B A B II

LANDASAN TEORI

2.1 Analisa Sistem

Merupakan sebuah teknik pemecahan masalah yang mengurangi sebuah sistem menjadi bagian – bagian komponen dengan tujuan mempelajari seberapa baik bagian – bagian komponen dengan tujuan yang diharapkan. Analisa sistem merupakan tahapan awal dengan proses pengembangan sistem, sehingga tahapan ini menjadi acuan pelaksanaan pada proses pengembangan sistem.

Penyelesaian sistem sering menghasilkan kebutuhan untuk memperbaharui dan mendapatkan penyelesaian dari banyak masalah yang ada. Tugas analisa sistem merupakan proses penemuan, perbaikan, pemodelan dan spesifikasi. Setiap model analisa harus dapat mencapai tiga sasaran utama : (1) menggambarkan apa yang dibutuhkan oleh pemakai, (2) membangun dasar bagi pembuatan desain perangkat lunak, (3) membatasai serangkai persyaratan yang dapat divalidasi begitu perangkat lunak diimplementasikan.

2.2 Desain Sistem

Desain sistem merupakan tahapan yang dilakukan setelah analisa sistem. Desain digambarkan sebagai proses multi langkah dimana representasi struktur data, struktur program, karakteristik *interface* dan detail *procedure disintesis* dari persyaratan informasi. Desain sistem merupakan teknik pemecahan masalah yang saling melengkapi (dengan analisa sistem) yang merangkai kembali bagian – bagian

menjadi sebuah sistem yang lengkap, hal ini melibatkan penambahan, penghapusan dan perubahan – perubahan bagian relatif pada sistem.

Pada prinsipnya tahapan desain sistem menekankan pada tahapan teknis untuk mewujudkan sistem dengan lebih nyata. Metode desain sistem yang digunakan pada penellitian ini menggunakan metode berorientasi obyek.

2.3 Teknik *Object Oriented*

Teknologi *Object Oriented* merupakan paradigma baru dalam rekayasa *software* yang didasarkan pada obyek dan kelas. *Object Oreinted* memandang *software* bagian per bagian, dan menggambarkan satuan bagian tersebut dalam satu obyek. Satu objek dalam satu mode merupakan satu fokus selama proses analisis, desain, dan implementasi dengan menekankan pada *state,* prilaku (*behavior*), dan interaksi objek – objek dalam model tersebut.

Object Oriented mencakup bidang aplikasi yang sangat luas. Efek yang ditimbulkan oleh *Object Oriented* dapat dalam bentuk meningkatnya aplikasi *software* yang mudah digunakan dan servis yang lebih fleksibel. Pemodelan dengan menggunakan *Unified Modelling Language* merupakan konsep *object oriented* yang berhubungan dengan *Object Oriented Analysis and Desaign* (OOAD). Kedua konsep ini merupakan bagian yag sangat penting dalam proses pengembangan sistem yang menggunakan pendekatan *Object Oriented*.

2.3.1. *Object Oriented Analysis* (OOA)

Merupakan metode analisis yang memeriksa *requirements* (syarat/ keperluan yang harus dipenuhi oleh sistem) dari sudut pandang kelas – kelas dan objek – objek yang ditemui dalan ruang lingkup permasalahan. Sasaran OOA adalah pengembangan sederetan yag menggambarkan perangkat lunak komputer pada saat perangkat itu bekerja untuk memenuhi serangkai persyaratan yang ditentukan oleh pemakai.

Proses OOA tidak dimulai dengan suatu pemikiran mengenai obyek, melainkan dengan memahami cara sistem akan digunakan oleh manusia atau sistem yang lain.

2.3.2. *Object Oriented Desaign* (OOD)

OOD mentranformasi model analisis yang dibuat dengan menggunakan OOA kedalam suatu model desain yang berfungsi sebagai *blue print* perangkat lunak. OOD menghasilkan desain yang mencapai sejumlah atau tingkat yang berbeda dari modularitas. OOD mempunyai sifat unik untuk membangun empat konsep desain perangkat lunak yang penting yaitu : abstraksi, penyembunyian informasi, indepedensi fungsional dan modularitas.

OOD merupakan metode untuk mengarahakan arsitektur *software* yang didasarkan pada manipulasi objek – objek sistem atau subsistem.

2.3.3. Komponen – komponen OOAD

Dalam konsep *Object Oriented* terdapat banyak komponen yang harus diketahui dan dipahami sebelum melakukan proses analisis dan desain. Komponen – komponen yang ada pada OOAD antara lain :

1. Object

Objek (*Object*) adalah "benda", secara fisik atau konseptual, yang terdapat disekeliling kita. Sebuah obyek memiliki keadaan sesaat (*state*) dan prilaku (*behavior*). State dari sebuah obyek adalah kondisi objek tersebut atau himpunan dari keadaan yang menggambarkan objek tersebut. Sebagai contoh, salah satu state dari obyek jam adalah waktu saat ini. *State* dinyatakan dengan nilai dari atribut (*atribute*) objeknya. *Atribute* adalah nilai internal suatu objek, kondisi sesaat, koneksi dengan objek yang lain dan identitas.

Perubahan state dicerminkan oleh perilaku (*behavior*) objek tersebut. Perilaku (*behavior*) suatu obyek mendefinisikan bagaimana suatu objek bertindak dan memberi reaksi. *Behavior* ditentukan oleh himpunan semua atau beberapa operasi yang dapat dilakukan dalam objek sendiri. *Behavior* dari sebuah objek dicerminkan oleh :

a. *interface*, adalah pintu untuk mengakses service objek.
b. *Service*, adalah fungsi yang bisa diemban oleh objek.
c. *Method*, adalah mekanisme internal objek yang mencerminkan perilaku (*behavior*) objek tersebut.

2. *Kelas (Class)*

Kelas (*class*) adalah definisi umum (pola, tamplate atau cetak biru) untuk himpunan objek sejenis. Kelas menetapkan spesifikasi perilaku (*behavior*) atau atribut objek – objek tersebut. Kelas adalah keniskalan atau abterasi dari entitas dalam dunia nyata. Objek adalah "contoh" (*instace*) dari sebuah kelas.

