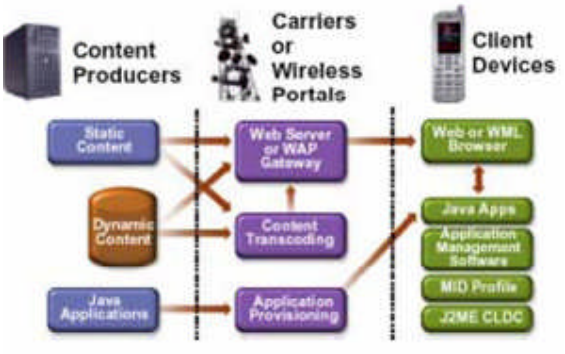
BAB II
LANDASAN TEORI

2.1 Teknologi *Wireless*

Secara konsep, teknologi *wireless* dapat dibagi dalam dua kategori, pertama untuk lokal dan kedua untuk area yang luas. Peralatan yang termasuk dalam katagori pertama misalnya adalah kendali jarak jauh untuk membuka atau mengunci mobil maupun garasi, telepon *cordless* 900 Mhz, peralatan mainan dengan *radio control*, atau jaringan nirkabel. Peralatan nirkabel jenis pertama ini hanya bekerja untuk daerah dengan jangkauan yang tidak terlalu jauh. Sedangkan peralatan jenis aplikasi yang kedua diantaranya adalah pager, telepon selular, PDA, dan sejenisnya. Jangkauan dari perangkat tersebut jauh lebih besar dari aplikasi jenis pertama. Karena jaringan yang ada di permukaan bumi berupa *cell – tower*, peralatan komunikasi bergerak seperti telepon selular menerima layanan dari sebuah *wireless carrier* atau perusahaan yang mengoperasikan *celltower* tersebut.



Gambar 2.1 Cara kerja jaringan *wireless* telepon selular

Pada gambar 2.1 bagian atas kiri terlihat konten berisi *static file* atau bisa juga berupa *dynamic content* atau juga aplikasi Java yang dijalankan oleh server. Lebih luas lagi, informasi dapat diekstrak dari basis data dan dimodifikasi menggunakan servlets, EJB, atau teknologi server lainnya.

Selanjutnya kita lihat pada bagian tengah, Java Application ditampilkan oleh *Web Server* dan seteruskan ditransmisikan melalui jaringan nirkabel menuju *mobile devices.* Web server atau WAP *gateway system* akan mendistribusikan isinya pada telepon selular atau peralatan lainnya.

Akhirnya isinya diterima oleh peralatan *client* dengan melakukan proses *transcoding*, baik itu mengambil data dari basis data, atau XML, atau HTML, dan selanjutnya diterjemahkan kedalam format WML atau cHTML bergantung pada format yang kita miliki.

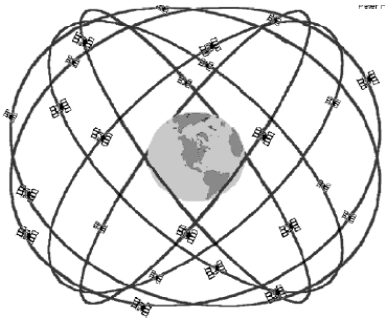
2.2 *Global Positioning System* (GPS)

Global Positioning System (GPS) adalah suatu sistem navigasi yang memanfaatkan satelit. Penerima GPS memperoleh sinyal dari beberapa satelit yang mengorbit bumi. Satelit yang mengitari bumi pada orbit pendek ini terdiri dari 24 susunan satelit, dengan 21 satelit aktif dan 3 buah satelit sebagai cadangan. Dengan susunan orbit tertentu, maka satelit GPS bisa diterima di seluruh permukaan bumi dengan penampakan antara 4 sampai 8 buah satelit. GPS dapat memberikan informasi posisi dan waktu dengan ketelitian sangat tinggi.

Assisted GPS atau A-GPS diperkenalkan untuk meningkatkan kinerja dari sistem navigasi GPS konvensional. Pengembangan dari A-GPS dilakukan di Amerika Serikat yang meletakan GPS pada telepon selular yang bertujuan untuk pengiriman panggilan darurat.

Pada GPS konvensional yang ada memiliki kesulitan dalam penyediaan data posisi yang dapat dipercaya dalam kondisi sinyal yang minim. Sebagai contoh ketika pemakai berada pada kondisi dikelilingi oleh gedung – gedung tinggi, atau ketika sinyal satelit lemah yang dikarenakan dalam keadaan di dalam ruangan atau di bawah pohon. Dalam hal inilah A-GPS memiliki keunggulan.

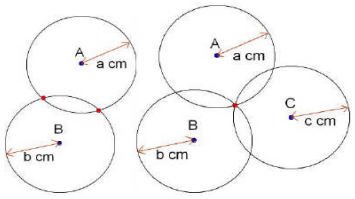
Beberapa A-GPS memerlukan sebuah aktivasi sambungan / koneksi jaringan ke telepon penyedia layanan data (GSM / CDMA) untuk mentransfer datanya, tetapi beberapa *device* tidak memerlukannya, hanya memposisikan dengan cepat dan akurat saja.



