Pembelajaran 2. Rekayasa Perangkat Lunak

A. Kompetensi

Penjabaran model kompetensi yang selanjutnya dikembangkan pada kompetensi guru bidang studi yang lebih spesifik pada pembelajaran 2. Rekayasa Perangkat Lunak, ada beberapa kompetensi guru bidang studi yang akan dicapai pada pembelajaran ini, kompetensi yang akan dicapai pada pembelajaran ini adalah guru P3K mampu melakukan pemrograman komputer dengan salah satu bahasa pemrograman berorientasi objek.

B. Indikator Pencapaian Kompetensi

Dalam rangka mencapai komptensi guru bidang studi, maka dikembangkanlah indikator - indikator yang sesuai dengan tuntutan kompetensi guru bidang studi.
Indikator pencapaian komptensi yang akan dicapai dalam pembelajaran 2. Rekayasa Perangkat Lunak adalah menggunakan prinsip-prinsip Rekayasa Perangkat Lunak beserta aplikasi terkait dalam pembelajaran bidang studi Teknik Komputer dan Informatika.

C. Uraian Materi

1. Konsep objek oriented dan Analisis dan desain berorientasi objek

a. Metodologi Berorientasi Objek

Metodologi berorientasi objek merupakan suatu strategi pembangunan perangkat lunak yang mengorganisasikan perangkat lunak sebagai kumpulan objek yang berisi data dan operasi yang diberlakukan terhadapnya. Metodologi berorientasi objek adalah suatu cara bagaimana sistem perangkat lunak dibangun melalui pendekatan objek secara sistematis. Metode berorientasi objek didasarkan pada penerapan prinsip-prinsip pengelolaan kompleksitas.

Metode berorientasi objek meliputi rangkaian aktivitas analisis berorientasi objek, perancangan berorientasi objek, pemrograman berorientasi objek, dan pengujian berorientasi objek. Ada teknik yang digunakan, produk yang dihasilkan, prosedur verifikasi, dan kriteria untuk setiap aktivitas yang dikerjakan. Ada alat bantu untuk memodelkan (mendokumentasikan)hasil dari setiap aktivitas.

Karakteristik metode berorientasi objek adalah:

1) Cara kerja yang sistematis untuk mengerjakan tahap analisis berdasarkan pendekatan objek.
2) Ada kumpulan aturan-aturan tertentu yang harus diikuti untuk menyelesaikan pekerjaan analisis tersebut.
3) Mempunyai urut-urutan aktivitas, teknik, dan alat bantu (*tools*) tertentu untuk memodelkan (mendokumentasikan) hasil dari setiap aktivitas.

b. Tahapan pengembangan sistem berorientasi obyek

Siklus pemodelan atau langkah-langkah pemodelan dalam mengembangkan suatu sistem adalah:

1) Rekayasa pemodelan sistem menyangkut pengumpulan kebutuhan (*requirement gathering*) pada level sistem dengan sejumlah analisis serta top desain.
2) Analisis merupakan kebutuhan perangkat Lunak, proses *requirement gathering* difokuskan, khususnya pada Perangkat lunak. Untuk memahami sifat program yang dibangun, analis harus memahami domain informasi, tingkah laku, unjuk kerja, dan *interface* yang diperlukan. Kebutuhan sistem maupun Perangkat Lunak didokumentasikan dan direview bersama *user.*
3) Desain

   Memiliki fokus terhadap 4 hal, yaitu:

   - Desain database
   - Arsitektur perangkat lunak
   - Arsitektur *interface*
   - Algoritma prosedural.

c. Teknik Dasar OOA/D (Object-Oriented Analysis/Design)

Dalam dunia pemodelan, metodologi implementasi obyek walaupun terikat kaidah-kaidah standar, namun teknik pemilihan obyek tidak terlepas pada subyektifitas software analyst dan designer. Beberapa obyek akan diabaikan dan beberapa obyek menjadi perhatian untuk diimplementasikan di dalam sistem. Ada 3 (tiga) teknik/konsep dasar dalam OOA/D, yaitu pemodulan (*encapsulation*), pewarisan (*inheritance*) dan polimorfisme (*polymorphism*).

1) Pemodulan (*Encapsulation*)

   Pada dunia nyata, seorang ibu rumah tangga menanak nasi dengan menggunakan rice cooker, ibu tersebut menggunakannya hanya dengan menekan tombol. Tanpa harus tahu bagaimana proses itu sebenarnya terjadi. Disini terdapat penyembunyian informasi milik rice cooker, sehingga tidak perlu diketahui seorang ibu. Dengan demikian menanak nasi oleh si ibu menjadi sesuatu yang menjadi dasar bagi konsep information hiding.

2) Pewarisan (*Inheritance*)

   Obyek-obyek memiliki banyak persamaan, namun ada sedikit perbedaan. Contoh dengan beberapa buah mobil yang mempunyai kegunaan yang berbeda-beda. Ada mobil bak terbuka seperti truk, bak tertutup seperti sedan dan minibus. Walaupun demikian obyek-obyek ini memiliki kesamaan yaitu teridentifikasi sebagai obyek mobil, obyek ini dapat dikatakan sebagai obyek induk (parent). Sedangkan minibus dikatakan sebagai obyek anak (child), hal ini juga berarti semua operasi yang berlaku pada mobil berlaku juga pada minibus.

3) Polimorfisme (*Polymorphism)*

   Pada obyek mobil, walaupun minibus dan truk merupakan jenis obyek mobil yang sama, namun memiliki juga perbedaan. Misalnya suara truk lebih keras dari pada minibus, hal ini juga berlaku pada obyek anak (child) melakukan metode yang sama dengan algoritma berbeda dari obyek induknya. Hal ini yang disebut *polymorphism*, teknik atau konsep dasar

lainnya adalah ruang lingkup / pembatasan. Artinya setiap obyek mempunyai ruang lingkup kelas, atribut, dan metode yang dibatasi.

d. Tools dalam pengembangan sistem berorientasi obyek

Unified Modeling Language (UML) adalah bahasa pemodelan untuk sistem atau perangkat lunak yang berparadigma berorientasi objek. Unified Modeling Language (UML) disebut bahasa pemodelan bukan metode.

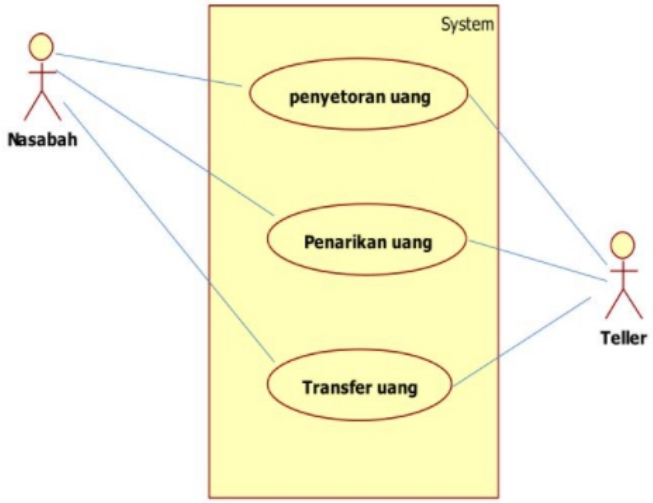
Untuk dapat memahami UML membutuhkan bentuk konsep dari sebuah bahasa model, dan mempelajari 3 (tiga) elemen utama dari UML seperti:

1) Benda / Things / Objek Objek merupakan bagian paling statik dari sebuah model, yang menjelaskan elemen–elemen lainnya dari sebuah konsep.
2) Hubungan / Relationship;
3) Bagan atau Diagram yaitu yang menggambarkan permasalahan maupun solusi dari permasalahan suatu model

Diagram pada UML

1) Use Case Diagram

   Use Case diagram merupakan suatu diagram yang menggambarkan fungsionalitas yang diharapkan dari sebuah sistem. Sebuah use case dapat memrepresentasikan interaksi antara aktor dengan sistem. Use Case Diagram adalah abstraksi dari interaksi antara sistem dan aktor. Use case bekerja dengan cara mendeskripsikan tipe interaksi antara user sebuah sistem dengan systemnya sendiri melalui sebuah cerita bagaimana sebuah sistem dipakai. Use case merupakan kontruksi untuk mendeskripsikan bagaimana system akan terlihat di mata user, sedangkn use case diagram memfalisitasi komunikasi di antara analis dan pengguna serta analis dan klien.



Gambar 15. Contoh use case diagram

Setiap use case diagram dilengkapi dengan skenario, skenario use case / use case skenario adalah alur jalannya proses use case dari sisi aktor dan system. Berikut adalah format tabel skenario use case.

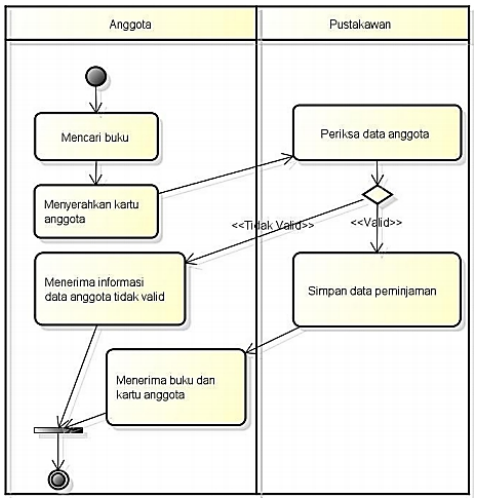
Tabel 10. Format tabel skenario use case

Nama Aktor	Reaksi Sistem
Skenario Normal	
Skenario Alternatif	

Skenario use case dibuat per use case terkecil, misalkan untuk generalisasi maka scenario yang dibuat adalah use case yang lebih khusus. Skenario normal adalah scenario bila system berjalan normal tanpa terjadi kesalahan atau error. Sedangkan skenario alternatif adalah scenario bila system tidak berjalan normal atau mengalami error. Skenario normal dan skenario alternatif dapat berjumlah lebih dari satu. Alur skenario inilah yang nantinya menjadi landasan pembuatan sequence diagram / diagram sekuen.

