BAB II

LANDASAN TEORI

Pada BAB II akan menjelaskan tentang teori-teori yang akan digunakan pada perancangan aplikasi, yang akan dibahas pada BAB III. Adapun teori-teori yang digunakan sebagai berikut :

2.1 Sastra Jawa[1]

2.1.1 Sejarah Sastra Jawa

Dimulai dengan sebuah prasasti yang ditemukan di daerah Sukabumi (Sukobumi), Pare, Kediri Jawa Timur. Prasasti yang biasa disebut dengan nama Prasasti Sukabumi ini bertarikh 25 Maret tahun 804 Masehi. Isinya ditulis dalam bahasa Jawa Kuno.

Setelah prasasti Sukabumi, ditemukan prasasti lainnya dari tahun 856 M yang berisikan sebuah sajak yang disebut kakawin. Kakawin yang tidak lengkap ini adalah sajak tertua dalam bahasa Jawa (Kuno). Biasanya sejarah sastra Jawa dibagi dalam empat masa :

1. Sastra Jawa Kuno
2. Sastra Jawa Tengahan
3. Sastra Jawa Baru
4. Sastra Jawa Modern

Terdapat pula kategori Sastra Jawa-Bali, yang berkembang dari Sastra Jawa Tengahan. Selain itu, ada pula Sastra Jawa-Lombok,

[1] "http://id.wikipedia.org/wiki/Sastra_Jawa"

Sastra Jawa-Sunda, Sastra Jawa-Madura, dan Sastra Jawa-Palembang.

Dari semua sastra tradisional Nusantara, sastra Jawa adalah yang paling berkembang dan paling banyak tersimpan karya sastranya. Tetapi setelah proklamasi RI, tahun 1945 sastra Jawa agak dianaktirikan karena di Negara Kesatuan RI, kesatuan yang diutamakan.

Bahasa Jawa pertama-tama ditulis dalam aksara turunan aksara Pallawa yang berasal dari India Selatan. Aksara ini yang menjadi cikal bakal aksara Jawa modern atau Hanacaraka yang masih dipakai sampai sekarang. Dengan berkembangnya agama Islam pada abad ke-15 dan ke-16, huruf Arab juga dipergunakan untuk menulis bahasa Jawa; huruf ini disebut dengan nama huruf pegon. Ketika bangsa Eropa datang ke Jawa, abjad Latin pun digunakan untuk menulis bahasa Jawa.

2.1.2 Kategori Sastra Jawa

Sastra Jawa secara global bisa dibagi menjadi dua kategori yaitu yang ditulis dalam bentuk prosa atau puisi. Dalam bentuk prosa biasanya disebut *gancaran* dan dalam bentuk puisi biasa disebut dengan istilah *tembang*. Sebagian besar karya sastra Jawa ditulis dalam bentuk *tembang* mulai dari awal bahkan sampai saat ini.

2.1.3 Bahasa Jawa[2]

Bahasa Jawa adalah bahasa yang digunakan penduduk suku bangsa Jawa terutama di beberapa bagian Banten terutama di kabupaten Serang dan Tangerang, Jawa Barat khususnya kawasan Pantai utara terbentang dari pesisir utara Karawang, Subang, Indramayu dan Cirebon, Jawa Tengah & Jawa Timur di Indonesia.

2.1.4 Penyebaran Bahasa Jawa

Penduduk Jawa yang berpindah ke Malaysia turut membawa bahasa dan kebudayaan Jawa ke Malaysia, sehingga terdapat kawasan pemukiman mereka yang dikenal dengan nama kampung Jawa, padang Jawa. Di samping itu, masyarakat pengguna Bahasa Jawa juga tersebar di berbagai wilayah Negara Kesatuan Republik Indonesia. Kawasan-kawasan luar Jawa yang didominasi etnis Jawa atau dalam persentase yang cukup signifikan adalah : Lampung (61%), Bengkulu (25%), Sumatra Utara (antara 15%-25%). Khusus masyarakat Jawa di Sumatra Utara ini, mereka merupakan keturunan para kuli kontrak yang dipekerjakan di berbagai wilayah perkebunan tembakau, khususnya di wilayah Deli sehingga kerap disebut sebagai *Jawa Deli* atau *Pujakesuma* (Putra Jawa Kelahiran Sumatera). Sedangkan masyarakat Jawa di daerah lain disebarkan melalui

[2] "http://id.wikipedia.org/wiki/Bahasa_Jawa"

program transmigrasi yang diselenggarakan semenjak jaman penjajahan Belanda.

Selain di kawasan Nusantara ataupun Malaysia. Masyarakat Jawa juga ditemukan dalam jumlah besar di Suriname, yang mencapai 15% dari penduduk secara keseluruhan, kemudian di Kaledonia Baru bahkan sampai kawasan Aruba dan Curacao serta Belanda. Sebagian kecil bahkan menyebar ke wilayah Guyana Perancis dan Venezuela.

Tabel 2.1 Tentang bahasa Jawa

Bahasa Jawa	
Dituturkan di:	Jawa (Indonesia), Jombang,Panguragan Suriname, Kaledonia Baru
Wilayah:	Jawa
Jumlah penutur:	80 –100 juta
Urutan ke:	11
Klasifikasi rumpun bahasa:	Austronesia Melayu-Polinesia Melayu-Polinesia Barat Sundik
Status resmi	**Bahasa Jawa**
Bahasa resmi di:	Hindia Belanda (hingga 1942)
Diatur oleh:	-

Kode bahasa	
ISO 639-1	Jv
ISO 639-2	Jav
SIL	JAN, JAS, JVS, OS, TES

2.1.5 Penjelasan Vokal

Tekanan kata (*stress*) direalisasikan pada suku kata kedua dari belakang, kecuali apabila sukukata memiliki sebuah pepet sebagai vokal. Pada kasus seperti ini, tekanan kata jatuh pada suku kata terakhir, meskipun sukukata terakhir juga memuat pepet. Apabila sebuah kata sudah diimbuhi dengan afiks, tekanan kata tetap mengikuti tekanan kata kata dasar. Contoh: /jaran/ (kuda) dilafazkan sebagai [j'aran] dan /pajaranan/ (tempat kuda) dilafazkan sebagai [paj'aranan].

Semua vokal kecuali /a/, memiliki alofon. Fonem /a/ pada posisi tertutup dilafazkan sebagai [a], namun pada posisi terbuka sebagai [o]. Contoh: /lara/ (sakit) dilafazkan sebagai [l'oro], tetapi /larane/ (sakitnya) dilafazkan sebagai [l'arane]

Fonem /i/ pada posisi terbuka dilafazkan sebagai [i] namun pada posisi tertutup lafaznya kurang lebih mirip [e]. Contoh: /panci/

dilafazkan sebagai [p'anci] , tetapi /kancil/ kurang lebih dilafazkan sebagai [k'ancel].

Fonem /u/ pada posisi terbuka dilafazkan sebagai [u] namun pada posisi tertutup lafaznya kurang lebih mirip [o]. Contoh: /wulu/ (bulu) dilafazkan sebagai [w'ulu] , tetapi /ʈuyul/ (tuyul) kurang lebih dilafazkan sebagai [ʈ'uyol].

Fonem /e/ pada posisi terbuka dilafazkan sebagai [e] namun pada posisi tertutup sebagai [ɐ]. Contoh: /lele/ dilafazkan sebagai [l'ele], tetapi /bebek/ dilafazkan sebagai [b'ɐbɐk].

Fonem /o/ pada posisi terbuka dilafazkan sebagai [o] namun pada posisi tertutup sebagai [ɔ]. Contoh: /loro/ dilafazkan sebagai [l'oro] , tetapi /boloŋ/ dilafazkan sebagai [b'ɔloŋ].

2.1.6 Penjelasan Konsonan

Fonem /k/ memiliki sebuah alofon. Pada posisi terakhir, dilafazkan sebagai [k̚]. Sedangkan pada posisi tengah dan awal tetap sebagai [k].

Fonem /n/ memiliki dua alofon. Pada posisi awal atau tengah apabila berada di depan fonem eksplosiva palatal atau retrofleks, maka fonem sengau ini akan berubah sesuai menjadi fonem homorgan. Kemudian apabila fonem /n/ mengikuti sebuah /r/, maka akan menjadi

[ɳ] (fonem sengau retrofleks). Contoh: /panjaŋ/ dilafazkan sebagai [p'aɲjaŋ], lalu /anɖap/ dilafazkan sebagai [k'aɳɖap]. Kata /warna/ dilafazkan sebagai [w'arɳo].

Fonem /s/ memiliki satu alofon. Apabila /s/ mengikuti fonem /r/ atau berada di depan fonem eksplosiva retrofleks, maka akan direalisasikan sebagai [ʂ]. Contoh: /warsa/ dilafazkan sebagai [w'arʂo], lalu /esʈi/ dilafazkan sebagai [k'eʂʈi].

2.1.7 Fonotaktik

Dalam bahasa Jawa baku, sebuah sukukata bisa memiliki bentuk seperti berikut: (n)-K_1-(l)-V-K_2.

Artinya ialah Sebagai berikut :

1. (n) adalah fonem sengau homorgan.
2. K_1 adalah konsonan eksplosiva ata likuida.
3. (l) adalah likuida yaitu /r/ atau /l/, namun hanya bisa muncul kalau K_1 berbentuk eksplosiva.
4. V adalah semua vokal. Tetapi apabila K_2 tidak ada maka fonem /ə/ tidak bisa berada pada posisi ini.
5. K_2 adalah semua konsonan kecuali eksplosiva palatal dan retrofleks; /c/, /j/, /ʈ/, dan /ɖ/.

   Contoh:

   1. a

2. an
3. pan
4. prang
5. njlen

2.1.8 Dialek-Dialek Bahasa Jawa

Bahasa Jawa pada dasarnya terbagi atas dua klasifikasi dialek, yakni :

1. Dialek daerah
2. Dialek sosial

Karena bahasa ini terbentuk dari gradasi-gradasi yang sangat berbeda dengan Bahasa Indonesia maupun Melayu, meskipun tergolong rumpun Austronesia. Sedangkan dialek daerah ini didasarkan pada wilayah, karakter dan budaya setempat. Perbedaan antara dialek satu dengan dialek lainnya bisa antara 0-70%. Untuk klasifikasi berdasarkan dialek daerah, pengelompokannya mengacu kepada pendapat E.M. Uhlenbeck, 1964, di dalam bukunya : "A Critical Survey of Studies on the Languages of Java and Madura", The Hague: Martinus Nijhoff[1].