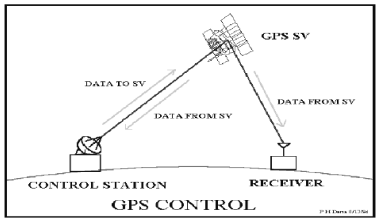
Gambar 2.2 Susunan orbit / konstelasi satelit-satelit GPS

2.2.1 Cara Kerja GPS

Untuk mengetahui posisi dari GPS, diperlukan minimal 3 satelit. Pengukuran posisi GPS didasarkan oleh sistem pengukuran matematika yang disebut dengan Triliterasi. Yaitu pengukuran suatu titik dengan bantuan 3 titik acuan. Misalnya Fulan berada di suatu kota A (disini kota kita anggap sebagai titik), tetapi dia tidak mengetahui dimana keberadaanya. Untuk mengetahuinya, Fulan bertanya kepada seseorang, dan orang tersebut menjawab bahwa dia 2 km dari kota B. Jawaban ini tidak memuaskan karena Fulan tidak tahu apakah anda di sebelah selatan, utara, barat, atau timur kota B. Kemudian Fulan bertanya kepada orang ke – 2 dan mendapat jawaban bahwa dia berada 5 km dari kota C. Dengan jawaban ini sudah dapat dibayangkan dimana posisi Fulan, hanya ada kemungkinan 2 titik berbeda yang berpotongan antara lingkaran dengan radius kota A dengan kota B dan lingkaran dengan radius kota A dengan kota C. Untuk lebih memperjelas lagi Fulan memerlukan orang ke – 3, misalnya Fulan berada di 1 km dari kota D. Dengan demikian Fulan mendapatkan perpotongan antara lingkaran dengan radius jarak kota A ke kota B, lingkaran antara kota A dan kota C, dan lingkaran antara kota A dan kota D. Dalam GPS kota A adalah alat penerima GPS, kota B, C, dan D adalah Satelit.



Gambar 2.3 Contoh Triliterasi pengukuran GPS



Gambar 2.4 Cara pengiriman sinyal GPS

2.2.2 Penerima GPS

Teknologi GPS telah membawa kemajuan yang besar pada sistem navigasi dan penentuan posisi. Dengan menggunakan 24 satelit dan stasiun – stasiun buminya, GPS mampu melacak keberadaan pesawat, kendaraan, kapal, laptop, ponsel, dan bahkan orang per orang. Ini semua menjadi mungkin karena teknologi GPS memiliki beberapa kemampuan sebagai berikut :

1. Memberikan data lokasi yang tepat untuk segala titik di muka bumi, pada segala kondisi cuaca.
2. Mampu menemukan lokasi dan memberi info tentang bagaimana menuju ke lokasi tersebut.
3. Menghemat waktu dan biaya *survey* pemetaan dengan hasil yang tetap serta akurat.
4. Ukuran alat yang semakin kecil dan murah, sehingga terjangkau oleh hampir semua orang.
5. Mampu melacak objek hingga titik lokasinya dengan akurat.

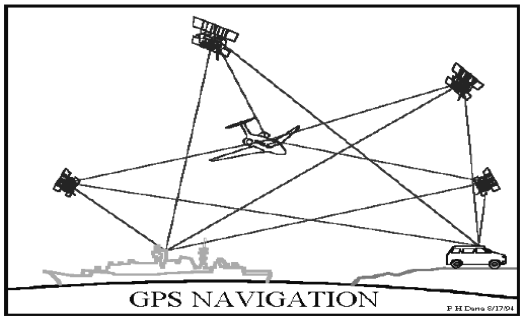
Di pasaran tersedia berbagai macam penerima GPS dengan *brand* dan fitur yang beragam, namun secara umum memiliki spesifikasi yang hampir sama contohnya sebagai berikut.

Tabel 2.1 Spesifikasi Penerima GPS secara Umum

GPS receiver	WAAS Enabled, Differential-ready, 12 parallel channel
Acquisition time	Approx. 15 seconds (warm start)
	Approx. 45 seconds (cold start)
	Approx. 5 minutes (First time)
Update rate	1/second, continuous
GPS accuracy	<15 meters, 95%
WAAS accuracy	<3 meters, 95%
Velocity accuracy	0.05 meter/sec
Interfaces	NMEA 0813, RTCM 104 (for DGPS)
	RS-232 for PC
Antenna	Quad-helix

2.2.3 Fitur GPS yang Dimanfaatkan

Data utama yang dihasilkan dari semua penerima GPS adalah data koordinat bujur dan lintang, namun ada dua fitur yang sangat berguna untuk melakukan pemetaan yaitu *waypoints* dan *track-log. Waypoints* digunakan untuk merekam titik – titik yang diinginkan secara manual. Sedangkan *track-log* digunakan untuk merekam jejak perjalanan secara otomatis dengan resolusi jejak yang bisa diatur. Fitur apa yang akan digunakan adalah tergantung pada peta apa yang ingin dibuat. Misalnya, fitur *waypoints* cocok dipakai untuk membuat peta letak kantor, landmark atau bangunan, dan fitur *track-log* cocok digunakan untuk membuat peta jalan atau batas wilayah.



Gambar 2.5 Navigasi menggunakan bantuan satelit-satelit GPS

2.3 Sistem Informasi Geografis (GIS)

2.3.1 Peta

Peta merupakan alat utama di dalam ilmu geografi, selain foto udara dan citra satelit. Melalui peta, seorang dapat mengamati kenampakan permukaan bumi lebih luas dari batas pandang manusia.

Peta adalah suatu gambaran atau representasi unsur – unsur ketampakan abstrak yang dipilih dari permukaan bumi, yang ada kaitannya dengan permukaan bumi atau benda – benda angkasa. Pada umumnya, peta digambarkan pada suatu bidang datar dan diperkecil atau diskalakan (menurut ICA / *International Cartographic Association*).

2.3.2 Komponen – komponen Peta

Peta yang baik biasanya dilengkapi dengan komponen – komponen peta, agar peta mudah dibaca, ditafsirkan dan tidak membingungkan. Adapun komponen – komponen yang harus dipenuhi dalam suatu peta antara lain.

