BAB II

LANDASAN TEORI

2.1 Pengertian Java

Java merupakan pemrograman yang menanjak popularitas saat ini. Selain gratis, mudah didapatkan dan tangguh. *Java* sendiri lahir pada tahun 1991 yang diprakrasai oleh Tim Sun yang bernama Proyek *Green* yang dipimpin oleh James Gosling. Java adalah bahasa yang dimana para pengembang mengekspresikannya dengan *Source Code* (program text), *Java Syntax* sebagian mengikuti pola dari bahasa C/C++ dimana untuk menyingkat proses pembelajaran bagi para pengembang bahasa C/C++. *Java* adalah *platform* untuk mengeksekusi program. *Java platform* terdiri dari *virtual machine* dan lingkungan dalam eksekusinya. *Virtual machine* adalah *prosessor* berdasarkan kumpulan instruksi *software* itu sendiri. Lingkungan eksekusilah yang terdiri dari *library* untuk menjalankan program dan berinteraksi dibalik operasi sistem tersebut. *(Jeff Friesen : 2010).*

Java menerapkan fitur-fitus bahasa pemrograman yang lain yang dianggap suatu kelebihan oleh Tim Sun misalnya JVM/JRE yang merupakan mesin maya pada bahasa Pascal, Sintaks, dan *Exception Handling* diambil dari C/C++ dan lain sebagainya *(WAHANA KOMPUTER,: 2010).*

Java adalah bahasa dan *platform* yang dibuat oleh Sun Microsystems, Sun membuat *Java* menjadi 3 edisi utama, untuk memenuhi kebutuhan yang bervariasi yaitu, *Java* SE, *Java* EE, dan *Java* ME. *(Jeff Friesen : 2010).*

Java SE, Java EE, Java ME

Para pengembang menggunakan *Java platform* berbeda untuk membuat program *java* yang berjalan di komputer *desktop*, *web browser*, *web server*, dan *handset* seperti *handphone*.

Java SE (*Java Standard Edition*)

Program *platform Java* untuk mengembangkan aplikasi, dimana program *stand-alone* yang berjalan di komputer *desktop*. *Java* SE juga digunakan untuk mengembangkan *applet* dimana berjalan di *web browser*.

Java EE (*Java Enterprise Edition*)

Program *platform Java* ini digunakan untuk mengembangkan program yang berorientasi ke aplikasi bidang *enterprise* dan juga *servlets*, dimana *server* untuk menjalankan *Server* API untuk *Java* EE. *Java* EE dibangun diatas *Java* SE.

Java ME (*Java Micro Editon*)

Platform java ini digunakan untuk mengembangkan MIDlet dimana berjalan di *mobile device* dan xLets dimana berada di *embedded device*.

JRE (*Java Runtime Enviroments*) dan JDK (*Java Development Kit*)

Dalam pengimplementasian sebuah program *java* JRE yang bekerja dan menjalankan instruksi *platform* Java SE Tetapi JRE tidak dapat berjalan sendiri saja dalam mengimlementasikan program *java*, JRE harus didampingi JDK. JDK versi 1.0 adalah yang pertama kali dirilis di bulan Mei 1995 sampai JDK 6,

sampai beberapa kali versi dirilis, dengan adanya JDK 7 yang dirilis pada tahun 2010, setiap versi JDK mengindentifikasi versi dari *java* tersebut. *(Jeff Friesen : 2010).*

Java adalah bahasa yang pengembang menggunakan objek untuk mewakili entitas (hal-hal yang ada, seperti kendaraan, memeriksa dan rekening bank tabungan dan lainnya. Keutamaan java dibanding dengan bahasa lainnya adalah :

1. *Cross platform*, dengan adanya *Java virtual machine*
2. *Robust & Secure*
3. Pengembangan didukung oleh programmer yang luas
4. *Automatic garbage collection*, membebaskan programmer dari manajemen memori.

2.1.1 Karakter dari bahasa pemrograman berbasis objek

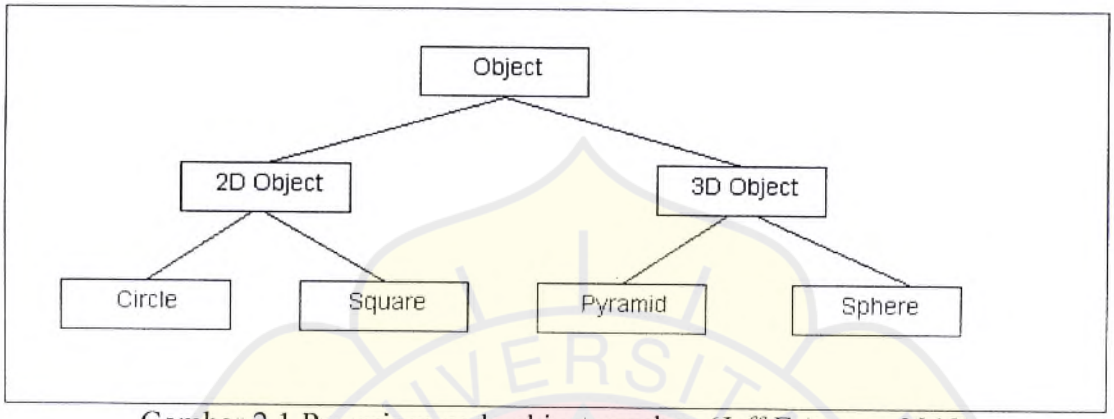
Dalam bahasa pemrograman berbasis objek, memiliki sebuah ciri khusu yang menjadikannya sebuah keuntungan dibandingkan bahasa pemrograman lainnya. Berikut adalah karakter atau cirri khas dari bahasa pemrograman, yaitu :

