BAB II

LANDASAN TEORI

2.1 Perancangan

Definisi perancangan menurut Azhar Susanto dalam bukunya yang berjudul*Sistem Informasi Manajemen Konsep dan Pengembangannya* adalah sebagaiberikut: “perancangan adalah spesifikasi umum dan terinci dari pemecahanmasalah berbasis komputer yang telah dipilih selama tahap analisis”(2004:332).

Definisi menurut John Burch dan Gary Grudnitski yang telah diterjemahkan olehJogiyanto HM dalam bukunya yang berjudul *Analisis dan Desain SistemInformasi* menyebutkan bahwa: ”desain sistem adalah penggambaran,perencanaan dan pembuatan sketsa atau pengaturan dari beberapa elemen yangterpisah dari suatu kesatuan yang utuh dan berfungsi.”(2005:196)

Berdasarkan dua definisi perancangan di atas maka penulis dapatmenyimpulkan bahwa perancangan adalah penggambaran, perencanaan danpembuatan sketsa yang kemudian diterjemahkan ke dalam sebuah konseprancangan sebagai pemecahan masalah berbasis komputer yang telah dipilihselama tahap analisis.

2.2 Parfume

Parfum atau minyak wangi adalah campuran minyak esensial dan senyawa aroma, fiksatif, dan pelarut yang digunakan untuk memberikan bau wangi untuk tubuh manusia, obyek, atau ruangan, Parfum benda yang pastinya sudah tidak asing lagi dalam kehidupan kita. Apalagi saat ini aroma parfum yang ditawarkan

sudah semakin beragam, baik yang dikhususkan untuk pria, wanita. Dan tidak jarang, aroma parfum bisa menjadi sebuah pick up line andalan seorang pria ketika ingin berkenalan dengan wanita, begitupula sebaliknya . ada beberapa katagori dalam parfume, yaitu :

1. Floral. Wewangian jenis floral umumnya memiliki aroma berkesan manis karena bahannya terdiri dari beragam jenis bunga. Dari semua jenis bunga tersebut, biasanya ada satu yang aromanya menonjol semisal mawar, melati, maupun bunga nila (lavender). Selain aroma manis, biasanya wewangian jenis floral juga memiliki aroma seperti bedak. Selain memberi rasa menenangkan, juga bisa mempercepat kerja otak dalam menyerap informasi, sekaligus melancarkan proses komunikasi. karenanya, jika anda hendak menghadiri meeting penting, kenakan parfum dengan aroma floral.
2. Oriental/Amber. Wewangian dengan keharuman oriental yang eksotis ini biasanya memiliki aroma yang tajam menggairahkan karena umumnya menggunakan ekstrak aroma rempah-rempah, seperti jahe (contohnya Bvlgari Blv), biji vanila, pala, bahkan juga dicampur dengan ekstrakhormon hewan seperti musk, vetiver (umumnya digunakan hormon sintesis karena hormon hewan yang asli sangatlah mahal). selain tajam menusuk, kesan dari wewangian oriental ini adalah hangat, menggairahkan, sensual serta tentunya seksi, sangat sesuai untuk kencan dengan pasangan Anda
3. Fresh/Citrus. Jika anda suka dengan kesegaran yang anda dapatkan setelah selesai mandi, maka wewangian dengan aroma Fresh sangatlah sesuai

untuk anda. Wewangian jenis ini biasanya bahan utamanya adalah buah-buahan seperti jeruk, lemon, ataupun jenis tanaman hijau seperti rerumputan pinus, cemara. Weangian jenis ini sesuai sekali untuk menemani aktivitas harian anda, sehingga tidak heran jika wewangian segar seperti L'Eau d'issey Miyake, Davidoff Cool Water for Woman digemari oleh mereka yang sportif, serta aktif dan memiliki berbagai kegiatan.

4. Woody. Wewangian ini didominasi oleh aroma kayu khas dari kepulauan Pasifik Selatan yang ciri khasnya ditandai oleh aroma sandalwood dan cedar. Bagi anda, yang suka berpetualang, independen dan menyukai tantangan, wewangian ini akan sangat cocok menemani hari-hari anda.
5. Aqua memiliki aroma yang mendiskripsikan pribadi yang enerjik,sporty,dan aktif namun secara bersamaan tanpa menghilangkan sisi feminin serta keanggunanya. keharumanya sangat natural dan mudah di gemari oleh pria maupun wanita.

2.2.1 Sejarah Parfume

Parfum sudah dikenal sejak ribuan tahun yang lalu - kata "parfum" berasal dari bahasa Latin per fume artinya "melalui asap". Salah satu kegunaan parfum tertua berupa bentuk pembakaran dupa dan herbal aromatik yang digunakan dalam pelayanan keagamaan, seringkali untuk aromatik gums, kemenyan dan mur, dikumpulkan dari pohon. Mesir adalah yang pertama memasukkan parfum ke budaya mereka diikuti oleh Cina kuno, Hindu, Israel, Carthaginians, Arab, Yunani, dan Romawi. Penggunaan awal dari botol parfum adalah di Mesir sekitar

1000 SM. Mesir menemukan gelas dan botol parfum adalah salah satu penggunaan umum pertama untuk kaca.

2.3 Android

Android adalah sebuah sistem operasi perangkat *mobile* berbasis linux yang mencakup sistem operasi, *middleware* dan aplikasi. Android menyediakan *platform* terbuka bagi para pengembang untuk menciptakan aplikasi mereka.