3. *Encapsulation*

Encapsulation adalah proses menyembunyikan detail implementasi sebuah objek. Untuk dapat berkomunikasi dengan objek diperlukan pesan (*message*). *Message* adalah permintaan untuk obyek penerimaan (*receiver object*) untuk membawa metode yang ditunjukkan atau perilaku dan mengembalikan result dari aksi tersebut kepada obyek pengirim (*sender object*).

4. *Association dan Aggregation*

Association adalah hubungan antara objek yang saling membutuhkan. *Aggregation* adalah bentuk khusus dari asosiasi yang menggambarkan seluruh bagian dari suatu objek dimana objek tersebut merupakan bagian dari objek lainnya.

2.4 *Unified Modelling Language* (UML)

UML merupakan suatu bahasa semi formal dengan spesifikasi semantik yang meliputi sintak secara abstrak, aturan – aturan dasar dan semantik dinamis. UML dapat digunakan untuk menerangkan sistem yang berorientasi pada obyek secara lebih jelas dan detail jika dibandingkan dengan menggunakan kode bahasa pemrograman. UML biasanya disajikan dalam bentuk diagram atau gambar yang

meliputi class beserta atribut dan operasinya, serta hubungan antar kelas yang meliputi *inherintance, association*, dan komposisi.

Pada dasarnya UML dikembangkan dengan tujuan – tujuan sebagai berikut :

1. Untuk menyediakan pemodelan visual bagi pengguna, dimana model ini siap digunakan dan dapat saling bertukar model obyek yang dapat dimengerti.
2. Agar konsep dasarnya dapat dikembangkan dengan mudah dan dibuat lebih spesifik.
3. Tidak tergantung pada bahasa pemrograman dan proses pengembangan.
4. Menyediakan dasar yang formal dengan aturan – aturan dasar didalam membuat diagram guna memodelkan suatu objek.
5. Untuk mendorong pertumbuhan alat bantu yang berorientasi objek sehingga diharapkan akan muncul cukup banyak alat bantu yang memudahkan pengembangan software berorientasi obyek untuk menerapkan konsep *object oriented* dengan mudah.
6. Memudahkan pengembangan konsep dasar tanpa harus terpaku pada kode dalam bahasa pemrograman.
7. Memudahkan penggabungan beberapa abstraksi, sudut pandang, teknologi untuk membuat suatu sistem dari permasalahan yang dihadapi menjadi suatu sistem yang lebih kompleks.
8. Dengan UML diharapkan dapat menggambarkan keseluruhan sistem yang hendak dikembangkan dengan benar dan tepat tanpa harus mengurangi maksud dan informasi yang terkandung didalamnya.

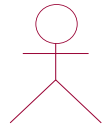
9. Memudahkan untuk memodelkan suatu masalah dari berbagai sudut pandang.
10. Memudahkan jika nantinya hendak melakukan pembuatan atau pengembangan dari obyek yang lebih selesai diimplemantasikan.
11. Dapat digunakan sebagai alat untuk mendokumentasikan obyek – obyek yang telah dikembangkan sehingga mudah untuk dibaca dan dimengerti oleh pihak lainnya.

2.4.1. Notasi UML

Adapun notasi yang dapat digunakan dalam *Unifief Modelling Language* (UML) antara lain :

1. Actor

Actor adalah segala sesuatu yang berinteraksi dengan sistem aplikasi komputer. Jadi actor ini bisa berupa orang, perangkat keras atau mungkin juga obyek lain dalam sistem yang sama. Biasanya dilakukan oleh aktor adalah memberikan informasi pada sistem dan memerintahkan sistem untuk melakukan sesuatu. Pada aktor ini dinotasikan dengan gambar dibawah ini :



Gambar 2.1. *Actor*

2. Class

Class merupakan pembentukan suatu sistem berorientasi obyek karena class menunjukkan kumpulan obyek yang memiliki atribut dan operasi yang sama. *Class* digunakan untuk mengimplementasikan *interface* dan mengabstraksi elemen – elemen dari sistem yang sedang dibangun, selain itu untuk mempresentasikan baik perangkat keras maupun perangkat lunak, baik konsep maupun benda nyata, dengan dinotasikan sebagai berikut :



Gambar 2.2. Notasi *Class*

3. Interface

Interface merupakan kumpulan operasi tanpa implementasi dari suatu *class*. Implementasi operasi *interface* dijabarkan oleh operasi dalam *class*. Oleh karena itu keberadaan *interface* selalu disertai oleh *class* yang mengimplementasikan operasinya. *Interface* ini merupakan salah satu cara mewujudkan prinsip *encapsulasi* dalam obyek. Pada *interface* dinotasikan seperti gambar dibawah ini :



Interface

Gambar 2.3. Notasi *Interface*

4. Use case

Use case menjelaskan urutan kegiatan yang dilakukan aktor dan sistem untuk mencapai suatu tujuan tertentu. Walaupun menjelaskan kegiatan, *use case* hanya menjelaskan apa yang dilakukan oleh aktor dan sistem, bukan bagaimana aktor dan sistem melakukan kegiatan dan dinotasikan seperti dibawah ini :



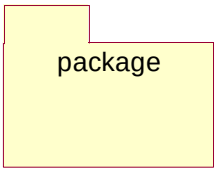
Gambar 2.4. Notasi *UseCase*

5. Interaction

Digunakan untuk menunjukkan baik aliran pesan atau informasi antara obyek maupun hubungan antar obyek. Biasanya interaksi ini dilengkapi juga dengan teks bernama *Operation Signature* yang tersusun dari nama operasi, parameter yang dikirim dan tipe parameter yang dikendalikan.