1. **Enkapsulasi**

Pembungkusan atau enkapsulasi adalah pelindung data dan program yang sedang diolah untuk diakses oleh program lain yang tidak diinginkan. Dalam dasar enkapsulasi adalah *class*. Dengan *class*, dideklarasikan *variable* yang dapat atau tidaknya diakses oleh *class* lainya. Dengan *private*, *variable* tidak dapat di akses oleh class lain atau p*rotected*, *variable* hanya dapat diakses oleh turunan dari *class* tersebut, atau *public* dimana *variable* tersebut dapat diakses oleh sembarang *class*. *(Dokumentasi Java, http://java.sun.com).*

2. **Inheritansi**

Pewarisan atau inherintansi adalah pemberian suatu sifat atau karakteristik untuk sebuah objek dari objek sebelumnya. Contohnya :



Gambar 2.1 Pewarisan pada object gambar *(Jeff Friesen : 2010)*

Sebuah program spesial yang dikenal sebagai *Java Compiler* menerjemahkan kode menjadi instruksi yang akan dieksekusi di *virtual machine,* instruksi ini umumnya berupa *bytecode*. *compiler* menyimpan *bytecode* dan mempunyai tipe file .class. *(Dokumentasi Java, http://java.sun.com)*

3. **Poliformisme**

Walau lingkaran dan kotak adalah sama-sama bentuk 2D tapi memiliki variable yang bebeda, dimana lingkaran memiliki jari-jari dan kotak memiliki panjang sisinya. *(Dokumentasi Java, http://java.sun.com)*

2.1.2 Objek dan Class

1. **Objek**

Objek adalah suatu perwakilan nyata dari dunia nyata, seperti buah mangga, mobil, rumah, dan lain-lainnya.*(Dokumentasi Java, http://java.sun.com)*

2. **Class**

Class adalah suatu *blueprint* dari objek itu sendiri dimana terdiri dari definisi *variable*, data dan fungsi dari suatu objek tesrsebut. *Class* adalah modularitas dan struktur dari pemrograman berbasis objek, dimana semua permasalahan bisa diselesaikan dengan modul modul ini. *(Dokumentasi Java, http://java.sun.com).*

2.2 Android

Android adalah Sisem Operasi untuk *platform* perangkat *mobile* semacam ponsel, *smartphone*, tablet dan *netbook*. Android dikembangkan oleh Google dengan *platform* dasar Kernel Linux dan *software* GPU. semula Android dikembangkan oleh Google, inc (sebuah lembaga yang dibeli Google) yang kemudian dikembangkan dan sekarang menjadi konsorsium aliansi handset yang dikenal sebagai OHA (*Open Handset Alliance*) yang menciptakan dan mengembangkan sebuah standard terbuka untuk perangkat mobile, anggota aliansinya sebagai berikut : Google, HTC, Dell, Motorola, QualComm, Texas Instruments, Samsung, LG, T-Mobile, NVidia, dan Wind River Systems. *(Chris Web :2009).* Android menggunakan aplikasi java seperti *platform framework* yang *Object-Oriented* berdasarkan *Java Core Libraries*, dan kode lainnya menggunakan XML, kode C, dan C++.

2.2.1 Komponen Android

Menurut Chris Web didalam buku Professional Android Application Development, komponen pada android meliputi sebagai berikut :

1. **Activity**

Struktur dari antarmuka pengguna adalah aktivitas. A*ctivity* dapat diartikan sebagai *window dialog* dalam aplikasi *desktop*. Meski dimungkinkan untuk activity tidak memiliki antarmuka pengguna, dimana kode program akan dikemas dalam bentuk penyedia konten atau layanan.

2. **Content Provider**

Penyedia konten memberikan tingkat abstraksi untuk setiap data yang tersimpan pada perangkat yang dapat diakses oleh beberapa aplikasi. Model pengembangan Android mendorong untuk membuat data sendiri tersedia untuk aplikasi lain, serta memungkinkan penyedia konten, sambil mempertahankan kontrol penuh atas bagaimana data akan diakses.

3. **Intents**

Intents adalah *message system*, berjalan didalam sistem perangkat, memberitahukan aplikasi berbagai *events*, dari perubahan keadaan *hardware* (misalnya, kartu SD dimasukkan), untuk data yang masuk (misalnya, pesan SMS tiba), dan juga untuk *events* aplikasi (misalnya, aktivitas dijalankan dari menu utama). *Intent* tidak hanya dapat ditanggapi atau direspon, tapi bisa juga membuat intent baru, untuk memulai *events* lainnya, atau untuk memberitahu ketika ada situasi khusus.

4. **Services**

Activities, *content providers*, and *intent recievers* adalah proses yang singkat dan bisa dimatikan kapan saja. Sedangkan *Services* dalam hal lain berbeda dan didesain untuk selalu berjalan, *services* dibutuhkan terbebas dari segala macam *activity*. Hal ini dilakukan untuk memungkinkannya *service* untuk mengecek *update* dari RSS *feed* atau *playback* dari musik bahkan *activity* untuk mengendalikan *activity* music sudah tidak berjalan lagi.

5. **Broadcast Receivers**

Broadcast Receivers adalah sebuah *intent* yang dikirim pengguna. Dengan membuat dan meregistrasi sebuah *Broadcast Receievers*, aplikasi dapat menanggapi *intent* yang dikirim secara spesifik dengan sebuah kriteria filter. Komponen ini akan secara otomatis mengaktifkan aplikasi anda dalam meresponse *intent* yang datang, menjadikan aplikasi menjadi *event-driven programming*.

6. **Notifikasi**

Notifikasi user digunakan untuk tidak mengganggu aktivitas user yang sedang menggunakan handset. Notifikasi adalah teknik untuk mendapatkan atensi *user* .