1. Judul peta
2. Skala peta
3. Legenda atau keterangan
4. Tanda arah atau orientasi
5. Simbol dan warna
6. Sumber dan tahun pembuatan
7. Proyeksi peta

2.3.2.1 Judul Peta

Judul peta merupakan komponen yang sangat penting. Biasanya, sebelum membaca memperhatikan isi peta, pasti terlebih dahulu judul yang dibacanya. Judul peta hendaknya memuat / mencerminkan informasi yang sesuai dengan isi peta.

Judul peta biasanya diletakkan di bagian tengah atas peta. Tetapi judul peta dapat juga diletakkan di bagian lain dari peta, asalkan tidak mengganggu kenampakkan dari keseluruhan peta.

2.3.2.2 Skala Peta

Skala adalah perbandingan jarak antara dua titik sembarang di peta dengan jarak sebenarnya di permukaan bumi, dengan satuan ukuran yang sama. Skala ini sangat erat kaitannya dengan data yang disajikan.

Berdasarkan skalanya peta dapat digolongkan menjadi :

1. Peta Kadaster / Teknik

   Adalah peta yang mempunyai skala antara 1 : 100 sampai 1 : 5.000. Peta ini digunakan untuk menggambarkan peta tanah atau peta dalam sertifikat tanah.

2. Peta Skala Besar

   Adalah peta yang mempunyai skala 1 : 5.000 sampai 1 : 250.000. Peta skala besar digunakan untuk menggambarkan wilayah yang relatif sempit, misalnya peta kelurahan, peta kecamatan.

3. Peta Skala Sedang

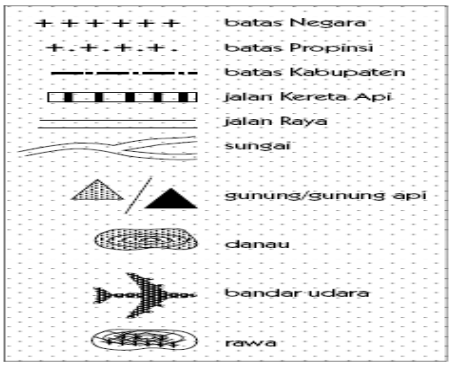
   Adalah peta yang mempunyai skala antara 1 : 250.000 sampai 1: 500.000. Peta skala sedang digunakan untuk menggambarkan daerah yang agak luas, misalnya peta propinsi Jawa Tengah, peta propinsi Maluku.

4. Peta Skala Kecil

   Adalah peta yang mempunyai skala 1 : 500.000 sampai 1 : 1.000.000 atau lebih. Peta skala kecil digunakan untuk menggambarkan daerah yang relatif luas, misalnya peta negara, benua bahkan dunia.

2.3.2.3 Legenda atau Keterangan

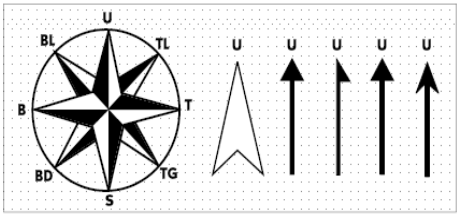
Legenda pada peta menerangkan arti dari simbol – simbol yang terdapat pada peta. Legenda itu harus dipahami oleh si pembaca peta, agar tujuan pembuatan peta itu mencapai sasaran. Legenda biasanya diletakkan di pojok kiri bawah peta. Selain itu legenda peta dapat juga diletakkan pada bagian lain peta, sepanjang tidak mengganggu kenampakan peta secara keseluruhan.



Gambar 2.6 Legenda atau Keterangan Peta

2.3.2.4 Tanda arah atau Orientasi

Tanda arah atau tanda orientasi penting artinya pada suatu peta. Gunanya untuk menunjukkan arah utara, selatan, timur dan barat. Tanda orientasi perlu dicantumkan pada peta untuk menghindari kekeliruan. Tanda arah pada peta biasanya berbentuk tanda panah yang menunjuk ke arah utara. Petunjuk ini diletakkan di bagian mana saja dari peta, asalkan tidak mengganggu kenampakan peta.



Gambar 2.7 Petunjuk arah atau Tanda Orientasi

2.3.2.5 Simbol dan Warna

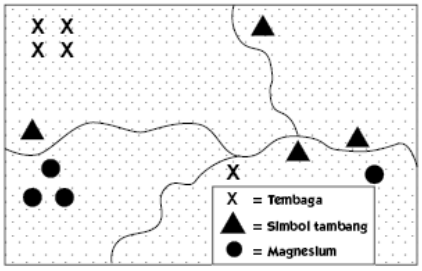
Agar pembuatan peta dapat dilakukan dengan baik, ada dua hal yang perlu mendapat perhatian, yaitu simbol dan warna.

Simbol

Pada peta terdapat simbol – simbol, gunanya agar informasi yang disampaikan tidak membingungkan. Simbol – simbol dalam peta harus memenuhi syarat, sehingga dapat menginformasikan hal – hal yang digambarkan dengan tepat. Adapun syarat simbol yang baik ialah sederhana, mudah dimengerti dan bersifat umum. Berikut merupakan jenis – jenis simbol, diantaranya.

1. Simbol titik

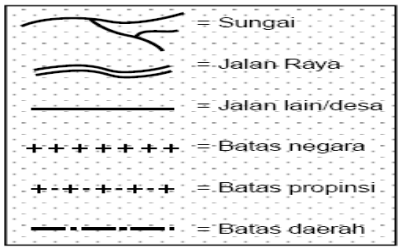
Digunakan untuk menyajikan tempat atau data posisi, seperti simbol kota, pertambangan, titik trianggulasi (titik ketinggian) tempat dari permukaan laut dan sebagainya.