2) Activity Diagram

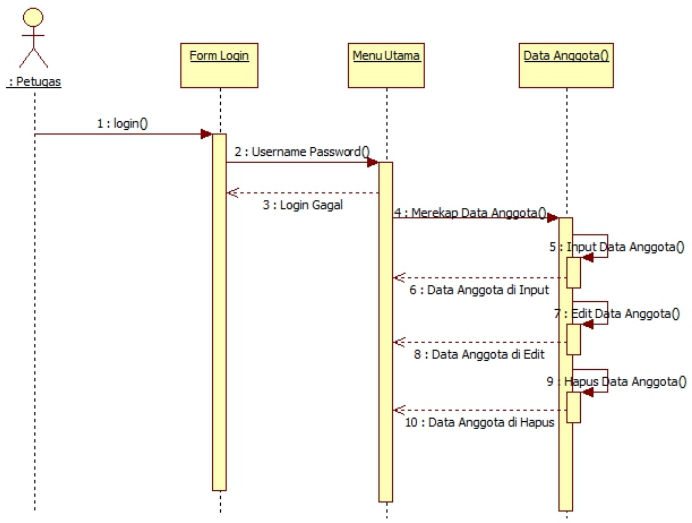
Activity diagram menggambarkan berbagai alur aktivitas dalam sistem yang sedang dirancang, bagaimana masing-masing alur berawal, decision yang mungkin terjadi, dan bagaimana mereka berakhir. Activity diagram juga dapat menggambarkan proses paralel yang mungkin terjadi pada beberapa eksekusi.



Gambar 16. Contoh activity diagram Sistem Informasi Perpustakaan

3) Sequence Diagram

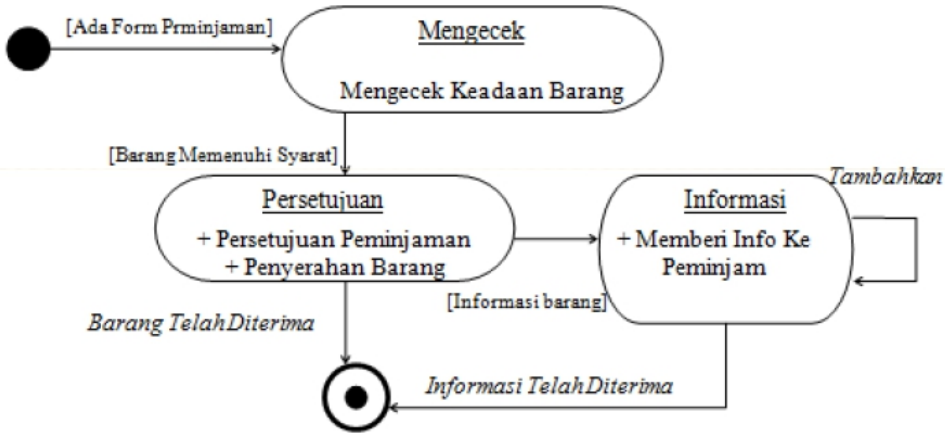
Sequence diagram menggambarkan interaksi antar objek di dalam dan di sekitar sistem (termasuk pengguna, display, dan sebagainya) berupa message yang digambarkan terhadap waktu. Sequence diagram terdiri atar dimensi vertikal (waktu) dan dimensi horizontal (objek-objek yang terkait). Sequence diagram dapat digunakan untuk menggambarkan skenario atau rangkaian langkah-langkah yang dilakukan sebagai respons dari sebuah event untuk menghasilkan output tertentu.



Gambar 17. Contoh squence diagram

4) Statechart diagram

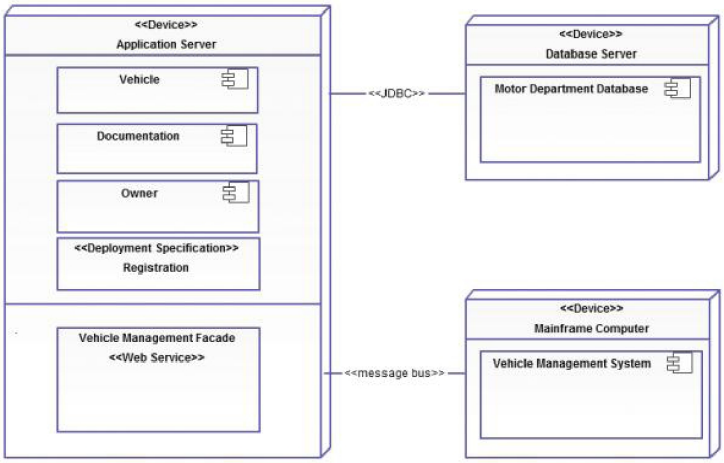
Statechart diagram menunjukkan siklus hidup dari obyek tunggal, dari saat dibuat sampai obyek tersebut dihapus.



Gambar 18. Contoh *statechart diagram*

5) Deployment diagram

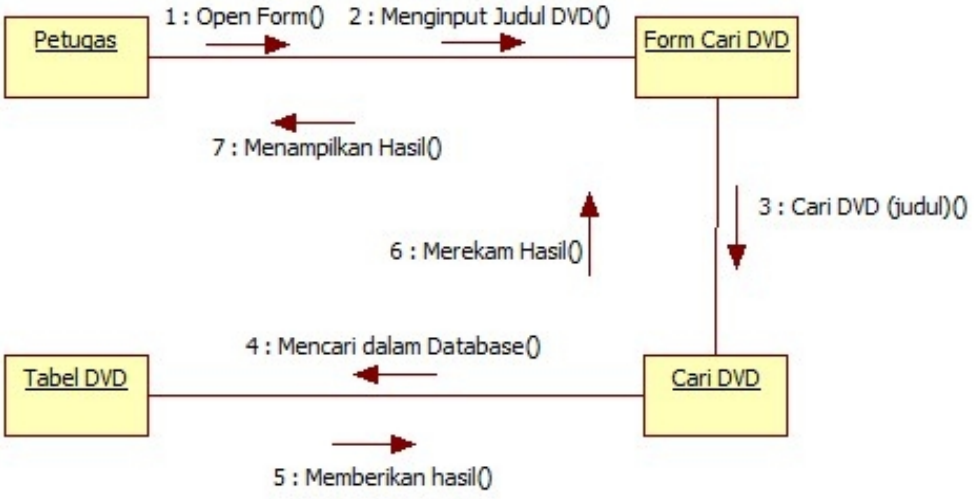
Deployment diagram merupakan gambaran proses-proses berbeda pada suatu sistem yang berjalan dan bagaimana relasi di dalamnya.



Gambar 19. Contoh *deployment diagram*

6) Collaboration Diagram

Kolaborasi diagram atau collaboration diagram adalah suatu diagram yang memperlihatkan/menampilkan pengorganisasian interaksi yang terdapat disekitar objek (seperti halnya sequence diagram) dan hubungannya terhadap yang lainnya.

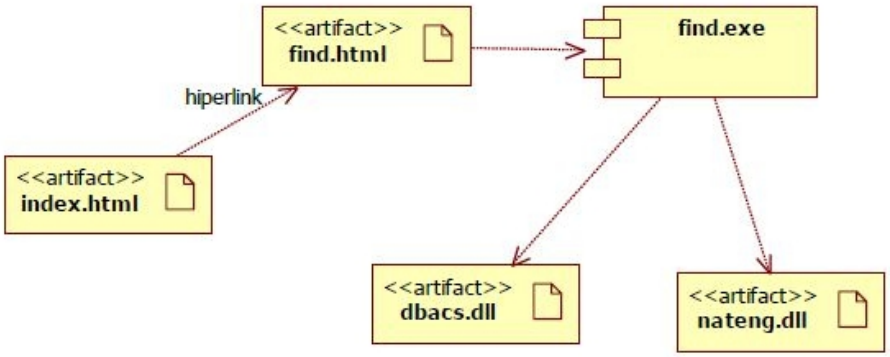


Gambar 20. Contoh collaboration diagram

7) Componen Diagram

Komponen adalah bagian fisik atau replaceable dari sistem yang bersesuaian dan menyediakan realisasi dari sekumpulan interface.

Component diagram menunjukkan organisasi dan ketergantungan antar komponen



Gambar 21. Contoh component diagram

2. Manajemen Sistem Basis Data (Database Management System/DBMS)

a. Konsep Basis Data dalam RDBMS

Sistem manajemen basis data atau database management system (DBMS), atau kadang disingkat SMBD, adalah suatu sistem atau perangkat lunak yang dirancang untuk mengelola suatu basis data dan menjalankan operasi terhadap data yang diminta banyak pengguna.

DBMS merupakan perangkat lunak yang dirancang untuk dapat melakukan utilisasi dan mengelola koleksi data dalam jumlah yang besar. DBMS juga dirancang untuk dapat melakukan manipulasi data secara lebih mudah. Sebelum adanya DBMS, data pada umumnya disimpan dalam bentuk flat file, yaitu file teks yang ada pada sistem operasi.

Dibandingkan dengan sistem data yang berbasis kertas, DBMS memiliki beberapa keunggulan:

1) Mengurangi redundancy, data yang sama pada beberapa aplikasi cukup disimpan sekali saja.
2) Integrity, data tersimpan secara akurat.

3) Menghindari inkonsisten, karena redundancy berkurang, maka update data jadi lebih efisien.
4) Penggunaan data bersama, data yang sama dapat diakses oleh beberapa user pada saat bersamaan.
5) Menyangkut keseragaman penyajian data.
6) Menyeimbangkan kebutuhan, dapat ditentukan prioritas suatu operasi, misal antara update dengan retrieval.