6. Package

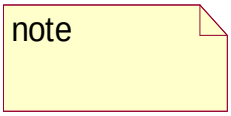
Package adalah kontainer atau wadah konseptual yang digunakan untuk mengelompokkan elemen – elemen dari sistem yang sedang dibangun, sehingga bisa dibuat model yang lebih sederhana. Tujuannya adalah untuk mempermudahkan penglihatan (*visibility*) dari model yang sedang dibangun, dinotasikan sebagai berikut :



Gambar 2.5. Notasi *Package*

7. Note

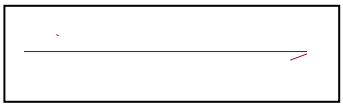
Digunakan untuk memberi keterangan dan komentar tambahan dari suatu elemen sehingga langsung terlampir dalam model. *Note* ini bisa ditempelkan ke semua elemen notasi lainnya. Dan dinotasikan seperti gambar dibawah ini :



Gambar 2.6. Notasi *Note*

8. Dependency

Merupakan relasi yang menunjukkan bahwa perubahan dari salah satu elemen memberikan pengaruh pada elemen lain. Elemen yang ada dibagian tanda panah adalah elemen yang tergantung pada elemen yang ada dibagian tanpa tanda panah, seperti gambar dibawah ini :



Gambar 2.7. Notasi *Dependency*

9. Association

Menggambarkan navigasi antar *class*, berupa banyak obyek lain yang berhubungan dengan satu obyek (*multiplicity* antar *class*), dan apakah suatu class menjadi bagian dari class lainnya (*aggregation*) , *navigation* dilambangkan dengan penambahan tanda panah di akhir garis.

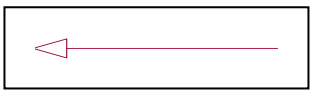
Bi-directional Navigation menunjukkan bahwa dengan mengetahui salah satu *class* bias didapatkan informasi dari *class* lainnya. *Underectional Navigation* hanya dengan mengetahui kelas diujung garis *association* tanpa panah kita bias mendapatkan informasi dari kelas di ujung dengan panah, tetapi tidak sebaliknya, dinotasikan seperti gambar dibawah ini :



Gambar 2.8. Notasi *Association*

10. Generalization

Adalah menunjukkan hubungan antara elemen yang lebih umum ke elemen yang lebih spesifik. Dengan *generalization class* yang lebih spesifik (*sub class*) akan menurunkan atribut dan operasi dari *class* yang lebih umum (*super class*). Notasi generalisasi sepeti gambar dibawah ini :



Gambar 2.9. Notasi *Generalization*

11. Realization

Menunjukkan hubungan bahwa elemen – elemen yang ada di bagian tanpa panah akan merealisasikan apa yang dinyatakan oleh elemen yang ada dibagian dengan tanda panah, seperti gambar dibawah ini :



Gambar 2.10. Notasi *Realization*

Misalnya kelas merealisasikan *package, component* merealisasikan *class* atau *interface*.

2.4.2. Artifak dalam UML

Artifak adalah sepotongan informasi yang digunakan atau menghasilkan dalam suatu proses rekayasa *software*. Artifak dapat berupa model, deskripsi, atau *software*. Untuk membuat suatu model, UML memiliki diagram grafis sebagai berikut :

1. *Use Case Diagram*

   Diagram ini memperlihatkan himpunan *use case* dan aktor – aktor. Diagram ini sangat penting untuk mengorganisasi dan memodelkan prilaku sesuatu sistem yang dibutuhkan serta diharapkan pengguna.

2. *Class Diagram*

   Diagram ini memperlihatkan himpunana kelas, antar muka, kolaburasi, serta relasi – relasi. Diagram ini umumnya dijumpai pada pemodelan sistem berorientasi obyek.

3. *Objek Diagram*

Diagram ini memperlihatkan obyek – obyek serta relasi – relasi antar obyek.

4. *Sequence Diagram*

Pada diagram ini menekankan pada pengiriman pesan ddalam suatu waktu tertentu.

5. *Collaboration Diagram*

Diagram interaksi yang menekankan organisasi struktural dari obyek yang menerima serta mengirim pesan.

6. *Statechart Diagram*

Diagram ini memperlihatkan *state* – *state* pada sistem, terutama untuk memperlihatkan sifat dinamis dari antar muka, kelas, kolaburasi dan terutama penting pada pemodelan sistem – sistem reaktif.

7. *Activity Diagram*

Pada diagram ini memperlihatkan aliran dari suatu aktivitas ke aktivitas lainnya dalam suatu sistem.

8. *Component Diagram*

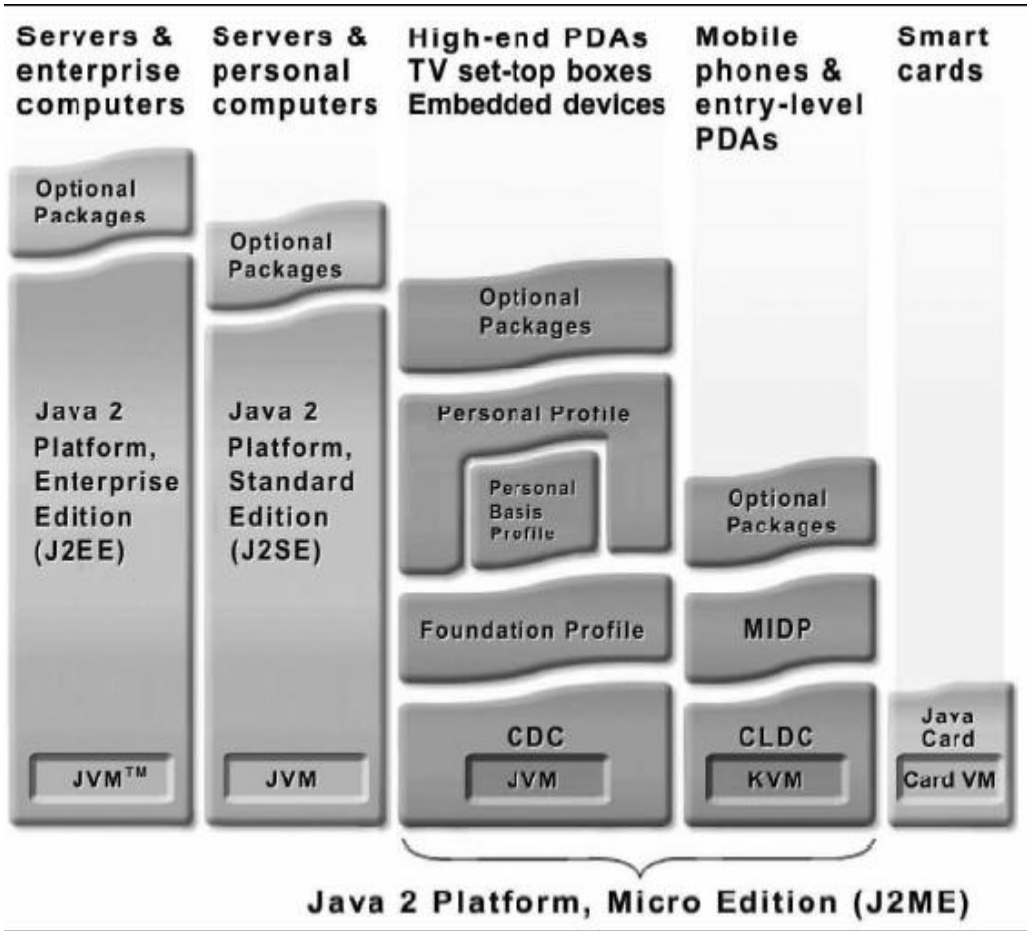
Diagram ini memperlihatkan organisasi serta kebergantungan sistem atau perangkat lunak pada komponen yang sudah ada sebelumnya.