Android merupakan generasi baru *platform mobile* yang memberikan kesempatan kepada pengembang untuk melakukan pengembangan sesuai dengan yang diharapkan. Sistem operasi yang mendasari Android merupakan lisensi di bawah naungan GNU, General *Public* License Versi 2(GPLv2), yang biasa dikenal dengan istilah *Copyleft*. Istilah *copyleft* ini merupakan lisensi yang setiap perbaikan oleh pihak ketiga harus terus jatuh di bawah *terms*.

Distribusi Android berada di bawah lisensi Apache Software (ASL/Apache2), yang memungkin untuk distribusi kedua atau seterusnya. Pengembang aplikasi Android diperbolehkan untuk mendistribusikan aplikasi mereka di bawah skema lisensi apapun yang mereka inginkan.

Pengembang memiliki beberapa pilihan dalam membuat aplikasi yang berbasis Android. Namun kebanyakan pengembang menggunakan Eclipse sebagai IDE untuk merancang aplikasi mereka. Hal ini diikarenakan Eclipse mendapat dukungan langsung dari Google untuk menjadi IDE pengembangan aplikasi Android.

Aplikasi Android dapat dikembangkan pada berbagai sistem operasi, diantaranya adalah:

a. Windows XP/Vista/7

b. Mac OS X (Mac OS X 10.48 atau yang lebih baru)

c. Linux

2.3.1 Sejarah Android

Pada saat perilisan perdana Android pada tanggal 5 November 2007, Android bersama Open Handset Alliance mendukung pengembangan standar terbuka pada perangkat seluler tersebut. Di sisi lain, Google merilis kode-kode Android di bawah lisensi Apache. Sehingga terdapat dua jenis *distributor* sistem operasi Android yaitu yang mendapat dukungan penuh dari Google dan yang mendapat dukungan penuh dari Open Handset Distribution (OHD).

Telepon selular pertama yang menggunakan sistem operasi Android adalah HTC Dream yang dirilis pada 22 Oktober 2008. Pada 9 Desember 2008, diumumkan anggota baru yang bergabung dalam program kerja Android ARM Holdings, Atheros Communication yang diproduksi oleh Asustek Computer Inc, Garmin Ltd, Softbank, Sony Ericsson, Toshiba Corp dan Vodafone Group Plc.

Hingga saat ini terdapat beberapa versi dari sistem operasi Android, antara lain:

a. Android versi 1.1

Dirilis pada 9 Maret 2009. Android versi ini dilengkapi dengan adanya jam, alarm, *voice search*, pengiriman pesan dengan Gmail dan pemberitahuan *email*.

b. Android versi 1.5 (Cupcake)

Dirilis pada Mei 2009. Terdapat pembaruan dari versi 1.1 diantaranya adalah fitur *upload* video ke Youtube dan gambar ke Picasa langsung dari telepon,

dukungan *bluetooth* A2DP, kemampuan terhubung secara otomatis ke headset *bluetooth*, animasi layar, dan *keyboard* pada layar yang dapat disesuaikan dengan sistem.

c. Android versi 1.6 (Donut)

Dirilis pada September 2009. Pembaruan yang terdapat pada versi ini diantaranya adalah proses pencarian yang lebih baik, penggunaan baterai indikator dan kontrol applet VPN. Fitur lainnya adalah memungkinkan pengguna untuk memilih foto yang akan dihapus, kamera, camcorder dan galeri yang diintegrasikan, CDMA/EVDO, 802.1x, VPN, Gestures, *Text-to-speech engine*.

d. Android versi 2.1 (Éclair)

Dirilis pada 3 Desember 2009. Perubahan yang dilakukan adalah pengoptimalan *hardware*, peningkatan Google Maps 3.1.2, perubahan UI dengan *browser* baru dan dukungan HTML5, daftar kontak yang baru, dukungan *flash* untuk kamera 3.2 MP, digital zoom dan *bluetooth* 2.1.

e. Android versi 2.2 (Froyo)

Dirilis pada 20 Mei 2010. Versi Android inilah yang sekarang banyak digunakan sebagai standar sistem operasi mereka. Terdapat perubahan yang cukup signifikan dari versi sebelumnya diantaranya adalah kerangka aplikasi memungkinkan penggunaan dan penghapusan komponen yang tersedia, Dalvik Virtual Machine (DVM) yang dioptimalkan untuk perangkat *mobile*, grafik di 2D dan 3D berdasarkan *libraries* OpenGL, SQLite, mendukung berbagai format audio dan video, GSM, *bluetooth*, EDGE, 3G, Wifi, kamera, *Global Positioning System* (GPS), kompas dan accelerometer.