Gambar 2.8 Simbol titik pada peta pertambangan

2. Simbol garis

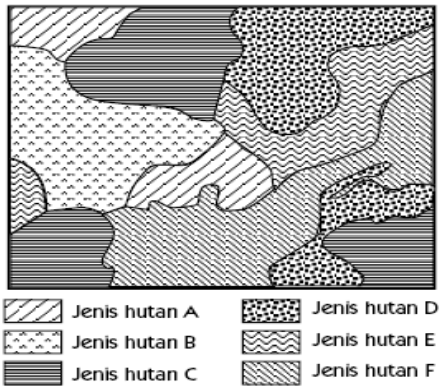
Digunakan untuk menyajikan data geografis misalnya sungai, batas wilayah, jalan, dan sebagainya.



Gambar 2.9 Contoh simbol garis

3. Simbol luasan (area)

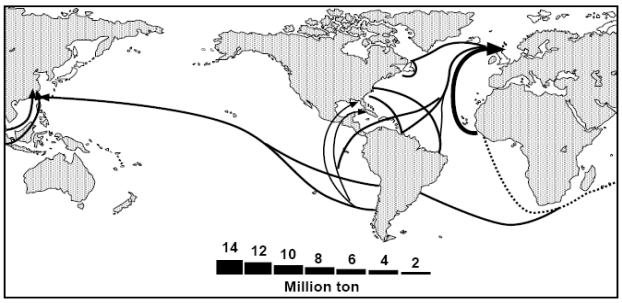
Digunakan untuk menunjukkan kenampakan area misalnya rawa, hutan, padang pasir dan sebagainya.



Gambar 2.10 Contoh simbol area

4. Simbol aliran

Digunakan untuk menyatakan alur dan gerak.



Gambar 2.11 Contoh peta pola pengiriman kopi

Warna

Tidak ada peraturan yang baku mengenai penggunaan warna dalam peta. Jadi penggunaan warna adalah bebas, sesuai dengan maksud atau tujuan si pembuat peta, dan kebiasaan umum.

Misalnya :

1. Untuk laut, danau digunakan warna biru.

2. Untuk temperatur (suhu) digunakan warna merah atau coklat.
3. Untuk curah hujan digunakan warna biru atau hijau.
4. Daerah pegunungan tinggi/dataran tinggi (2000 - 3000 meter) digunakan warna coklat tua.
5. Untuk dataran rendah (pantai) ketinggian 0 sampai 200 meter dari permukaan laut digunakan warna hijau.

2.3.2.6 Sumber dan Tahun Pembuatan

Sumber memberi kepastian kepada pembaca peta, bahwa data dan informasi yang disajikan dalam peta tersebut benar benar absah (dipercaya / akurat), dan bukan data fiktif atau hasil rekaan. Hal ini akan menentukan sejauh mana si pembaca peta dapat mempercayai data / informasi tersebut. Selain sumber, perhatikan juga tahun pembuatannya. Pembaca peta dapat mengetahui bahwa peta itu masih cocok atau tidak untuk digunakan pada masa sekarang atau sudah kadaluarsa karena sudah terlalu lama.

2.3.2.7 Proyeksi Peta

Untuk menghindari terjadinya kesalahan yang lebih besar, dalam ukuran (luas, jarak) bentuk permukaan bumi pada peta, maka dalam pembuatan peta digunakan proyeksi peta. Proyeksi peta adalah teknik pemindahan bentuk permukaan bumi yang lengkung (bulat) ke bidang datar.

2.3.3 Skala Peta

Dalam menggambar peta, model pesawat, dena gedung dan sebagainya didasarkan pada perbandingan senilai. Oleh karena itu masalah peta dan model berskala dapat diselesaikan dengan menggunakan perbandingan senilai. Yang dimaksud dengan skala adalah perbandingan ukuran gambar atau peta dengan ukuran sebenarnya (biasanya dalam ukuran *centimeter*).

Contoh :

Jarak Jakarta – Bandung 220 km, sedangkan jarak pada peta 11 cm, maka skalanya dapat dicari sebagai berikut.

Skala = 11 cm : 220 km
= 11 cm : 22.000.000 cm
= 1 : 2.000.000

2.4 *Extensible Markup Language* (**XML**)

XML ialah sebuah tujuan umum *markup language*. XML diklasifikasikan sebagai sebuah bahasa perluasan / *extensible* karena hal ini mengijinkan pengguna untuk mendefenisikan elemen – elemennya. Tujuan utamanya ialah untuk memfasilitasi pembagian dari struktur data diantara sistem informasi yang berbeda, terutama melalui internet, dan digunakan pada dokumen *encode* ataupun data serial. Dalam konteks terakhir, dapat dibandingkan dengan bahasa serialisasi berbasis teks lainnya seperti JSON dan YAML.

Dimulai sebagai sebuah *subset* dari *Standard Generalized Markup Language* (SGML) dan dirancang untuk menjadi secara relatif dapat terbaca oleh manusia. Dengan menambahkan batasan semantik, bahasa aplikasi dapat dimplementasikan dalam XML. XML biasanya dipakai sebagai bahasa spesifikasi untuk bahasa aplikasi tertentu.

XML direkomendasikan oleh *World Wide Web Consortium* (W3C). Merupakan bebas lisensi, direkomdendasikan bahwa menspesifikasikan baik itu *grammar* leksikal dan persyaratan *parsing.*

Keuntungan XML :

1. Berbasiskan teks.
2. Berdasarkan standar internasional.
3. Mendukung *Unicode*, memperbolehkan informasi apapun yang dibuat dapat dikomunikasikan.
4. Merupakan platform independen, yang relatif tidak terpengaruh pada perkembangan teknologi.

2.5 Pemrograman Berorientasi Objek

Java adalah sebuah bahasa pemrograman berorientasi objek yang diciptakan oleh *Sun Microsystem*, sebuah perusahaan besar yang berada di negara Amerika Serikat. Bahasa ini berkembang sangat pesat terutama untuk web-programming. Dalam beberapa tahun terakhir, Java telah merambah dunia *mobile*

dengan J2ME (Micro Edition : MIDlet, dipakai dalam *mobile-phone*, PDA, *smart-phone* dan sebagainya).