Dalam implementasinya terdapat empat komponen utama DBMS, yaitu: perangkat keras (hardware), data, perangkat lunak (software) dan pengguna.

b. Abstraksi Data

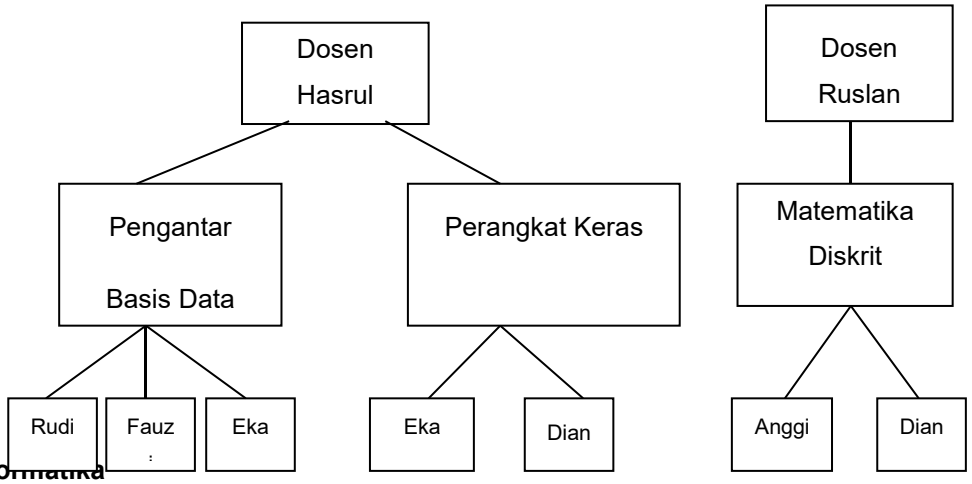
Untuk mendukung kepraktisan, DBMS menyediakan pandangan abstrak terhadap data bagi pengguna. DBMS berusaha menyembunyikan detail tentang bagaimana data disimpan dan dipelihara. Abstraksi data dalam DBMS biasanya dibagi menjadi 3 lapis, yaitu Lapis fisis, Lapis konseptual dan Lapis pandangan.

c. Model Basis Data

Model basis data menyatakan hubungan antar rekaman yang tersimpan dalam basis data. Beberapa literatur menggunakan istilah struktur data logis untuk menyatakan keadaan ini. Model dasar yang paling umum ada 3 macam, yaitu model hierarkis, jaringan, dan relasional.

1) Model Hierarkis

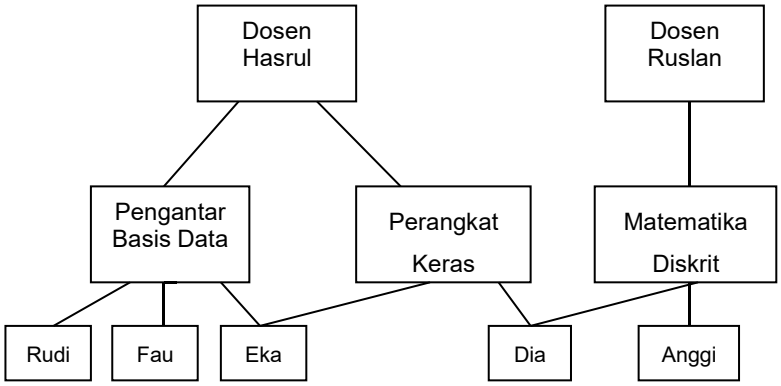
   Model Hirarkis biasa disebut model pohon, karena menyerupai pohon yang dibalik. Model ini menggunakan pola hubungan orang tua-anak. Setiap simpul (biasa dinyatakan dengan lingkaran atau kotak) menyatakan sekumpulan medan.



Gambar 22. Contoh model hirarkis

2) Model Jaringan

Model ini menyerupai model hirarkis, dengan perbedaan suatu simpul anak bisa memiliki lebih dari satu orang tua.



Gambar 23. Model Jaringan

3) Model Relasional

Model relasional merupakan model yang paling sederhana sehingga mudah digunakan dan dipahami oleh pengguna, serta merupakan yang paling populer saat ini. Model ini menggunakan sekumpulan tabel berdimensi dua (yang disebut relasi atau tabel), dengan masing-masing relasi tersusun atas tupel atau baris dan atribut. Relasi dirancang sedemikian rupa sehingga dapat menghilangkan kemubaziran data dan menggunakan kunci tamu untuk berhubungan dengan relasi lain.

Pada model relasional, seluruh data terstruktur secara logika di dalam sebuah relasi (tabel). Setiap relasi mempunyai nama dan terdiri dari atribut-atribut bernama (kolom). Setiap tupel (baris) berisikan satu nilai per atribut. Kekuatan yang besar dari model data relasional adalah struktur logikal yang sederhana.

Tabel 11. Padanan istilah relasi, tupel, dan atribut.

Model Relasional	Pemrogram	Pengguna
Relasi	Berkas	Tabel
Tupel (baris)	Rekaman	Baris
Atribut	Medan	Kolom

Istilah yang digunakan dalam basis data relasional

a) Relasi

   Relasi merupakan sebuah tabel yang terdiri dari beberapa kolom dan beberapa baris.

b) Entitas

   Entitas *(entity)* adalah sebuah objek yang keberadaannya dapat dibedakan terhadap objek lain. Entitas dapat berupa orang, benda, tempat, kejadian, konsep

c) Atribut

   Atribut adalah sifat atau karakteristik yang melekat dalam sebuah entitas. Setiap atribut akan memiliki nilai *(values).* Domain *(value Set)*

d) Tupel atau baris

   Tupel merupakan baris pada sebuah relasi atau kumpulan elemen-elemen yang saling berkaitan menginformasikan tentang suatu entitas secara lengkap. Satu *record* mewakili satu data atau informasi tentang seseorang, misalnya : NIM, nama mahasiswa, alamat, kota, dan lain-lain.

e) Domain

   Domain adalah kumpulan nilai yang valid untuk satu atau lebih atribut.

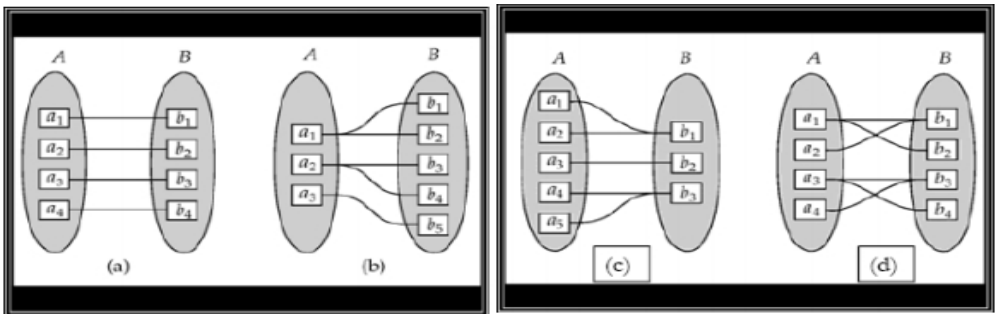
f) Derajat *(degree)*

   Degree menunjukan banyaknya himpunan entitas yang saling berelasi.

g) Kardinalitas relasi

   Kardinalitas relasi menggambarkan banyaknya jumlah maksimum entitas dapat berelasi dengan entitas pada himpunan entitas yang lain.

Jenis relasi antara dua entitas dapat berupa relasi: *One to One (1:1), One to Many (1:M), Many to One (M:1)* dan *Many to many (M:M)*



Gambar 24. Kardinalitas relasi

Tabel berikut merupakan bentuk relasional berdasarkan contoh model hirarkis dan jaringan sebelumnya.

Tabel 12. Contoh konversi model jaringan ke model relasional

Nama Dosen	Mata Kuliah	Mahasiswa
Hasrul	Pengantar Basis Data	Rudi
Hasrul	Pengantar Basis Data	Fauji
Hasrul	Pengantar Basis Data	Icca
Hasrul	Perangkat Keras	Icca
Hasrul	Perangkat Keras	Phoo
Al Imran	Matematika Diskrit	Anggi
AlImran	Matematika Diskrit	Phoo

Pada prakteknya, relasi pada gambar cardinal relasi akan dinormalisasikan sehingga akan terbentuk beberapa tabel yang saling terhubung. Contoh model relasional seperti ini dapat dilihat pada tabel berikut ini,

NO_MHS	NAMA_MHS
55	Ahmadi
56	Rina
57	Budi

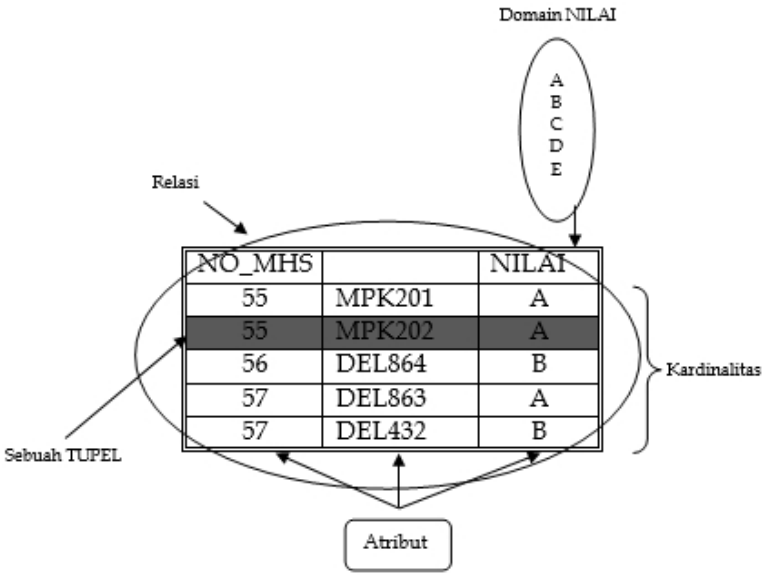
KODE_MK	NAMA_MK
DB001	Pengantar Basis Data
DB002	Basis Data Lanjut
P1001	Teknik Multimedia

NO_MHS	KODE_MK	NILAI
55	MPK201	A
55	MPK202	A
56	DEL864	B
57	DEL863	A
57	DEL432	B

Gambar 25. Contoh beberapa relasi pada model relasional

Pada gambar ini terdapat tiga buah relasi. Relasi yang terbawah menggunakan kunci tamu berupa nomor mahasiswa (NO_MHS)

dan kode matakuliah (KODE_MK) untuk menghubungkan diri ke kedua relasi di atasnya. Dengan kata lain, berdasarkan data pada terbawah, informasi seperti nama mahasiswa (NAMA_MHS) dan nama mata kuliah (NAMA_MK) bisa diperoleh.