2.2.2 Fitur Android

1. **Storage (Penyimpanan)**

Dalam android, memungkinkan untuk membuat paket file data di aplikasi yang telah dibuat, untuk hal-hal yang tidak berubah, seperti ikon atau file pembantu. Dan juga dapat membuat ruang pada perangkat itu sendiri, untuk

database atau *file content* pengguna untuk meng-*insert* atau diambil data yang diperlukan oleh aplikas. *(Chris Web :2009)*

2. **Network (Jaringan)**

Perangkat Android umumnya siap menggunakan *Internet*, melalui satu media komunikasi atau lain. *(Chris Web :2009)*

3. **Multimedia**

Perangkat android memiliki kemampuan untuk memutar dan merekam audio dan video. Sedangkan secara spesifik mungkin berbeda setiap perangkat, tetapi semua kemampuan multimedia sebagian besar dapat digunakan oleh pengguna android. Apakah itu adalah untuk memutar musik, mengambil gambar dengan kamera dan lain-lainnya. *(Chris Web :2009)*

4. **GPS**

Perangkat android sering mengakses penyedia lokasi, seperti GPS, yang dapat memberitahu ada di mana perangkat ini. Pada akhirnya, fitur yang dapat digunakan dalam android adalah menampilkan peta atau ataupun melakukan pelacakan perangkat jika perangkat telah dicuri. *(Chris Web :2009)*

5. **Phone Service**

Tentu saja, perangkat android adaah sebuah ponsel, memungkinkan untuk memulai panggilan, mengirim dan menerima pesan SMS, dan segala sesuatu yang diharapkan dari sebuah teknologi telepon modern. *(Chris Web :2009)*

2.2.3 File Manifest

Setiap *project* android memiliki *file manifest* yang tersimpan di *Root* aplikasi. File Manifest berfungsi untuk mendefinisikan struktur dan metadata dari aplikasi serta komponen-komponen didalam aplikasi. Komponen dari file manfiest adalah sebagai berikut :

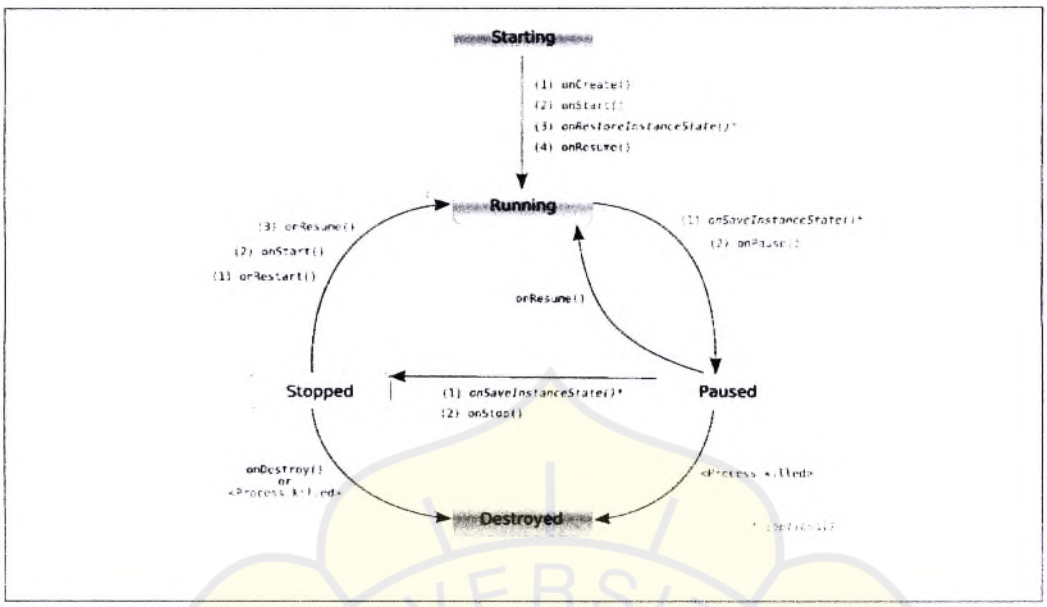
1. Activities
2. Services
3. Content Providers
4. Broadcast Receivers. *(Chris Web :2009).*

2.2.4 Daur Hidup Android

Aplikasi android tidak dapat mengkontrol daur hidupnya sendiri, setiap komponen harus mendengar pada setiap perubahan melalui pernyataan didalam aplikasi tersebut. Secara *default*, setiap aplikasi android berjalan dalam proses masing-masing secara terpisah didalam mesin *virtual dalvik*. Memori dan manajemen proses pada setiap aplikasi ditangani secara eksklusif pada saat *runtime*.

Sistem android secara agresif melakukan manajemen terhadap sumber yang dimiliki dan akan memastikan bahwa perangkat tetap responsif, artinya akan mematikan proses, jika perlu tanpa adanya peringatan. *(Chris Web :2009)*

Berikut adalah daur hidup didalam android :



Gambar 2.2. Diagram daur hidup android. . *(Ed Burnet : 2010)*

2.2.5. Android Java Package

Salah satu cara untuk mendapatkan gambaran singkat dari *platform* android adalah dengan melihat struktur paket-paket Java. Karena android menyimpang dari distribusi JDK standar, penting untuk mengetahui apa yang didukung dan apa yang tidak. Berikut adalah paket yang disertakan dalam SDK Android :

1. android.app

   Mengimplementasikan model aplikasi untuk android. Terutama kelas yang termasuk aplikasi , semantik, dan sejumlah kegiatan yang berhubungan dengan kelas, kontrol, dialog, peringatan, dan pemberitahuan.