f. Android versi 2.3 (GingerBread)

Dirilis pada 6 Desember 2010. Beberapa perbaikan fitur dari versi sebelumnya adalah SIP-*based* VoIP, *Near Field Communications* (NFC), *gyroscope* dan sensor, *multiple cameras support*, *mixable audio effect* dan *download manager*.

g. Android versi 3.0 (Honeycomb)

Dirilis tahun 2011. Android versi ini dirancang khusus untuk tablet, sehingga terdapat perbedaan dari fitur UI (*User Interface*). Honeycomb sengaja dibuat untuk layar yang lebih besar dan juga dapat mendukung *multiprocessor*.

h. Android versi 4.0 (Ice Cream)

Versi ini masih dalam pengembangan. Dari berbagai informasi menyebutkan bahwa versi Ice Cream merupakan gabungan antara versi Gingerbread dengan Honeycomb. Sehingga bisa digunakan untuk ponsel maupun tablet. Dan kemungkinan dirilis pada quarter ke 4 tahun 2011.

2.3.2 *Dalvik Virtual Machine*

Salah satu elemen penting dari Android adalah *Dalvik Virtual Machine* (DVM). Android berjalan di dalam DVM bukan di *Java Virtual Machine (JVM).* DVM dirancang untuk memastikan bahwa beberapa fitur berjalan lebih efisien pada perangkat *mobile*.

Tidak seperti JVM yang "*stack base*", DVM adalah sebuah "*register-base architecture*". DVM ditulis oleh Dan Bornsten dan beberapa *engineers* dari Google lainnya. Jadi jelas bahwa DVM tidak sama dengan JVM.

DVM menggunakan kernel Linux untuk menangani fungsionalitas tingkat rendah termasuk keamanan, threading dan proses serta manajemen memori. DVM mengeksekusi *executable file*, sebuah format yang dioptimalkan untuk memastikan memori yang digunakan sangat kecil. *Executable file* diciptakan dengan mengubah kelas bahasa Java dan dikompilasi menggunakan *tools* yang disediakan dalam SDK Android.

2.3.3 Android SDK (*Software Development Kit*)

Android SDK adalah *tools* API (*Application Programming Interface*) yang diperlukan untuk memulai mengembangkan aplikasi pada *platform* Android menggunakan bahasa pemrograman Java.

Beberapa fitur Android yang paling penting adalah sebagai berikut:

a. *Framework* aplikasi yang mendukung penggantian komponen dan reusable

b. DVM dioptimalkan untuk perangkat *mobile*

c. Integrated *browser* berdasarkan engine *open source* WebKit

d. Grafis yang dioptimalkan dan didukung oleh *libraries* grafis 2D, grafis 3D berdasarkan spesifikasi OpenGL ES 1.0

e. SQLite untuk penyimpanan data

f. Dukungan untuk audio, video dan gambar

g. *Bluetooth*, EDGE, 3G, Wifi

h. Kamera, GPS, kompas dan *accelerometer*

i. Lingkungan development yang lengkap dan kaya termasuk perangkat *emulator*, *tools* untuk *debugging*, profil dan kinerja memori serta plugins untuk IDE Eclipse.

2.3.4 Arsitektur Android

Secara garis besar arsitektur Android dapat dijelaskan dan digambarkan sebagai berikut:

a. *Application* dan *Widget*

Application dan *Widget* ini adalah *layer* dimana kita berhubungan dengan aplikasi saja. Di *layer* terdapat aplikasi inti termasuk klien *email*, program SMS, kalender, peta, *browser*, kontak, dan lain-lain. Semua aplikasi ditulis dengan menggunakan bahasa pemrograman JAVA.

b. *Application Framework*

Application Framework adalah *layer* untuk melakukan pengembangan / pembuatan aplikasi yang akan dijalankan di sistem operasi Android, karena pada *layer* inilah aplikasi dapat dirancang dan dibuat, seperti *content provider* yang berupa SMS dan panggilan telepon.

Komponen-komponen yang termasuk di dalam *Application Framework* adalah sebagai berikut:

a) Views

b) *Content Provider*

c) *Resource Manager*

d) *Notification Manager*

e) *Activity Manager*

c. *Libraries*

Libraries adalah *layer* tempat fitur-fitur Android berada, biasanya para pengembang aplikasi mengakses *libraries* untuk menjalankan aplikasinya.

d. Android *Runtime*

Layer yang membuat aplikasi Android dapat dijalankan di mana dalam prosesnya menggunakan implementasi Linux. Dalvik *Virtual Machine* merupakan mesin yang membentuk dasar kerangka aplikasi Android.