Gambar 26. Relasi, tupel, atribut, dan berbagai istilah lainnya

Ada beberapa sifat yang melekat pada suatu relasi, yaitu:

- Tidak ada tupel (baris) yang kembar
- Urutan tupel tidaklah penting (tupel-tupel yang dipandang dalam sembarang urutan)
- Setiap atribut memiliki nama yang unik
- Letak atribut bebas (urutan atribut tidak penting)
- Setiap atribut memiliki nilai tunggal dan jenisnya sama untuk semua tupel

Pada model relasional, jumlah tupel suatu relasi disebut kardinalitas dan jumlah atribut suatu relasi disebut derajat (*degree*) atau terkadang disebut *arity*. Relasi berderajat satu (hanya memiliki satu atribut) disebut *unary*. Relasi yang berderajat dua disebut *binary* dan relasi yang berderajat tiga disebut *ternary*. Relasi yang berderajat n disebut *n-ary*. Istilah lainnya yang

terdapat pada model relaisional adalah domain. Domain adalah himpunan nilai yang berlaku bagi suatu atribut.

d. Key (Atribut Kunci)

Penggunaan key merupakan cara untuk membedakan suatu entitas dalam himpunan entitas dengan entitas lain. Secara konsep, masing-masing entitas memiliki nilai yang berbeda, perbedaannya terlihat pada isi masing-masing atributnya. *Key* adalah satu atau gabungan dari beberapa atribut yang dapat membedakan semua baris dalam relasi secara unik. Dalam RDBMS, key terbagi menjadi beberapa jenis yaitu: *primary key, foreign key, candidate key, super key, alternate key* dan *composite key.*

1) Primary key (kunci primer), merupakan sebuah aturan dimana fungsinya adalah untuk membedakan anatara baris satu dengan baris lainnya yang ada pada tabel dan bersifat unik. Ada ketentuan yang harus diperhatikan ketika field yang menjadi primary key yakni, yaitu data tidak boleh sama atau ganda (unik) dan data tidak boleh bernilai null
2) Foreign key (kunci tamu), merupakan suatu atribut untuk melengkapi hubungan yang menunjukan ke induknya, itu artinya field pada tabel merupakan kunci tamu dari tabel lain. Dan biasanya penggunaan foreign key akan sangat dibutuhkan ketika menemukan banyak tabel dan ingin menghubungkan satu tabel dengan tabel lainnya.
3) Super key (kunci super), merupakan satu atau lebih atribut (kumpulan atribut) yang dapat membedakan setiap baris data dalam table secara unik.
4) Candidate key (kunci kandidat), merupakan suatu atribut ataupun super key yang mengidentifikasi secara unik untuk kejadian spesifik dari entitas.
5) Composite key (kunci gabungan), merupakan kunci yang terdiri dari 2 atau lebih atribut yang secara unik mengidentifikasi suatu kejadian entitas.
6) Alternative key (kunci alternatif),merupakan candidate key yang tidak dipilih sebagai primary key.

7) Sekunder key (kunci sekunder) adalah sebuah atribut atau kombinasi yang digunakan hanya untuk tujuan pengambilan data.

3. Konsep dan implementasi pemrograman berorientasi objek dalam pengembangan apikasi atau sistem informasi

a. Konsep OOP dalam pengembangan aplikasi/sistem informasi menggunakan Program Java

Bahasa java dibuat oleh James Gosling saat masih bergabung di Sun Microsystems dan dirilis tahun 1995. Bahasa Java dapat dijalankan pada berbagai komputer dan platform sistem operasi. Slogan Java: Write once, run anywhere! (Tulis sekali, jalankan di manapun). Java adalah bahasa pemrograman bersifat umum (general purpose). Sintaks Bahasa Java diadopsi dari Bahasa C dan C++ tetapi lebih sederhana. Nama “java” diambil dari jenis kopi yang diminum oleh James Gosling saat itu. Bahasa java memiliki karakteristik: sederhana, berorientasi objek, interpreted, terdistribusi, tangguh, portable, robus (memiliki kinerja tinggi), aman, ArchitectureNeutral, Portable, Multithreaded dan dinamis.

Compiler java mengubah kode program menjadi bahasa intermediate yang mengkompilasi kode program Java dirancang untuk menghasilkan kode yang netral terhadap semua arsitektur perangkat keras disebut java bytecode. Kemudian interpreter Java bernama JVM (JavaVirtual Machine) melakukan interpretasi bytecode setiap kali bytecode tersebut dijalankan.

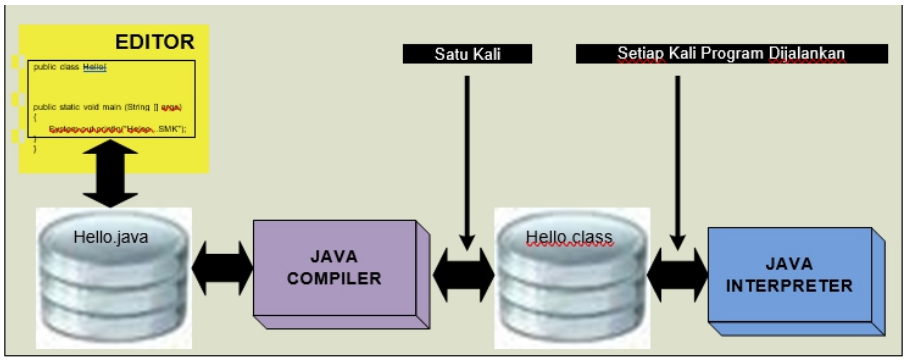
Platform java terrdiri atas dua bagian utama, yaitu Java Virtual Machine (JVM) dan Java Application Programming Interface (JavaAPI).

Sun membagi arsitektur Java membagi tiga bagian, yaitu:

- Enterprise Java (J2EE) untuk aplikasi berbasis web, aplikasi sistem tersebar dengan beraneka ragam klien dengan kompleksitas yang tinggi. Merupakan superset dari Standar Java
- Standar Java (J2SE), ini adalah yang biasa dikenal sebagai bahasa Java.
- Micro Java (J2ME) merupakan subset dari J2SE dan salah satu aplikasinya yang banyak dipakai adalah untuk wireless device / mobile device.

Fase-fase pemrograman java

Gambar di bawah ini menjelaskan aliran proses kompilasi dan eksekusi sebuah program Java.



Gambar 27. Fase dari sebuah program Java

Langkah pertama dalam pembuatan sebuah program berbasis Java adalah menuliskan kode program pada text editor. Contoh text editor yang dapat digunakan antara lain: notepad, vi, emacs dan lain sebagainya. Kode program yang dibuat kemudian tersimpan dalam sebuah berkas berekstensi .java. Setelah membuat dan menyimpan kode program, kompilasi file yang berisi kode program tersebut dengan menggunakan Java Compiler. Hasil dari kompilasi berupa berkas byte code dengan ekstensi *.class. Berkas yang mengandung byte code tersebut kemudian akan dikonversikan oleh Java Interpreter menjadi bahasa mesin sesuai dengan jenis dan platform yang digunakan.

Platform java yang sering digunakan adalah Java SDK dan java editor NetBeans. JavaSDK adalah platform dasar Java yang diperlukan agar komputer atau laptop dapat digunakan untuk mengeksekusi kode-kode programbahasa Java, sedangkan NetBeans adalah aplikasi editor terpadu (IDE atau Integrated Develepment Environment) yang akan banyak mempermudah dalam membuat aplikasi karena menyediakan kontrol-kontrol visual yang penting dalam pemrograman desktop (atau lebih dikenal sebagai p emrograman visual). IDE NetBeans mengharuskan membuat new Project terlebih dahulu sebelum menulis script program java. Dengan cara klik File new Project , langkah berikutnya memilih aplikasi Java Aplication. File dengan extension .java dibuat untuk memulai menulis program java.

b. Java Keywords

Di bawah ini ditampilkan semua kata-kunci (Java keywords) :

Tabel 13. Java Keywords

Abstract	Default	If	private	this
Boolean	Do	implements	protected	throw
Break	Double	import	public	throws
Byte	Else	instanceof	return	transient
Case	Extends	int	short	try
Catch	Final	interface	static	void
Char	Finaly	long	strictfp	volatile

c. Tipe Data Dalam Java

Sebagaimana bahasa pemrograman yang lain, di dalam Java juga dikenal istilah tipe data. Tipe data ini digunakan untuk pengalokasian memory guna menyimpan nilai/valuenya. Di dalam Java, ada beberapa tipe data sebagai berikut:

Tabel 14. Tipe Data dalam Java

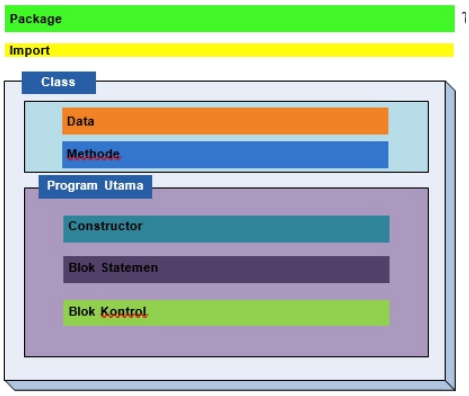
Tipe Data	**Range nilai**	**Keterangan**
Byte	-128 ... 127	Bilangan bulat
Short	-32768 ... 32767	Bilangan bulat
Int	- 2147483648 ... 2147483647	Bilangan bulat
Long	-9223372036854775808 ... 9223372036854775807	Bilangan bulat
Float		Bilangan riil
Double		Bilangan riil
Char		Karakter
String		String (beberapa karakter)
Boolean	true/false	-

d. Variabel

Variabel adalah item yang digunakan data untuk menyimpan pernyataan objek. Variabel memiliki tipe data dan nama. Tipe data menandakan tipe nilai yang dapat dibentuk oleh variabel itu sendiri. Nama variabel harus mengikuti aturan untuk identifier.

e. Bagan dasar program Java

Berikut ini gambar bagan program java.



Gambar 28. Bagan program java

- Package

  Perintah java yang digunakan untuk memberitahukan bahwa suatu class adalah anggota dari package, sedangkan nama Package dapat berupa susunan direktori tempat dimana file class disimpan atau nama folder. Package adalah sebuah sarana untuk mengelompokkan atau mengorganisasikan kelas dan interface yang sama atau sekelompok menjadi satu unit tunggal dalam library. Alasan menggunakan package pada java ialah untuk menghindari tabrakan nama kelas yang akan dibuat dengan nama kelas yang sudah ada.masing-masing kelas tersebut dalam package tersebut dikompilasi menjadi byte code (*.class). Path hirarki package, didaftarkan sebagai salah satu nilai variabel lingkungan yang bernama Classpath. Classpath diset dengan aturan.