9. *Deployment Diagram*

Diagram ini memperlihatkan konfigurasi saat aplikasi dijalankan. Digram ini memuat simpul – simpul beserta komponen – komponen yang ada didalamnya. Diagram ini sangat erat dengan diagram komponen, diagram ini sangat berguna saat aplikasi yang kita buat digunakan banyak mesin.

2.5 J2ME (*JAVA 2 MICRO EDITION*)

J2ME dirancang untuk dapat menjalankan program Java pada perangkat-perangkat semacam telepon genggam atau PDA yang memiliki memori terbatas. Pada gambar 2.1 akan ditunjukkan lingkungan kerja J2ME pada pemrograman Java.



Gambar 2.11 Lingkungan Kerja J2ME

J2ME terdiri atas komponen-komponen sebagai berikut :

1. *Java Virtual Machine* (JVM)

   Komponen ini untuk menjalankan program-program Java pada emulator atau *handheld devices*.

2. Java API (*Application Programming Interface*)

   Komponen ini merupakan kumpulan *library* untuk menjalankan dan mengembangkan program Java pada *handheld devices*.

3. *Tools* lain untuk mengembangkan aplikasi Java, semacam emulator Java phone, dapat kita lihat contohnya pada gambar 2.2.



Gambar 2.12 Contoh beberapa *handheld devices* dari J2ME *wireless toolkit*

Dalam J2ME dibagi menjadi dua bagian yang dikenal dengan istilah *configuration* dan *profile*. Dua istilah tersebut sangat penting dalam pengembangan aplikasi *wireless* dengan Java.

2.5.1. J2ME Configuration

J2ME *Configuration* mendefinisikan lingkingan kerja J2ME *runtime*. Oleh karena setiap *handheld devices* memiliki fitur-fitur yang berbeda-beda. J2ME *Configuration* ini dirancang untuk menyediakan *library* standar yang mengimplementasikan fitur standar dari sebuah h*andheld devices*.

Saat ini ada dua kategori J2ME Configuration, yakni :

a. CLDC (*Connected Limited Devices Configuration*)

   Kategori ini umumnya digunakan untuk aplikasi Java pada perangkat keras yang memiliki memori terbatas, yaitu 160 sampai dengan 512 Kilobyte seperti telepon genggam atau PDA.

b. CDC (*Connected Device Configuration*)

   Kategori ini umumnya digunakan untuk aplikasi Java pada perangkat-perangkat *handheld devices* dengan ukuran memori paling tidak 2 MegaBytes.Contohnya adalah internet TV, Telepon genggam *Communicator*, dan *Car Television* (TV pada mobil).

   Sederhananya, perbedaan antara CDC dan CLDC adalah seperti pada tabel 2.1.

Tabel 2.1 Perbandingan antara CDC dan CLDC

CLDC	**CDC**
Mengimplementasikan subset dari J2SE	Mengimplementasikan seluruh fitur pada J2SE
JVM yang digunakan dikenal dengan nama KVM	JVM yang digunakan dikenal dengan nama CVM
Digunakan pada perangkat *handheld* dengan ukuran memori terbatas (160-512 Kilobyte)	Digunakan pada perangkat *handheld* dengan ukuran memori minimal 2 Megabyte
Prosesor : 16 atau 32 bit	Prosesor : 32 bit

Pada proyek akhir ini akan dibahas lebih lanjut mengenai CLDC, karena yang dibuat pada proyek akhir ini adalah aplikasi pada telepon genggam yang memiliki memori terbatas. CLDC digunakan untuk implementasi program Java pada perangkat-perangkat keras dengan ukuran memori yang sangat terbatas, yaitu antara 160 sampai dengan 512 Kilobyte. Sehingga fitur-fitur yang kurang penting untuk diimplementasikan dalam *handheld device* dari Java 2 harus dihilangkan. Fitur-fitur yang dibuang tersebut antara lain :

1. Tidak ada dukungan untuk *floating point*

Kelas-kelas untuk perhitungan *floating point*, yakni *java.lang.Float* dan *java.lang.Double* dibuang dari CLDC

2. Tidak ada dukungan untuk finalisasi objek

Garbage collector yang secara sederhananya digunakan untuk ”bersih-bersih memori” membuang fungsi *finalize* pada java.lang.Object, sekalipun fungsi ini sangat penting pada Java 2

3. Tidak ada dukungan untuk JNI

Pembuatan Sistem Organizer Pribadi Pada Telepon Genggam Berbasis J2ME 8 Kelas JNI yang memungkinkan Java mengakses *library* yang dibuat dengan bahasa selain Java, tidak didukung oleh CLDC

4. Penanganan kesalahan/*exception* yang terbatas

CLDC hanya mendefinisikan tiga kelas berikut untuk penanganan kesalahan/*exception* :

a. Kelas *java.lang.Error*

b. Kelas *java.lang.OutOfMemory*

c. Kelas *java.lang.VirtualMachineError*

2.5.2. J2ME *Profile*

Jika J2ME *Configuration* menyediakan *library-library* Java untuk implementasi fiyur-fitur standar dari sebuah *handheld devices*, J2ME Profile menyediakan implementasi-implementasi tambahan yang sangat spesifik dari sebuah *handheld devices*.

