referensi ini yang dapat diaktifkan oleh pengembang untuk menulis kode aplikasi Android dengan bahasa pemrograman Java. Dalvik *Virtual Machine* aktif setiap kali aplikasi Android berproses (aplikasi Android dikompilasi menjadi Dalvik *executable*). Dalvik adalah mesin semu yang dirancang khusus untuk Android yang dapat megoptimalkan daya *battery* perangkat bergerak dengan memori dan CPU (*Central Processing Unit)* terbatas.

d) *Application Framework*

Merupakan sebuah kumpulan *class built-in* yang tertanam dalam sistem operasi Android sehingga *developer* dapat memanfaatkannya untuk aplikasi yang sedang dibangun.

e) *Applications*

Pada tingkat inilah kita akan bekerja, contoh aplikasi ini banyak ditemui seperti: *Phone, Contact, Browse* dan lain-lain. Seperti pada aplikasi Android

pada umumnya yang dapat di-*download* dan di-*install* dari Market Android. Semua aplikasi yang dibuat oleh pengguna terletak pada tingkat *Applications.*

2.7 PEMROGRAMAN JAVA

2.7.1 Defenisi

Java merupakan bahasa pemrograman yang dapat dijalankan di berbagai jenis komputer dan berbagai sistem operasi termasuk *handphone*. Pada tahun 1995, Java dirilis dan dikembangkan oleh Sun Microsystems. Selain itu, Java merupakan suatu *platform* yang memiliki *virtual mahchine* atau JVM (*Java Virtual Machine)* dan *library* yang diperlukan untuk menulis dan menjalankan (*running)* suatu program. Java juga merupakan *multiplatform.*

Adapun mengapa bahasa pemrograman Java ini dapat diminati/dikuasai oleh *programmer-programmer* yang ada karena adanya kelebihan dri Java yaitu :

1. *Multiplatform,* yang berarti Java dapat dijalankan di beberapa *platform*/sistem operasi komputer, sesuai dengan prinsip Java "*Write Once, Run Anywhere* (WORA)". Adanya prinsip tersebut, berarti *programmer* cukup menuliskan sekali saja sebuah program Java dan dapat dikompilasi (diubah, dari bahasa yang dimengerti manusia menjadi bahasa mesin) sekali, kemudian hasilnya dapat dijalankan diatas beberapa *platform.*

2. OOP, pemrograman berorientasi obyek merupakan semua aspek yang terdapat pada Java adalah Obyek. Hal ini sangat memudahkan *programmer* dalam merancang, membuat dan mengembangkan sebuah program dengan basis Java secara tepat dan mudah.

3. *Class Library, Library* yang lengkap sangat dikenal dalam Java, yang berarti kumpulan program yang disertakan dalam pemrograman Java terdapat pada suatu perpustakaan untuk melingkupi seluruh kebutuhan pembangunan aplikasi.

4. Bergaya C++, yang berarti Java memiliki sintaks seperti bahasa pemrograman C++.

Dari kelebihan yang ada, Java juga memiliki kekurangan yaitu, mudah didekompilasi, dekompilasi merupakan proses membalikkan dari kode yang sudah

jadi menjadi kode sumber. Hal ini dapat terjadi dimungkinkan karena kode jadi Java merupakan *bytecode* yang menyimpan banyak atribut bahasa tingkat tinggi, seperti nama-nama kelas, metode dan tipe data.

2.7.2 Aturan Penamaan

Ada beberapa aturan untuk membuat suatu kode program Java agar dasar penamaan tersebut menghasilkan kode program yang *readable.*

a) Penamaan Paket (*Package)*

Pemberian nama paket ini bertujuan untuk mencegah terjadinya konflik paket, dengan asumsi tidak menggunakan nama *domain* orang lain. Cara penulisan nama paket, nama awal paket sebaiknya terdiri atas dua atau tiga huruf kecil, dan biasanya menggunakan nama domain Internet, seperti com, org, dan edu. Contoh, com.aplikasi.java.

b) Penamaan Kelas dan *Interface*

Nama kelas dan *interface* sebaiknya berupa kata benda atau ungkapan kata benda yang tidak terlalu panjang namun deskriptif. Penulisan nama kelas, diawali dengan huruf besar dan tidak ada spasi diantara dua kata benda tersebut. Contoh, KoneksiData.

c) Penamaan *Method*

Penamaan *method* seharusnya berupa kata kerja atau ungkapan kata kerja, dimana penulisan *method* mengacu pada sintaks Camel. Huruf pertama adalah huruf kecil dan huruf kedua huruf besar dan tidak menggunakan spasi. Contoh, setData dan getData.

d) Penamaan Variabel

Penamaan variabel kelas hampir sama dengan penamaan *method,* namun untuk penamaan *method* menggunakan suatu singkatan atau akronim. Contoh, s (String), d (double).

e) Penamaan Konstanta

Penamaan ini berupa kata-kata yang setiap abjadnya berupa huruf besar dan untuk memisahkan antara satu kata dengan kata yang lain menggunakan tanda garis bawah. Contoh, MAX_DATA atau MAX_LEN_DATA.