Kelompok bahasa Jawa bagian Barat :

1. Dialek Banten
2. Dialek Indramayu-Cirebon
3. Dialek Tegal
4. Dialek Banyumasan

5. Dialek Bumiayu (peralihan Tegal dan Banyumas)

Kelompok pertama di atas sering disebut bahasa Jawa ngapak-ngapak.

Kelompok bahasa Jawa bagian Tengah :

1. Dialek Pekalongan
2. Dialek Kedu
3. Dialek Bagelen
4. Dialek Semarang
5. Dialek Pantai Utara Timur (Jepara, Rembang, Demak, Kudus, Pati)
6. Dialek Blora
7. Dialek Surakarta
8. Dialek Yogyakarta
9. Dialek Madiun

Kelompok kedua di atas sering disebut Bahasa Jawa Standar, khususnya dialek Surakarta dan Yogyakarta.

Kelompok bahasa Jawa bagian Timur :

1. Dialek Pantura Jawa Timur (Tuban, Bojonegoro)
2. Dialek Surabaya
3. Dialek Malang
4. Dialek Jombang
5. Dialek Tengger
6. Dialek Banyuwangi (atau disebut Bahasa Osing)

Kelompok ketiga di atas sering disebut Bahasa Jawa Timuran.

Dialek sosial dalam Bahasa Jawa berbentuk sebagai berikut :

1. Ngoko
2. Ngoko andhap
3. Madhya
4. Madhyantara
5. Kromo
6. Kromo Inggil
7. Bagongan
8. Kedhaton

Kedua dialek terakhir digunakan di kalangan keluarga Kraton dan sulit dipahami oleh orang Jawa kebanyakan.

2.1.9 Bilangan Dalam Bahasa Jawa

Bila dibandingkan dengan bahasa Melayu atau Indonesia, bahasa Jawa memiliki sistem bilangan yang agak rumit.

Tabel 2.2 Bilangan dalam bahasa Jawa

Bahasa	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**
Jawa Kuno	Sa	rwa	telu	Pat	lima	nem	pitu	wwalu	sanga	sapuluh
Kawi	Eka	dwi	Tri	Catur	panca	sad	sapta	as.t.a	nawa	dasa
Krama	setunggal	kalih	tiga	sekawan	gangsal	nem	pitu	wolu	sanga	sedasa
Ngoko	Siji	loro	telu	Papat	lima	nem	pitu	wolu	sanga	sepuluh

Angka	Ngoko	Krama
11	Sewelas	setunggal welas
12	Rolas	kalih welas
13	Telulas	tiga welas
14	Padbelas	sekawan welas
15	Limalas	gangsal welas
16	Nembelas	sekawan welas
17	Pitulas	Pitulas
18	Wolulas	Wolulas
19	Sanga-las	sanga-las
20	Rongpuluh	kalih dasa
21	Selikur	Selikur
22	Rolikur	kalih likur
23	Telulikur	tigang likur
24	Padlikur	sekawan likur
25	Selawe	Selangkung
26	Nemlikur	Nemlikur
30	Telung puluh	tigang dasa
31	Telung puluh siji	tigang dasa setunggal
32	Telung puluh loro	tigang dasa kalih

40	Patang puluh	sekawan dasa
41	Patang puluh siji	sekawan dasa setunggal
42	Patang puluh loro	sekawan dasa kalih
50	Skeet	Skeet
51	Skeet siji	sèket setunggal
52	Skeet loro	sèket kalih
60	Swidak	Swidak
61	Swidak siji	swidak setunggal
62	Swidak loro	swidak kalih
70	Pitung puluh	pitung dasa
100	Satus	setunggal atus
101	Satus siji	setunggal atus tunggal
102	Satus loro	setunggal atus kalih
120	Satus rong puluh	setunggal atus kalih dasa
121	Satus rong puluh siji	setunggal atus kalih dasa setunggal
200	Rong atus	kalih atus
500	Limang atus	gangsal atus
1.000	Sèwu	setunggal èwu
1.001	Sèwu siji	setunggal èwu setunggal
1.002	Sèwu loro	setunggal èwu kalih

1.500	Sèwu limang atus	setunggal èwu gangsal atus
1.520	Sèwu limang atus rong puluh	setunggal èwu gangsal atus kalih dasa
1.550	Sèwu limang atus skeet	setunggal èwu gangsal atus sèket
1.551	Sèwu limang atus skeet siji	setunggal èwu gangsal atus sèket setunggal
2.000	Rong èwu	kalih èwu
5.000	Limang èwu	gangsal èwu
10.000	sepuluh èwu	Sedasa èwu
100.000	Satus èwu	setunggal atus èwu
500.000	Limang atus èwu	gangsal atus èwu
1.000.000	Sayuta	setunggal yuta
1.562.155	sayuta limang atus swidak loro èwu satus sèket lima	setunggal yuta gangsal atus swidak kalih èwu setunggal atus sèket gangsal

Fraksi

1. 1/2 setengah, separo, sepalih (Krama)
2. 1/4 saprapat, seprasekawan (Krama)
3. 3/4 telung prapat, tigang prasekawan (Krama)
4. 1,5 karo tengah, kalih tengah (Krama)

2.1.10 Perbedaan Gaya Dalam Bahasa Jawa

Bahasa Jawa adalah bahasa yang membedakan gaya bahasa secara sosial menjadi tiga tingkatan, yaitu: ngoko, madya dan krama. Selain itu dikenal pula apa yang disebut kata-kata honorifik untuk merendahkan diri dan meninggikan lawan bicara. Kata-kata ini disebut kata-kata krama andhap dan krama inggil. Bahasa-bahasa lain yang juga membedakan gaya-gaya bahasa adalah bahasa Sunda, bahasa Madura dan bahasa Bali. Di bawah ini disajikan contoh sebuah kalimat dalam beberapa gaya bahasa yang berbeda-beda ini.

- Bahasa Indonesia: “Maaf, saya mau tanya rumah kak Budi itu, di mana?”
    1. Ngoko kasar: “Eh, aku arep takon, omahé Budi kuwi, nèng*ndi?’
    2. Ngoko alus: “Aku nyuwun pirsa, dalemé mas Budi kuwi, nèng endi?”
    3. Ngoko meninggikan diri sendiri: “Aku kersa ndangu, omahé mas Budi kuwi, nèng ndi?”
    4. Madya: “Nuwun sèwu, kula ajeng tanglet, griyané mas Budi niku, teng pundi?”
    5. Madya alus: “Nuwun sèwu, kula ajeng tanglet, dalemé mas Budi niku, teng pundi?”
    6. Krama andhap: “Nuwun sèwu, dalem badhé nyuwun pirsa, dalemipun mas Budi punika, wonten pundi?”
    7. Krama: “Nuwun sewu, kula badhé takèn, griyanipun mas Budi punika, wonten pundi?”

8. Krama inggil: “Nuwun sewu, kula badhe nyuwun pirsa, dalemipun mas Budi punika, wonten pundi?”

*nèng adalah bentuk percakapan sehari-hari dan merupakan kependekan dari bentuk baku ana ing yang disingkat menjadi (a)nêng.

Dengan memakai kata-kata yang berbeda, dalam sebuah kalimat yang secara tatabahasa berarti sama, seseorang bisa mengungkapkan status sosialnya terhadap lawan bicaranya dan juga terhadap yang dibicarakan. Namun harus diakui bahwa tidak semua penutur bahasa Jawa mengenal semuanya. Biasanya mereka hanya mengenal *ngoko* dan sejenis *madya*.

2.2 Aplikasi

Aplikasi merupakan program yang dibuat dengan suatu bahasa pemrograman tertentu yang bersifat umum dengan menggunakan metode algoritma tertentu.

2.3 Kamus

Kamus adalah kumpulan kata – kata berupa tulisan dengan menggunakan kaidah tata bahasa yang berlaku atau ejaan katanya sudah disempurnakan, sesuai dengan aturan yang dibuat oleh pemerintah.

2.4 Informasi

Informasi merupakan sesuatu yang nyata / setengah nyata dan dapat mengurangi derajat ketidakpastian tentang suatu keadaan atau kejadian. Pengertian lain bahwa informasi adalah data yang diolah menjadi bentuk yang lebih berguna bagi yang menerimanya. Informasi tersebut digunakan untuk membuat suatu keputusan dan melakukan tindakan yang berarti menghasilkan suatu tindakan yang lain yang akan membuat sejumlah data kembali yang baru. Data tersebut diolah kembali dan seterusnya membentuk suatu siklus data.

Sasaran yang harus dicapai agar suatu informasi dapat dikatakan sebagai informasi yang berkualitas adalah:

1. Akurat

   Bahwa informasi harus bebas dari kesalahan dan bersifat tidak bias (menyesatkan). selain itu juga berarti harus jelas mencerminkan maksuk dan tujuannya.

2. Tepat waktu

   Bahwa informasi tidak boleh terlambat dalam penerimaannya. Infromasi yang telah usang tidak akan mempunyai nilai lagi karena informasi merupakan landasan dalam pengambilan keputusan

3. Relevan

   Bahwa informasi harus mempunyai manfaat bagi pemakainya. Relevansi informasi untuk setiap orang yang berbeda-beda.

2.5 Konsep Pemrograman Berorientasi Objek[3]

Java merupakan salah satu bahasa pemrograman berbasis *object* murni. Semua tipe data diturunkan dari kelas dasar *Object*. Hal ini sangat memudahkan *programmer* untuk mendesain, membuat, mengembangkan dan mengalokasi kesalahan sebuah program dengan basis Java secara cepat, tepat, mudah dan terorganisir. Java adalah bahasa pemrograman berorientasi objek yang dikembangkan oleh *Sun Microsystems* sejak tahun 1991. Bahasa ini dikembangkan dengan model yang mirip dengan bahasa *C++* dan *Smalltalk*, namun dirancang agar lebih mudah dipakai dan *platform independent*, yaitu dapat dijalankan di berbagai jenis sistem operasi dan arsitektur computer. Untuk dapat menguasai pemrograman Java, harus mengerti dengan baik pemrograman berorientasi objek, karena Java merupakan bahasa pemrograman berorientasi objek. Pada bagian ini akan dibahas konsep-konsep penting pemrograman berorientasi objek, sehingga diharapkan kita akan lebih mudah mempelajari bahasa Java.