Saat ini ada lima kategori J2ME Profile, yakni:

a. *Mobile Information Device Profile* (MIDP)

b. *Foundation Profile* (FP)

c. *Personal Profile*

d. RMI *Profile*

e. *Personal Digital Assistance Profile*

2.5.3. MIDP (Mobile Information Device Profile)

MIDP menyediakan *library-library* Java untuk implementasi dasar *interface* (GUI), implementasi jaringan (*networking*), *database*, dan *timer*. MIDP dirancang khusus untuk *wireless phone* dan *pager*.

2.5.4. Arsitektur Aplikasi MIDlet

Paket javax.microedition.midlet memiliki kelas MIDlet yang mendefinisikan MIDP (Mobile Information Device Profile) dan interaksinya dengan perangkat handheld yang bersangkutan. Kelas ini merupakan kelas utama dalam sebuah MIDlet,

yaitu sebuah aplikasi MIDlet harus merupakan turunan dari kelas ini. Kelas ini merupakan kelas abstrak, sehingga kelas turunannya harus mengimplementasikan fungsi-fungsi abstrak di dalamnya.

Perhatikan kode Java kelas MIDlet pada listing program berikut :

```
/*
* Kelas Abstract MIDlet dalam paket
javax.microedition.midlet
*
*/
package javax.microedition.midlet;
import com.sun.midp.midlet.MIDletState;
import com.sun.midp.midlet.MIDletSuite;
public abstract class MIDlet{
protected MIDlet(){
state = new MIDletProxy(this);}
protected abstract void startApp()
throws MIDletStateChangeException;
protected abstract void pauseApp();
protected abstract void destroyApp (Booleanflag) throws
MIDletStateChangeException;
public final void notifyDestroyed(){
state.notifyDestroyed();
}
```

```
public final void notifyPaused(){
state.notifyPaused();
}
public final String getAppProperty(String Key){
return.state.getMIDletSuite().getProperty(key);
public finall void resumeRequest(){
state.resumeRequest();
}
private MIDletProxy state;
}
```

Ada beberapa istilah penting dalam daur hidup sebuah MIDlet, yakni :

1. *Application Management Software* (AMS)

Sebagaimana diketahui bahwa MIDlet berjalan dalam sebuah perangkat *handheld*. AMS ini merupakan *software* atau program bawaan yang ada dalam perangkat *handheld* tersebut yang mengatur manajemen aplikasi MIDlet yang ada.

2. MIDlet

Merupakan aplikasi yang dijalankan pada sebuah perangkat *handheld*. MIDlet tidak berinteraksi langsung dengan hardware dari *handphone*, Palm, atau perangkat *handheld* Anda yang lain, melainkan berinteraksi melalui AMS.

AMS inilah yang akan menerima sinyal dari MIDlet bahwa MIDlet akan dijalankan atau berhenti.

3. MIDlet *states*

Merupakan status dari sebuah MIDlet yang akan dijelaskan kemudian.

2.5.5. Status MIDlet

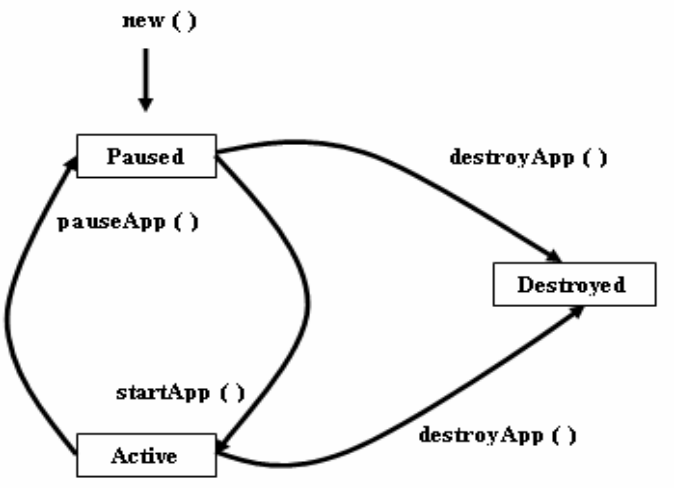
Terdapat tiga status dalam daur hidup sebuah MIDlet, seperti pada tabel 2.2.

Tabel 2.2 Status dalam daur hidup sebuah MIDlet

Nama Status	**Keterangan**
Paused	Status ini terjadi ketika MIDlet selesai diinisialisasi dan tidak melakukan aksi apapun. Pada status ini MIDlet tidak boleh sedang mengunci suatu sumber daya, misalnya tidak boleh sedang melakukan *locking* atau mengunci sebuah file untuk ditulis yang menyebabkan proses lain tidak bisa menulis ke file tersebut. a. Secara garis besar, status MIDlet adalah *Paused* pada saat :

	b. Setelah MIDlet dibuat dengan konstruktor *new()* c. Dari status *Active*, kemudian berhasil memanggil fungsi *MIDlet.pauseApp()* d. Dari status *Active*, kemudian berhasil memanggil fungsi *MIDlet.notifyPaused()* e. Dari status *Active*, namun ketika akan start () terjadi kesalahan berupa *exception*:*MIDletStateChangeException*
Active	Status ini terjadi ketika MIDlet sedang aktif/berjalan dengan normal, yakni setelah memanggil fungsi *MIDlet.startApp().*
Destroyed	Status ini terjadi ketika MIDlet berhenti

Jika digambarkan, perubahan status dalam daur hidup sebuah MIDlet adalah seperti pada gambar 2.13.