2.7.3 Karakteristik Java

Semboyan utama Java yang terkenal adalah *Write Once Run Anymwhere*, yang berarti cukup menuliskan *source code* sekali saja pada sebuah komputer dan dapat menjalakannya pada komputer lainnya yang memiliki Java. Berikut adalah karakteristik Java seperti yang terdapat di dalam programnya, yaitu :

1) *Java is Simple*

Sebenarnya tidak ada satu bahasa pemrograman yang dapat dikatakan *simple*. Sebagian besar Java dibuat menggunakan bahasa C++, tetapi dibuat lebih sederhana dan kemampuannya lebih ditingkatkan.

2) *Java is Object Oriented*

Pemrograman berorientasi obyek merupakan metodologi perancangan program berdasarkan obyek. Dalam pemrograman berorientasi obyek semua hal dapat dianggap obyek.

3) *Java is Distributed*

Distributed computing merupakan metode komputerisasi dengan menggunakan beberapa komputer yang dihubungkan dengan jaringan untuk mengatur tugas-tugas tertentu. Java telah memiliki kemampuan *networking* yang bagus.

4) *Java is Interpeted*

Java merupakan bahasa yang menggunakan *interpreter* (penerjemah) agar dapat menjalankan program. Jadi, agar program dapat dijalankan, maka pada komputer tujuan harus memiliki *interpreter*-nya. *Interpreter* Java menerjemahkan kode *bytecode* ke dalam bahasa mesin.

5) *Java is Robust*

Robust (dapat diandalkan) merupakan bahasa pemrograman Java yang dapat diandalkan untuk segala macam keperluan seperti, Java telah menghilangkan berbagai macam gangguan (*bug*)

6) *Java is Secure*

Secure yang dimaksud adalah pengguna tidak perlu khawatir mengenai kerusakan, karena Java tidak menyediakan akses secara bebas ke sistem komputer lainnya.

7) *Java is Architecture-Neutral*

Program yang dihasilkan oleh Java tidak bergantung kepada aristektur komputer tertentu. Karena program Java berjalan dalam lingkungan JVM (*Java Virtual Machine*), yang berarti program Java dapat dijalankan pada asritektur komputer yang berbeda-beda.

8) *Java is Portable*

Program Java dapat dibawa dan dijalankan dimanapun, sehingga pengguna dapat meng-*compile* program Java dan dapat menjalankannya pada komputer lainnya tanpa harus melakukan *compile* ulang.

9) *Java Performance*

Performa dari bahasa pemrograman Java dianggap lambat oleh beberapa *developer*. Hal ini disebabkan, karena program dijalankan melalui JVM.

10) *Java is Multithreated*

Contoh pemrograman *multithreading* merupakan pengguna dapat melakukan *download* sebuah *file* video sambil memainkan bagian *file* yang sudah ter-*download*. Kemampuan *multithreading* ini digunakan pada pemrograman GUI (*Graphical* Pengguna *Interface)*.

11) *Java is Dynamic*

Java didesain sedemikian rupa, seperti pengguna dapat mengakses sebuah *class* secara langsung tanpa melakukan *recompile* ulang.

2.8 JAVA 2 MICRO EDITION (J2ME)

J2ME dikhususkan pada perangkat-perangkat kecil seperti *mobile phone,* Palm, dan PDA (*Personal Digital Assintant)*. Untuk membuat agar Java dapat berjalan pada perangkat-perangkat tersebut maka Sun Mycrosystems menghadirkan teknologi Java khusus untuk mengembangkan *software* dalam perangkat tersebut. Teknologi tersebut dikemas dalam suatu *software* yang dikenal dengan Java 2 Micro Edition atau sering disebut J2ME.

2.9 *MOBILE APPLICATION*

Mobile application merupakan aplikasi yang berjalan pada sebuah *mobile device.* Contoh *mobile application* yang sangat dikenal adalah *game,* di mana *game* yang secara tetap sudah di-*install* oleh *vendor* ponsel tersebut.

Ada dua fitur yang secara *default* sudah harus ada pada sebuah ponsel/*handphone* yaitu fitur GPRS dan Java. Fitur GPRS yang memiliki kemampuan terjadinya pengiriman data dengan model paket atau *volume based*. Sedangkan fitur Java memungkinkan pengguna untuk memasukkan program baru ke dalam sebuah ponsel untuk melengkapi *software* yang diberikan oleh *vendor* ponsel.

Sistem Java yang sering digunakan untuk membuat sebuah *mobile application* adalah Java 2 Micro Edition (J2ME).