2.5.1 Objek

Pada dasarnya semua benda yang ada di dunia nyata dapat sebagai sebuah objek. Jika perhatikan lebih lanjut, pada dasarnya karakteristik yang utama pada sebuah objek , yaitu Setiap objek memiliki atribut sebagai status yang kemudian disebut sebagai ***state.*** Setiap objek memiliki tingkah laku yang kemudian akan sebagai ***behaviour.*** Contoh sederhananya adalah : objek sepeda Sepeda memiliki atribut (***state***) : pedal, roda, jeruji, dan warna. Sepeda

3 http://asep-hs.web.ugm.ac.id/Artikel/MODUL%20PEMROGRAMAN%20JAVA/MODUL%20FINAL/BAB%20IV%20KONSEP%20PEMROGRAMAN%20BERORIENTASI%20OBJEK.pdf

memiliki tingkah laku (***behaviour***) : kecepatannya kecepatannya menurun, dan perpindahan gigi sepeda. Dalam pengembangan perangkat lunak berorientasi objek, dalam perangkat lunak akan menyimpan ***state***-nya dalam variabel menyimpan informasi tingkah laku (***behaviour***) dalam ***method-method*** fungsi-fungsi/prosedur.

2.5.2 Class

Class berbeda dengan objek. *Class* merupakan *prototype* mendefinisikan variabel-variabel dan *method-method* secara Sedangkan objek pada sisi yang lain merupakan instansiasi dari suatu kelas.

2.5.3 Enkapsulasi

Dalam sebuah objek yang mengandung variabel-variabel dan *method -method*, dapat ditentukan hak akses pada sebuah variabel atau method dari objek. Pembungkusan variabel dan *method* dalam sebuah objek dalam bagian yang terlindungi inilah yang disebut dengan ***enkapsulasi***. Jadi, ***enkapsulasi*** dapat diartikan sebagai bungkusan (*wrapper*) pelindung program dan data yang sedang diolah. Pembungkus ini mendefinisikan perilaku dan melindungi program dan data yang sedang diolah agar tidak diakses sembarangan oleh program lain.

Manfaat dari proses enkapsulasi adalah :

1. Modularitas

Kode sumber dari sebuah objek dapat dikelola secara *independen* dari kode sumber objek yang lain.

2. Information Hiding

Karena kita dapat menentukan hak akses sebuah *variabel/method* dari objek, dengan demikian kita bisa menyembunyikan informasi yang tidak perlu diketahui objek lain.

2.5.4 Inheritance

Class dapat didefinisikan dengan referensi pada *class* yang lain yang telah terdefinisi. ***Inheritance*** merupakan pewarisan atribut dan *method* pada sebuah *class* yang diperoleh dari *class* yang telah terdefinisi tersebut. Setiap ***subclass*** akan mewarisi ***state*** (variabel-variabel) dan ***behaviour*** (*method-method*) dari ***superclass***-nya. ***Subclass*** kemudian dapat menambahkan ***state*** dan ***behaviour*** baru yang spesifik dan dapat pula memodifikasi (***override***) ***state*** dan ***behaviour*** yang diturunkan oleh ***superclass***-nya.

Keuntungan dari *inheritance* adalah :

1. ***Subclass*** menyediakan ***state***/***behaviour*** yang spesifik yang membedakannya dengan ***superclass***, hal ini akan memungkinkan programmer Java untuk menggunakan ulang ***source code*** dari ***superclass*** yang telah ada.

2. Programmer Java dapat mendefinisikan ***superclass*** khusus yang bersifat generik, yang disebut ***abstract class***, untuk mendefinisikan ***class*** dengan ***behaviour*** dan ***state*** secara umum.

Istilah dalam ***inheritance*** yang perlu diperhatikan :

1. Extends

Keyword ini harus kita tambahkan pada definisi *class* yang menjadi *subclass*.

2. Superclass

Superclass digunakan untuk menunjukkan *hirarki class* yang berarti *class* dasar dari *subclass/class* anak.

3. Subclass

Subclass adalah *class* anak atau turunan secara *hirarki* dari *superclass*.

4. Super

Keyword ini digunakan untuk memanggil *konstruktor* dari *superclass* atau menjadi variabel yang mengacu pada *superclass*.

5. Methode ***Overriding***

Pendefinisian ulang *method* yang sama pada *subclass*.

Dalam ***inheritance***, *method* ***overriding*** berbeda dengan *method* ***overloading***. Kalau *method* ***overriding*** adalah mendefinisikan kembali *method* yang sama, baik nama *method* maupun *signature* atau *parameter* yang diperlukan dalam subclass, kalau method ***overloading***

adalah mendefinisikan method yang memiliki nama yang sama, tetapi dengan *signature* yang berbeda dalam definisi *class* yang sama.

2.5.5 Polimorfisme

Kata ***polimorfisme*** yang berarti satu objek dengan banyak bentuk yang berbeda, adalah konsep sederhana dalam bahasa pemrograman berorientasi objek yang berarti kemampuan dari suatu variabel referensi objek untuk memiliki aksi berbeda bila *method* yang sama dipanggil, dimana aksi *method* tergantung dari tipe objeknya.

Kondisi yang harus dipenuhi supaya ***polimorfisme*** dapat diimplementasikan adalah :

1. *Method* yang dipanggil harus melalui variabel dari *basis class* atau *superclass*.
2. *Method* yang dipanggil harus juga menjadi method dari basis class.
3. *Signature method* harus sama baik pada *superclass* maupun *subclass*.
4. *Method access attribute* pada *subclass* tidak boleh lebih terbatas dari *basis class*.

2.5.6 Interface

Pada Java juga dikenal konsep ***interface***, yang merupakan *device* yang digunakan untuk komunikasi antar objek berbeda yang

tidak memiliki hubungan apapun. ***Interface*** bisa dikatakan sebagai protokol komunikasi antar objek tersebut.

2.6 Model Pengembangan Perangkat Lunak

Model rekayasa perangkat lunak yang biasa digunakan, salah satunya yaitu :

The Classic Life Cycle[4]

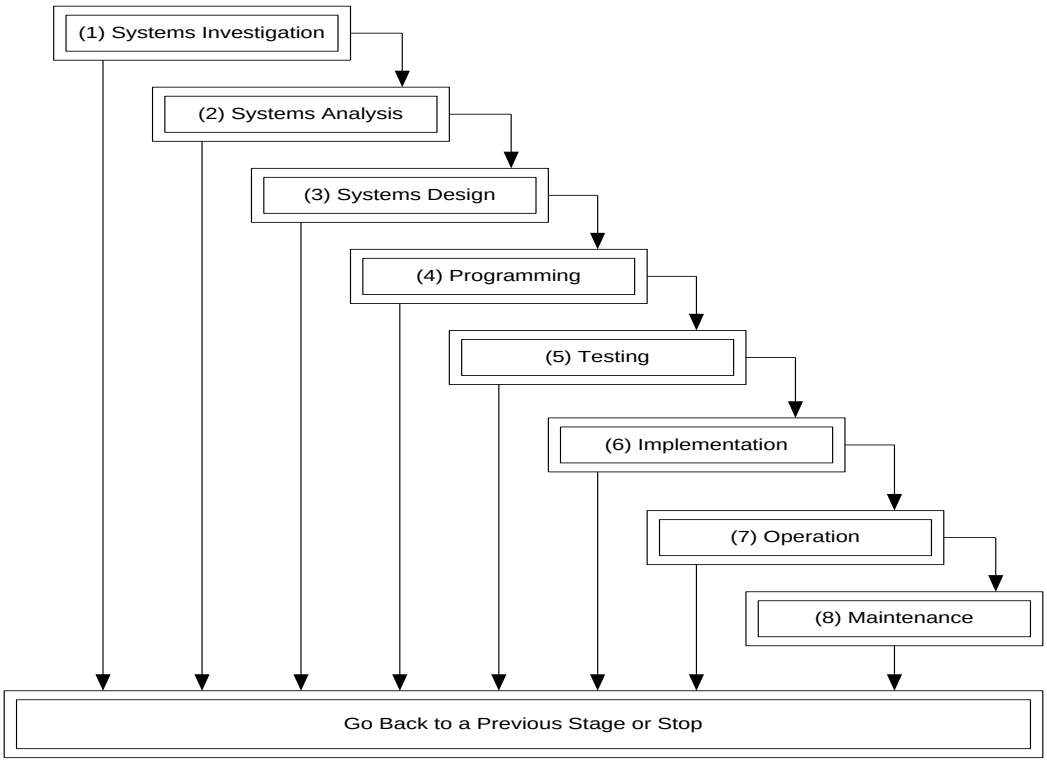
Daur hidup rekayasa perangkat lunak dibagi berdasarkan pendekatan sistematis dan berurutan sedangkan untuk pengembangan perangkat lunak dimana didalamnya terdapat metode *System Development Life Cycle*.

System Development Life Cycle (SDLC) adalah tahapan operasi yang terstruktur yang dibutuhkan untuk menyusun, membangun, dan membuat beroperasinya sebuah sistem informasi baru. SDLC dapat dipecah menjadi 4 tahap utama yaitu analisis, desain, implementasi dan perawatan.

Para pengembang menggunakan pendekatan *waterfall* untuk SDLC, di mana tugas-tugas pada satu tahapan diselesaikan sebelum pengembangan dilanjutkan ke tahapan berikutnya. Kini, apabila dibutuhkan, pengembang bisa bergerak maju mundur dari berbagai tahapan seperti ditunjukkan oleh gambar 2.1.

---

[4] Pressman, Roger S.(1992). *Software Engineering A Practitioner's Approach*, 3rd edition, McGraww-Hill, Inc., Singapore, p24



Gambar 2.1 Siklus Hidup Pengembangan Sistem (SDLC) 8 level

Contoh dari penerapan SDLC adalah model air terjun (*the waterfall cycle*). metode rekayasa perangkat lunak ini memberikan secara teknis bagaimana membangun sebuah *software*. Adapun tahap-tahap dari *classic life cycle* adalah sebagai berikut :

1. Investigasi Sistem (*systems Investigation*)

   Pada Tahap ini, dilakukan studi kelayakan untuk menentukan kemungkinan suksesnya proyek pengembangan sistem yang diajukan dan menaksir kelayakan proyek dari segi teknis, ekonomis, dan perilaku.