Di dalam Android *Runtime* dibagi menjadi dua bagian yaitu:

1) Core *Libraries*

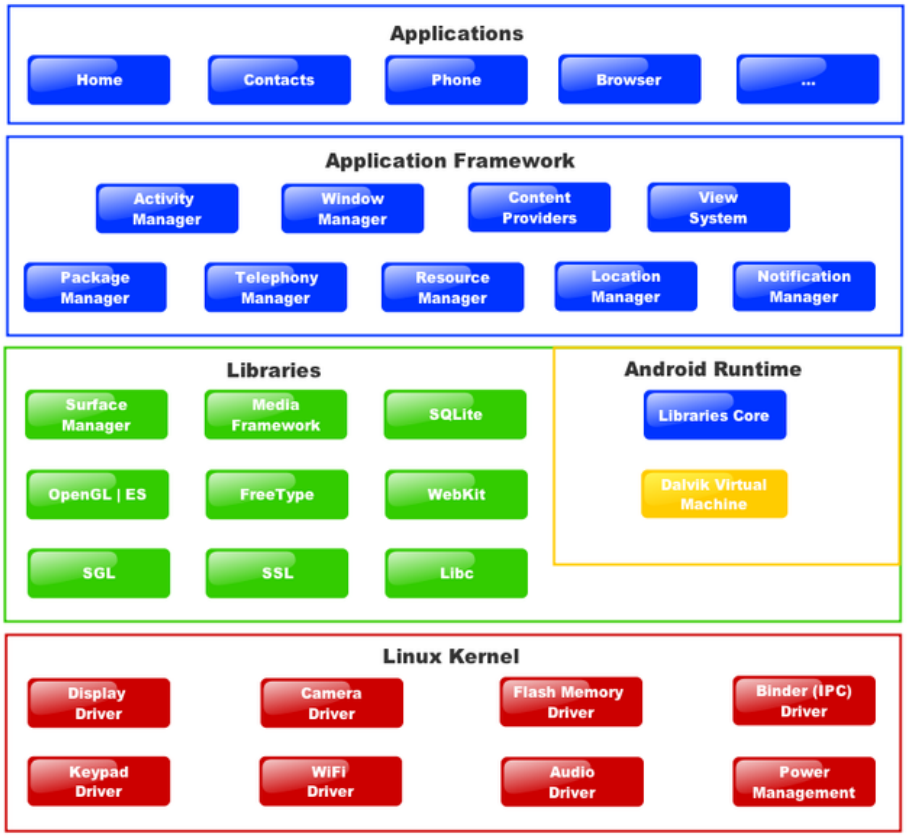
Aplikasi Android dibangun dalam bahasa Java, sementara DVM bukan merupakan *virtual machine* untuk Java. Sehingga diperlukan *libraries* yang berfungsi untuk menterjemahkan bahasa Java/C yang ditangani oleh *Core Libraries*

2) *Dalvik Virtual Machine*

Virtual Machine berbasis *register* yang dioptimalkan untuk menjalankan fungsi-fungsi secara efisien, dimana merupakan pengembangan yang mampu membuat Linux kernel untuk melakukan *threading* dan manajemen tingkat rendah.

e. Linux Kernel

Linux Kernel adalah *layer* dimana inti sistem operasi dari Android itu berada. Berisi file system yang mengatur system processing *memory*, *resource*, *drivers*, dan sistem-sistem operasi Android lainnya. Linux Kernel yang digunakan Android adalah Linux Kernel *release* 2.6.



Gambar 2.1 Arsitektur Android

(Sumber : NazruddinSafaat H.2012.*Pemrograman Aplikasi Mobile Smartphone dan Tablet PC berbasis Android*)

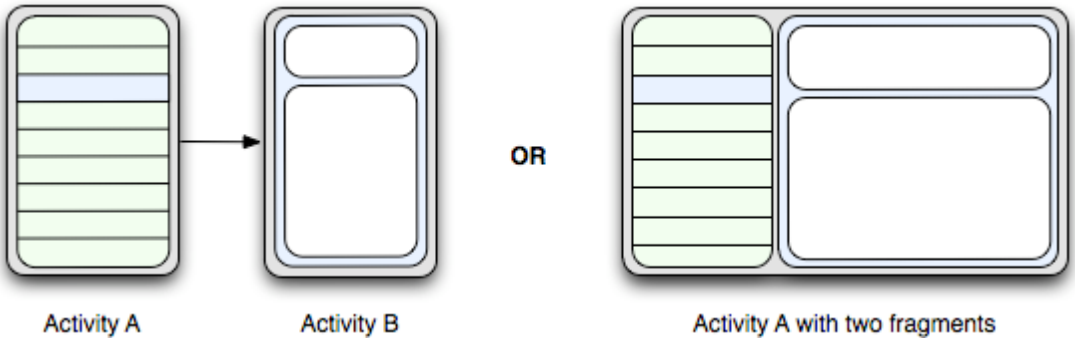
2.3.5 Fundamental Aplikasi

Aplikasi Android ditulis dengan menggunakan bahasa pemrograman Java. Kode Java dikompilasi bersama dengan data file *resource* yang dibutuhkan oleh aplikasi, dimana prosesnya di-*package* oleh *tools* yang dinamakan apt *tools* ke dalam paket Android sehingga menghasilkan *file* dengan ekstensi apk. File apk itu lah yang disebut dengan aplikasi dan siap diinstall di perangkat *mobile*.