2. android.bluetooth

   Menyediakan sejumlah kelas untuk bekerja dengan fungsionalitas *bluetooth.* Kelas-kelas utama meliputi *Bluetooth Adapter, Bluetooth*

Device, Bluetooth Socket, Bluetooth Server Socket, dan *Bluetooth Class*. *Bluetooth Adapter* digunakan untuk mengontrol secara lokal dipasang adaptor *bluetooth*. Misalnya, mengaktifkannya, menonaktifkannya, dan memulai proses penemuan. *Bluetooth Device* tersebut merupakan *Bluetooth remote* perangkat yang terhubung dengan kedua Soket *Bluetooth* digunakan untuk membangun komunikasi antara perangkat. Sebuah kelas *Bluetooth* merupakan jenis perangkat *Bluetooth* yang terhubung.

3. android.content

   Mengimplementasikan konsep penyedia konten. Isi abstrak dari akses data dari toko penyedia data. Paket ini juga menerapkan ide-ide sentral di sekitar maksud dan *Android Uniform Resource Identifier* (URI).

4. android.content.pm

   Implementasi *Package Manager* yang berhubungan dengan kelas. Sebuah manajer paket tentang perizinan, paket terinstal, diinstal penyedia, jasa diinstal, komponen yang terpasang seperti kegiatan, dan diinstal aplikasi.

5. android.content.res

   Memberikan akses ke file sumber daya baik terstruktur dan tidak terstruktur. Kelas-kelas utama adalah *Asset Manager* (untuk sumber daya tidak terstruktur) dan Sumber Daya.

6. android.database

   Mengimplementasikan ide database abstrak.

7. android.database.sqlite

   Mengimplementasikan konsep-konsep dari paket android.database menggunakan SQLite sebagai *database* fisik. Kelas primer SQLite Cursor,

SQLite Database, SQLite Query, SQLite *Query Builder*, dan SQLite *Statement*. Namun, sebagian besar interaksi akan menjadi dengan kelas dari paket abstrak android.database.

8. android.gesture

   Paket ini adalah rumah untuk semua kelas dan *interface* yang perlu untuk bekerja dengan user-defined workflow. Kelas primer *Gesture, Gesture Library, Gesture Overlay View, Gesture Store, GestureStroke, GesturePoint. Gesture* adalah kumpulan *dari Gesture Strokes* dan *Gesture Points. Gestures* dikumpulkan dalam *Gesture Library*. Perpustakaan *Gesture* disimpan dalam *Gesture Store. Gestures* diberi nama sehingga mereka dapat diidentifikasi sebagai tindakan.

9. android.graphics

   Berisi bitmap kelas, kanvas, kamera, warna, matriks, film, *paint, path, rasterizer, shader, sweep gradient,* dan jenis huruf.

10. android.graphics.drawable

    Mengimplementasi menggambar protokol dan latar belakang gambar, animasi dan memungkinkan objek ditarik.

11. android.graphics.drawable.shapes: Implements bentuk termasuk *Arc Shape, Oval Shape, Path Shape, Rect Shape, dan Round Rect Shape.*

12. android.hardware

    Implementasi kamera terkait kelas fisik. Kamera ini merupakan kamera perangkat keras, sedangkan android.graphics, camera merupakan konsep grafis yang tidak terkait dengan kamera fisik sama sekali.

13. android.location

Berisi alamat kelas, geocoder, lokasi, *location manager*, dan location provider. Kelas alamat mewakili yang disederhanakan XAL (Bahasa alamat diperluas). geocoder memungkinkan untuk mendapatkan lintang / bujur mengkoordinasikan diberikan alamat. Lokasi mewakili lintang / bujur.

14. android.media

Berisi kelas *Media Player, Media Recorder, Ringtone, Audio Manager, dan Face Detector*. *MediaPlayer*, yang mendukung streaming, digunakan untuk memutar audio dan video. *Media Recorder* adalah digunakan untuk merekam audio dan video. Class *ringtone* digunakan untuk memainkan suara pendek potongan yang dapat digunakan sebagai nada dering dan pemberitahuan. Audio manager bertanggung jawab untuk kontrol volume. Penggunakan *face detector* untuk mendeteksi wajah orang di bitmap.

15. android.net

Implementasi API soket jaringan tingkat dasar. Utama kelas termasuk *Uri Connectivity Manager, Local Socket, dan Local Server Socket*. Hal ini juga dicatat di sini bahwa android mendukung HTTPS pada tingkat browser dan juga pada tingkat jaringan. Android juga mendukung *JavaScript* di *browser*.

16. android.net.wifi

Mengatur konektivitas WiFi. Kelas primer meliputi *Wifi Manager* dan *Wifi Configuration*. *Wifi Manager* bertanggung jawab untuk daftar jaringan dikonfigurasi dan jaringan WiFi yang sedang aktif.

17. android.opengl

Berisi kelas utilitas sekitarnya OpenGL ES operasi. Kelas-kelas utama dari OpenGL ES diimplementasikan dalam berbeda set paket dipinjam dari JSR 239. Paket-paket ini javax.microedition.khronos.opengles, javax.microedition.khronos.egl , dan javax.microedition.khronos.nio. Paket-paket ini pembungkus tipis sekitar pelaksanaan chronos OpenGL ES di C dan C + +.

18. android.os

Merupakan layanan OS diakses melalui Java bahasa pemrograman. Beberapa kelas penting *termasuk Battery Manager, Binder, File Observer, Handler, Looper,* dan *Power Manager. Binder* adalah kelas yang memungkinkan interprocess komunikasi. File Observer membuat tab pada perubahan file. Penggunakan Handler kelas untuk menjalankan tugas-tugas di thread pesan, dan Looper untuk menjalankan thread pesan.

19. android.preference

Memungkinkan aplikasi kemampuan untuk memiliki pengguna mengelola preferensi mereka untuk aplikasi dengan cara yang seragam. Itu kelas utama adalah *Preference Activity, Preference Screen,* dan Preferensi yang diturunkan dari berbagai kelas seperti *Check Box Preference* dan *Shared Preferences.*

20. android.provider

Terdiri satu set penyedia konten *prebuilt* mengikuti antarmuka android.content. Content Provider. Itu penyedia konten termasuk Kontak,

MediaStore, Browser, dan Pengaturan. Ini set interface dan kelas menyimpan metadata untuk struktur data yang mendasarinya.