2. Analisa dan perancangan sistem

   Software selalu merupakan bagian dari sebuah sistem yang besar, maka pekerjaan dimulai dengan mengumpulkan kebutuhan bagi semua elemen-elemen sistem kemudian mengalokasikan beberapa *subset* dari

kebutuhan-kebutuhan tersebut ke *software*. Hal ini sangat penting ketika *software* harus berhubungan dengan elemen lain seperti *hardware*, manusia, dan basis data. Tahap ini meliputi pengumpulan kebutuhan pada tingkat sistem dengan sedikit analisa dan perancangan ditingkat atas.

3. Analisa Kebutuhan Perangkat Lunak

   Proses pengumpulan elemen sistem ditingkatkan dan dipusatkan secara khusus pada *software* untuk mengerti karakteristik dari program yang akan dibuat sistem analis *software* harus mengerti ruang lingkup informasi yang ingin dicakup dalam pembuatan *software* seperti fungsi-fungsi yang dibutuhkan, karakteristik, kinerja dan tampilan.

4. Perancangan

   Desain *software* adalah proses bertahap yang berfokus pada 4 atribut program yang berbeda yaitu struktur data, arsitektur perangkat lunak, implementasi kebutuhan dan detail procedural (algoritma). Proses perancangan menerjemahkan kebutuhan elemen sistem yang direpresentasikan kedalam sebuah *software* yang diperkirakan kualitasnya sebelum dilakukan pengkodean.

5. Pengkodean

   Pada tahap ini rancangan diterjemahkan kedalam bentuk yang dapat dibaca oleh mesin jika perancangan dilaksanakan secara detail. Pengkodean dapat dilakukan secara mekanis.

6. Pengujian

   Jika kode telah dibuat atau ditulis maka diadakan pengujian program. Pengujian juga dilakukan untuk memastikan masukan yang diberikan menghasilkan keluaran yang diinginkan.

7. Operasi (*operation*)

   Setelah konversi, sistem baru akan beroperasi dalam periode tertentu. Saat operasi sistem baru telah stabil, audit dilakukan guna menaksir kapabilitas sistem dan menentukan apakah penggunaannya tepat.

8. Pemeliharaan

   Software yang telah dihasilkan secara tidak langsung akan mengalami perbaikan hal ini disebabkan masih ditemukan adanya perubahan-perubahan di dalam lingkungan eksternalnya (seperti perubahan yang disebabkan oleh sistem operasi atau peralatan yang baru) dengan adanya pemeliharaan *software* perubahan yang terjadi akan lebih mudah dilakukan dibandingkan harus membuat ulang program baru.

2.7 Sejarah Java

Sejarah java berawal pada tahun 1991 ketika perusahaan Sun Microsystem memulai *Green Project*, yakni proyek penelitian untuk membuat bahasa yang akan digunakan pada chip-chip *embedded* untuk *device intelegent consumer electronic*. Bahasa tersebut haruslah bersifat *multiplatform*, tidak tergantung kepada yang memanufaktur *chip* tersebut.

Dalam penelitiannya, *Project Green* berhasil membuat *prototype* semacam PDE (*Personal Data Assistence*) yang dapat berkomunikasi antara satu dengan yang lain dan diberi nama *Star* 7. Ide berawal untuk membuat sistem operasi bagi *star* 7 berbasis C dan C++. Setelah berjalan beberapa lama, James Glosing, salah seorang anggota *team*, merasa kurang puas dengan beberapa karakteristik dari kedua bahasa tersebut berusaha mengembangkan bahasa lain. Bahasa tersebut kemudian dinamakan Oak, diinspirasi ketika dia melihat pohon diseberang kaca ruang kantornya. Belakangan Oak beralih nama menjadi Java.

Karena pada awalnya ditujukkan untuk pemrograman *device* kecil, Java memiliki karakteristik beruukuran kecil, efisien, dan *portable* untuk berbagai *hardware*. *Green Project* sempat terancam terhenti karena dalam perkembangannya, *device* ini memiliki pasar seperti yang diramalkan semula. Selanjutnya Java diarahkan untuk pemrograman internet. Secara kebetulan, fitur-fitur Java yang telah disebutkan sebelumnya sangat sesuai bagi pengembangan *internet* sehingga dalam beberapa tahun belakangan ini Java telah menjadi primadona untuk pemrograman yang berbasis internet.

2.8 Keunggulan Java

Java memiliki beberapa keunggulan bila dibandingkan dengan bahasa pemrograman lainnya. Ada beberapa aspek, yaitu :

1. Java bersifat sederhana dan *relative* mudah

   Java dimodelkan sebagian dari bahasa C++, namun dengan memperbaiki beberapa karakteristik C++, seperti mengurangi kompleksitas beberapa

fitur, penambahan fungsionalitas, serta penghilangan beberapa aspek pemicu ketidakstabilan sistem pada C++.

Sebagai contoh, java menggantikan konsep pewarisan lebih dari satu (*multiple inheritance*) dengan *interface*, menghilangkan konsep *pointer* yang sering membingungkan, otonatisasi sistem alokasi *memory*, dan sebagainya. Ini membuat java menjadi relatif sederhana dan mudah untuk dipelajari dibandingkan bahasa pemrograman lainnya.

2. Java berorientasi pada objek (*Object Oriented*)

   Java adalah bahasa pemrograman yang berorientasi objek (OOP), bukan seperti Pascal, Basic, atau C yang berbasis prosedural. Dalam memecahkan masalah, Java membagi program menjadi objek-objek, kemudian memodelkan sifat dan tingkah laku masing-masing. Selanjutnya Java menentujan dan mengatur interaksi antar objek yang satu dengan lainnya.

3. Java bersifat terdistribusi

   Pada dekade awal perkembangan PC (*Personal Computer*), komputer hanya bersifat sebagai *workstation* tunggal, tidak terhubung satu sama lain. Saat ini, sistem komputerisasi cenderung terdistribusi, mulai dari *workstation client*, *e-mail server*, *database server*, *web server*, dan sebagainya.

4. Java bersifat *Multiplatform*

   Dewasa ini kita mengenal banyak *platform Operating System*, mulai dari windows, Apple, berbagai varian UNIX dan Linux, dan lainnya. Umumnya, program yang dibuat dan dikompile di suatu *platform* hanya bisa dijalankan

di*platform* tersebut. Java bersifat *multiplatform* yakni dapat di-"terjemahkan" oleh Java *interpreter* pada berbagai sistem operasi.

5. *Thread* adalah proses yang dapat dikerjakan oleh program dalam suatu waktu. Java bersifat *multi threaded*, artinya dapat mengerjakan beberapa proses dalam waktu yang hampir bersamaan.

2.9 Jenis Program Java

Program java dapat dibedakan menjadi dua jenis, yaitu aplet dan aplikasi. Aplet adalah program yang dibuat oleh Java, dapat diletakan pada *web server* dan diakses melalui *web browser*. dalam hal ini *browser* yang digunakan adalah *browser* yang memiliki kemampuan java (misalnya *Netscape Navigator, Internet Explorer* dan *HotJava*).

Aplikasi adalah program yang dibuat dengan Java yang bersifat umum. Aplikasi dapat dijalankan secara langsung, tidak perlu perangkat lunak *browser* untuk menjalankannya. Aplikasi dapat dibayangkan seperti program yang anda tulis dengan bahasa C atau pascal. Setelah dikompilasi, program dapat dieksekusi secara langsung.

2.10 Dasar Bahasa Java

2.10.1 Karakter

Elemen terkecil pada pemrograman java adalah karakter. Yang dimaksud karakter bisa berupa :

1. Huruf (A sampai Z, a sampai z).
2. Angka (0 sampai dengan 9).

3. Simbol (misalnya * dan !).
4. Kode control (misalnya *formfeed* dan *newline*).

Berbeda dengan bahasa pendahulunya (misalnya C dan C++), java tidak menggunakan himpunan kode ASCII (*American Standard Code for Information Interchange*) untuk menyatakan karakter melainkan memakai himpunan *Unicode*.

Dengan menggunakan *Unicode*, tidak hanya huruf latin yang dapat dicakup, melainkan juga huruf-huruf yang digunakan oleh berbagai bangsa seperti Korea, Jepang, dan bahkan Bengali. *Unicode* dapat menampung berbagai macam huruf karena setiap karakter ditampung dengan 16 bit (ASCII standar hanya menggunakan 7 bit).

2.10.2 Kata Kunci

Java memiliki sejumlah kata yang bermakna khusus. Kata-kata ini digolongkan sebagai kata kunci atau kata tercadang. Kata kunci tidak dapat digunakan sebagai pengenal.

2.10.3 Pengenal

Pengenal (*identifier*) adalah nama yang diciptakan oleh pemrogram dan digunakan dalam program untuk memberi nama kelas atau variabel pada program.

Aturan pemberian nama pengenal pada java adalah sebagai berikut :

1. Karakter pertama berupa huruf, tanda garis bawah (_), atau tanda *dollar* ($).
2. Karakter kedua dan seterusnya dapat berupa sembarang huruf atau angka.
3. Panjang pengenal bebas (bisa berapa saja).
4. Huruf kapital dan huruf kecil diperlakukan berbeda.

2.10.4 Tipe Data Primitif

Java memiliki delapan tipe data primitif, meliputi empat tipe untuk bilangan bulat, dua tipe untuk bilangan tipe pengembang, dan sisanya untuk karekter dan *boolean*.

2.10.5 Literal

Literal adalah suatu nilai yang dituliskan pada kode sumber java. Misalnya, menulis 4 untuk menyatakan nilai bulat 4 dan “Selamat belajar java” untuk menyatakan suatu deretan karakter (*string*).

Literal pada java dapat dibedakan menjadi :

1. Literal bilangan
2. Literal karakter
3. Literal *boolean*
4. Literal *string*

2.10.6 Variabel

Variabel menyatakan suatu lokasi di dalam memori komputer yang digunakan untuk menyimpan suatu nilai dan nilai yang ada di dalamnya bisa diubah.

Variabel yang digunakan dalam program java perlu dideklarasikan. Didalam pendeklarasian nama variable dan tipe yang dikandung disebutkan. Bentuk pendeklarasian variabel :

tipe *namaVar* [,*namaVar*]

Tanda [ ] dalam [,*namaVar*] berarti bahwa yang berada di dalam tanda tersebut bersifat opsional.

Setelah variabel dideklarasikan variabel dapat diberi nilai. Caranya adalah dengan menggunakan operator =. Kaidahnya :

variabel = nilai;

Nilai yang diberikan dapat berupa suatu literal, variabel, atau bahkan suatu ekspresi (yang melibatkan operator).