2.10 KECEPATAN AKSES INTERNET

Kecepatan akses internet adalah kemampuan yang dimiliki oleh sebuah perangkat keras ataupun perangkat lunak untuk mengakses sebuah alamat *website* ataupun layanan-layanan internet yang ada seperti *download*, *upload*, ataupun *browsing* dalam setiap detik. Kecepatan akses internet memiliki satuan KBps (*Kilo Byte persecond*). Faktor-faktor yang mempengaruhi kecepatan akses internet seperti, kecepatan modem, komputer pengguna, jaringan telepon dan ISP (*Internet Service Provider*).

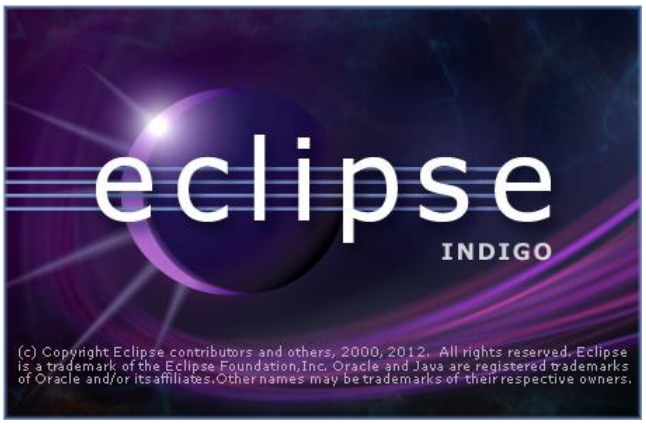
2.11 IDE ECLIPSE 3.7.2 (Indigo)

Untuk pengembangan aplikasi Android, IDE (*Integrated Development Environment*) yang diperlukan adalah IDE Eclipse. Eclipse merupakan IDE *software* yang digunakan oleh banyak bahasa pemrograman seperti Java, C++, Phyton dan lain-lain. Pada IDE Eclipse terdapat suatu layanan *system extensible* (seperti sistem penambahan *plugins*), *editor*, *debugger*, *control tools*, dan lain-lain.

IDE Eclipse sebenarnya adalah suatu perangkat lunak, di mana lingkungannya dikondisikan agar memudahkan *developer* membangun sebuah aplikasi. Dengan kata lain, Eclipse merupakan sebuah IDE untuk mengembangkan perangkat lunak dan dapat dijalankan di semua *platform* (*platform-independent*) dan dikembangkan dengan

bahasa pemrograman Java. Kelebihan dari Eclipse yaitu kemampuannya dapat dikembangkan oleh *developer* dengan komponen yang dinamakan *plugin.*

IDE Eclipse dapat diperoleh dengan cara mengunduh melalui alamat resmi Eclipse yaitu http://www.Eclipse.org/downloads/.



Gambar 2.11 Aplikasi Eclipse Indigo

Eclipse memiliki karakteristik yaitu sebagai berikut :

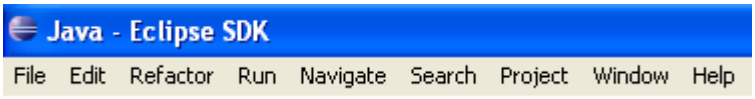
1. *Multi-platform* : Pencapaian sistem operasi Eclipse adalah Microsoft Windows, Solaris, Linux, AIX, dan Mac OS X.
2. *Multi-languange* : Eclipse mendukung bahasa pemrograman Java, dan mendukung pengembangan aplikasi berbasis bahasa pemrograman lainnya seperti, C++, Phyton, PHP dan lain sebagainya.
3. *Multi-role* : IDE ini dapat berfungsi sebagai pengembangan web, tes perangkat lunak dan sebagai pengembangan aplikasi tentunya.

Konsep Eclipse adalah, selain sebagai IDE yang terbuka (*open source*), Eclipse juga mudah diperluas (*extensible*) untuk *project* apa saja, dan tidak untuk sesuatu yang spesifik. Jadi, Eclipse tidak hanya mengembangkan program Java, akan tetapi dapat digunakan untuk berbagai macam keperluan seperti menambahkan *plugin*, cukup dengan memasang *plugin* yang dibutuhkan. Apabila pengguna ingin mengembangkan program C/C++ terdapat *plug-in* CDT (*C/C++ Development Tools*). Salah satu situs yang menawarkan *plug-in* secara gratis seperti Eclipse *downloads by project*.

Penjelasan lebih jauh mengenai fitur dari IDE Eclipse 3.7.2 dan mengenai fasilitas-fasilitas yang ditawarkan oleh IDE Eclipse 3.7.2, antara lain :

1. Menu

Menu merupakan daftar perintah-perintah yang dikelompokkan dalam kriteria tertentu, di mana fungsi menu untuk melaksanakan sebuah perintah yang di-*input* pengguna. Eclipse 3.7.2 terdapat sembilan menu utama yaitu, *File, Edit, Refractor, Run, Navigate, Search, Project, Window* dan *Help,* terlihat pada Gambar 2.12.