21. android.sax

    Berisi serangkaian efisien API sederhana untuk XML (SAX) parsing kelas utilitas. Kelas primer termasuk *Element, Root Element*, dan sejumlah antarmuka *Element Listener.*

22. android.speech

    Berisi konstanta untuk digunakan dengan pengenalan suara. Paket ini hanya tersedia di rilis 1.5 dan kemudian.

23. android.speech.tts

    Menyediakan dukungan untuk mengkonversi text to speech. Kelas utama adalah Text To Speech. Anda akan dapat mengambil teks dan meminta sebuah instance dari kelas ini ke antrian teks yang akan diucapkan. Akses ke sejumlah callback untuk memantau ketika pidato tersebut diharuskan, misalnya. Android menggunakan TTS Pico (Text to Speech) mesin dari SVOX.

24. android.telephony

    Berisi Cell Location kelas, Phone Number Utils, dan Telephony Manager. Telephony Manager sebuah memungkinkan untuk menentukan lokasi sel, nomor telepon, nama operator jaringan, jaringan, jenis telepon, dan subscriber Identity Module (SIM) serial nomor.

25. android.telephony.gsm

    Paket ini untuk mengumpulkan lokasi sel berdasarkan GSM dan juga menjadi tuan rumah kelas bertanggung jawab untuk pesan SMS. Paket ini

disebut GSM karena Global System for Mobile Komunikasi adalah teknologi yang awalnya didefinisikan SMS data messaging standar.

26. android.telephony.cdma

    Menyediakan dukungan untuk telepon CDMA.

27. android.text

    Berisi teks-pengolahan kelas.

28. android.text.method

    Menyediakan kelas untuk memasukkan input teks untuk berbagai kontrol.

29. android.text.style

    Menyediakan sejumlah mekanisme styling untuk teks.

30. android.utils

    Berisi Login kelas, DebugUtils, TimeUtils, dan Xml.

31. android.view

    Berisi Menu kelas, View, ViewGroup, dan serangkaian pendengar dan callback.

32. android.view.animation

    Menyediakan dukungan untuk animasi tweening. Kelas-kelas utama meliputi Animasi, serangkaian interpolators untuk animasi, dan satu set kelas animator tertentu yang mencakup Alpha Animation, Scale Animation, Translation Animation, dan Rotation Animation.

33. android.view.inputmethod

    Mengimplementasikan kerangka input-metode arsitektur. Paket ini hanya tersedia di rilis 1.5 dan kemudian.

34. android.webkit

Berisi kelas-kelas yang mewakili web browser. Itu kelas utama termasuk Web View, Cache Manager, dan Cookie Manager.

35. android.widget

Widget berisi widget primer termasuk Button, Checkbox, Kronometer, Analog Clock, Date Picker, Digital Clock, Edit Text, List View, Frame Layout, Grid View, Image Button, Media Controller, Progress Bar, Radio Button, Radio Group, Rating Button, Scroller, Scroll view, Spinner, Tab Widget, Text View, Time Picker, Video View, dan Zoom Button.

36. *com.google.android.*

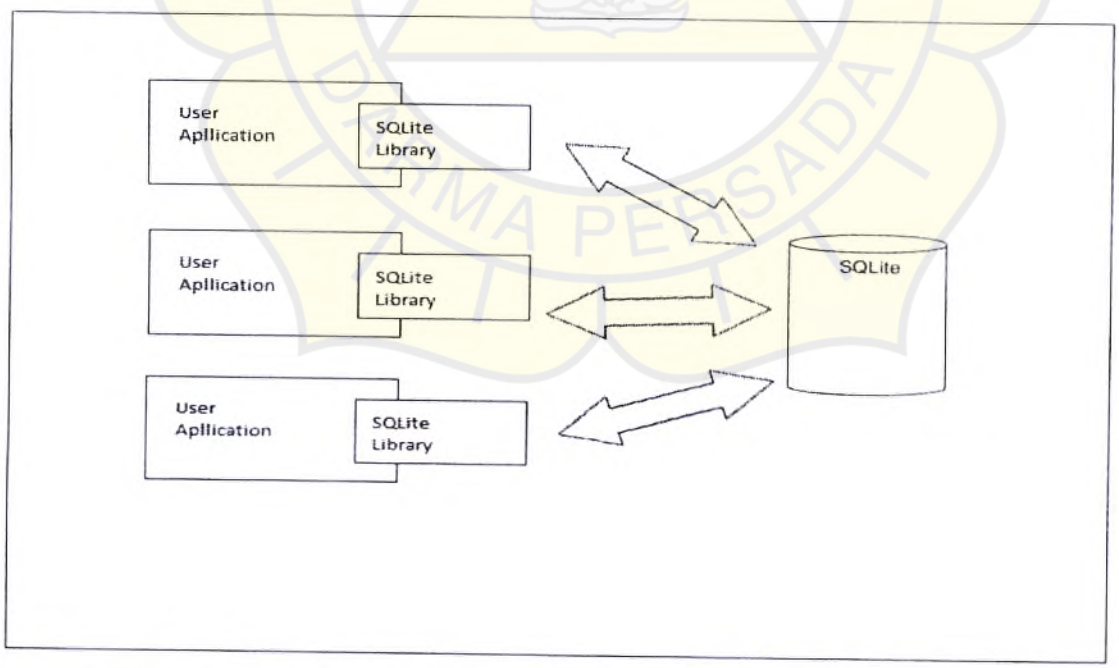
Berisi Map View kelas, Map Controller, dan Map Activity, pada dasarnya kelas dituntut untuk bekerja dengan peta Google. *(Mark L. Murphy : 2009)*

2.2 SQLite

SQLite adalah paket aplikasi yang menyediakan sistem *database relational* (RDBMS) sebagaimana vendor DBMS lainnya seperti Oracle,MySQL, PostreSQL, dan lain-lain. Kata "*Lite*" mengacu pada keringanan/kemudahan dan kemampuan yang sedikit minim. Karakter SQLite bisa diuraikan sebagai berikut :

1. *Serverless*, SQLite tidak memerlukan proses pada *server* atau sistem untuk menjalankan, melainkan hanya sebuah file yang diakses oleh *Library* SQLite.
2. *Zero Configuration*. Tidak ada *server* berarti tidak perlu *setup*, membuat sebuah *database* instan adalah sebuah membuat *file* biasa.