- Import

  Perintah import digunakan untuk memberitahukan kepada program untuk mengacu pada class-class yang terdapat pada package tersebut dan bukan menjalankan class-class tersebut**.** Dalam program, dapat diimport class-class tertentu saja dan dapat pula mengimport semua class yang terdapat pada package.

- Class

  Merupakan bentuk logis yang menjadi landasan bangun seluruh bahasa pemograman berorientasi object. Class mendefinisikan bentuk dan perilaku object.

  Class merupakan contoh abstrak dari sebuah object yang telah terbentuk dari

proses penyederhanaan. Dengan kata lain class merupakan cikal bakal dari object. Kemudian contoh nyata atau perwujudan dari sebuah object dinamakan instance.

- Data dan Metode
  Data merupakan identitas yang berupa variabel yang menjelaskan properti dari class. Metode adalah sekumpulan instruksi untuk menjalankan data yang diberi nama dan dapat dipanggil dari manapun di dalam program dengan menuliskan nama metoda tersebut.
- Program utama
  Salah satu metode yang paling penting di dalam bahasa Java adalah metoda main. Metode main harus dideklarasikan sendiri oleh programmer di dalam sebuah kelas. Kelas yang mempunyai metoda main disebut dengan kelas main (main class), akan tetapi tidak semua kelas Java harus mempunyai metoda main. Interpreter Java akan meminta metoda main saat program aplikasi dieksekusi.

Pemrograman Java menggunakan konsep Pemrograman Berorientasi Obyek (PBO) atau *Object Oriented Programming* (OOP). Semua program Java merupakan suatu obyek. Dasar-dasar OOP meliputi istilah yaitu*: class, object, attribute* dan *method*.

Secara umum, OOP adalah teknik yang memfokuskan desain program pada obyek dan class berdasarkan pada skenario di dunia nyata. Sebagai contoh, misalkan mobil. Sebuah mobil secara umum tentunya memiliki beberapa karakteristik, yaitu misalnya memiliki sejumlah roda, memiliki warna, memiliki beberapa pintu dsb. Selanjutnya mobil ini bisa terdapat berbagai macam merek, misalnya mobil Suzuki Ertiga, Toyota Avanza dsb. Sebuah mobil tentunya juga bisa dijalankan, baik maju maupun mundur atau dihentikan. Dalam OOP, mobil tersebut identik dengan Class, mobil Suzuki Ertiga, Avanza dll itu merupakan obyek. Jumlah roda, warna mobil, jumlah tempat duduk dll identik dengan atribut dari suatu obyek, serta proses untuk mengendalikan mobil (maju, mundur dan berhenti) itu dalam OOP identik dengan metode dari suatu obyek. Untuk lebih memahami perbedaan class, objek, atribut dan metode dapat dilihat pada tabel berikut.

Tabel 15. Contoh class, objek, atribut dan metode

Class	Objek	Atribut	metode
Mobil	Suzuki, Ertiga, Avanza	Jumlah roda, warna mobil, jumlah tempat duduk	maju, mundur dan berhenti

1) Objek

Objek adalah abstraksi dan sesuatu yang mewakili dunia nyata seperti benda, manusia, satuan organisasi, tempat, kejadian, struktur, status, atau hal-hal lain yang bersifat abstrak. Objek merupakan suatu entitas yang mampu menyimpan informasi (status) dan mempunvai operasi (kelakuan) yang dapat diterapkan atau dapat berpengaruh pada status objeknya. Objek mempunyai siklus hidup yaitu diciptakan, dimanipulasi, dan dihancurkan.

2) Atribut

Atribut adalah elemen data dari suatu class. Atribut menyimpan informasi tentang class. Atribut dapat diartikan sebagai data, variabel, properti atau sebuah field. Atribut dapat berupa nilai atau elemen-elemen data yang dimiliki oleh Objek dalam kelas objek. Atribut dipunyai secara individual Oleh sebuah objek, misalnya berat, jenis, nama, dan sebagainya

3) Class

Class adalah model dari suatu obyek yang menjelaskan karakteristik (sifat) serta fungsi yang dimiliki dari suatu obyek. Class merupakan wadah (tempat) yang digunakan untuk menciptakan suatu obyek. Dengan kata lain sebuah Class merupakan *blueprint* dari suatu obyek.

Class adalah struktur dasar dari OOP, class terdiri dari dua tipe anggota dimana disebut dengan field dan method. Field merupakan tipe data yang didefiniskan, sementara method merupakan operasi. Untuk membuat class, sebelum menulis nama pertimbangkan dulu nama class dan dimana class tersebut digunakan. Dalam pendeklarasian atribut untuk menggunakan tipe data integer untuk nama siswa, atau tipe data string

pada nilai siswa. Jika anda menginginkan bahwa atribut-atribut tersebut unik, maka dideklarasikan sebagai instance variable. Class variable atau static variable, variabel ini sama pada semua object di class yang sama. Anda dapat mendeklarasikan satu static variable yang akan menampung nilai tersebut.

Berikut ini adalah aturan pembuatan class dalam Java:

```
public class namaclass
{
        .
        .
}
```

Aturan pemberian nama class:

- Dimulai dengan huruf, atau tanda _ atau tanda $
- Tidak boleh menggunakan reserved word dalam Java
- Tidak boleh memuat operator aritmatika
- Bersifat case sensitif

4) Method

Method (metode) adalah sebuah function atau fungsi yang ada dalam suatu class. Setiap metode memiliki tugas sendiri. Operasi atau metode atau method pada sebuah kelas hampir sama dengan fungsi atau prosedur pada metodologi struktural. Sebuah kelas boleh memiliki lebih dari satu metode atau operasi. Metode atau operasi yang berfungsi untuk memanipulasi objek itu sendiri.

Di dalam Java ada 2 jenis method yaitu void dan non void method. Method void adalah method yang tidak mengembalikan nilai, sedang non void method adalah method yang mengembalikan suatu nilai. Jika diperhatikan, ketika membuat project baru misalnya ‘project1’, maka akan di dalam class ‘project1’ ini akan dibuat pula method dengan nama main()

```
public class Project1 {
      public static void main(String[] args) {
            .
            .
      }
}
```

Method main() dalam suatu class menunjukkan method tersebut adalah method utama yang akan dijalankan pertama kali ketika program Java dijalankan. Khusus method main(), perlu diberikan 'static' setelah modifiernya. Pada suatu class, bisa dibuat method berapapun sesuai keigninan.

Perlu diingat juga bahwa di dalam Java, beberapa class itu bisa digabung atau disimpan menjadi satu dalam sebuah paket atau package jika diperlukan. Hal ini dimaksudkan untuk memudahkan pengelolaan class saja.

Contoh

Sebagai contoh dari penerapan konsep OOP dalam pemrograman Java, misalkan akan dibuat sebuah program untuk menjumlahkan dua buah bilangan. Untuk langkah awal, desain terlebih dahulu bentuk class untuk penjumlahan bilangan tersebut. Misalkan dibuat class dengan nama 'operasiBilangan'. Di dalam class tersebut, misalkan dibuat atribut yaitu 'bilangan1' dan 'bilangan2', merupakan kedua bilangan yang akan dioperasikan, serta 'hasil' yang merupakan hasil dari operasi kedua bilangan. Selanjutnya di dalam class 'operasiBilangan' tersebut dibuat sebuah method 'jumlah' untuk menjumlahkan kedua bilangan, serta method untuk menampilkan hasil operasi bilangan.

1) buat Class dengan nama 'operasiBilangan'

kode dalam class operasiBilangan

```
/*
 * To change this template, choose Tools | Templates
 * and open the template in the editor.
 */
package contoh2;

/**
 *
 * @author acer
 */
public class operasiBilangan {

}
```

2) kode berikut ini di dalam class operasiBilangan

```
public class operasiBilangan {

    // deklarasi atribut atau
    properties public int
    bilangan1;

    public
    int
    bilangan2;
    private
    int hasil;


    // method
    jumlah()
    public
    void
    jumlah()

    {

        this.hasil = this.bilangan1 + this.bilangan2;

    }

    // method
    tampilHasil
    () public
    void
    tampilHasil
    ()

    {

        System.out.println("Hasil operasi bilangan : " +
        this.hasil);

    }

}
```

Keterangan:

- Bilangan1, bilangan2 dan hasil merupakan atribut atau properties dari class operasibilangan, sedangkan jumlah() dan tampilHasil() adalah methodnya.
- Perhatikan, di depan atribut atau method ada ‘public’ atau ‘private’. Jika diberikan ‘public’ maka atribut atau method tersebut bisa diakses dari class manapun (jika terdapat lebih dari satu class). Namun jika ‘private’, maka atribut atau method hanya bisa diakses di dalam class itu saja. Selain ‘public’ dan ‘private’ sebuah atribut atau method bisa juga diset dengan sifat ‘protected’ yang artinya hanya bisa diakses dalam class itu saja

ata class lain yang masih dalam satu package yang sama. Keterangan public', 'private' dan 'protected' dalam OOP disebut modifier yang digunakan untuk menentukan aksesibilitas method atau atribut.

- Perintah 'this.' digunakan untuk mengakses atribut atau method yang ada dalam class tersebut.

3) Kemudian, di class 'Contoh2' nya (di file 'Contoh2.java'), tulis kode program sebagai berikut

```
public class Contoh2 {
    public static void main(String[] args) {
        operasiBilangan op1 = new operasiBilangan();
        op1.bilangan1 = 10;
        op1.bilangan2 = 20;
        op1.jumlah();
        op1.tampilHasil();
    }
}
```

Keterangan:

Perintah:

```
operasiBilangan op1 = new operasiBilangan();
```

digunakan untuk instantisasi, yaitu proses membuat obyek baru dengan nama 'op1'. Obyek ini termasuk dalam class 'operasiBilangan'.

Perintah:

```
op1.bilangan1 = 10;
```

adalah mengeset atribut 'bilangan1' pada obyek 'op1' dengan suatu nilai.