Gambar 2.12 Tampilan Menu

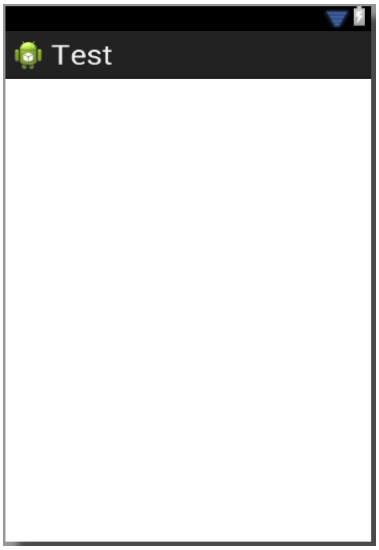
2. *Toolbar*

Toolbar merupakan kumpulan *icon* yang dapat melakukan sebuah perintah dengan cepat. Fungsi *toolbar* hampir sama dengan menu, hanya saja *toolbar* berbentuk *icon-icon* yang susunannya tidak secara bertingkat. Contoh beberapa *toolbar* tersebut antara lain, *New, Save, Save All, Print, Run,* dan lain sebagainya, terlihat pada Gambar 2.13.

Gambar 2.13 Tampilan *Toolbar*

3. *Layout*

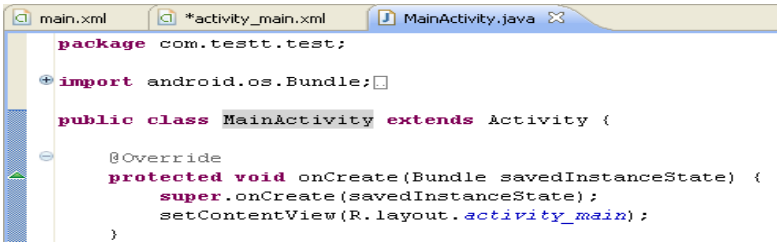
Layout merupakan tempat untuk meletakkan obyek-obyek yang digunakan untuk melaksanakan perintah yang diberikan. Pada bagian atas terdapat *icon* sinyal, baterai, logo, dan judul aplikasi yang akan dibuat, contoh *Layout Test.* Tampilan *layout* kosong seperti pada Gambar 2.14.



Gambar 2.14 Tampilan *Layout*

4. *Editor*

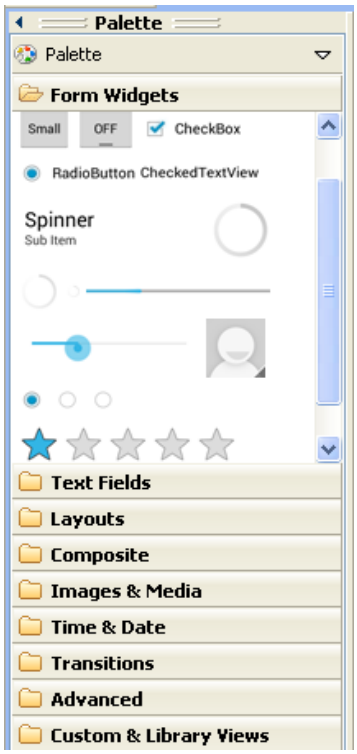
Editor merupakan tempat menuliskan kode-kode program. Semua kode perintah ditulis pada *editor* ini. Pada bagian *editor* terdapat fasilitas *focus* terhadap *code* program yang tidak dimengerti. Contoh tampilan *editor* seperti pada Gambar 2.15.



Gambar 2.15 Tampilan *Editor*

5. *Pallete*

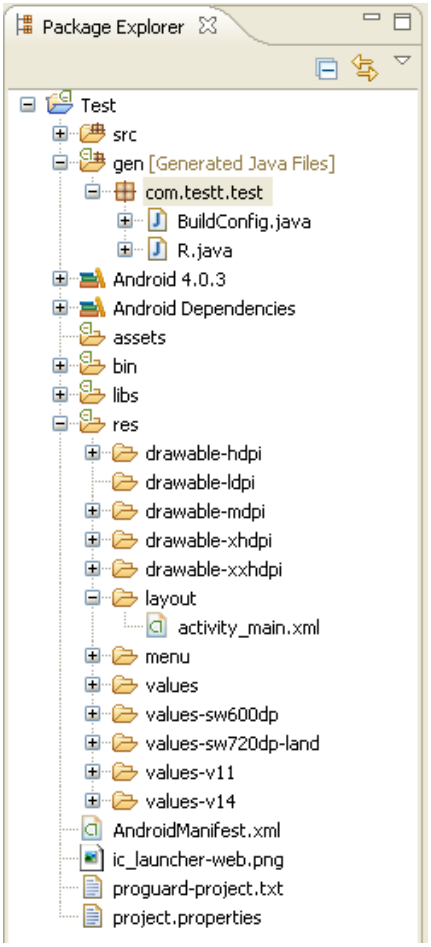
Pallete merupakan kumpulan obyek yang digunakan untuk mengontrol sebuah program. *Pallete* terdiri atas beberapa *tool* yang ditempatkan dalam satu *folder*, sesuai dengan fungsinya masing-masing. *Pallete* yang selanjutnya ditempatkan pada sebuah *layout. Pallete* terdiri dari *Form Widgets, Text Fields, Layouts, Composite, Image &Media, Time & Date, Transitions, Advanced,* dan *Custom & Library Views*, seperti pada Gambar 2.16.