Gambar 2.13 Daur hidup dan perubahan status sebuah MIDlet

2.5.6. J2ME Emulator

J2ME emulator merupakan program emulator yang digunakan untuk menampilkan file-file Java dengan tampilan menyerupai wireless device (ponsel).

Dengan menggunakan J2ME emulator, dapat mempermudah untuk mencoba program Java yang telah dibuat. Untuk membuat program Java cukup dengan menggunakan editor biasa semacam Notepad, Ultraedit, kawa, atau Gel.

2.5.7. Eksekusi Sebuah MIDlet

Umumnya eksekusi sebuah MIDlet adalah sebagai berikut :

1. AMS (*Application Management Software*) yang menginisialisasi sebuah objek MIDlet, sehingga MIDlet berada pada status *Paused*.

2. Ketika AMS memutuskan bahwa MIDlet sudah waktunya dijalankan, maka AMS memanggil fungsi *MIDlet.startApp()* sehingga MIDlet akan mulai aktif berfungsi.
3. Ketika AMS memutuskan bahwa MIDlet harus dinonaktifkan sementara, maka AMS akan memanggil fungsi *MIDlet.pauseApp()* sehingga MIDlet akan berhenti bekerja.
4. Ketika AMS memutuskan bahwa MIDlet harus dinonaktifkan selamanya, maka AMS akan memanggil fungsi *MIDlet.destroyApp()* sehingga alokasi memori yang digunakan oleh MIDlet yang bersangkutan akan dibersihkan.

2.5.8. Jenis Fungsi-Fungsi API pada MIDP untuk GUI

Fungsi-fungsi untuk pembuatan interface berbasis window (GUI) yang disediakan oleh MIDP terbagi atas dua level, yaitu *high level* dan *low level*. Semua fungsi-fungsi untuk interface berbasis window ini, ditangani oleh paket *javax.microedition.lcdui. Low level* API berbasis pada *lass Canvas* (*javax.microedition.lcdui.Canvas*) *dan Graphic* (*javax.microedition.lcdui.Graphics*) sedangkan *high level* API berbasis pada *screen* (*javax.microedition.lcdui.Screen*). Contoh dari *high level* API adalah *alert, form, list, dan text box*.

2.5.9. MIDlet Suite dan Application Descriptor

Ketika selesai membuat sebuah aplikasi, aplikasi tersebut harus dikemas dalam sebuah kemasan yang berekstensi *.JAR. Selain itu, dikenal pula file *.JAD yang disebut sebagai application descriptor yang berguna untuk mendeskripsikan isi dari file JAR tadi. Kedua file ini haruslah di-upload ke ponsel pengguna agar aplikasi dapat dijalankan.

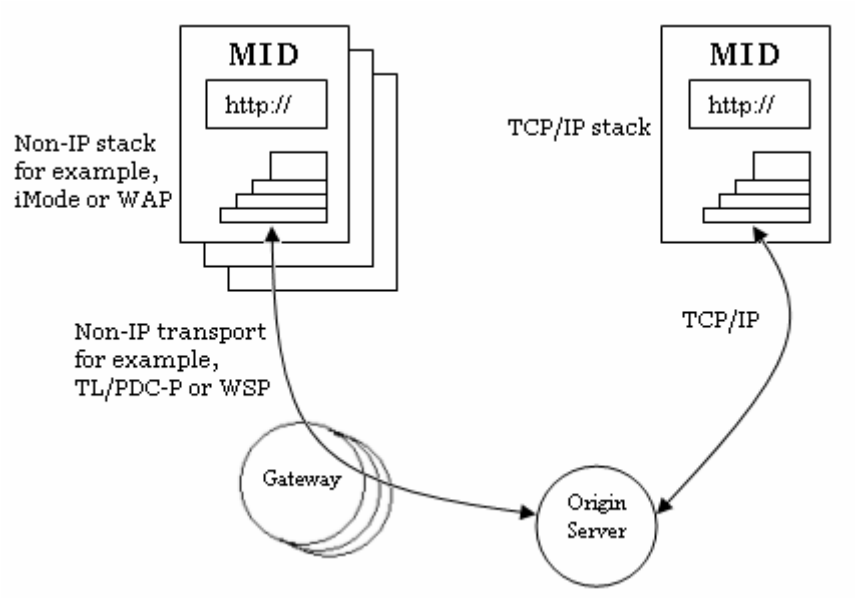
2.5.10. *Network* MIDlet

Selain kelebihannya dalam hal akses internal *resource*, MIDP juga menyediakan fasilitas khusus untuk koneksi ke eksternal *resource* melalui fungsionalitas khusus untuk *GenericConnectionFramework* (GCF).

Dukungan MIDP ini merupakan subset dari protokol HTTP yang dapat diimplementasikan menggunakan protokol IP misalnya TCP/IP maupun protokol non-IP semacam WAP dan i-mode, serta memanfaatkan sebuah gateway untuk menyediakan akses server HTTP yang ada di internet.

Generic Connection Framework (GCF) digunakan untuk mendukung client-server dan jaringan datagram. Dengan hanya menggunakan protokol yang ditentukan oleh MIDP, menyebabkan aplikasi ini akan memiliki probabilitas untuk semua MIDP. Implementasi MIDP harus menyediakan dukungan untuk mengakses server dan layanan HTTP 1.1. Pada gambar 2.4 akan ditunjukkan konfigurasi network MIDlet.

GFC mendeskripsikan sebuah kelas fundamental yang bernama *Connector* yang digunakan untuk menyelenggarakan semua koneksi ke jaringan. Tipe-tipe khusus dari koneksi jaringan dimodelkan oleh *interface-interface* GCF yang diperoleh melalui kelas Connector. Kelas Connector ini dan *interface-interface* koneksi berada dalam package *javax.microedition.io*.



Gambar 2.14 Konfigurasi Network MIDlet

2.6 PHP (*Hypertext Preprocessor*)

PHP merupakan script yang menyatu dengan HTML dan berada pada server (*server side HTML embedded scripting*). Dengan PHP ini Anda dapat membuat beragam aplikasi berbasis web, mulai dari halaman web yang sederhana sampai aplikasi komplek yang membutuhkan koneksi ke *database*.