3. *Cross Platform*, sebuah instan *database* berada dalam sebuah *file* yang *cross platform*, dimana tidak memerlukan administrasi.
4. *Self-Contained*, seluruh *library* mengandung keseluruhan dari sistem *database*, yang langsung terintegrasi pada sebuah aplikasi program.
5. *Small-Runtime Footprint*, untuk membangun *database* SQLite hanya butuh kurang dari 1 *Megabyte memory library* dan membutuhkan sedikit *Megabyte memory*.
6. *Transactional*, SQLite *Transactional* memperbolehkan aksi penyimpanan melalui beberapa proses *thread*.
7. *Full Featured*, SQLite mensupport hampir sebagian besar standar SQL92(SQL2).
8. *Highly Reliable*, Tim Pengembang SQLite mengembangkan melalui kode program yang sangat serius telah melalui proses *testing*. *(Mark L. Murphy : 2009)*



Gambar 2.3. Arsitektur dari *Server-less* SQLite *(Mark L. Murphy : 2009)*

2.4 Pengertian UML

Unified Modelling Language (UML) adalah sebuah "bahasa" yang telah menjadi standar dalam industri untuk visualisasi, merancang dan mendokumentasikan sistem piranti lunak. UML menawarkan sebuah standar untuk merancang model sebuah sistem.

Martin Fowler dan Steve Mellor menggolongkan tiga cara/metode menggunakan UML:

1. Sketsa

Developer menggunakan UML untuk membantu menyampaikan beberapa aspek dari sebuah sistem . Sketsa dapat dalam bentuk *forward engineering* atau *reverse engineering*.

Forward engineering menggambarkan sebuah diagram UML terlebih dahulu sebelum membuat kode, sedangkan *reverse engineering* adalah kebalikannya, yaitu membuat diagram dari kode yang sudah ada untuk membantu memahaminya.

2. *Blueprint*

Penggunaan UML sebagai *blueprint* menyampaikan suatu keutuhan. Seperti halnya sketsa, blueprint juga bisa dalam sebuah *forward engineering* maupun dalam *reverse engineering*.

Dalam *forward engineering*, *blueprint* dibuat oleh seorang desainer yang pekerjaannya membuat desain yang detail untuk pengembangan oleh seorang *programmer*.

Dalam *reverse engineering*, *blueprint* bertujuan untuk menyampaikan informasi detail tentang kode baik dalam bentuk dokumen cetak atau sebagai *browser* grafis interaktif. *Blueprint* dapat menunjukan setiap detail sebuah *class* dalam sebuah bentuk grafis yang memudahkan *developer* dalam memahaminya.

Blueprint membutuhkan piranti yang lebih canggih dari pada yang dibutuhkan sketsa untuk menangani detailnya. Piranti *forward engineering* mendukung penggambaran diagram dan menyimpannya dalam suatu tempat untuk menyimpan informasi. Piranti *reverse engineering* membaca *source code* dan menginterpretasinya untuk dimasukan ke dalam ruang penyimpanan, lalu membuat diagram. Piranti yang dapat melakukan kedua *engineering* tersebut, baik *forward* maupun *reverse*, disebut piranti *roundtrip*.

Batas yang memisahkan *blueprint* dan sketsa sebenarnya kabur, tetapi perbedaannya terletak pada fakta bahwa sketsa dibuat tidak lengkap, hanya menunjukan informasi penting, sedangkan *blueprint* cenderung komprehensif dan bertujuan meringkas pemrograman menjadi ktivitas yang simple dan sedikit mekanis. Dengan kata lain, sketsa bersifat *eksploratif* sedangkan *blueprint* bersifat *definitive*.

3. Bahasa pemrograman

UML sebagai bahasa pemrograman adalah tentang bagaimana memodelkan logika *behavioral*. UML 2 menawarkan tiga cara untuk pemodelan *behavior*: *interaction diagram*, *state diagram*, dan *activity diagram*.

1. *Use case diagram*
2. *Class diagram*
3. *Statechart diagram*
4. *Activity diagram*
5. *Sequence diagram*
6. *Collaboration diagram*
7. *Component diagram*
8. *Deployment diagram*

(Munawar, 2005)

Pada peracangan aplikasi ini penulis mengunakan beberapa *diagram* UML yang relevan, di antaranya : *use case diagram, activity diagram, component diagram* dan *deployment diagram*.