Gambar 2.16 Tampilan *Pallete*

6. *Package Explorer*

Package Explorer digunakan untuk melihat bagian-bagian proyek pembuatan aplikasi. Bagian-bagian tersebut dapat berupa *project*, di mana dalam satu *folder project* mengandung banyak *folder-folder* sebagai tempat *source* dari aplikasi yang dibuat. *Project Explorer* ini berbentuk menu *tree* sehingga mempermudah dalam pengaksesannya. Pada *package explorer* terdapat dua *icon* kontrol, yaitu *collapse all* untuk membuat *project* tampil dalam satu *folder* (pengelompokan) dan *Link with editor* untuk menampilkan *folder layout* yang berekstensikan .xml. Contoh tampilan *package explorer* seperti pada Gambar 2.17.

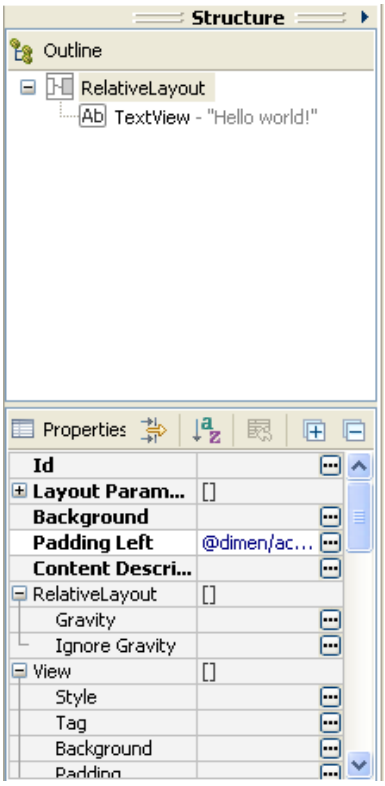


Gambar 2.17 Tampilan *Package Explorer*

7. *Window Structure*

Window structure menampilkan dua *window* yaitu *outline* dan *properties. Structure outline* merupakan tampilan pengerjaan aplikasi yang dibuat pada sebuah *layout* dalam bentuk menu *tree*, seperti *text view*, *buttons* dan sebagainya. *Properties* menampilkan semua properti dari obyek yang digunakan. Kita dapat

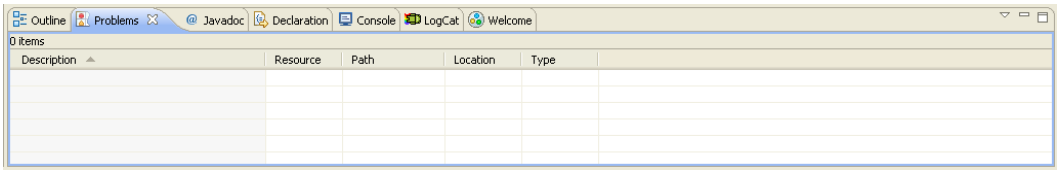
mengubah setiap properti dari obyek yang ada melalui jendela ini misalnya, *background*, *style* dan sebagainya, seperti pada Gambar 2.18.



Gambar 2.18 Tampilan *Windows Structure*

8. *Group Tab*

Group tab merupakan *tab-tab* yang menampilkan hasil apa yang telah dilakukan selama *project* aplikasi dijalankan. Pada group tab ini terdapat tab *Outline, Problems, Javadoc, Declaration, Console, LogCat,* dan *Welcome*. *Tab Console* merupakan *command prompt* Eclipse, *LogCat* merupakan *log* dari *emulator* Android, yang berisi *log* dari *process* yang dilakukan pada *emulator* serta *log* proses yang dilakukan oleh aplikasi. *Problems* merupakan hasil dari *project* yang berada di Eclipse. Setiap *project* yang *error* akan tampil, sehingga kesalahan yang ada pada *project* bisa diketahui dengan mudah. *Declarations* memuat fungsi-fungsi yang telah dibuat didalam *file* .java atau *code* java. Javadoc memuat dokumentasi dari fungsi-fungsi yang dibuat dalam bentuk HTML, seperti pada Gambar 2.19.