2.5.1 Kelas dan Objek

Dalam paradigma pemrograman berorientasi objek dikenal kelas dan objek. Kelas merupakan *blue print* dari objek – objek yang akan dibuat. Analogi kelas dan objek seperti *rancangan* model rumah dan *pembangunan* rumah – rumah, adapun proses pembuatan objek dari kelas dikenal dengan *instantiasi.* Sedangkan objek merupakan representasi nyata dari sebuah objek. Dalam *object oriented programming* (OOP) dengan Java, tidak mendefenisikan objek secara langsung melainkan bekerja dengan kelas dari sebuah objek.

2.5.2 Pewarisan (*Inheritance*)

Salah satu kelebihan pemrograman berorientasi objek adalah penggunaan ulang kode – kode yang telah dibuat. Pewarisan adalah salah satu cara untuk menggunakan kode – kode yang telah dibuat sebelumnya.

Untuk membuat suatu kelas dapat kita turunkan dengan pewarisan field – field dan metode pada kelas dasarnya. Dalam Java tidak dikenal pewarisan ganda, sehingga digunakan *interface*.

2.5.3 Polimorfisme (*Polymorphism*)

Salah satu pilar Pemrograman Berorientasi Objek yang lain adalah polymorfisme yaitu kemampuan beberapa objek bertipe sama bereaksi secara

berbeda terhadap "pesan" yang sama. Dalam pemrograman berorientasi obek Java, karakteristik ini berlaku untuk suatu tingkah laku (metode).

2.5.4 Pengkapsulan (*Encapsulation*)

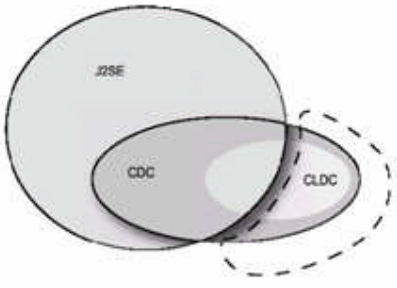
Pilar lain dari Pemrograman Berorientasi Objek adalah pengkapsulan, dimana pengembang software dapat menyembunyikan detail suatu objek. Hak akses *public* memungkinkan semua kelas mengaksesnya, hak akses *protected* hanya diberikan kepada kelasnya sendiri dan turunannya, serta kelas – kelas dalam satu paket sedangkan *private* hanya boleh diakses oleh kelasnya sendiri.

2.6 *Java 2 Micro Edition* (J2ME)

Dalam pemrograman, J2ME ialah sebuah spesifikasi dari subset *platform* Java yang diperuntukkan untuk menyediakan koleksi Java API tersertifikat untuk pengembangan perangkat lunak skala kecil, sumber daya peralatan seperti telepon selular, PDA dan sebagainya.

J2ME dirancang oleh *Sun Microsystem*, pada mulanya dikembangkan di bawah *Java Community Process* sebagai JSR 68 (*Java Specification Request*), perbedaan kebutuhan dari J2ME telah menghasilkan JSR yang terpisah. Perusahaan Sun Microsystem menyediakan referensi untuk implementasinya. Pada tanggal 22 Desember 2006, *source code* dari J2ME telah bersifat *open source* dibawah lisensi GNU (*General Public Lisence*).

Seperti yang ditampilkan pada gambar 2.1, J2ME terbagi menjadi dua yaitu CDC dan CLDC. Pada CLDC (*Connected Limited Device Configuration*) umumnya untuk aplikasi java pada *handphone* seperti produk-produk Siemens, Nokia, Motorola dll. Sedangkan CDC (Connected Device Configuration) umumnya digunakan pada perangkat dengan memori setidaknya 2 Megabyte. Subset API pada CLDC dijelaskan pada gambar di bawah ini.



Gambar 2.12 Struktur API pada J2ME

2.6.1 Perangkat Pengembangan

Selain JDK dalam pengembangan aplikasi dengan J2ME diperlukan juga *software* tambahan yakni *Sun Java Wireless Toolkit*. Dalam Toolkit ini terdapat API J2ME dan emulator beberapa perangkat nirkabel. *Toolkit* ini berfungsi untuk mengkompile dan mensimulasikan aplikasi pada emulator.

Keterangan direktori – direktori yang ada didalam *Sun Java Wireless Toolkit* adalah sebagai berikut.

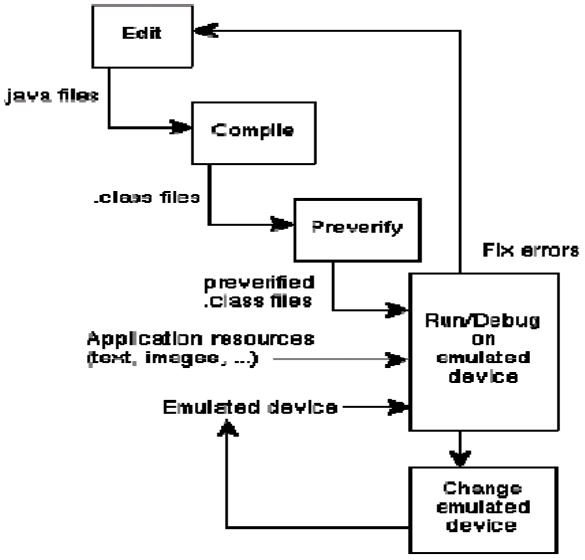
Tabel 2.2 Keterangan Direktori *Sun Java Wireless Toolkit*