2.10.7 Komentar

Komentar biasa dipakai dalam program dengan tujuan untuk memberikan penjelasan atau informasi kepada pembaca program, tanggal pembuatan program, fungsi program, ataupun penjelasan untuk bagian tertentu dalam program.

Komentar dapat dibuat dengan pasangan tanda /* dan */. Semua tulisan yang berada dalam pasangan tanda tersebut akan diperlakukan sebagai komentar. Selain itu komentar juga dapat

dibuat dengan menggunakan awalan //. Ini berarti semua tulisan yang berada di belakang tanda ini sampai akhir baris merupakan komentar.

2.10.8 Pernyataan

Pernyataan adalah suatu instruksi lengkap yang akan diproses oleh java. Beberapa contoh pernyataan :

final double PI = 3.14;

double radius = 30;

System.out.println(“selamat belajar java”);

Yang perlu dingat, sebuah pernyataan selalu diakhiri dengan tanda titik koma (;)

2.11 Teknologi Wireless Java[5]

Teknologi *mobile* merupakan trend teknologi TI yang sudah mulai matang. *Vensor-vendor* raksasa sudah mengembangkan teknologi ini. Diantaranya : Microsft, Sun, IBM, Oracle. Salah satu teknologi yang populer adalah teknologi Java. Konsep *“write once run everywhere”* membuat banyak pengembang aplikasi *mobile* mengadopsi Java. Sun Java menyusun bingkai kerja pengembangan aplikasi *mobile* ke dalam J2ME (*Java 2 Micro Edition*). Bingkai kerja J2ME memiliki dua cabang, yaitu CLDC (*Connected Limited Device Configuration*) dan CDC (*Connected Device Configuration*). Di atas CLDC, J2ME menspesifikasi MIDP sebagai

5 Tri Mardiono, Membangun solusi mobile business dengan Java, Elex Media komputindo, 2006, hal 9

platform pada peranti *mobile* dengan kekuatan *processor* dan memori yang relatif kecil.

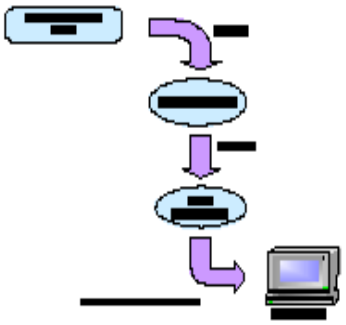
Selain itu, J2ME mengadopsi konsep *smart client*. Konsep ini berbicara tentang kemampuan *client* untuk meminta dan menangkap pesan dari *server*. Dalam proses meminta dan menangkap itu, *client* melakukan proses-proses validasi dan verifikasi data.

Secara konsep, teknologi *wireless* dapat dibagi dalam dua katagori, pertama untuk *local* dan kedua untuk area yang luas. Peralatan yang termasuk dalam katagori pertama misalnya adalah *remote control* untuk membuka atau mengunci mobil maupun garasi, telepon *cordless* 900Mhz, peralatan mainan dengan *radio control*, atau jaringan *wireless*. Peralatan *wireless* jenis pertama ini hanya bekerja untuk daerah dengan jangkauan yang tidak terlalu jauh. Sedangkan peralatan jenis aplikasi yang kedua diantaranya adalah *pager, handphone,* PDA *(Personal Digital Access),* 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 *handphone* menerima layanan dari sebuah *wireless carrier* atau perusahaan yang mengoperasikan *celltower* tersebut. Aplikasi komunikasi bergerak, dalam perkembangan awal masing-masing *vendor* menghasilkan *platform* aplikasi dan sistem operasi sendiri. Sehingga sebuah peralatan *handphone* Nokia dan Siemens mempunyai *platform* aplikasi masing-masing. Perbedaaan aplikasi menyebabkan suatu *platform* aplikasi maupun sistem operasi dalam *handphone* Nokia tidak dapat dijalankan dalam peralatan *handphone*

Siemens misalnya. Sehingga berakibat memperburuk pengembangan aplikasi-aplikasi yang baru. Standarisasi yang dilakukan untuk membuat suatu bahasa pemrograman yang memiliki kebebasan *platform* atau *platform independence*. Salah satu teknologi Java adalah "*write once run everywhere*", sehingga protabilitas Java merupakan suatu kekuatan yang dimiliki Java. Java dijalankan pada sistem operasi apapun tanpa perlu kompilasi ulang program Java yang dibuat. Untuk komunikasi bergerak, *Sun Microsystem* mengenalkan *Java* 2 *Micro Edition* (J2ME) yang merupakan salah satu bagian teknologi Java yang digunakan untuk aplikasi Java yang berjalan pada perangkat *mobile device* dan teknologi aplikasi *wireless*.

2.11.1 Java Virtual Machine (JVM)

Java *Virtual Machine* adalah *software* yang berfungsi untuk menjalankan program Java supaya dapat dimengerti oleh komputer. Kode program Java ditulis menggunakan *editor teks* seperti Notepad, Textpad, Editplus, Jcreator dan lainnya. Java *Compiler* yang digunakan untuk mengkompilasi kode program Java dirancang untuk menghasilkan kode yang netral terhadap semua arsitektur perangkat keras (*hardware*) yang disebut sebagai Java *Bytecode (*.class)*. Dan JVM merupakan basis dari Java *platform* dan menjembatani antara *bytecode* dengan *hardware.*



Gambar 2.2 Java *Virtual Machine*

2.11.2 Java Application Programming Interface (Java API)

Java API merupakan komponen-komponen dan kelas Java yang sudah jadi, yang memiliki berbagai kemampuan. Kemampuan untuk menangani objek, *string*, angka dan sebagainya.

2.11.3 Applet

Java *Applet* merupakan program Java yang berjalan di atas *browser.* Penggunaan *applet* ini akan membuat halaman *HTML* lebih dinamis dan menarik.

2.11.4 Java Database Connectivity (JDBC).

JDBC API terdiri atas class dan interface yang ditulis dalam bahasa Java untuk sebagai alat Bantu bagi pembuat program (*developer*) dan menyediakan sekumpulan API untuk mengatur keamanan mengakses *database* seperti *Oracle, MySQL, PostgreSQL, Microsoft SQL Server*. Jadi keunggulan *API JDBC*

dapat mengakses sumber data dan berjalan pada semua *Platform* yang mempunyai *Java Virtual Machine* (JVM).

2.11.5 Java 2 Platform

Pada Java 2 *Platform* yang digunakan terbagi dalam empat *Platform*, yaitu :

2.11.5.1 Java 2 Platform, Standard Edition (J2SETM)

Platform digunakan untuk menjalankan dan mengembangkan aplikasi Java pada level *Personal Computer* (PC). *Platform* ini berisi *class-class* inti pada Java dan *Graphical User Interface* (GUI).

2.11.5.2 Java 2 Platform, Micro Edition (J2METM)

Platform ini digunakan untuk menjalankan dan mengembangkan aplikasi-aplikasi Java pada *handheld devices* atau perangkat-perangkat semacam *handphone*, *Personal Digital Assistance* (PDA) dan *PocketPC*

2.11.5.3 Java 2 Platform, Enterprise Edition (J2EETM)

Platform ini berupa paket yang berisi *class - class* dan interface *-interface* yang digunakan untuk menjalankan dan mengembangkan aplikasi Java berbasis *web*, seperti *class* –class *Servlet*, *Java Server Pages* (JSP) dan *EnterpriseJavaBeans (EJB)* serta Java CORBA.

2.11.5.4 Java 2 Platform, Micro Edition (J2METM)

Komponen-komponen J2ME terdiri dari Java *Virtual Machine* (JVM) yang digunakan untuk menjalankan aplikasi Java pada emulator atau *handheld device*, Java API (*Aplication Programming Interface*) dan *tools* lain untuk pengembangan aplikasi Java semacam emulator *Java Phone*, *emulator* Motorolla dari J2ME *wireless toolkit*.

2.12 Keunggulan J2ME [6]

Salah satu kelebihan Java yang paling signifikan adalah *run everywhere*. Dengan kelebihan ini, para developer yang sudah terbiasa mengembangkan aplikasi dalam bingkai kerja J2SE dan J2EE, akan mampu bermigrasi dengan mudah untuk mengembangkan aplikasi J2ME. Selain itu, Java merupakan *platform* yang memiliki banyak keunggulan lain. Keunggulan Java secara umum adalah :

1. Multiplatform

Aplikasi J2ME bisa berjalan di atas banyak *platform* yang didalam nya terdpat JVM (*Java Virtual Machine*). Beberapa *platform* yang tersedia *installer mobile* JVM-nya antara lain : Windows CE, Symbian, embedded Linux dan sebagainya.

6 Tri Mardiono, Membangun solusi mobile business dengan Java, Elex Media komputindo, 2006, hal 10

2. **Robust**

   Kode-kode Java adalah kode -kode yang robust, karena *Virtual Machine* mengatur keamanan proses eksekusi aplikasi. *Java Virtual Machine* menyediakan *garbage collector* yang bertugas untuk mencegah kebocoran *memory*.

3. **Terintegrasi dengan baik**

   J2ME bisa terhubung dengan *back end J2EE serve*r dan *Web services* dengan mudah, karena J2ME menyediakan *library* API RMI dan *web services.*

4. **Berorientasi objek**

   Java merupakan salah satu bahasa pemrograman yang ,murni berorientasi obyek. Hal ini mempermudah dan mempercepat pengembangan sistem yang dikembangkan dengan metode analisa dan dalam berorientasi obyek.

2.13 Arsiterktur J2ME

Banyaknya jenis dan tipe peranti *mobile* membuat sulit pencapaian standar kinerja dan portabilitas. Meskipun J2ME menerapkan konsep *run anywhere*, pengembang J2ME menspesifikasikan beberapa arsitektur yang terbagi atas : konfigurasi, profil dan paket opsi (*optional package*). tujuan dari spesifikasi J2ME itu demi mencapai kinerja dengan memanfaatkan kelebihan peranti sekaligus mencapai portabilitas.

Konfigurasi : *virtual machine* yang menyediakan beberapa pustaka kelas. Konfigurasi menyediakan fungsi dasar dengan karakteristik yang sama. Contohnya : fungsi koneksi jaringan dan manajemen *memory*.