Gambar 2.19 Tampilan *Group Tab*

Setelah memahami mengenai fasilitas yang terdapat dalam IDE Eclipse 3.7.2, maka selanjutnya pemahaman lebih jauh mengenai pembuatan program Aplikasi Eclipse 3.7.2.

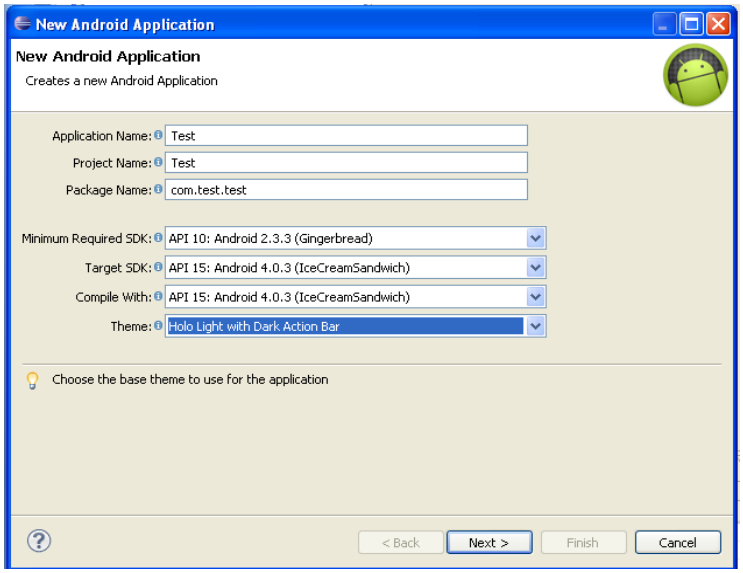
Pembuatan sebuah program aplikasi dengan Eclipse 3.7.2 yang harus dilakukan adalah membuat sebuah *project*, menambahkan *layout* ke dalam proyek apabila program aplikasi membutuhkan lebih dari satu *layout*, meletakkan atau menambahkan obyek kontrol pada *layout*, menulis kode program dan mengeksekusi atau menjalankan program tersebut untuk melihat hasilnya.

Berikut ini adalah cara-cara untuk membangun sebuah aplikasi, yakni :

a. Membuat *Project*

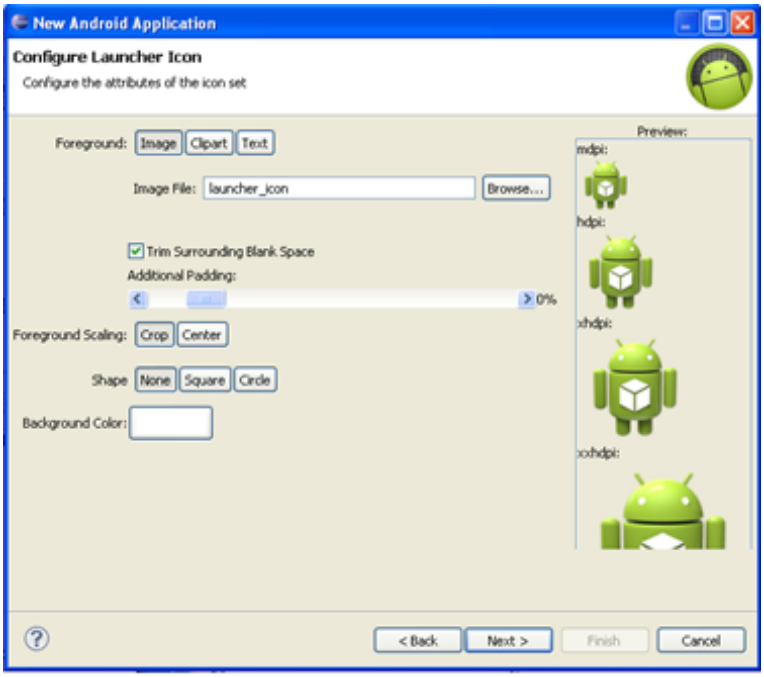
Pada waktu *project* dibuat, secara *default* telah terdapat sebuah *layout* yaitu *layout* di dalam *project* tersebut. Untuk membuat *project* dapat dilakukan dengan langkah berikut :

1. Dari *Windows*, menjalankan program Eclipse, kemudian memilih menu *File*. Memilih *New*. Memilih *Android Application Project*. Maka akan tampil kotak *dialog New Android Application*, kemudian mengisi semua *text field* yang ada pada kotak dialog sesuai dengan keperluan, seperti pada Gambar 2.20.