File/Direktori	Keterangan
appdb\	Direktori berisi file-file database, seperti file-file RMS file
apps\	Direktori yang berisi demo aplikasi dan aplikasi-aplikasi yang dibuat dengan KToolbar
bin\	Batch file dan file-file executable tool
docs\	Direktori yang berisi dokumtasi API
lib\midpapi.zip	Arsip kelas-kelas API CLDC dan MIDP. File-file ini digunakan pada proses kompilasi dan preverification
sessions\	Direktori default yang berisi profiling, monitoring memori dan session file pada monitoring network
wktlib\devices\	Direktori yang berisi file-file properti device-device yang digunakan pada emulator

2.6.2 Tahapan Pengembangan

Setelah semua fungsional aplikasi sesuai dengan yang direncanakan, aplikasi perlu disimulasikan ke beberapa emulator sesuai dengan target pemakainya. Secara umum semua perangkat *wireless* yang telah mendukung Java (J2ME) maka aplikasi yang kita buat akan bisa dijalankan di perangkat tersebut, namun terkadang untuk tiap perangkat mempunyai kekhususan seperti lebar dan tinggi layar / *display*, sehingga mungkin kita perlu membuat beberapa macam distribusi aplikasi kita yang disesuaikan dengan perangkat targetnya.

Pengembangan aplikasi dengan J2ME memerlukan tahapan – tahapan seperti yang dijelaskan dalam gambar berikut ini.

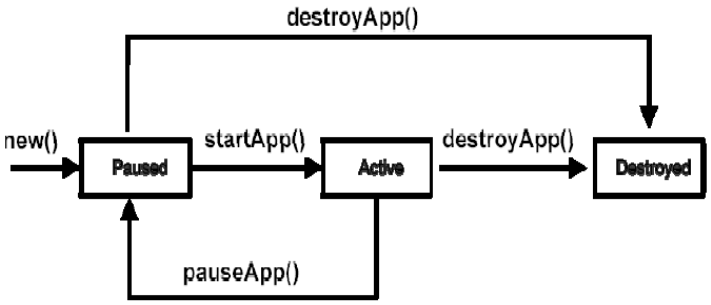


Gambar 2.13 Tahapan Pengembangan aplikasi dengan J2ME

2.6.3 Tahapan Pembuatan Program

2.6.3.1 MIDlet

MIDlet (*Mobile Information Device Applet*) mempunyai 3 kemungkinan kondisi yaitu *pause*, *active* dan *destroyed*. Ketika MIDlet mulai dijalankan, maka MIDlet berada pada kondisi aktif. Jika terdapat interupsi seperti adanya panggilan pada ponsel maka MIDlet berada pada kondisi *pause*. Dan ketika selesai menjalankan MIDlet maka diperlukan metode untuk membunuh MIDlet dengan *destroyApp().* Ketiga kondisi ini digambarkan pada diagram di bawah ini:



Gambar 2.14 Siklus hidup sebuah MIDlet

2.6.3.2 *Event Handling*

Event berguna untuk menangani interaksi *user* dengan program, misalnya *user* memilih sebuah menu dalam aplikasi MIDlet. Untuk menangani *event* perlu mengimplementasikan *interface CommandListener* dan atau *ItemListener*. *CommandListener* berfungsi untuk menangani jika *user* memilih perintah tertentu sedangkan *ItemListener* berfungsi untuk menangani jika *user* mengubah nilai seperti misalnya mengubah pilihan pada *ChoiceGroup.*

2.6.3.3 Antarmuka Aplikasi

Dalam sebuah aplikasi seringkali membutuhkan antarmuka seperti misalnya dibutuhkan masukan pada aplikasi yang akan dibangun. Sebagai contoh ialah melakukan *event input* dengan sebuah *TextField* dan keluaran dengan *StringItem*.

2.7 Pustaka JSR 179

Pustaka JSR 179 ialah spesifikasi yang mendefenisikan sebuah paket pilihan J2ME yang membuat aplikasi berbasis – lokasi dapat memperoleh sumber perangkat yang terbatas dan dapat dipakai oleh banyak profil aplikasi J2ME. Pustaka ini dirancang secara padat dan pustaka umum yang menghasilkan informasi tentang lokasi geografi posisi dari terminal (perangkat) ke aplikasi java yang dibuat. Pustaka ini mencakup perolehan informasi tentang lokasi geografi posisi dan orientasi dari terminal dan mengakses sebuah *database* dari landmark yang diketahui yang tersimpan di dalam terminal (perangkat).

Platform minimal yang diperlukan oleh pustaka ini ialah J2ME berjenis *Connected Limited Device Configuration* (CLDC) versi 1.1 dan *Connected Device Configuration* (CDC).

Pustaka ini terletak dalam paket *javax.microedition.location* mengandung kelas – kelas dasar yang diperlukan untuk meminta dan mendapatkan hasil lokasi. Sebagai contohnya yaitu kelas *LocationProvider* yang mencerminkan sebuah modul yang dapat menentukan lokasi terminal (perangkat). Hal ini mungkin dapat diimplementasikan menggunakan metode lokasi apapun, seperti satelit berdasarkan metode GPS.

Lokasi digambarkan oleh objek *Location* yang mengandung sebuah objek *QualifiedCoordinates* berupa koordinat geografi (garis lintang, garis bujur serta ketinggian dari permukaan laut) dan informasi tentang akurasinya. Untuk beberapa metode lokasi, objek *Location* juga mengandung sebuah objek

AddressInfo yang mencakup informasi tekstual informasi alamat seperti nama jalan. *Location* memberikan keakuratan koordinat sebagai jangkauan dari area *circular* yang mengindikasikan wilayah cakupan lokasi.

Dalam paket ini juga terdapat *database* landmark. Sebuah landmark juga diketahui sebagai sebuah lokasi fisik yang diassosiasikan dengan sebuah nama yang merepresentasikan lokasi dari si pengguna perangkat.