2.4.1 Use Case

Use case diagram menggambarkan fungsionalitas yang output dari sebuah sistem, tujuan utamanya adalah "apa" yang sistem lakukan, dan bukan "bagaimana". Sebuah *use case* merepresentasikan sebuah interaksi antara *actor*

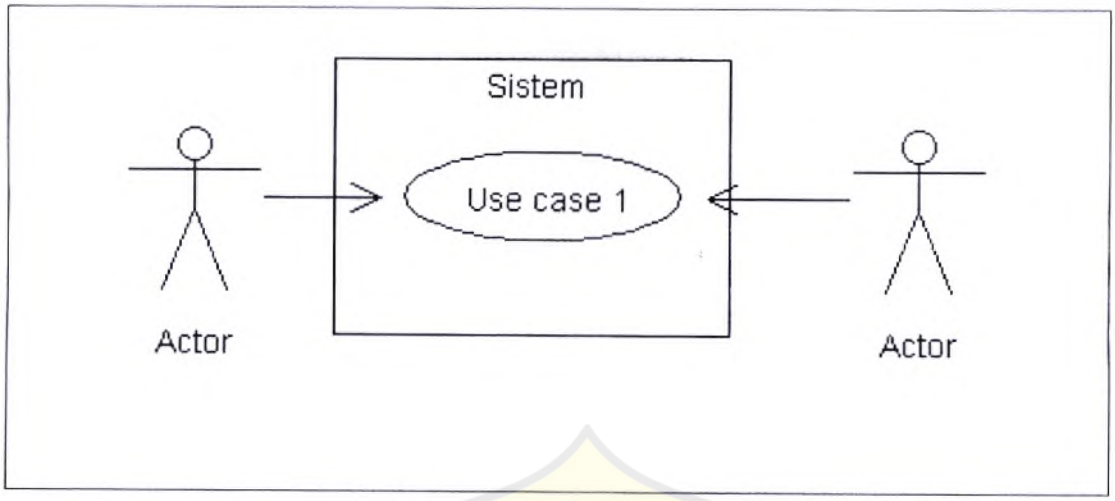
dengan sistem. *Use case* merupakan sebuah pekerjaan tertentu, misalnya *login* ke sistem, meng-*create* sebuah daftar belanja, dan sebagainya. Seorang / sebuah *actor* adalah sebuah entitas manusia atau mesin yang berinteraksi dengan sistem untuk melakukan pekerjaan-pekerjaan tertentu. (Dharwiyanti,Sri & W Satria, Romi. Ilmukomputer.com, 2003).

Use case mendeskripsikan interaksi antara para pengguna sistem itu sendiri, dengan memberi sebuah narasi tentang bagaimana sistem tersebut digunakan. Dua hal (atau istilah) yang tidak dapat dipisahkan dari *use case*, yakni *scenario* dan *actor*. *Scenario* adalah rangkaian langkah-langkah yang menjabarkan sebuah interaksi antara seorang pengguna dengan sebuah sistem .

Dimulai dengan diambilnya salah satu *scenario* sebagai sekenario keberhasilan utama (MSS – *main success scenario*). Isi *use case* dibuat dengan dituliskannya skenario keberhasilan utama sebagai serangkaian langkah-langkah bernomor, kemudian diambil *scenario* lain dan ditulis sebagai ekstensi, hasilnya adalah variasi sekenario keberhasilan utama. Ekstensi-ekstensi tersebut dapat berupa keberhasilan pengguna mencapai keberhasilan.

Sebuah *use case* dapat meng-*include* fungsionalitas *use case* lain sebagai bagian dari proses dalam dirinya. Secara umum *use case* yang di-*include* akan dipanggil setiap kali *use case* yang meng-*include* dieksekusi secara normal.

Sebuah *use case* juga dapat meng-*extend use case* lain dengan *behaviour*-nya sendiri. Sementara hubungan generalisasi antar *use case* menunjukkan bahwa *use case* yang satu merupakan spesialisasi dari yang lain.



Gambar 2.4. Contoh *use case diagram* UML (Munawar, 2005)

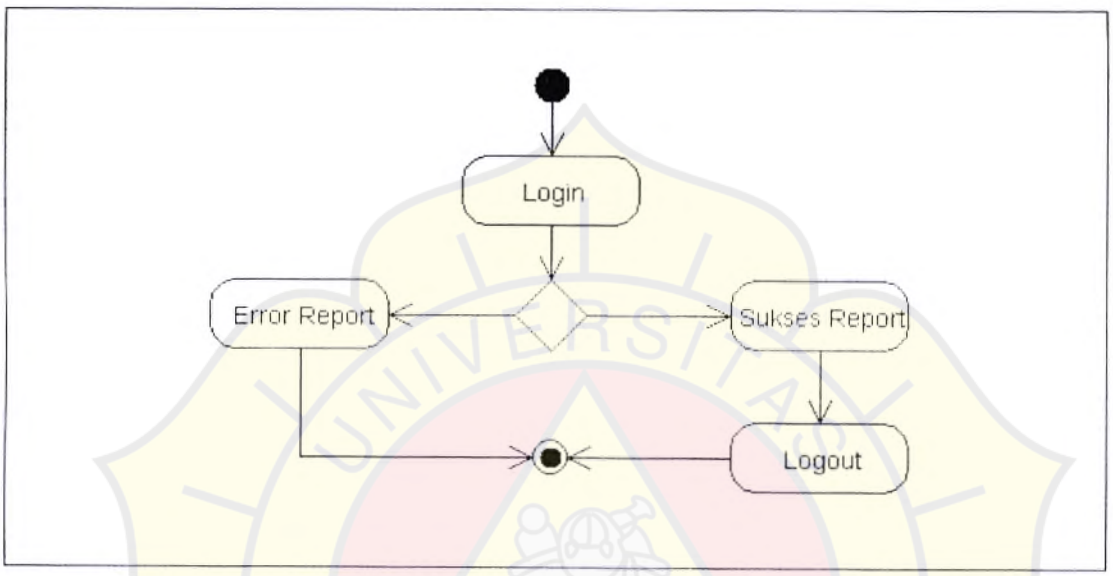
2.4.2 Activity Diagram

Activity diagrams menggambarkan berbagai aliran aktivitas dalam sistem yang berjalan, bagaimana masing-masing alir berawal, *decision* yang mungkin terjadi, dan bagaimana mereka berakhir. *Activity diagram* juga dapat menggambarkan proses paralel yang mungkin terjadi pada beberapa eksekusi.