Ada empat jenis komponen pada aplikasi Android, yaitu:

a. Activities

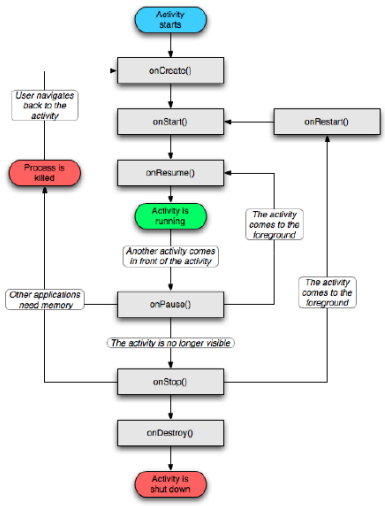
Suatu *activity* akan menyajikan *user interface* (UI) kepada pengguna, sehingga pengguna dapat melakukan interaksi. Sebuah aplikasi Android bisa jadi hanya memiliki satu *activity*, tetapi umumnya aplikasi memiliki banyak *activity* tergantung pada tujuan aplikasi dan desain dari aplikasi tersebut.



Gambar 2.2 Contoh Penggunaan *Activity*

(Sumber : NazruddinSafaat H.2012.*Pemrograman Aplikasi Mobile Smartphone dan Tablet PC berbasis Android***)**

Satu *activity* biasanya akan dipakai untuk menampilkan aplikasi yang bertindak sebagai *user interface*(UI) saat aplikasi diperlihatkan kepada *user*. Untuk pindah dari satu *activity* ke *activity* lain dapat dilakukan dengan satu *even*, misalnya klik tombol, memilih opsi atau menggunakan trigger tertentu. Secara hirarki sebuah window *activity* dinyatakan dengan *method Activity.setContentView*(). *Content View* adalah objek yang berada pada *root* hirarki.



Gambar 2.3 *Activity Lifecycle*

(Sumber : NazruddinSafaat H.2012.*Pemrograman Aplikasi Mobile Smartphone dan Tablet PC berbasis Android***)**

b. Service

Service tidak memiliki *Graphic User interface* (GUI), tetapi service berjalan secara background. Misalnya ketika ponsel sedang memainkan music di media p*layer*, namun user juga sedang mengetik SMS. *Activity* yang bisa terlihat oleh user adalah *activity* SMS, sedangkan *media player* yang sedang memainkan musik tidak terlihat, maka media p*layer* tersebut sedang menjalankan sebuah *service*.

c. Broadcast Receiver

Broadcast Receiver berfungsi menerima dan bereaksi untuk menyampaikan notifikasi. Sebagai contoh ketika baterai dalam keadaan *low* atau misalnya sudah berhasil menerima data hasil *download* atau pun *bluetooth*. *Broadcast Receiver* tidak memiliki *user interface* (UI), tetapi memiliki sebuah

activity untuk meresponn informasi yang mereka terima, atau mungkin menggunakan *Notification Manager* untuk memberitahu kepada pengguna, seperti lampu layar atau vibrating dan lain sebagainya.

d. Content Provider

Content Provider membuat kumpulan aplikasi data secara spesifik sehingga bisa digunakan oleh aplikasi lain. Data disimpan dalam file sistem seperti database SQLite. *Content Provider* menyediakan cara untuk mengakses data yang dibutuhkan oleh suatu *activity*, misalnya ketika menggunakan aplikasi yang membutuhkan peta(*Map*), atau aplikasi yang membutuhkan untuk mengakses data kontak dan navigasi, maka disinilah fungsi *Content Provider*.

2.4 JAVA

Java merupakan bahasa pemrograman berorientasi objek yang bersifat *open source* dan bisa dijalankan di berbagai *platform*. Java mengadopsi sintaks dari bahasa C dan C++. Setiap program yang ditulis dengan bahasa Java awalnya dikompilasi menjadi sebuah kode objek yang disebut *bytecode*. Hasil dari *bytecode* adalah file berekstensi *.*class*, selanjutnya file *.*class* tersebut akan diterjemahkan baris demi baris(*interpreter*).

Dalam terminologi Java terdapat istilah "*write once, run anywhere*". Ini berarti bahwa sekali menulis program Java dan melakukan kompilasi terhadap program Java tersebut, maka *bytecode*nya bisa dijalankan di dalam *platform* manapun selama *platform* tersebut terdapat *Java Virtual Machine (JVM).*

2.4.1 Sejarah Java

Bahasa pemrograman terlahir dari sebuah projek yang diberi nama The Green Project. Proyek tersebut berlangsung selama 18 bulan, mulai dari awal tahun 1991 hingga musim panas 1992. Proyek ini dilakukan oleh Patrick Naughton, Mike Sheridan, James Gosling dan Billy Joy beserta sembilan pemrogram lainnya dari Sun Microsystems.