Sementara itu, *profile* menyediakan lingkungan pustaka-pustaka API untuk membangun aplikasi *mobile*. Paket opsi dibuat untuk menyediakan fungsi-fungsi pada peranti *mobile* yang lebih spesifik. Contohnya peranti yang memiliki akses *buetooth* memerlukan API *bluetooth*. Dalam pengembangan aplikasi *wireless* dengan Java, J2ME dibagi menjadi dua buah bagian diantaranya ialah bagian *configuration* dan *profile*.

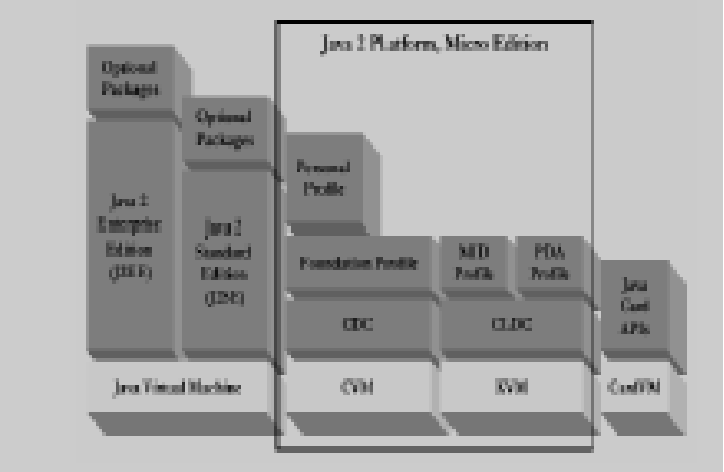
2.13.1 Lapisan Konfigurasi (*Configuration Layer*)

J2ME mempunyai dua konfigurasi yaitu `*Connected Limited Device Configuration* (CLDC) dan *Connected Device Configuration* (CDC).

2.13.2 Lapisan Profil (*Profile Layer*)

J2ME mempunyai beberapa profil antara lain :

1. *MOBILE INFORMATION DEVICE PROFILE* (MIDP)
2. *Foundation Profile* (FP)
3. *Personal Profile*
4. *Personal Digital Assistance* (PDA) *Profile*



Gambar 2.3 Konfigurasi pada J2ME

2.13.3 Connected Limited Device Configuration (CLDC)[7]

Merupakan konfigurasi untuk peranti *mobile* dengan kapasitas terbatas. *CLDC* memiliki keterbatasan fungsi dan fitur, seperti operasi matematika, *string*, I/O, JNI dan sebagainya. Dalam arsitektur J2ME, CLDC sebagai *virtual machine*

Pustaka CLDC terbagi atas dua kategori, yaitu :

1. Pustaka yang merupakan subset dari J2SE

   CLDC mendukung kelas-kelas yang diturunkan dari J2SE 1.3.3. CLDC tidak mengganti sintaks, semantik, dan identitas dari kelas J2SE. Kelas-kelas yang diturunkan dari J2SE, antara lain :

   1. kelas-kelas sistem
   2. kelas-kelas tipe data
   3. kelas-kelas *collection*
   4. kelas-kelas I/O

7 Tri Mardiono, Membangun solusi mobile business dengan Java, Elex Media komputindo, 2006, hal 13

5. kelas-kelas tanggal dan jam
6. kelas-kelas tambahan
7. kelas-kelas *exception*

2. Pustaka CLDC yang spesifik tapi bisa di-*mapping* ke dalam J2SE.

   CLDC hanya mengambil sedikit fitur yang ada pada J2SE, namun keterbatasan itu membuat CLDC mampu diadopsi oleh banyak tipe peranti *mobile*. Popularitas dan keberhasilan J2ME berjalan bersama profil MIDP yang berdiri di atas CLDC. Selain itu , CLDC juga mempunyai banyak paket opsional yang berjalan di atas profil MIDP. Pada awalnya, MIDP (versi 1.0) memiliki banyak kekurangan, terutama dalam hal keamanan data dan user interface. Namun, kekurangan-kekurangan itu telah diperbaiki dalam MIDP 2.0. Kekurangan itu menjadi lebih banyak lagi yang tertutup,karena pada MIDP 2.0 telah tersedia paket-paket opsional, yaitu :

   1. Paket opsional untuk PDA: paket ini didesain untuk produk-produk PDA seperti Pal. Paket ini menambah fungsi API yang spesifik seperti PIM
   2. *Mobile media* API: pustaka pendukung untuk membangun aplikasi yang mengakses *audio-video*.
   3. *Wireless messaging* API : pustaka pendukung untuk embangun aplikasi messaging seperti SMS

4. *Location* API : pustaka pendukung untuk mencari lokasi peranti *mobile*.

   Spesifikasi *CLDC* adalah sebagai berikut

   1. Mengimplementasikan subset dari J2SE
   2. JVM yang digunakan dikenal dengan nama *KVirtual Machine* (KVM)
   3. Digunakan pada perangkat *handheld* dengan ukuran memori terbatas (160 -512 Kbytes)
   4. Prosesor : 16 Bit atau 32 Bit

      Pada bagian ini secara detail diperlukan untuk pengembangan aplikasi *wireless* dengan MIDP implementasinya CLDC digunakan untuk program Java pada perangkat keras dengan ukuran memori yang terbatas, pada 160 dengan 512 Kilobyte. Akibatnya, fitur fitur yang kurang penting diimplementasikan dalam *handheld device* bersangkutan dari Java 2 harus dibuang.

2.13.4 Connected Device Configuratuion (CDC)[8]

CDC mampu menggunakan seluruh fitur Java *2 Virtual Machine*. CDC juga mampu menggunkan hampir seluruh fitur J2SE, besarnya permintaan CDC akan sumber daya, membuat CDC tidak bsa digunakan di banyak peranti *mobile*. Hanya peranti *mobile* berkapisitas besar saja yang mampu menggunakan CDC. Diatas

8 Tri Mardiono, Membangun solusi mobile business dengan Java, Elex Media komputindo, 2006, hal 15

CDC ada profil yang memiliki fitur-fitur yang kaya. Profil tersbut adalah :

1. *Foundation profile* : menyediakan lingkungan koneksi jaringan yang sangat kaya, mampu mendukung banyak protokol komunikasi dan keamanan data, tidak menyediakan pustaka GUI
2. *Personal basis profile* : menyediakan pustaka GUI yang berjalan di atas *Foundation Profile*. Targetnya adalah PDA *high-end*, dan aplikasi *internet*.
3. *Personal profile* : fungsinya sama dengan *personal basis profile*, *naun Personal profile* digunakan pada peranti yang lebih besar (minimum 2,5 Mb).
4. *Game profile* : berguna untuk mebangun aplikasi *game* pada peranti *mobile*.

Selain profil-profil standar, CDC juga mempunya paket-paket opsional. Paket-paket opsional ini adalah bagian ari paket opsional J2SE. Paket-paket tersebut adalah :

1. Paket RMI : berfungsi untuk mendukung aplikasi J*2SE* dan *J2EE* berbasiskan RMI. Palet RMI juga mampu mengakses teknologi ini untuk peranti jaringan yang terdistribusi.
2. Paket JDBC : berfungsi untuk akses basis data secara langsung dari peranti *mobile*. Paket JDBC merupakan bagian dari JDBC 3.0

3. Paket grafis dan UI tambahan : paket ini menyediakan pustaka Swing, Java 2D, I/O dan IMF API. Paket ini juga mengadopsi beberapa kelas baru dari J2SE 1.4.1

Tabel 2.3 Perbandingan CLDC dan CDC[9]

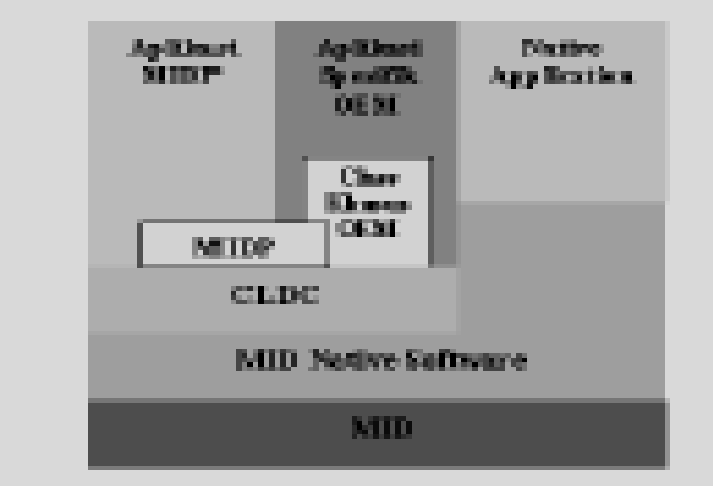
CLDC	CDC
Mengimplementasikan subset dari J2SE	Mengimplementasikan seluruh fitur dari J2SE
JVM yang digunakan adalah KVM	JVM yang digunakan adalah CVM
Digunakan pada perangkat *handled* (*handphone*, PDA, *two way pager*) dengan memory terbatas (160- 512 kb)	Digunakan pada perangkat *handled* (internet TV, Nokia *Communicator*, *cat* TV) dengan memory minimal 2 Mb
Procesor : 16-32 bit	Prosecor : 32 bit

2.14 MIDlets

Aplikasi yang berjalan pada sebuah perangkat yang mendukung MIDP disebut dengan MIDlets, atau lebih singkatnya MIDlet merupakan aplikasi yang dibuat menggunakan *Java 2 Micro Edition* dengan *profile Mobile Information Device Profile* (MIDP). MIDP dikhususkan untuk

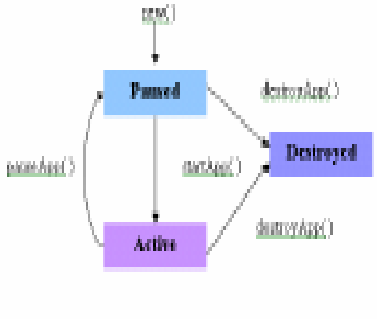
9 http://inf.untag-sby.ac.id/pustaka/public_html/Campuran/PDF/mic-j2me-01.pdf

digunakan pada *handset* dengan kemampuan *CPU, memori, keyboard* dan *layer* terbatas, seperti *handphone, pager*, PDA dan sebagainya. Pada Gambar dibawah Menunjukkan bahwa aplikasi yang mendukung perangkat MIDP adalah aplikasi MIDlet yang juga termasuk bagian dari *Java 2 Micro Edition.*



Gambar 2.4 MIDLET

2.14.1 Daur Hidup (*LifeCycle*) MIDlet



Gambar 2.5 Daur Hidup MIDLET

Lifecycle dari sebuah MIDlet ditangani oleh *Application Management Software* (AMS). AMS adalah sebuah lingkungan tempat siklus dari sebuah MIDlet, mampu untuk diciptakan, dijalankan, dihentikan maupun dihilangkan. AMS sering pula disebut dengan *Java Application Manager* (JAM). MIDlet memiliki beberapa state, yaitu *Pause, Active* dan *Destroy*. Ketika masing-masing state dipanggil, beberapa *method* yang bersesuaian dipanggil. *Method-method* tersebut merupakan bawaan dari J2ME. Untuk menjelaskan proses MIDlet dalam *Java Aplication Manager* (JAM)

2.14.2 Status MIDlet

1. ***High Level API***

   Kelas-kelas yang menyediakan fungsionalitas untuk pembuatan GUI pada MIDP ada pada paket *javax.microedition.lcdui*. Pada paket tersebut terdapat tiga *interface* dan 21 kelas. *Interface* tersebut adalah *Display, Screen* dan *Form*.