Demikian juga dengan perintah

```
op1.bilangan2 = 20;
```

Perintah

```
op1.jumlah();
```

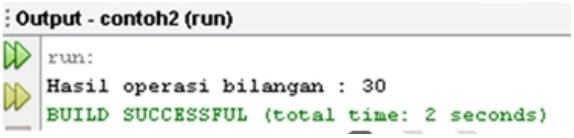
dimaksudkan untuk menjalankan method jumlah() yaitu menjumlahkan kedua nilai atribut 'bilangan1' dan 'bilangan2' pada obyek 'op1'.

Sedangkan perintah,

```
op1.tampilHasil();
```

digunakan untuk menjalankan method tampilHasil() yaitu menampilkan hasil penjumlahan.

4) Untuk melihat hasil output program, Anda bisa mengcompilenya dahulu kemudian menjalankan RUN PROJECT. Adapun outputnya adalah sebagai berikut:



Gambar Output project contoh2

Dalam sebuah program, bisa dibuat instantisasi beberapa obyek dari class yang sama.

f. Input dan Output dalam Java

Dalam bagian ini akan dibahas cara membaca data input melalui console serta menampilkan outputnya juga melalui console.

1) Input Data Via Console

Untuk keperluan input data via console, perlu dibuat class khusus.

Contoh:

Sebagai contoh, berikut ini adalah sebuah pembuatan program Java untuk perhitungan gaji karyawan yang beberapa datanya diinput lewat *console*:

inputConsole.java

```
import java.io.*;
public class inputConsole {
    // membaca data string public String bacaString()
    {
        BufferedReader bfr = new BufferedReader(new
                        InputStreamReader(Sys
        tem.in), 1); String string = "";
        try
        {
            string = bfr.readLine();
        }
        catch (IOException ex)
        {
            System.out.println(ex);
        }
        return string;
    }
    // membaca
    data
```

```
        integer
        public int
        bacaInt()
        {
            return Integer.parseInt(bacaString());
                }
        // membaca data float public float  bacaFloat()
        {
            return Float.parseFloat(bacaString());
        }
        // membaca data
        long integer
        public long
        bacaLong()
        {
            return  Long.parseLong(bacaString());
        }
    }
```

Secara umum, di dalam class 'inputConsole' tersebut, mekanisme method-method untuk membaca input dalam berbagai tipe data itu adalah membaca setiap input dalam bentuk string kemudian input string tersebut diubah ke tipe data yang bersesuaian.

Selanjutnya buat beberapa kode di bawah ini pada class Gajikaryawan

```
public class Gajikaryawan {
    public static void main(String[] args) {
        String nama, kodekar; int gapok, jmlanak; float gaber,
        tunjanak;
        inputConsole input1 = new inputConsole();
        // input kode karyawan System.out.print("KODE
        KARYAWAN : "); kodekar = input1.bacaString();
        // input nama karyawan System.out.print0;("NAMA
        KARYAWAN : "); nama = input1.bacaString();
        // input gaji pokok karyawan
        System.out.print("GAJI  POKOK: ");
        gapok = input1.bacaInt();
        // input jumlah anak System.out.print("JML  ANAK: ");
        jmlanak = input1.bacaInt();
        // hitung tunjangan anak -> setiap anak 10% dari
        gaji pokok tunjanak = (float) ((float) gapok *
        0.1 *  jmlanak);
        // hitung gaji bersih = gaji pokok + tunj anak
        gaber = gapok + tunjanak;
        // output
        System.out.println("NAMA KARYAWAN : "+nama+"("+
        kodekar +")"); System.out.println("GAJI  BERSIH :
        Rp. "+gaber);
```

```
    }
}
```

2) Output Via Console

Secara umum perintah untuk menampilkan output ke layar console adalah

```
System.out.println(string);
```

atau

```
System.out.print(string);
```

Perbedaan keduanya adalah jika dengan println() setelah menampilkan suatu string ke output console, maka terjadi perpindahan baris pada pointernya. Sedangkan untuk System.output.print() tidak terjadi perpindahan baris pointernya.

3) Mengatur Digit Presisi Bilangan Riil (Float)

Secara default, Java akan menampilkan bilangan riil atau float dalam bentuk 15 digit di belakang koma, misalnya:

```
System.out.print(22./7);
```

akan muncul hasil di layar, bilangan 3.142857142857143

4) Input Data Via GUI (Graphics User Interface)

Selain via console, input data juga bisa dilakukan via GUI. Di dalam Java, untuk membuat aplikasi berbasis GUI bisa menggunakan SWING sebagai package nya, sehingga di dalam program perlu melakukan import dengan perintah sebagai berikut:

```
import javax.swing.*;
```

Berikut ini contoh kode Java untuk menerima input melalui form GUI kemudian outputnya melalui console.

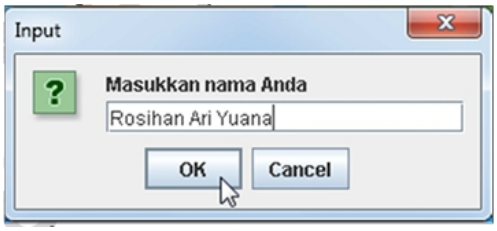
Contoh

Contoh program Java yang menerima input berupa nama (string) kemudian menampilkan nama yang tadi diinputkan via console.

```
import
javax.swing.
*; public
```

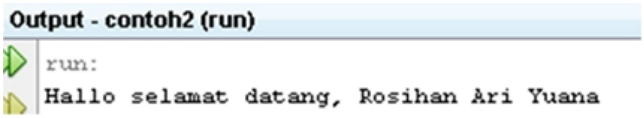
```
class
Contoh2 {
    public static void main(String[] args) {
        String nama;
        nama = JOptionPane.showInputDialog("Masukkan nama
        Anda"); System.out.println("Hallo selamat datang, " +
        nama);
    }
}
```

Tampilan dari kode di atas setelah dirunning adalah sebagai berikut:



Gambar 29. Contoh tampilan input melalui GUI

dan outputnya:



Gambar 30. Output contoh GUI

g. Struktur Kontrol Proses

Struktur kontrol proses bertujuan untuk dapat menentukan urutan statement/perintah yang akan dikerjakan atau diproses. Struktur kontrol proses ini antara lain:

1) Struktur Kontrol Kondisional

Struktur kontrol ini untuk menyatakan proses yang berbentuk persyaratan/kondisional.

Struktur Kontrol IF

Tata cara penulisan statement IF:

```
if (syarat)
{
      statement; statement;
      .
      .
}
```

Bisa juga berbentuk sebagai berikut,

```
if (syarat)
{
       statement; statement;
       .
       .
}
else
{
       statement; statement;
       .
       .
}
```

2) Statement SWITCH

Struktur penulisan statement SWITCH adalah sebagai berikut:

```
switch(ekspresi)
{
    case variabel1  :statement;
                     statement;
                      .
                      .
                     break;
    case variabel2  :  statement;
                       statement;
                      .
                      .
                       break;
                      .
                      .
     Default          : statement;
                       statement;
                        .
                        .
}
```

3) Struktur Kontrol Perulangan (Looping)

Struktur kontrol perulangan digunakan untuk mengatur proses yang dijalankan secara berulang-ulang. Berikut ini beberapa statement yang dapat digunakan untuk mengatur proses perulangan:

a) Statement FOR

Aturan penulisan (syntax) nya adalah:

```
for(ekspresiawal; syarat; ekspresiakhir)
{
       statement;
       statement;
       .
       .
}
```

b) Statement WHILE

Aturan penulisannya:

```
while(syarat)
{
      statement;
      statement;
      .
      .
}
```

c) Statement DO WHILE

Aturan penulisanya:

```
do
{
      statement;
      statement;
      .
      .
}
while (syarat);
```

h. *Constructor*

Di dalam OOP, ada istilah ‘constructor’. ‘Constructor’ ini melekat pada suatu class, sehingga bisa menset beberapa nilai atribut sekaligus dari suatu obyek ketika proses instansiasi. Constructor sangatlah penting pada pembentukan sebuah object. Constructor adalah method dimana seluruh inisialisasi object ditempatkan. Default constructor adalah constructor yang tidak memiliki parameter apapun. Constructor tidak dapat dipanggil secara langsung, namun harus dipanggil dengan menggunakan operator *new* pada pembentukan sebuah *class*. Pemanggilan constructor dapat dilakukan secara berangkai, dalam arti Anda dapat memanggil constructor di dalam constructor lain. Pemanggilan dapat dilakukan dengan referensi this(). Referensi this digunakan untuk mengakses instance variable yang dibiaskan oleh parameter

i. *Larik (Array)*

Seperti halnya bahasa pemrogaman yang lain, di dalam Java juga ada penggunaan Array. Di dalam java nomor indeks suatu array dimulai dari 0.

Berikut ini cara mendeklarasikan sebuah array dengan n buah elemen

```
tipedata[] namaarray = new tipedata[n];
```

Sebagai contoh, perhatikan perintah berikut ini untuk membuat array dengan nama arrayku bertipe data integer dengan jumlah elemennya 10.

```
int[] arrayku  = new int[10];
```

Pada contoh di atas, pendeklarasian tersebut akan memberitahukan kepada compiler Java, bahwa identifier arrayku akan digunakan sebagai nama array yang berisi data bertipe integer, dan dilanjutkan dengan membuat atau men-instantiate sebuah array baru yang terdiri dari 10 elemen.

j. *Enkapsulasi dan modifier*

Enkapsulasi merupakan teknik yang membuat variabel/field class menjadi bersifat private dan menyediakan akses ke variabel/field melalui public method. Jika field di deklarasikan sebagai private, maka field ini tidak bisa diakses oleh siapapun diluar class, dengan demikian field disembunyikan di dalam class.

Manfaat utama teknik enkapsulasi adalah kita mampu memodifikasi kode tanpa merusak kode yang telah digunakan pada class lain.

Enkapsulasi memiliki manfaat sebagai berikut:

- Modularitas
  Source code dari sebuah *class* dapat dikelola secara independen dari *source code class* yang lain. Perubahan internal pada sebuah *class* tidak akan berpengaruh bagi *class* yang menggunakannya.
- *Information Hiding*
  Penyembunyian informasi yang tidak perlu diketahui objek lain.