Pada sekitar bulan Maret 1995, untuk pertama kali kode sumber Java versi 1.0 dibuka. Awalnya produk tersebut dinamakan Oak. Namun ternyata nama tersebut telah terdaftar sebagai merek dagang lain, sehingga mereka merubahnya menjadi Java yang diambil dari kopi murni yang digiling langsung yang juga merupakan kesukaan James Gosling.

Sekitar tahun 1996 mereka mengeluarkan Java versi 1.0 Release. Java versi ini menyertakan banyak paket standar awal yang terus dikembangkan pada versi selanjutnya:

a. Java.lang

b. Java.io

c. Java.util

d. Java.net

e. Java.awt

f. Java.applet

2.4.2 Konsep Pemrograman Berorientasi Objek

Pemrograman berorientasi objek atau lebih lazim dikenal dengan sebutan asing *Object Oriented Programming* (OOP) adalah inti dari pemrograman Java.

Semua program Java adalah objek. Berikut ini adalah ciri dari pemrograman berbasis objek:

2.4.2.1 Abstraksi

Abstraksi adalah pengabstrakan atau penyembunyian kerumitan dari suatu proses. Sebagai contoh sebuah perangkat telepon seluler terdiri dari puluhan atau ribuan komponen kecil seperti transistor, baut, lampu indikator dan lain-lain. Namun pengguna tidak perlu mengetahui bagaiman setiap komponen bisa bekerja sesuai fungsinya. Pengguna hanya cukup mengetahui bagaimana cara mengoperasikan telepon seluler tersebut. Ini artinya si pembuat telepon seluler tersebut telah menyembunyikan kerumitan-kerumitan proses yang terdapat pada telepon selular tersebut.

2.4.2.2 Pengkapsulan/Enkapsulasi

Dalam melakukan pembungkusan kode dan data di dalam Java terdapat tiga tingkat akses, yaitu:

a. Private

Dengan mendeklarasi data dan *method* menggunakan tingkat akses *private*, maka data dan *method* tersebut hanya dapa diakses oleh kelas yang memilikinya saja. Ini artinya data dan *method* tersebut tidak boleh diakses atau digunakan oleh kelas-kelas lain yang terdapat di dalam program. Untuk mendeklarasikan data atau *method* dengan tingkat akse *private* maka harus menggunakan kata kunci *private*.

b. *Protected*

Data atau *method* yang dideklarasikan dengan tingkat akses *protected* dapat diakses oleh kelas yang memilikinya dan juga oleh kelas-kelas yang masih turunan. Untuk mendeklarasikan data atau *method* yang menggunakan tingkat akses *protected* maka harus menggunakan kata kunci *protected*.

c. *Public*

Data atau pun *method* yang dideklarasikan dengan tingkat akses *Public* artinya data tersebut bisa diakses oleh semua bagian di dalam program. Untuk mendeklarasikan data atau *method* yang menggunakan tingkat akses p*ublic* maka harus menggunakan kata kuci *public*.

2.4.2.3 Pewarisan

Pewarisan (*inheritance*) adalah salah satu ciri pemrograman berorientasi objek, yang menyatakan bahwa suatu kelas dapat diturunkan lagi menjadi kelas-kelas baru lainnya sehingga dapat membentuk sebuah hirarki.

Pewarisan yang dimaksud dalam pemrograman berorientasi objek sama seperti pewarisan di dunia nyata. Sebagai contoh, setiap anak pasti mewarisi karakteristik atau beberapa sifat dari orang tuanya. Begitu juga di dalam pemrograman berorientasi objek, setiap kelas dapat diturunkan menjadi kelas-kelas baru dan mewarisi beberapa sifat atau perilakunya.

2.4.2.4 Polimorfisme

Polimorfisme adalah kemampuan suatu objek untuk mengungkapkan banyak hal melalui satu cara yang sama. Polimorfisme merupakan hal paling esensial dalam konsep pemrograman berorientasi objek karena polimorfisme

mengizinkan kelas induk untuk mendefinisikan sebuah *method* genera(bersifat umum) untuk semua kelas turunannya dan selanjutnya kelas-kelas turunan dapat memperbarui implementasi dari *method* tersebut secara lebih spesifik sesuai dengan karakteristiknya masing-masing.

2.5 *Unified Modeling Language* (UML)

2.5.1. Definisi *Unified Modeling Language* (UML)

Berikut ini definisi *Unified Modeling Language* (UML) menurut para ahli:

1. Menurut (Hend, 2006) "*Unified Modeling Language* (UML)adalah bahasa yang telah menjadi standard untuk visualisasi, menetapkan, membangun dan mendokumentasikan artifak suatu sistem perangkat lunak".
2. Menurut (Adi Nugroho : 2005). "*Unified Modeling Language* (UML) adalah alat bantu analisis serta perancangan perangkat lunak berbasis objek".
3. Menurut(Joomladarihttp://soetrasoft.com:2007). "*Unified Modeling Language* (UML)merupakan standard modeling language yang terdiri dari kumpulan-kumpulan diagram, dikembangkan untuk membantu para pengembang sistem dan *software* agar bisa menyelesaikan tugas-tugas seperti: Spesifikasi, Visualisasi, Desain Arsitektur, Konstruksi, Simulasi dan testing serta Dokumentasi".