Sampai saat ini telah banyak *database* yang telah didukung oleh PHP dan kemungkinan akan terus bertambah. *Database* tersebut adalah :

dBase

DBM

FilePro

mSQL

MySQL

ODBC

Oracle

Postgres

Sybase

Velocis

Selain itu, PHP juga mendukung koneksi dengan protokol IMAP, SNMP, NNTP, dan POP3.

2.6.1. *PHP Script*

Kode PHP umumnya diawali dengan <? dan diakhiri dengan ?>. Pasangan kedua kode inilah yang berfungsi sebagai tag kode PHP. Berdasarkan tag inilah pihak server dapat memahami kode PHP dan kemudian memrosesnya, dan hasilnya dikirim ke browser.

Untuk memberikan komentar (tulisan pada program yang tidak dieksekusi) pada program PHP dapat dilakukan dengan tiga macam cara penulisan, yaitu :

1. /* komentar */

Tulisan apapun yang berada di antara ’*/’ dan ’*/’ akan dianggap sebagai komentar. Cara seperti ini Sangay berguna dan efisien untuk pemberian komentar yang memakan banyak baris.

2. // komentar

Tulisan di baris yang sama setelah ’//’ akan dianggap sebagai komentar. Cara ini berguna untuk pemberian komentar singkat yang tidak lebih dari satu baris saja.

3. #komentar

Sama seperti ’//’, tulisan di baris yang sama setelah ’#’ akan dianggap sebagai komentar. Cara ini berguna untuk pemberian komentar singkat yang tidak lebih dari satu baris saja.

Dalam PHP, penulisan variabel diawali dengan karakter ’$’, kemudian diikuti dengan huruf sebagai karakter pertama. Setelah itu, dapat dilanjutkan dengan kombinasi huruf dan angka. Variabel tidak boleh mengandung spasi maupun tanda baca di salamnya kecuali *underscore* (’_’).

Variabel PHP bersifat ”*case sensitif*”, yang berarti kita harus memperhatikan penulisan huruf besar dan huruf kecil. Variabel $jumlah berbeda dengan $ Jumlah.

Contoh pemakaian variabel yang benar adalah sebagai berikut :

$nama_pemakai

$kota_3

Sedangkan contoh pemakaian variabel yang salah adalah sebagai berikut :

$nama pemakai

$3kota

Untuk menampilkan data dapat digunakan ’print’ maupun ’echo’. Sintaks yang digunakan antara lain :

print (data);

atau

print data;

atau

echo (data);

atau

echo data;

2.7 MySQL

MySQL adalah sebuah perangkat lunak sistem manajemen basis data SQL. Dalam bahasa Inggris disebut *database management system* atau DBMS yang *multithread, multi-user*. MySQL tersedia sebagai perangkat lunak gratis di bawah lisensi GNU *General Public License* (GPL), tetapi mereka juga menjual lisensi komersial untuk kasus-kasus dimana penggunaannya tidak cocok dengan penggunaan GPL.

MySQL dimiliki dan disponsori oleh sebuah perusahaan komersial Swedia MySQL AB, dimana memegang hak cipta hampir atas semua kode sumbernya.

Kedua orang Swedia dan satu orang Finlandia yang mendirikan MySQL AB adalah David Axmark, Allan Larsson, dan Michael “Monty” Widenius.

Terdapat beberapa API tersedia yang memungkinkan aplikasi-aplikasi komputer yang ditulis dalam berbagai bahasa pemrograman untuk dapat mengakses *database* MySQL, antara lain bahasa pemrograman C, C++, C#, Eiffel, Smaltalk, Java, Lisp, Perl, PHP, Python, Ruby, REALbasic, dan Tcl. Kebanyakan kode sumber MySQL dalam ANSI C.

Perangkat lunak gratis untuk administrasi *database* MySQL berbasis web yang sangat populer adalah phpMyAdmin. Untuk perangkat lunak untuk administrasi *database* MySQL yang dijual secara komersial antara lain *MySQL front*, *Navicat*, dan *EMS SQL Manager for MySQL*.

2.7.1. Dasar-dasar MySQL

Dalam bahasa SQL pada umumnya informasi tersimpan dalam tabel-tabel yang secara logic merupakan struktur dua dimensi terdiri dari baris (row atau record) dan kolom (column atau field). Sedangkan dalam sebuah *database* dapat terdiri dari beberapa table. Beberapa tipe data dalam MySQL yang sering dipakai dapat dilihat pada tabel 2.3

Tabel 2.3 Beberapa tipe data yang sering dipakai dalam MySQL

Tipe Data	**Keterangan**
INT (M) [UNSIGNED]	Angka -2147483648 s/d 2147483647
FLOAT (M,D)	Angka pecahan
DATE	Tanggal Format : YYYY-MM-DD
DATETIME	Tanggal dan Waktu Format : YYYY-MM-DD HH:MM:SS
CHAR (M)	String dengan panjang tetap sesuai dengan yang ditentukan. Panjangnya 1-255 karakter
VARCHAR (M)	String dengan panjang yang berubah-ubah sesuai dengan yang disimpan saat itu. Panjang 1-255 karakter
BLOB	Teks dengan panjang maksimum 65535 karakter
LONGBLOB	Teks dengan panjang maksimum 4294967295 karakter

Cara untuk membuat sebuah *database* baru adalah dengan perintah :

create database namadatabase;

Perintah untuk membuat tabel baru adalah :

```
create table namatabel
 (
truktur
);
```

2.8 Teknologi GPRS

General Packet Radio Service (GPRS) merupakan teknologi generasi kedua yang akan meramaikan maraknya standar jaringan *mobile* yang sudah ada, seperti GSM dan TDMA. Kecepatan transmisi data diharapkan dapat naik dari 9.6 Kbps menjadi 115Kbps. GPRS memiliki kemampuan foto dan *video* dengan kecepatan tinggi.