Encapsulation (Enkapsulasi) adalah suatu cara untuk menyembunyikan implementasi detail dari suatu class. Enkapsulasi mempunyai dua hal mendasar, yaitu:

- information *hiding*
- menyediakan suatu perantara (*method*) untuk pengaksesan data

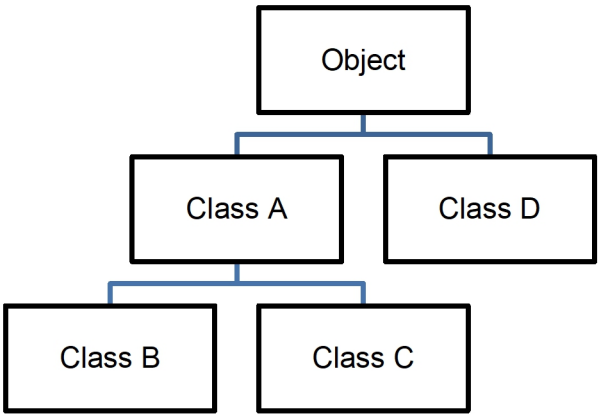
contoh

Listing Program

```
public class Siswa {
private int nrp;
public void setNrp(int n) {
nrp=n;
}
}
```

k. *inheritance*

Dalam Java, semua class, termasuk class yang membangun Java API, adalah subclasses dari superclass Object. Contoh hirarki class diperlihatkan di bawah ini. Beberapa class di atas class utama dalam hirarki class dikenal sebagai superclass. Sementara beberapa class di bawah class pokok dalam hirarki class dikenal sebagai sub class dari class tersebut.



Gambar 31. Hierarki Class di Java

Pewarisan adalah keuntungan besar dalam pemrograman berbasis object karena suatu sifat atau method didefinisikan dalam superclass, sifat ini secara otomatis diwariskan dari semua subclasses. Jadi, Anda dapat menuliskan kode method hanya sekali dan mereka dapat digunakan oleh semua subclass. Subclass hanya perlu mengimplementasikan perbedaannya sendiri dan induknya. Dengan konsep inheritance, sebuah class dapat mempunyai class turunan. Suatu class yang mempunyai class turunan dinamakan parent class atau base class. Sedangkan class turunan itu sendiri sering kali disebut subclass atau child class. Suatu subclass dapat Mewari siapa-apa yang dipunyai oleh parent class-nya, sehingga member dari suatu subclass adalah terdiri dari apa- apa yang ia punyai dan juga apa-apa yang diawarisi dari classparent-nya. Kesimpulannya, boleh dikatakan bahwa suatu subclass adalah tidak lain hanya memperluas (extend) parentclass-nya.

Berikut adalah contoh deklarasi inheritance:

```
public class B extends A{
.....
}
```

Subclass juga dapat memanggil constructor secara eksplisit dari superclass terdekat. Hal ini dilakukan dengan pemanggil constructor super.

Overriding adalah suatu keadaan dimana method pada subclass menolak method pada parent class-nya. Subclass dapat mengesampingkan method yang didefinisikan dalam superclass dengan menyediakan implementasi baru dari method tersebut. Dalam Java, juga memungkinkan untuk mendeklarasikan class-class yang tidak lama menjadi subclass. Class ini dinamakan class final. Untuk mendeklarasikan class untuk menjadi final kita hanya menambahkan kata kunci final dalam deklarasi class. Beberapa class dalam Java API dideklarasikan secara final untuk memastikan sifatnya tidak dapat di-override.

I. *Polimorfisme*

Polymorphism merupakan salah satu konsep penting dalam object oriented programming (OOP) khusunya di bahasa Java setelah abstraction dan inheritance. Polymorphism berarti banyak bentuk. Polymorphism sering dikaitkan dengan penggunaan lebih dari satu metoda dengan nama sama. Penggunaan

metoda dengan nama sama dapat diterapkan dengan method *overloading* dan method *overriding*. Pada saat obyek yang sudah dibuat tersebut memanggil *overridden* method pada parent class, compiler Java akan melakukan *invocation* (pemanggilan) terhadap overriding method pada subclass dimana yang seharusnya dipanggil adalah overridden method.

4. Program untuk mengatasi kesalahan (error handling)

Error handling atau sering juga disebut *exception handling* merupakan mekanisme yang paling diperlukan dalam menangani *error* yang terjadi pada saat runtime (program berjalan) atau yang lebih dikenal dengan sebutan *runtime error.* Secara umum, adanya kesalahan yang terjadi pada program pada saat *runtime* dapat menyebabkan program berhenti atau hang. Untuk itulah diperlukan mekanisme untuk memastikan bahwa program tetap dapat berjalan meskipun terdapat kesalahan yang terjadi.

Secara umum, *exception handling* yang dijelaskan pada modul ini adalah *exception handling* pada pemrograman java yang dilakukan menggunakan *keyword try-catch.*

Tabel 16. Fungsi pada exception handling

Kata kunci	Deskripsi
try	Digunakan untuk menentukan bagian statement program dimana akan terjadi pengecualian. Blok dari try ini harus diikuti dengan catch atau finally
catch	Digunakan untuk menangani kesalahan/pengecualian yang terjadi. Blok catch ini dapat berdiri sendiri tanpa blok try. Blok catch dapat diikuti oleh blok finally
finally	Digunakan untuk mengeksekusi bagian code yang penting dari program. Bagian iji akan tetap dijalankan baik terjadi pengecualian maupun tidak.
throw	Digunakan untuk melempar pengecualian yang terjadi, dimana throw digunakan dalam body dari code yang ada
throws	Digunakan untuk mendeklarasikan pengecualian yang akan terjadi pada bagian fungsi tersebut.

Exception adalah peristiwa yang terjadi ketika proses running program yang mengakibatkan program berhenti, ditandai dengan munculnya pesan *error.*

Untuk mengantisipasi munculnya exception tersebut, bisa dilakukan penanganan dengan statement:

```
try
   {
  ...
   }
   catch (namaexception var)
   {
  ...
   }
```

Keterangan:

- ‘namaexception’ nantinya diisikan dengan nama exception yang muncul, dalam contoh kasus sebelumnya yang merupakan nama exception adalah ‘**NumberFormatException**’
- Secara umum, bisa digunakan keyword ‘**Exception**’ pada ‘namaexception’ untuk menangkap semua exception yang terjadi. Keyword ‘Exception’ adalah top level dari semua exception.

```
run:
Exception in thread "main" java.lang.NumberFormatException: For input
        at java.lang.NumberFormatException.forInputString(NumberFormat
        at java.lang.Integer.parseInt(Integer.java:492)
```

- Sedangkan ‘var’ diisikan dengan sembarang nama variabel.

Dengan statement try-catch di atas, maka pesan error exception tidak akan muncul namun akan digantikan dengan pesan atau prosedur lain yang dituliskan dalam bagian catch.

5. Koneksi database server melalui client-server

Dalam konteks basis data, client mengatur *interface* berfungsi sebagai *workstation* tempat menjalankan aplikasi basis data. Client menerima permintaan pemakai, memeriksa sintaks dan generate kebutuhan basis data dalam SQL atau bahasa yang lain. Kemudian meneruskan pesan ke server, menunggu response dan bentuk response untuk pemakai akhir. Server menerima dan memproses permintaan basis data kemudian mengembalikan hasil ke client.

Proses-proses ini melibatkan pemeriksaan autorisasi, jaminan integritas, pemeliharaan data dictionary dan mengerjakan query serta proses update. Selain itu juga menyediakan kontrol terhadap concurrency dan recovery.

Sedangkan fungsi dari masing-masing Client Server

Client

1. Mengatur *user interface*
2. Menerima dan memeriksa sintaks input dari pemakai
3. Memproses aplikasi
4. Generate permintaan basis data dan memindahkannya ke server
5. Memberikan response balik kepada pemakai
6. Menyediakan akses basis data secara bersamaan
7. Menyediakan *control recovery*

Server

- Menerima dan memproses basis data yang diminta dari client
- Memeriksa autorisasi
- Menjamin tidak terjadi pelanggaran terhadap integrity constraint
- Melakukan query/pemrosesan update dan memindahkan response ke client
- Memelihara data dictionary

6. Desain user interface

User interface (UI) merupakan cara sebuah program dengan pengguna untuk saling berkomunikasi atau bisa dikatakan sebagai segala sesuatu yang dirancang menjadi sebuah perangkat informasi, dimana pengguna dapat melakukan sebuah interaksi dengan sebuah program dengan lebih mudah. Media yang dapat digunakan pengguna untuk berinteraksi dengan program (apliksi atau *website*) dapat berupa tampilan layar (*layout*), *keyboard*, dan *mouse*.

a. Desain *Output*

Salah satu cara untuk menggolongkan output adalah dengan melihat distribusinya apakah ke dalam atau ke luar perusahaan, dan orang-orang yang membaca dan menggunakan output.

Internal output digunakan untuk para pemilik dan pengguna sistem dalam sebuah perusahaan.

Eksternal output bersifat keluar organisasi. Output ini ditujukan kepada

konsumen, pemasok, mitra bisnis dan badan pemerintahan. Output eksternal menyimpulkan dan melaporkan transaksi bisnis. Contoh faktur, nota pembelian, jadwal kursus, tiket pesawat, tagihan telepon dan lain sebagainya.

b. Proses Desain Output

Langkah-langkah atau proses desain *output* adalah:

1) Mengidentifikasi output sistem dan meninjau persyaratan logis
2) Menentukan persyaratan output fisik

c. Desain Input

Untuk menginput data ke dalam komputer, analis sistem harus mendesain dokumen sumber, screen input dan metode serta prosedur untuk memasukkan data ke dalam komputer (dari konsumen ke form ke staf entry data ke komputer).

d. Kontrol Internal – Data Editing untuk Input

Kontrol internal merupakan persyaratan yang ada di seluruh sistem berbasis computer. Kontrol input internal menjamin input data pada komputer tersebut akurat dan bahwa sistem tersebut aman terhadap suatu kesalahan incidental dan penyalahgunaan.

e. Proses Desain Input

Langkah-langkahnya adalah sebagai berikut:

1) Mengidentifikasi input sistem dan memberikan persyaratan logika
2) Memilih control GUI yang sesuai
3) Mendesain, memvalidasi dan mengetes input dengan menggunakan beberapa kombinasi dari : Peralatan layout dan Prototyping peralatan.
4) Jika perlu, mendesain source document

f. Desain Antar Muka

Pada desain antarmuka, audiens adalah *system user. System user* dapat diklasifikasikan secara luas baik sebagai pakar atau orang baru dan baik secara terikat dan tidak terikat. *Expert user (dedicated user)* adalah pengguna komputer yang berpengalaman yang banyak menghabiskan waktunya untuk menggunakan program aplikasi khusus. *Novice user (casual user)* adalah pengguna komputer yang pengalamannya lebih sedikit yang biasanya menggunakan komputer pada

frekuensi sedikit atau bahkan pada saat-saat tertentu saja.