Berdasarkan beberapa pendapat yang dikemukakan diatas dapat ditarik kesimpulan bahwa "*Unified Modeling Language* (UML)adalah sebuah bahasa yang berdasarkan grafik atau gambar untuk menvisualisasikan, menspesifikasikan, membangun dan pendokumentasian dari sebuah sistem pengembangan perangkat lunak berbasis OO(*Object Oriented*)".

Unified Modelling Language (UML) adalah sebuah bahasa untuk menentukan, visualisasi, konstruksi, dan mendokumentasikan *artifacts* dari sistem *software*, untuk memodelkan bisnis, dan sistem *nonsoftware* lainnya. *Artifacts* adalah sepotong informasi yang digunakan atau dihasilkan dalam suatu proses rekayasa *software*. *Artifacts* dapat berupa model, deskripsi, atau software. Untuk membuat suatu model, UML memiliki diagram grafis yang diberi nama berdasarkan sudut pandang yang berbeda-beda terhadap sistem dalam proses analisa atau rekayasa. Diagram grafis tersebut antara lain

1. Use Case diagram

Use Case diagram adalah jenis diagram perilaku yang didefinisikan dan diciptakan dari analisis kasus. Tujuannya adalah untuk menyajikan gambaran grafis dari fungsionalitas yang disediakan oleh sistem dalam hal pelaku, tujuan mereka (direpresentasikan sebagai kasus penggunaan), dan setiap ketergantungan antara kasus-kasus digunakan. Dan tujuan utama dari sebuah diagram *Use Case* adalah untuk menunjukkan apa yang dilakukan fungsi sistem yang aktor. Peran aktor dalam sistem dapat digambarkan. *Use Case* diagram secara formal dimasukkan dalam dua bahasa pemodelan yang didefinisikan oleh OMG: Unified Modeling Language (UML) dan Sistem Modeling Language(SysML).

Sebuah *Use Case* menggambarkan suatu urutan tindakan yang memberikan sesuatu yang bernilai terukur untuk aktor. Dalam *Use Case* diagram ada empat jenis hubungan yang saling terkait satu sama dan sering digunakan dalam praktek. Empat hubungan ini adalah

a. Include

Relasi ini menyatakan bahwa satu *Use Case* selalu menggunakan fungsionalitas yang disediakan oleh *Use Case* lainnya

b. Extend

Relasi ini menyatakan bahwa suatu *Use Case* tidak selalu menggunakan fungsionalitas yang disediakan oleh *Use Case* lainnya.

c. Generalization

Generalization ada 2 jenis yaitu : generalisasi antar aktor dan generalisasi antar *Use Case*. Generalisasi antar actor adalah relasi antara satu actor dengan actor yang lain. Dimana salah satu actor berperan sebagai Actor yang diwariskan sifatnya (Base Actor) dan Actor yang lain sebagai Actor yang mewarisi sifatnya (Child Actor). Sedangkan Generalisasi antar *Use Case*adalah relasi antara satu *Use Case* dengan *Use Case* yang lain. Dimana salah satu *Use Case* berperan sebagai *Use Case* yang diwariskan kerjanya (Base *Use Case*) dan *Use Case* yang lain mewarisi kerja (Child *Use Case*).

d. Associations

Adalah komunikasi antara aktor dengan use yang berada dalam sebuah sistem. Asosiasi ini menggambarkan apa yang actor bisa lakukan dengan sistem. Dimana setiap use menggambarkan satu buah perilaku yang bisa actor lakukan.

2. Class diagram

Class diagram adalah jenis diagram struktur statis yang menggambarkan struktur sistem dengan menunjukkan kelas sistem, atribut mereka, operasi (atau) metode dan hubungan antara kelas-kelas. Class diagram digunakan untuk

pemodelan konseptual sistematiska aplikasi, dan untuk pemodelan rinci yang menerjemahkan midel kedalan kode program. Kelas dalam diagram kelas mewakili kedua objek utama dan atau interaksi dalam aplikasi dan objek yang akan diprogram. Dalam class diagram kelas – kelas ini diwakili dengan kotak – kotak yang berisi tiga bagian. Sebuah kelas dengan tiga bagian :

- Bagian atas memegang nama kelas
- Bagian tengah berisi attribut kelas
- Bagian bawah memberikan metode atau operasi kelas dapat mengambil atau melakukan

3. Activity diagram

Activity diagram adalah representasi grafis dari alur kerja tahapan aktifitas. Diagram ini mendukung pilihan tindakan, iterasi dan *concurrency*. *Activity diagram* memodelkan alur kerja sebuah proses bisnis dan urutan aktivitas dalam suatu proses. Diagram ini sangat mirip dengan *flowchart* karena dapat memodelkan sebuah alur kerja satu aktivitas ke aktivitas lainnya atau dari satu aktivitas ke dalam keadaan sesaat (*state*). Seringkali bermanfaat bila membuat sebuah proses *activity diagram* terlebih dahulu dalam memodelkan sebuah proses untuk membantu memahami proses secara keseluruhan.