Pustaka ini terdiri dari beberapa kelas yaitu *AddressInfo, Coordinates, Criteria, Landmark, LandmarkException, LandmarkStore, Location, LocationException, LocationListener, LocationProvider, Orintation, ProximityListener, QualifiedCoordinates*. Dari semua kelas itu dipakai sesuai kebutuhan daripada aplikasi yang dibuat.

2.8 *Unified Modeling Language* (UML)

UML adalah bahasa grafis untuk mendokumentasikan, menspesifikasikan, dan membangun sistem perangkat lunak. UML berorientasi objek, menerapkan banyak level abstraksi, tidak bergantung proses pengembangan, tidak bergantung bahasa dan teknologi, pemaduan beberapa notasi di beragam metodelogi, usaha bersama dari banyak pihak, didukung oleh kakas – kakas yang diintegrasikan lewat XML. Standar UML dikelola oleh OMG (*Object Management Group*).

UML adalah bahasa pemodelan untuk menspesifikasikan, memvisualisasikan, membangun dan mendokumentasikan artifak – artifak dari sistem.

UML bukanlah :

1. Bahasa pemrograman visual, tapi bahasa pemodelan visual.
2. Spesifikasi kakas, tetapi spesifikasi bahasa pemodelan.
3. Proses, tetapi yang memungkinkan proses – proses.

2.8.1 Tujuan UML

Tujuan utama perancangan menggunakan UML adalah :

1. Menyediakan bahasa pemodelan visual yang ekspresif dan siap pakai untuk mengembangkan dan pertukaran model – model yang berarti.
2. Menyediakan mekanisme perluasan dan spesifikasi untuk memperluas konsep – konsep inti.
3. Mendukung spesifikasi independen bahasa pemrograman dan proses pengembangan tertentu.
4. Menyediakan basis formal untuk pemahaman bahasa pemodelan.
5. Mendorong pertumbuhan pasar kakas berorientasi objek.
6. Mendukung konsep – konsep pengembangan level lebih tinggi seperti komponen, kolaborasi, *framework* dan *pattern.*

2.8.2 Diagram dan Teknik Pemodelan UML

Diagram mengemukakan banyak hal, penggunaan notasi yang terdefenisi baik dan ekspresif adalah penting pada proses pengembangan perangkat lunak, yaitu :

1. Notasi standar memungkinkan pengembang mendeskripsikan skenario atau rumusan arsitektur dan kemudian mengkomunikasikan secara tidak ambigu.
2. Notasi yang bagus membebaskan otak untuk berkosentrasi pada masalah – masalah yang lebih lanjut.
3. Notasi yang baik memungkinkan mengeleminasi keperluan pemeriksaan konsistensi dan kebenaran keputusan dengan menggunakan *tool* terotomatisasi.

2.8.2.1 Diagram Struktur

Diagram ini untuk memvisualisasi, menspesifikasikan, membangun dan mendokumentasikan aspek statik dari sistem. Diagram struktur pada UML terdiri dari:

1. Diagram Kelas (*Class diagram*)

   Diagram ini menunjukkan sekumpulan kelas, *interface* dan kolaborasi serta keterhubungannya. Diagram kelas ditujukan untuk pandangan statik terhadap sistem.

2. Diagram Objek (*Objek diagram*)

   Diagram ini menunjukkan sekumpulan objek dan keterhubungannya. Diagram ini menunjukkan potongan statik dari instan – instan yang ada pada diagram kelas. Diagram ini memperlihatkan satu prototipe atau kasus tertentu yang mungkin terjadi. Diagram objek menyediakan notasi grafis formal guna memodelkan objek, kelas dan saling keterhubungan. Diagram objek berguna untuk *abstract modelling* dan perancangan program – program sesungguhnya.

3. Diagram Komponen (*Component diagram*)

   Diagram ini menunjukkan organisasi dan kebergantungan diantara sekumpulan komponen. Diagram ini merupakan pandangan statik terhadap implementasi sistem.

4. Diagram Pengembangan (*Deployment diagram*)

   Diagram ini menunjukkan konfigrasi pemrosesan saat jalan dan komponen – komponen yang terdapat di dalamnya. Diagram ini merupakan pandangan statik dari arsitektur. Pilihan model dan diagram yang digunakan dipengaruhi oleh bagaimana persoalan ditangani dan bagaimana solusi dibentuk.

2.8.2.2 Diagram Perilaku

Diagram ini untuk memvisualiasi, menspesifikasi, membangun dan mendokumentasikan aspek dinamis dari sistem. Diagram perilaku pada UML terdiri dari :

1. Diagram *Use – case* (*Use – case diagram*)

   Diagram ini menunjukkan sekumpulan kasus fungsional dan aktor (jenis kelas khusus) dan keterhubungannya.

2. Diagram Sekuen (*Sequen diagram*)

   Diagram ini menunjukkan interaksi yang terjadi antar objek. Diagram ini merupakan pandangan dinamis terhadap sistem. Diagram ini menekankan pada basis keberurutan waktu dari pesan – pesan yang terjadi.

3. Diagram Kolaborasi (*Collaboration diagram*)

   Diagram ini merupakan diagram interaksi. Diagram ini menekankan pada organisasi struktur dari objek – objek yang mengirim dan menerima pesan.

4. Diagram *Statechart* (*Statechart diagram*)

   Diagram ini berisi *state*, transisi, kejadian dan aktivitas. *Statechart* merupakan pandangan dinamis dari sistem. Diagram ini penting dalam memodelkan perilaku antarmuka, kelas, kolaborasi dan menekankan pada urutan kejadian. Penting untuk sistem reaktif yang dipicu kejadian di dunia nyata.