2. ***Low Level API***

   Pada level pemrograman yang lebih rendah (*low level*), akan ditemukan fungsionalitas yang lebih spesifik ke jenis *handheld* yang digunakan. Kelas –kelas untuk pemrograman GUI pada level yang lebih rendah ini diimplementasikan oleh kelas*javax.microedition.lcdui.Canvasjavax.microedition.lcdui.Graphics*. Kelas *Canvas* memungkinkan pengguna untuk menggambar garis, titik dan elemen-elemen dasar lain.

2.15 Basis Data

Basis data adalah kumpulan dari data-data yang saling berkaitan. Dalam pengertian lain basis data dapat diartikan sebagai kumpulan *record-record* atau *file-file* yang terorganisir dengan baik untuk kegunaan tertentu. Basis data dapat juga didefinisikan sebagai kumpulan informasi yang diorganisasikan dengan cara tertentu untuk bisa dipanggil kembali dan digunakan. Dengan menggunakan basis data, banyak keuntungan yang didapat seperti mengurangi pengulangan yang sama, menjaga integritas data, menjaga keamanan data, membuat data tetap bebas, serta menjaga konsistensi data. Menurut Mcleod, basis data adalah suatu koleksi data komputer yang terintegritasikan, diorganisasikan, dan disimpan dalam suatu cara yang memudahkan pengambilan kembali.

Tujuan dibuatnya basis data adalah meminimalkan pengulangan dan mencapai indepedensi data. Indepedensi data adalah kemampuan untuk membuat perubahan dalam struktur data tanpa membuat perubahan pada program yang memproses data. Indepedensi data dicapai dengan menempatkan spesifikasi dalam tabel dan kamus data yang terpisah secara fisik dari program.

Ada empat tipe model basis data yaitu:

1. Hierarki

   Hubungan data secara hierarki atau seperti struktur pohon yang merefleksikan hubungan 1 ke 1 atau 1 kebanyak pada tipe-tipe record.

2. *Network*

   Sama seperti hierarki, tetapi setiap record bisa memiliki lebih dari satu orang tua, selain itu terdapat relasi banyak kebanyak antara *record*.

3. *Relational*

   Model basis data yang banyak dipakai, dibuat dalam bentuk tabel sebagai tempat datanya dan relasi basis data dapat terjadi antara data pada tabel. Setiap tabel terdiri dari *record* dan *filed*.

4. Beroriantasi Objek

   Merupakan model relational dirancang untuk mengani data seperti angka-angka dan kata-kata, model ini bisa menangani objek seperti *video*, *image*, gambar dan lain-lain.

Secara garis besar dalam merancang basis data meliputi dua hal yaitu perancangan logika basis data dan perancangan fisik basis data. Perancangan logika merupakan pendiskrisian detail dari basis data dengan jalan mengetahui data apa saja yang digunakan untuk mengidentifikasikan, mengelompokan dan menghubungkan data. Perancangan fisik merupakan transformasi perancangan logika kedalam perancangan fisik, dari kamus data kedalam basis data yang sesungguhnya.

Sistem basis data memiliki empat kelompok, yaitu:

1. Data harus dapat dibagi (*shared*) dan juga terintegrasi.
2. Perangkat keras (*Hardware*) sebagai media penyimpanan.
3. Perangkat lunak (*Software*).
4. Pemakai (*User*).

Pada dasarnya basis data terdiri dari beberapa atau banyak tabel, tabel-tabel terdiri atas beberapa *field* dan *record-record*. *Record* adalah kumpulan *field-field* yang saling terkait dengan berisi elemen-elemen data itu. *Field-field* tersebut terkait satu sama lain karena mereka menjelaskan obyek yang spesifik. Dengan digunakannya basis data pada suatu perusahaan, maka akan diperoleh beberapa keuntungan yaitu:

1. Dapat mengurangi perulangan (*redundancy*), karena data yang sama pada beberapa aplikasi cukup disimpan sekali saja.
2. *Integrity*, data yang disimpan bersifat akurat.
3. Menghindari inkonsistensi, karena *redundancy* berkurang.
4. Standardisasi terhadap keseragaman penyajian data.
5. Penggunaan data bersama.
6. Adanya jaminan keamanan data (*Security*), karena data hanya dapat diakses oleh pihak yang berhak.

2.16 RMS Sebagai Basis Data Yang Digunakan[10]

Pada umumnya kita memerlukan data yang dapat kita isi dari aplikasi pada saat proses eksekusi. Pada sebuah aplikasi PC tradisional akan menyimpan data ke dalam file sistem yang terdapat dalam *hardisk* lokal. Apabila data memerlukan data yang lebih terstruktur, maka data tersebut dapat disimpan kedalam sebuah *database server.* Namun dalam kebanyakan ponsel, hal tersebut jelas tidak dapat dilakukan. Dalam ponsel

[10] Budi raharjo, imam heryanto, arif haryono, " *Tuntunan Pemrograman Java Untuk Handphone* ", penerbit informatika (bandung 2007), hal 115

hanya disediakan ruang penyimpanan lokal yang dapat dibaca dan ditulis (*read-write*) yang disebut dengan RMS (*Record Management System*).

Pada pemrograman *Java 2 Micro Edition* ini, perancangan aplikasi *database*nya menggunakan RMS dengan memakai kelas *recordstore,* yaitu satu – satunya kelas konkrit yang terdapat dalam paket ***javax . microedition . rms***. Sebuah *recordstore* akan berisi sekumpulan *record* yang diidentifikasi dengan suatu ID yang bersifat unik. ID tersebut akan berperan sebagai *primary key* dari *record – record* yang dimasukan.

Recordstore ini mempunyai fungsi untuk memanipulasi *database* yang ada meliputi, seperti membuat *record* baru, menambahkan ke dalam *recordstore*, membaca data dari *recordstore* dan menghapus *record* dari *recordstore*.

2.17 PEMROGRAMAN PERANGKAT BERGERAK

Merupakan bahasa pemrograman yang digunakan untuk membuat aplikasi yang bisa dijalankan pada perangkat bergerak. Bahasa pemrograman tersebut dapat menghasilkan aplikasi sesuai dengan spesifikasi perangat keras dari sebuah perangkat bergerak. Karakteristik dari pemrograman bergerak diantaranya:

1. Berjalan pada perangkat bergerak yang memiliki ukuran memori terbatas antara 160-512 Kbytes
2. Berjalan pada perangkat bergerak yang memiliki procesor 16 bit atau 32 bit

3. Berjalan pada perangkat bergerak yang sistem operasinya telah terintegrasi didalamnya.

Komponen yang terdapat pada pemrograman bergerak diantaranya adalah :

1. Koneksi tanpa kabel untuk jaringan atau internet
2. Tampilan pemakai (*user interface*)
3. Siklus hidup (*liife cycle*)

Adapun bahasa pemrograman yang termasuk dalam pemrograman perangkat bergerak adalah J*2ME, E- Visual C*++ yang merupakan turunan dari bahasa pemrograman *Visual C++, gcc Cygwin* merupakan turunan dari bahasa pemrograman *Basic.* Program aplikasi yang dihasilkan bahasa pemrograman tersebut hanya bisa berjalan pada sistem operasi tertentu, seperti prigram aplikasi yang dihasilkan oleh bahasa pemrograman *E-Visual C*++ dan *gcc Cygwin* hanya dapat berjalan pada sistem operasi *Windows CE,* kecuali bahasa pemrograman J2ME yang merupakan turunan dari bahasa pemrograman Java dengan sifatnya tulis sekali jalankan dimana saja (" *write once run every where* ").

2.18 Alat Bantu Perancangan Sistem Yang Digunakan

2.18.1 State Transition Diagram (STD)

State transition diagram adalah suatu *modelling tool* yang menggambarkan sifat ketergantungan terhadap suatu *realtime system* dan *human interface* pada *online system*. Kowal. J

(1988,p31)[11] mengatakan bahwa *state transition diagram* pada mulanya hanya digunakan untuk mengambarkan suatu sistem yang bersifat *real time*. *Realtime system* adalah suatu kondisi untuk mengoperasikannya secara bersama-sama dalam waktu bersamaan dengan waktu reaksi yang teratur atau sudah diprediksi dengan keadaan yang sebenarnya.

Cara kerja sistem pada umumnya terbagi dua, yaitu:

1. Aktif

   Sistem melakukan kontrol terhadap lingkungan secara aktif. Sistem mampu menerima sumber data eksternal dengan kecepatan tinggi dan dalam jangka waktu yang singkat atau *real time* memberikan respon terhadap lingkungan sesuai dengan program yang telah ditentukan.

2. Pasif

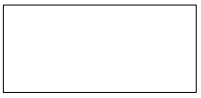
   Sistem tidak melakukan kontrol terhadap lingkungan tetapi lebih bersifat memberikan reaksi atau hanya menerima data saja.