7. Model MVC dalam pengembangan aplikasi

Model View Controller atau MVC adalah sebuah metode untuk membuat sebuah aplikasi dengan memisahkan data *(Model)* dari tampilan (View) dan cara bagaimana memprosesnya *(Controller).* Dalam implementasinya kebanyakan *framework* dalam aplikasi dan website adalah berbasis MVC. MVC memisahkan pengembangan aplikasi berdasarkan komponen utama yang membangun sebuah aplikasi seperti manipulasi data, antarmuka pengguna, dan bagian yang menjadi kontrol dalam sebuah aplikasi web.

Komponen MVC, terdiri atas:

a. *Model*, model mewakili struktur data. Biasanya model berisi fungsi-fungsi yang membantu seseorang dalam pengelolaan basis data seperti memasukkan data ke basis data, pembaruan data dan lain-lain.
b. *Viewers,* viewers adalah bagian yang mengatur tampilan ke pengguna. Bisa dikatakan berupa halaman web.
c. *Controller, controller* merupakan bagian yang menjembatani model dan view. Controller berisi perintah-perintah yang berfungsi untuk memproses suatu data dan mengirimkannya ke halaman web.

D. Rangkuman

1. Rangkuman Konsep objek oriented dan Analisis dan desain berorientasi objek

   Tahap atau skema pelaksanaan analisis berorientasi objek , yaitu mentukan kebutuhan pemakai untuk sistem berorientasi objek, mengidentifikasi kelas dan objek, mengidentifikasi atribut dan layanan untuk setiap objek, mendefinisikan efinisikan struktur dan hirarki, mebuat uat model hubungan objek dan membuat model perilaku objek.

   Unified Modeling Language (UML) adalah bahasa pemodelan untuk sistem atau perangkat lunak yang berparadigma berorientasi objek. Unified Modeling Language (UML) disebut bahasa pemodelan bukan metode. Bahasa pemodelan (sebagaian besar grafik) merupakan notasi dari metode yang

digunakan untuk mendesain secara cepat. Bahasa pemodelan merupakan bagian terpenting dari metode. Ini merupakan bagian kunci untuk komunikasi. Pemodelan ini merupakan bahasa standar untuk digunakan dalam visualisasi, spesifikasi, pembentukan dan pendokumentasian alat – alat dari sistem perangkat lunak. Dengan demkian UM sebagai bahasa pemodelan, sebagai bahasa untuk menggambarkan sistem, sebagai bahasa untuk menspesifikasi sistem, dan sebagai bahasa untuk pendokumentasian sistem.

2. Manajemen Sistem Basis Data (Database Management System/DBMS)

   Sistem manajemen basis data atau database management system (DBMS), atau kadang disingkat SMBD, adalah suatu sistem atau perangkat lunak yang dirancang untuk mengelola suatu basis data dan menjalankan operasi terhadap data yang diminta banyak pengguna. komponen utama DBMS, yaitu: perangkat keras (hardware), data, perangkat lunak (software) dan pengguna.

   RDBMS tidak hanya menjada salah satu model basis data, tetapi telah menjadi software pemrosesan data yang dominan saat ini. Software ini menggambarkan generasi kedua dari DBMS dan berbasiskan model data relasional yang diajukan oleh E.F. Codd (1970). Pada model relasional, seluruh data terstruktur secara logika di dalam sebuah relasi (tabel). Setiap relasi mempunyai nama dan terdiri dari atribut-atribut bernama (kolom). Setiap tuple (baris) berisikan satu nilai per atribut. Kekuatan yang besar dari model data relasional adalah struktur logikal yang sederhana. RDBMS dapat mengatasi semua kekurangan pada model data sebelumnya.

   Untuk mendukung kepraktisan, DBMS menyediakan pandangan abstrak terhadap data bagi pengguna. DBMS berusaha menyembunyikan detail tentang bagaimana data disimpan dan dipelihara.

   Untuk menjaga database dari pengrusakan data dan pemakaian data oleh pemakai yang tidak punya kewenangan, penerapan keamanan database adalah hal wajib yang harus dilakukan. Keamanan database adalah suatu cara untuk melindungi database dari ancaman, baik dalam bentuk kesengajaan atau pun bukan. Secara garis besar keamanan database

dikategorikan sebagai berikut: keamanan server, trusted Ip Access, koneksi database dan kontrol akses tabel. Pengamanan basis data dapat dilakukan dengan cara otorisasi, tabel view, backup data dan recovery , integritas data dan enkripsi

Replikasi adalah suatu teknik untuk melakukan copy dan pendistribusian data dan objek-objek database dari satu database ke database lain dan melaksanakan sinkronisasi antara database sehingga konsistensi data dapat terjamin. Dengan menggunakan teknik replikasi ini, data dapat didistribusikan ke lokasi yang berbeda melalui koneksi jaringan lokal maupun internet. Replikasi juga memungkinkan untuk mendukung kinerja aplikasi, penyebaran data fisik sesuai dengan penggunaannya, seperti pemrosesan transaksi online dan DSS (Desiscion Support System) atau pemrosessan database terdistribusi melalui beberapa server. Replikasi dapat dilakukan baik secara synchronous maupun asynchronous.

3. Konsep dan implementasi pemrograman berorientasi objek dalam pengembangan apikasi atau sistem informasi

   Model data berorientasi objek dikatakan dapat memberi fleksibilitas yang lebih, kemudahan mengubah program, dan digunakan luas dalam teknik peranti lunak skala besar. Lebih jauh lagi, pendukung OOP mengklaim bahwa OOP lebih mudah dipelajari bagi pemula dibanding dengan pendekatan sebelumnya, dan pendekatan OOP lebih mudah dikembangkan dan dirawat.

   Pemrograman Java menggunakan konsep Pemrograman Berorientasi Obyek (PBO) atau Object Oriented Programming (OOP). Semua program Java merupakan suatu obyek. Dasar-dasar OOP meliputi istilah yaitu: class, object, attribute dan method. Perhatikan contoh berikut.

Class	Object	Atribute	method
Mobil	Suzuki, Ertiga, Avanza	Jumlah roda, warna mobil, jumlah tempat duduk	maju, mundur dan berhenti

- Class

  Class adalah model dari suatu obyek yang menjelaskan karakteristik (sifat) serta fungsi yang dimiliki dari suatu obyek. Class merupakan wadah (tempat) yang digunakan untuk menciptakan suatu obyek. Dengan kata lain sebuah Class merupakan blueprint dari suatu obyek.

- Object adalah abstraksi dan sesuatu yang mewakili dunia nyata seperti benda, manusia, satuan organisasi, tempat, kejadian, struktur, status, atau hal-hal lain yang bersifat abstrak.
- Atribute

  Atribut adalah elemen data dari suatu class. Atribut menyimpan informasi tentang class. Atribut dapat diartikan sebagai data, variabel, properti atau sebuah field. Atribut dapat berupa nilai atau elemen-elemen data yang dimiliki oleh Objek dalam kelas objek. Atribut dipunyai secara individual Oleh sebuah objek, misalnya berat, jenis, nama, dan sebagainya
- Method

  Method (metode) adalah sebuah function atau fungsi yang ada dalam suatu class. Setiap metode memiliki tugas sendiri. Operasi atau metode atau method pada sebuah kelas hampir sama dengan fungsi atau prosedur pada metodologi struktural. Sebuah kelas boleh memiliki lebih dari satu metode atau operasi. Metode atau operasi yang berfungsi untuk memanipulasi objek itu sendiri.

4. Koneksi database server melalui client-server

   Klien-server atau client-server merupakan sebuah paradigma dalam teknologi informasi yang merujuk kepada cara untuk mendistribusikan aplikasi ke dalam dua pihak: pihak klien dan pihak server. Dalam model klien/server, sebuah aplikasi dibagi menjadi dua bagian yang terpisah, tetapi masih merupakan sebuah kesatuan yakni komponen klien dan komponen server. Komponen klien juga sering disebut sebagai front-end, sementara komponen server disebut sebagai back-end. Komponen klien dari aplikasi tersebut dijalankan dalam sebuah workstation dan menerima masukan data dari pengguna. Komponen klien tersebut akan menyiapkan data yang dimasukkan oleh pengguna dengan menggunakan teknologi pemrosesan tertentu dan mengirimkannya kepada komponen server yang dijalankan di atas mesin server, umumnya dalam bentuk request terhadap beberapa layanan yang dimiliki oleh server. Komponen server akan menerima request dari klien, dan langsung memprosesnya dan mengembalikan hasil pemrosesan tersebut kepada klien. Klien pun menerima informasi hasil pemrosesan data yang dilakukan server dan menampilkannya kepada pengguna, dengan menggunakan aplikasi yang berinteraksi dengan pengguna.

5. Desain user interface

   User interface (UI) merupakan cara sebuah program dengan pengguna untuk saling berkomunikasi atau bisa dikatakan sebagai segala sesuatu yang dirancang menjadi sebuah perangkat informasi, dimana pengguna dapat melakukan sebuah interaksi dengan sebuah program dengan lebih mudah. Media yang dapat digunakan pengguna untuk berinteraksi dengan program (apliksi atau website) dapat berupa tampilan layar (layout), keyboard, dan mouse.

6. Model View Controller atau MVC adalah sebuah metode untuk membuat sebuah aplikasi dengan memisahkan data (Model) dari tampilan (View) dan cara bagaimana memprosesnya (Controller). Dalam implementasinya kebanyakan framework dalam aplikasi dan website adalah berbasis MVC. MVC memisahkan pengembangan aplikasi berdasarkan komponen utama yang membangun sebuah aplikasi seperti manipulasi data, antarmuka pengguna, dan bagian yang menjadi kontrol dalam sebuah aplikasi web.