Kemampun tambahan yang dimiliki GPRS adalah pertama dapat memelihara keutuhan komunikasi data dan suara pada saat sedang bergerak. Kedua *user* dapat segera terhubung ke nomor yang dituju kapan saja jika diinginkan tidak tergantung pada lokasi mana berada sekarang serta tanpa mengalami *delay* yang lama. Ketiga dengan tingkat kecepatan yang dimiliki GPRS sangat memungkinkan untuk men-*download file.*

Fasilitas yang dimiliki oleh GPRS adalah dapat mengakses *intranet, internet*, menggunakan *E-Mail* dan *Fax, unified Messaging.* Selain itu terdapat beberapa

layanan tambahan seperti *e-commerce, banking, balance checking*, transfer antar rekening, pembayaran tagihan dan perdagangan valas.

Teknologi transmisi data GSM berupa GPRS (*General Packet Radio Services*) adalah sebuah teknologi yang dipergunakan untuk pelayanan data *wireless* seperti pada *wireless internet* atau *intranet* serta pelayanan multimedia.

Alat komunikasi dengan GSM GPRS mempunyai fasilitas pertukaran data sampai dengan kecepatan 170kbps. Teknologi ini juga biasanya disebut sebagai GSM-IP (*internet Protocol*), karena akan menghubungkan pengguna dengan ISP (*Internet* Service Provider).

Salah satu keuntungan dari teknologi baru ini adalah, pengguna akan selalu terhubung atau connected. Selalu *online*, tetapi akan dikenai biaya hanya dari besarnya data yang ditransmisi. Dengan teknologi ini panggilan secara *voice* dapat dilakukan secara bersamaan dengan transmisi data. Tetapi ini tergantung dari jenis hp yang digunakan. Banyak pembuat hp telah menggunakan teknologi ini pada hpnya. Seperti Ericsson, Telepon genggam, Siemens, Sagem, Alcatel, Samsung, Motorola dan sebagainya.

GPRS merupakan suatu tambahan pelayanan baru pada *network* yang telah ada. Dengan GPRS , para operator *network* GSM dapat memberikan pelayanan kompetitif untuk pertukaran data, sehingga melengkapi jasa pelayanan yang mereka berikan.

Teknologi ini sedang berkembang, dan tentu saja seperti teknologi pada dunia *wireless* lainnya, teknologi ini akan berkembang sangat cepat. Seperti GPRS *roaming* yang merupakan kebutuhan dasar untuk terwujudnya global *mobile internet* bagi

pelanggan GPRS pada berbagai operator GPRS *network* pada masa yang akan datang. Ini diterapkan pada GRX, yang merupakan suatu *IP routing network* tersentralisasi untuk interkoneksi antar GPRS *network.* Sistem GRX yang berdasarkan GPRS *roaming* telah berhasil dilakukan antara Sistem *roaming* Sonera dengan *network* dari Telepon genggam.

Tentu saja banyak perusahaan besar lainnya terus mengembangkan sistem ini, yang pada akhirnya mudah-mudahan dapat menguntungkan konsumen.

2.8.1. Fitur-fitur Penting GPRS

General Packet Radio Service (GPRS) adalah sebuah jasa pelayanan tambah berupa *nonvoice* yang memungkinkan informasi dapat dikirim dan diterima pada sebuah *mobile telephone network*. Sistem ini menambahkan fasilitas baru dari apa yang ada sekarang yaitu *Circuit Switched Data and Short Message Service*. GPRS tidak ada hubungannya dengan GPS (*the Global Positioning System*). Beberapa fitur unik GPRS adalah sebagai berikut :

a. Kecepatan

Secara teoritis kecepatan maksimum sistem ini 171.2 kilobytes per second (kbps). Ini adalah adalah tiga kali lebih cepat dari transmisi data sistem *network* telekomunikasi *fixed* yang ada dan 10 kali lebih cepat dari *Circuit Switched Data* pada sistem *network* telekomunikasi *wireless* GSM saat ini.Dengan kecepatan pengiriman data, maka sistem ini tentu saja akan

memberikan pelayanan yang lebih murah dibanding dengan SMS atau *Circuit Switched Data*.

b. Selalu tersedia

GPRS adalah sebuah sistem yang terhubung terus, dimana informasi dapat segera dikirim atau diterima saat diperlukan. Tidak perlu adanya dial-up modem. Karena itu seperti telah diterangkan diatas sebelumnya, GPRS adalah sebuah sistem yang "*always connected*".

c. Aplikasi baru yang lebih baik

GPRS memberikan fasilitas baru untuk beberapa aplikasi yang sebelumnya tidak ada pada *network* GSM, karena adanya keterbatasan pada kecepatan di *Circuit Switched Data* (9.6 kbps) dan panjang *message*/berita pada *Short Message Service* (160 characters). GPRS akan dapat secara penuh mengaplikasi *internet* seperti yang biasa kita dapatkan di *desktop* komputer, dari *Web* browsing sampai dengan *chat.* Selain itu GPRS akan memberikan kemudahan *transfer file*, dan *home automation* yaitu dengan dapatnya kita mengontrol alat-alat di rumah kita secara *remote* dari daerah di luar rumah kita sendiri.

2.8.2. Penggunaan GPRS

Untuk dapat menggunakan GPRS, diperlukan beberapa hal yaitu:

a. Telepon genggam atau terminal dengan sistem GPRS (banyak telepon baru yang telah menyediakan sistem ini)
b. Mendaftar pada *operator network* dengan sistem GPRS
c. Penggunaan GPRS harus telah diizinkan oleh *operator network* bersangkutan.
d. Pengetahuan tentang bagaimana mengirim dan menerima informasi GPRS dengan telepon genggam yang dimiliki.
e. Sebuah tujuan untuk mengirim atau menerima informasi melalui GPRS. Kalau pada SMS biasanya tujuan ini adalah telepon genggam orang lain, maka pada GPRS kemungkinan tujuan yang paling banyak adalah alamat pada *internet*, karena GPRS didisain agar *internet* dapat digunakan oleh pemegang telepon genggam.