   Notasi yang umumnya digunakan pada STD antara lain adalah :

   1. State

   State ialah kumpulan keadaan atau atribut yang mencirikan seseorang atau benda pada waktu tertentu, bentuk keadaan tertentu maupun kondisi tertentu. *State* disimbolkan dengan empat persegi panjang atau segi empat

11 Kowal, James A (1988), Analyzing System, Prentice Hall, New Jersey, p31



Gambar 2.6 notasi tampilan

2. Transition State

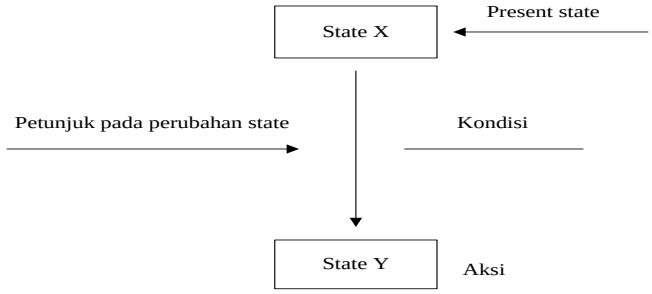
Transisi *state* atau perubahan *state* disimbolkan dengan anak panah

Gambar 2.7 notasi tindakan

Untuk melengkapi STD dibutuhkan kondisi yang dimaksud dengan kondisi adalah suatu kejadian pada lingkungan luar yang dapat dideteksi oleh sistem, sedangkan aksi adalah yang dilakukan oleh sistem apabila terjadi perubahan *state* atau merupakan reaksi terhadap kondisi. Aksi akan menghasilkan keluaran, tampilan pada layar, menghasilkan kalkulasi, dan lain-lain.

Terdapat 2 cara pendekatan untuk membuat *State Transition Diagram* yaitu:

1. Identifikasi setiap kemungkinan *state* dari sistem menggambarkan masing-masing *state* pada sebuah kotak, kemudian buatlah hubungan *state-state* tersebut.
2. Dimulai dengan state pertama dan kemudian dilanjutkan dengan *state-state* yang berikutnya sesuai dengan *flow* yang diinginkan.



Gambar 2.8 Komponen dasar STD12

2.18.2 Flowchart

Flowchart adalah bagan (*chart*) yang menunjukkan alir (*flow*) didalam program atau prosedur sistem secara logika. Diagram alir populer pada awal-awal era pemrograman dengan komputer. Diagram alir lebih menggambarkan aliran instruksi didalam program secara visual ketimbang memperlihatkan struktur program. Notasi algoritma ini juga cocok untuk masalah kecil, tidak cocok untuk masalah besar karena akan memerlukan berlembar halaman kertas untuk menggambarkan aliran proses program. Bagan alir (*Flow Chart*) adalah Bagan yang menunjukan didalam Program atau Prosedur sistem secara logika. Bagan Alir digunakan untuk Alat bantu Komunikasi dan untuk Dokumentasi [12].

Ada 5 Macam Bagan Alir Yaitu :

1. Bagan alir Program (*Program Flowchart*)
2. Bagan Alir Sistem (*System Flowchart*)
3. Bagan Alir Dokumen (*Document Flowchart*)

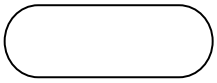
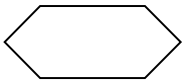
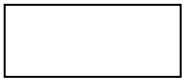
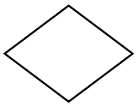
12 Kowal, James A (1988), Analyzing System, Prentice Hall, New Jersey, p329

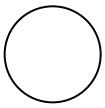
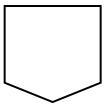
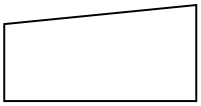
4. Bagan Alir Skematik (*Schematic Flowchart*)

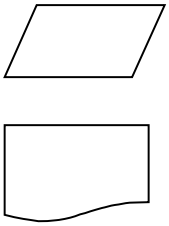
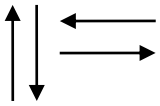
5. Bagan Alir Proses (*Process Flowchart*)

Simbol-simbol yang digunakan pada bagan alir program atau *flowchart*, disajikan pada tabel dibawah ini

Tabel 2.4 Simbol-simbol pada *flowchart*

No.	Gambar	Nama	Keterangan
1.		Simbol terminal	Menunjukkan awal dan akhir dari program.
2.		Simbol persiapan	Memberikan nilai awal pada suatu varabel.
3.		Simbol pengolahan	Digunakan untuk pengolah aritmatika dan pemindahan data.
4.		Simbol keputusan	Digunakan untuk mewakili operasi perbandingan logika.

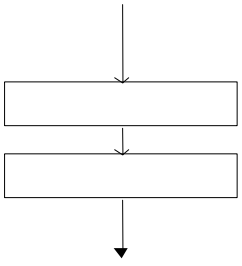
5.		Simbol proses terdefinisi	Digunakan untuk proses yang detilnya dijelaskan terpisah, misalnya dalam bentuk *subroutine*.
6.		Simbol Proses	Menunjukkan kegiatan proses dari operasi program Komputer.
7.		Simbol penghubung	Menunjukkan hubungan arus proses yang terputus masih di halaman yg sama.
8.		Simbol penghubung halaman lain	Menunjukkan hubungan arus proses yang terputus dengan sambungannya ada di halaman yang lain.
9.		Simbol *Keyboard*	Menggunakan input yang menggunakan on-line keyboard

10.		Simbol *input* / *output*	Menunjukkan Digunakan untuk mewakili data *input* / *output.*
11.		Simbol Garis Alir	Menunjukkan arus dari proses

2.19 Bentuk Dasar struktur logika

1. Struktur urut sederhana (*simple sequence structure*)

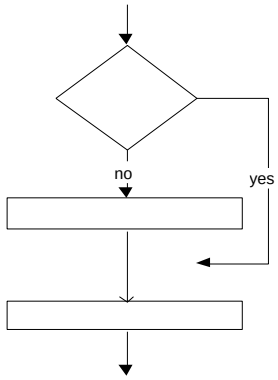
   Struktur ini semata-mata hanya berisi langkah-langkah yang urut saja, satu diikuti yang lainnya.



Gambar 2.9 *Simple Sequence*

2. Struktur loncat (*branch structure*)

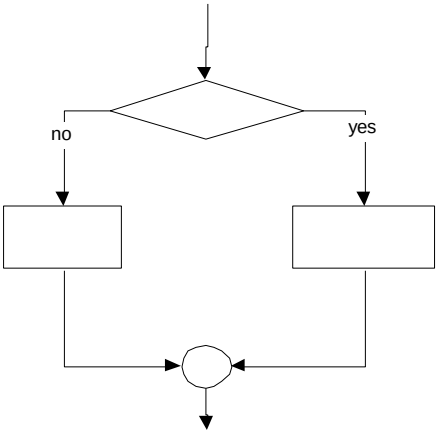
   Struktur ini suatu loncatan ke proses tertentu oleh statement *GOTO* atau *statement IF*.



Gambar 2.10 Branch Structure

3. Struktur seleksi (*selection structure*)

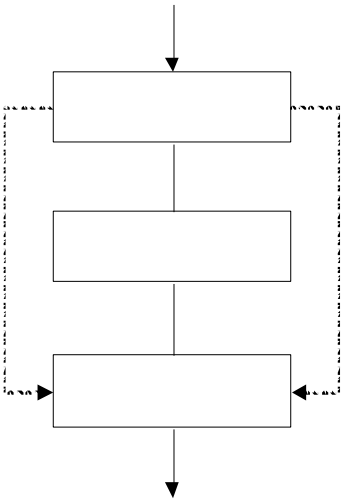
   Struktur ini merupakan penyeleksian kondisi yang menggunakan statement *IF-THEN-ELSE.*



Gambar 2.11 *Selection Structure*

4. Struktur perulangan *FOR* (*FOR loop structure*)

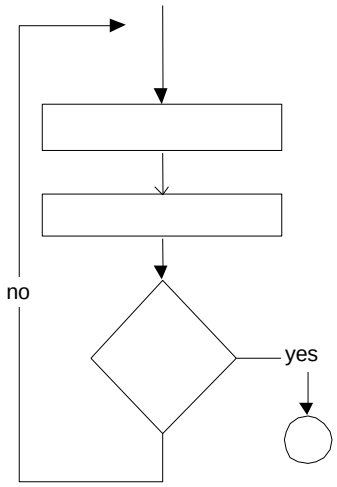
   Struktur ini merupakan prulangan beberapa *blok statement* yang dibentuk dengan *statement FOR*.



Gambar 2.12 *For Loop Structure*

5. Struktur perulangan *DO WHILE* (*DO WHILE loop structure*)

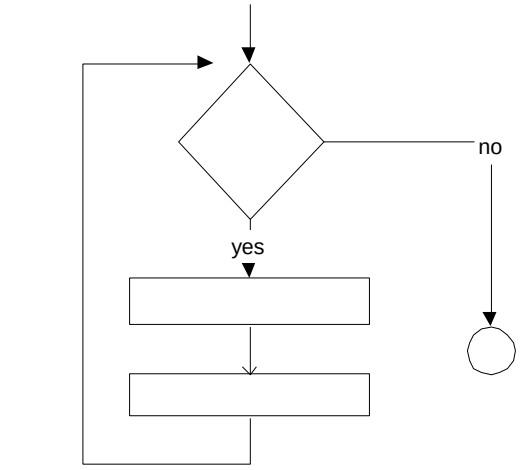
   Struktur ini menunjukkan suatu *blok statement* akan dikerjakan (*DO*) berulang-ulang selama (*WHILE*) kondisi yang diseleksi masih terpenuhi dan akan keluar dari lingkungan *Loop* bila kondisi sudah terpenuhi.



Gambar 2.13 *DO WHILE loop structure*

6. Struktur perulangan *DO UNTIL* (*DO UNTIL loop structure*)

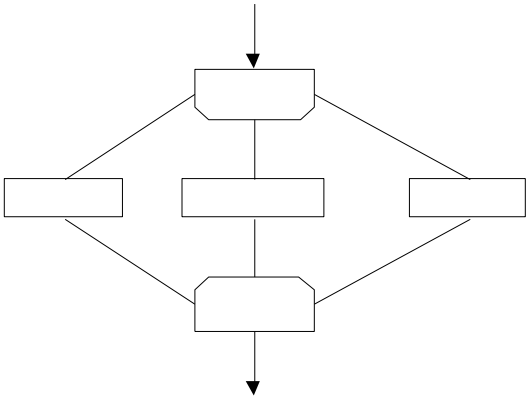
   Struktur ini menunjukkan suatu blok *statement* akan dikerjakan (*DO*) sampai (*UNTIL*) kondisi yang diseleksi tidak terpenuhi.



Gambar 2.14 *Do UNTIL (DO UNTIL loop structure)*

7. Struktur *Case* (*case structure*)

   Struktur ini akan memproses sebuah blok *statement* pada salah satu kondisi *case* yang terpenuhi dari sejumlah *case* yang ada.



Gambar 2.15 *CASE structure*