5. Diagram Aktivitas (*Activity diagram*)

   Diagram ini untuk menunjukkan aliran aktivitas di sistem. Diagram ini adalah pandangan dinamis terhadap sistem. Diagram ini penting untuk memodelkan fungsi sistem dan menekankan pada aliran kendali diantara objek – objek.

2.8.3 Notasi dalam UML

UML menyediakan beberapa notasi dan artifak standar yang bisa digunakan sebagai alat komunikasi bagi para pelaku dalam proses analisis dan desain. Artifak di dalam UML didefenisikan sebagai informsi dalam berbagai bentuk yang digunakan atau dihasilkan dalam proses pengembangan perangkat lunak. Yang perlu diperhatikan untuk menjaga konsisten antar artifak, selama proses analisis dan desain adalah bahwa setiap perubahan yang terjadi pada satu artifak harus juga dilakukan pada artifak sebelumnya.

1. Aktor

   Aktor adalah segala sesuatu yang berinteraksi dengan sistem aplikasi komputer. Jadi aktor dapat berupa manusia, perangkat keras, atau mungkin objek lain dalam sistem yang sama. Biasanya yang dilakukan aktor adalah memberikan informasi pada sistem dan atau memerintahkan sistem untuk melakukan sesuatu. Notasi Aktor dapat dilihat pada gambar 2.15 di bawah ini.

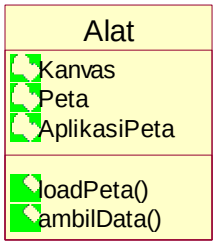


Gambar 2.15 Notasi Aktor

2. Kelas

   Kelas merupakan pembentuk utama dari sistem berorientasi objek karena kelas menunjukkan kumpulan objek yang memiliki atribut dan operasi yang sama. Kelas digunakan untuk mengabstraksikan elemen –

elemen dari sistem yang sedang dibangun. Kelas bisa untuk merepresentasikan baik perangkat lunak maupun perangkat keras, baik konsep maupun benda nyata.



Gambar 2.16 Notasi Kelas

3. *Interface*

Interface merupakan kumpulan operasi tanpa implementasi dari suatu kelas. Implementasi operasi dalam *interface* dijabarkan oleh operasi dalam kelas. Oleh karena itu keberadaan *interface* selalu disertai oleh kelas yang mengimplementasikan operasinya. *Interface* ini merupakan salah satu cara mewujudkan prinsip enkapsulasi dalam objek. Notasi *Interface* dapat dilihat pada gambar 2.17 di bawah ini.



Gambar 2.17 Notasi *Interface*

4. *Use case*

Use – case menjelaskan urutan kegiatan yang dilakukan aktor dan sistem untuk mencapai suatu tujuan tertentu walaupun menjelaskan

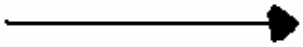
kegiatan. Namun hanya menjelaskan “apa” yang dilakukan oleh aktor dan sistem, bukan “bagaimana” aktor dan sistem melakukan kegiatan tersebut. Notasi *Use – case* dapat dilihat pada gambar 2.18 di bawah ini.



Gambar 2.18 Notasi *Use case*

5. Interaksi

Interaksi digunakan untuk menunjukkan baik aliran pesan atau informasi antar objek maupun hubungan antar objek. Biasanya interaksi ini dilengkapi juga dengan teks bernama *operation signature* yang tersusun dari nama operasi, parameter yang dikirim dan tipe parameter yang dikembalikan. Notasi Interkasi dapat dilihat pada gambar 2.19 di bawah ini.

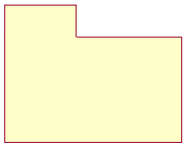


Gambar 2.19 Notasi Interaksi

6. Paket

Paket 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 mempermudah penglihatan (*visibility*) dari model yang

sedang dibangun. Notasi Paket dapat dilihat pada gambar 2.20 di bawah ini.



Gambar 2.20 Notasi Paket

7. Catatan (*Note*)

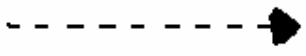
Note digunakan untuk memberikan keterangan dan komentar tambahan dari satu elemen sehingga bisa langsung terlampir dalam model. Catatan ini bisa ditampilkan ke semua elemen notasi yang lain. Notasi *Note* dapat dilihat pada gambar 2.21 di bawah ini.



Gambar 2.21 Notasi *Note*

8. Ketergantungan (*Depedency*)

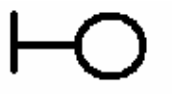
Ketergantungan merupakan relasi yang menunjukkan bahwa perubahan pada salah satu elemen memberi pengaruh pada elemen lain. Elemen yang ada di bagian tanda panah adalah elemen yang tergantung pada elemen yang ada di bagian tanpa tanda panah.



Gambar 2.22 Notasi *Depedency*

9. *Boundry Class*

Boundry Class adalah kelas yang menghubngkan *user* dengan sistem. Oleh karena itu sering disebut juga sebagai *user interface class.* Notasi *Boundry Class* dapat dilihat pada gambar 2.23 di bawah ini.



Gambar 2.23 Notasi *Boundry Class*

10. *Control Class*

Control class adalah kelas yang mengkoordinasi aktivitas dalam sistem. Kelas ini menghubungkan *boundry class* dengan entitas kelas. Notasi *Control Class* dapat dilihat pada gambar 2.24 di bawah ini.



Gambar 2.24 Notasi *Control Class*

11. *Entity Class*

Entity Class adalah kelas yang menghubungkan dengan data atau informasi yang digunakan oleh sistem. *Entity Class* ini adalah kelas yang menyimpan dan mengelola data. Notasi *Entity Class* dapat dilihat pada gambar 2.25 di bawah ini.



Gambar 2.25 Notasi *Entity Class*