4. Sequence diagram

Sequence diagram adalah suatu diagram yang menggambarkan interaksi antar obyek dan mengindikasikan komunikasi diantara obyek-obyek tersebut. Diagram ini juga menunjukkan serangkaian pesan yang dipertukarkan oleh obyek-obyek yang melakukan suatu tugas atau aksi tertentu. Obyek-obyek tersebut kemudian

diurutkan dari kiri ke kanan, aktor yang menginisiasi interaksi biasanya ditaruh di paling kiri dari diagram.

2.5.2. Langkah-langkah Penggunaan *Unified Modeling Language* (UML)

Menurut (Afif Amrullah:2002). "Langkah-langkah penggunaan *Unified Modeling Language* (UML) sebagai berikut:

1. Buatlah daftar *business process* dari *level* tertinggi untuk mendefinisikan aktivitas dan proses yang mungkin muncul.
2. Petakan *Use Case* untuk setiap *business process* untuk mendefinisikan dengan tepat fungsional yang harus disediakan oleh sistem, kemudian perhalus *Use Casediagram* dan lengkapi dengan *requirement, constraints* dan catatan-catatan lain.
3. Buatlah*deployment diagram* secara kasar untuk mendefinisikan arsitektur fisik sistem.
4. Definisikan *requirement* lain *non fungsional, security*dansebagainya yang jugaharusdisediakanolehsistem.
5. Berdasarkan*Use Case diagram,*mulailahmembuat*activity diagram.*
6. Definisikan obyek-obyek level atas *package* atau *domain* dan buatlah *sequence* dan/atau *collaboration* utuk tiap alir pekerjaan, jika sebuah *Use Case* memiliki kemungkinan alir normal dan *error*, buat lagi satu diagram untuk masing-masing alir.
7. Buatlah rancangan *user interface model* yang menyediakan antamuka bagi pengguna untuk menjalankan *skenario Use Case.*

8. Berdasarkan model-model yang sudah ada, buatlah *class diagram*. Setiap *package* atau *domian* dipecah menjadi *hirarki class* lengkap dengan *atribut*dan metodenya. Akan lebih baik jika untuk setiap *class* dibuat *unit test* untuk menguji *fungsionalitas class* dan interaksi dengan *class* lain.
9. Setelah *class diagram* dibuat, kita dapat melihat kemungkinan pengelompokkan *class* menjadi komponen-komponen karena itu buatlah *component diagram* pada tahap ini. Juga, definisikan *test integrasi* untuk setiap komponen meyakinkan ia bereaksi dengan baik.
10. Perhalus*deployment diagram* yang sudahdibuat. Detilkan kemampuan dan *requirement* piranti lunak, sistem operasi, jaringan dan sebagainya. Petakankomponen ke dalam node.
11. Mulailah membangun sistem. Ada dua pendekatan yang tepat digunakan:

a. Pendekatan *Use Case* dengan meng*assign* setiap *Use Case* kepada tim pengembang tertentu untuk mengembangkan unit kode yang lengkap dengan *test.*
b. Pendekatan*komponen* yaitu *mengassign* setiap komponen kepada tim pengembang tertentu.

12. Lakukan uji modul dan uji integrasi serta perbaiki model beserta *code*nya. Model harus selalu sesuai dengan *code* yang aktual.
13. Perangkat lunak siap dirilis".

2.6 Eclipse

Eclipse adalah sebuah IDE (*Integrated Development Environment*) untuk mengembangkan perangkat lunak dan dijalankan di semua *platform*. Eclipse sendiri juga merupakan sebuah komunitas *open source*, yang memiliki proyek yang berfokus pada membangun sebuah *platform* pengembangan terbuka dari extensible *Framework, tools* dan runtime untuk membangun, menyebarkan dan mengelola perangkat lunak di seluruh siklus hidup perangkat lunak tersebut.

Eclipse juga bisa digunakan untuk penggunaan bahasa pemrograman lainnya seperti C, C++, COBOL, Perl, PHP, Python, dan sebagainya.

2.6.1 Versi Eclipse

Berikut ini adalah versi Eclipse yang telah dirilis:

Tabel 2.1 Versi Eclipse

Nama	**Tanggal rilis**	**Versi**
Eclipse 3.0	21 Juni 2004	3.0
Eclipse 3.1	28 Juni 2008	3.1
Callisto	30 Juni 2006	3.2
Europa	29 Juni 2007	3.3
Ganymade	25 Juni 2008	3.4
Galileo	24 Juni 2009	3.5
Helios	23 Juni 2010	3.6
Indigo	22 Juni 2011	3.7

Sumber: Eclipse *Foundation* (2011)