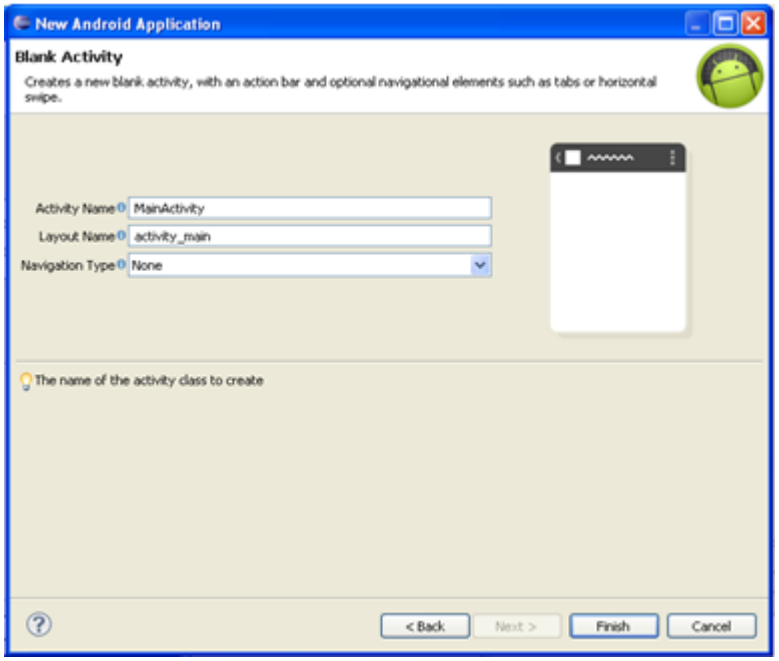
Gambar 2.20 Jendela *New Project*

2. Memilih *Next.* Setelah tampil *configure project* kemudian memilih *Next*. Pada *configure launcher icon setting image*, *clipart*, dan *text* sesuai keperluan, seperti pada Gambar 2.21.



Gambar 2.21 Jendela *Configure Launcher Icon*

3. Memilih *Next.* Pada kotak dialog *Create Activity*, *setting Blank Activity*. Pada *Blank Activity* mengisi *activity name*, *layout name*, dan *navigation type* sesuai keperluan, lalu memilih *Finish*, seperti pada Gambar 2.22.



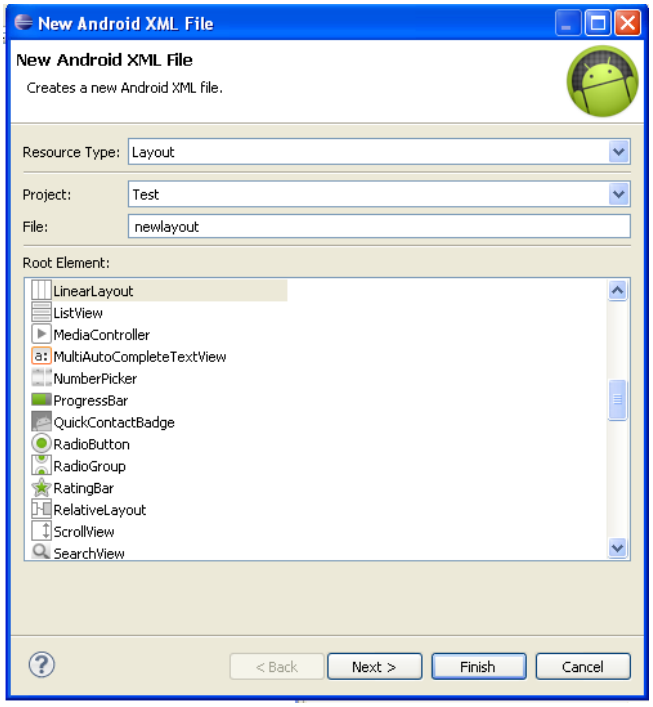
Gambar 2.22 Jendela *New Activity*

b. Menambah *Layout*

Setelah membuat sebuah *project*, maka selanjutnya program aplikasi dapat dibuat. Apabila program aplikasi yang dibuat membutuhkan lebih dari satu *layout*, maka *layout* penunjang tersebut harus ditambahkan ke dalam proyek tersebut.

Berikut ini adalah cara untuk menambah *layout* pada suatu *project* :

1. Menekan tombol kanan pada *project*, kemudian memilih *New* dan memilih Android XML *File*. Mengisi *text field* pada kotak dialog lalu memilih *finish*, seperti pada Gambar 2.23.

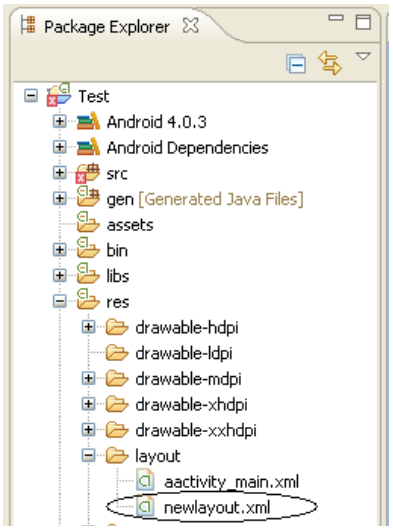


Gambar 2.23 Jendela *New Layout*

2. Menekan tombol kanan pada *folder layout* yang berada pada *project package explorer*, kemudian memilih *New* dan memilih Android XML *File*. Mengisi *text field* pada kotak dialog lalu memilih *finish.*

3. Pada *toolbar*, memilih *icon New*. Setelah itu memilih Android XML *File* Mengisi *text field* pada kotak dialog lalu memilih *finish*, seperti pada gambar sebelumnya.