Activity diagram merupakan *state diagram* khusus, di mana sebagian besar *state* adalah *action* dan sebagian besar transisi di-*trigger* oleh selesainya *state* sebelumnya (*internal processing*). *Activity diagram* lebih menggambarkan proses-proses dan jalur-jalur aktivitas secara umum.

Sebuah aktivitas dapat direalisasikan oleh satu *use case* atau lebih. Aktivitas menggambarkan proses yang berjalan, sementara *use case* menggambarkan bagaimana *actor* menggunakan sistem untuk melakukan aktivitas.

Standar UML menggunakan segi empat dengan sudut membulat untuk menggambarkan aktivitas. *Activity diagram* dapat dibagi menjadi beberapa *object swimlane* untuk menggambarkan objek mana yang bertanggung jawab untuk aktivitas tertentu.



Gambar 2.5. Contoh *Activity diagram* UML (Romi Satrio Wahono, 2003).

2.5. IDE Eclipse

Program android dibangun dengan bahasa java. dan android itu sendiri bukanlah suatu bahasa, melainkan lingkungan yang ada untuk menjalankan aplikasi. Secara teori dibutuhkanlah suatu IDE atau *Integrated Development Environment* untuk memulai pengembangan aplikasi. Bahkan sebenarnya bisa saja tidak menggunakan IDE satupun. *(Jerome Dimarzio , 2008).*

Jika dengan menggunakan aplikasi IDE yang gratis berbasis Java Builder seperti Jbuilder, NetBeans, dan Borland mungkin saja, tetapi Open Handset Alliance dan Google mendukung satu IDE daripada yang lainnya, yaitu Eclipse.

Dalam menjaga pengembangan mobile market, Open Handset Alliance's. Eclipse adalah salah satu yang didukung, gratis, dan IDE yang tersedia. Eclipse juga mudah digunakan, dengan pembelajaran yang ringkas. Menjadikan Eclipse IDE yang menarik dan solid untuk pengembangan program Java.

Open Handset Alliance's merilis Android Plugin untuk Eclipse yang memungkinkan menciptakan spesifik proyek Android, mencompile-nya dan menggunakan emulatornya dan juga men-debug-nya.

Untuk dapat menjalankan program Eclipse, diharuskan melengkapi IDE Eclipse dengan beberapa komponen berikut untuk dalam proses pengembangan proyek Android dapat berjalan dengan seharusnya :

1. JRE (Java Runtime Environment)

JRE adalah lingkungan atau Environment yang digunakan agar program java dapat berjalan.

2. Android Development Tools (ADT)

ADT digunakan sebagai Library tambahan selain Library Native yang ada di IDE, dimana semua Library/package digunakan dan dipanggil hampit disetiap class.

3. JDK (Java Development Kit)

JDK digunakan untuk mendukung pengembangan selanjutnya setelah proses coding, dimana JDK akan berjalan untuk mengcompile, mendebug, dan menjalankan aplikasi yang sudah dibuat. Dengan bantuan Emulator, proyek

android dapat dijalankan sebagaimana nanti akan berjalan di dalam handset nantinya. *(Jerome Dimarzio , 2008).*

Android menggunakan Java Virtual Machine yang khusus bernama Davik yang menggunakan bytecote special, karenanya kita tidak dapat menjalankan Android dengan bytecode standar. Android menyediakan sebuah alat bernama "dx" yang memungkinkan untuk mengkonversi file-file class Java menjadi sebuah file eksekusi Dalvik tersebut "dex". Android akan mempaket program menjadi aplikasi berupa file apk yang siap diinstal dalam sebuah ponsel cerdas.

Versi JDK yang diperlukan untuk lingkungan Android SDK adalah JDK 5 atau JDK 6. Dalam sistem operasi linux tidak dapat menggunakan GNU Java Compiler (gcj) yang biasa dalam terdapat pada paket distribusi system operasi linux. (Chris Web, *2009).*

2.6. XML

XML berasal dari kata eXtensible Markup Language. Ini bahasa yang dikembangka oleh W3C (World Wide Web Consortium), utamanya untuk mengatasi keterbatasan dari HTML. Organisasi W3C bertugas untuk mengembangkan dan memelihara sebagian besar situs standar, yang kebanyakannya adalah HTML.

HTML adalah bahasa yang termasuk popular. Berdasarkan beberapa penelitian, ada 800 juta halaman situs berdasarkan HTML. HTML didukung oleh

ribuan aplikasi termasuk browser, editor, email, software, database, pengolah kontak dan masih banyak lagi.

Awalnya, situs adalah solusi dari penyebaran dokumen ilmiah. Sekarang, menjadi media yang sangat berkembang, setara dengan kertas print dan TV. Dan yang lebih pentingnya lagi, Situs menjadi media interaktif karena mendukung seperti took online, perbankan elektronik, dan perdagangan dan forum. Untuk mengakomodasi populeritas ini, HTML ditambahkan beberapa tahun ini. Banyak Tag baru yang telah dikenalkan. Pada versi awal, HTML memiliki selusin tag, dan HTML 4.0 yang terbaru telah mencapai 100 tag. (Benoit Marchal, 1999).

XML digunakan dibeberapa hal, sebagai berikut :

1. Pemeliharaan situs yang besar untuk penyederhanaan dokumen.
2. Pergantian informasi antara dua organisasi
3. Penggunaan database.
4. Konten yang terhubung dengan situs yang lain
5. Aplikasi komersial diantara organisasi yang berbeda untuk kolabosasi para konsumen.
6. Aplikasi ilmiah dengan tag baru untuk tujuan matematikal.
7. Buku elektronik untuk mengekspresikan hak dan kepemilikan.
8. Lingkungan handheld dan smartphones (Benoit Marchal, 1999).