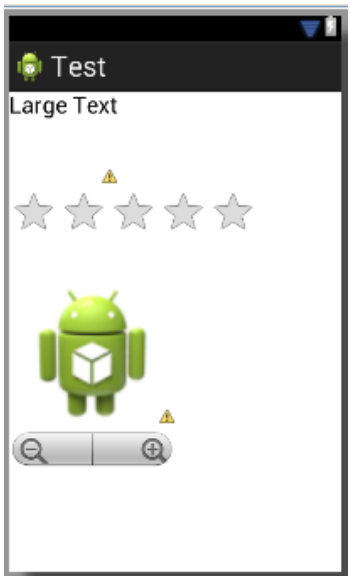
Setelah menambahkan sebuah *layout* pada *project*, maka pada *package explorer* akan bertambah sebuah *layout* seperti yang terlihat seperti pada Gambar 2.24.



Gambar 2.24 Tampilan *Package Explorer* Setelah Ditambah *Layout* Baru

c. Menambah Obyek Kontrol pada *Layout*

Dalam hal mengatur tampilan *layout*, perlu ditambahkan obyek kontrol pada sebuah *layout*. Penambahan obyek kontrol pada *layout*, dapat dilakukan dengan menahan obyek kontrol pada *pallete* lalu meletakkan pada *layout.* Contoh *layout* yang ditambahkan beberapa obyek kontrol terlihat pada Gambar 2.25.

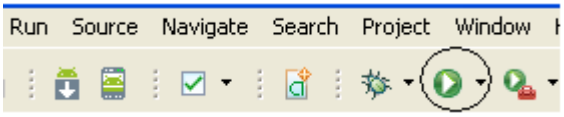


Gambar 2.25 Tampilan *Layout* Sesudah Ditambahkan Obyek Kontrol

d. Menjalankan Program Aplikasi

Jika ingin melihat hasil program aplikasi yang telah dibuat, program aplikasi tersebut harus dieksekusi atau dijalankan. Cara untuk menjalankan program aplikasi yang telah dibuat, dapat dilakukan dengan cara berikut :

1) Memilih menu *Run*, kemudian memilih Android *Application*
2) Menekan tombol kanan *project*, lalu memilih Run As Android *Application*
3) Memilih *icon Run* pada *toolbar* seperti Gambar 2.26.



Gambar 2.26 Tampilan *Icon Run*

2.12 ADT (Android *Development Tools*) *Plugins*

Android *Development Tools* berfungsi sebagai pengenal Android di dalam IDE Eclipse. *Plugin* ini yang dapat digunakan untuk mendukung pembuatan *project* aplikasi Android baru, mengakses *tools emulator*, melakukan kompilasi, proses *debugging* aplikasi, mengekspor aplikasi menjadi file paket Android *Packages* (APK) dan membuat sertifikasi digital untuk *code-signing* atas aplikasi yang telah dibuat. *Plugin* ADT dapat diperoleh dengan cara mengunduh pada alamat *http://developer.android.com/sdk/eclipse-adt.html.*

2.13 ANDROID SDK MANAGER & AVD *MANAGER*

Android SDK (*Software Development Kit) Manager* adalah sebuah alat pengembangan perangkat lunak yang berguna untuk mengembangkan dan membuat aplikasi untuk *platform* Android. SDK Android merupakan mesin utama untuk mengembangkan aplikasi Android. Pada Android SDK, terdapat *packages* sesuai dengan *platform* Android, terdapat *project sample* dengan kode sumber, sebuah *emulator* Android *Virtual Device* (AVD), dan *library* yang dibutuhkan untuk membangun aplikasi Android.

Aplikasi Android menggunakan bahasa pemrograman Java dan berjalan di Dalvik dan mesin virtual yang dirancang khusus untuk penggunaan *embedded* yang berjalan diatas kernel Linux. Android SDK dapat diperoleh dengan cara mengunduh pada alamat *http://developer.android.com./sdk/index.html.*

2.14 JDK (*Java Development Kit)*

Sebuah perangkat lunak yang berfungsi sebagai *software* pendukung dalam pembuatan aplikasi Android, karena bahasa pemrograman Android menggunakan bahasa Java, sehingga dibutuhkan JDK. JDK versi terbaru, dapat diperoleh dengan cara mengunduh pada alamat *http://www.java.com/en/download/maual.jsp.*