# BAB II
# LANDASAN TEORI

## 2.1 Spesifikasi ISO-8583

### 2.1.1 Pengenalan

Kegiatan transaksi di industri perbankan kian banyak dilakukan secara non-tunai. Masyarakat makin akrab dengan alat pembayaran menggunakan kartu (APMK) (Channel, 2010).

Layanan dalam industri finansial membutuhkan pertukaran data pesan elektronis terkait dengan transaksi finansial. Kesepakatan penggunaan spesifikasi standar merupakan keharusan agar interkoneksi antar sistem institusi finansial dapat terwujud. Pertukaran data di antara sistem yang berbeda perlu mengikuti suatu format standar agar dapat terintegrasi dan beroperasi.

Spesifikasi ISO-8583 merupakan standar spesifikasi yang diterbitkan oleh ISO (International Organization for Standardization) suatu lembaga internasional, untuk kebutuhan standar spesifikasi pertukaran data pesan elektronis transaksi finansial.



ISO-8583 adalah spesifikasi untuk transaksi finansial berbasis kartu antar sistem yang diadaptasi oleh Industri Pembayaran pada umumnya. ISO-8583 menjelaskan struktur, format dan nilai elemen data.

Transaksi berbasis kartu sendiri saat ini beragam. Namun pada umumnya adalah transaksi pembelian, penarikan uang, cash deposit, pemindah bukuan, cek saldo, pembayaran tagihan dan transfer dana antar bank.

Berikut adalah istilah-istilah dalam pengenalan ISO-8583 :

- *Acquirer* : institusi finansial yang memperoleh data transaksi dari pembaca kartu, dan memulai proses data tersebut ke sistem lain yang terhubung. Biasa juga dikenal sebagai institusi pemilik terminal.

- *Card Issuer* : institusi yang mengeluarkan kartu kepada pemegang kartu.
- *Advice* : message yang menandakan sejumlah aksi yang telah dilakukan, tidak perlu persetujuan.
- *Authorization* : persetujuan atau garansi yang diberikan oleh issuer kepada acquirer (atau ke pembaca kartu).
- *Card Acceptor* : komponen yang membaca kartu dan memberi data transaksi kepada acquirer.
- *Card Holder* : nasabah yang terkait dengan nomor rekening yang melakukan transaksi dari pembaca kartu.
- *Interactive Message* : message yang dikirim dan direspon pada saat transaksi sedang berlangsung.
- *Message*: sekumpulan elemen data yang digunakan untuk bertukar informasi antar institusi.
- *Non-interactive Message* : message yang dikirim setelah transaksi berlangsung dan tidak berpengaruh oleh respon transaksi tersebut.
- *Point of Service (POS)* : lokasi tempat dimulainya transaksi.
- *Reversal* : message yang menyatakan bahwa transaksi original tidak dapat diproses sebagaimana mestinya, misal : tidak dapat dikirim, tidak dapat diproses, ataupun dibatalkan oleh penerima.
- *Settlement* : pemindahan dana untuk menyelesaikan transaksi yang sudah pernah dilakukan, sesuai dengan perhitungan akuntansi akhir.



### 2.1.2 Struktur Message ISO-8583

Message ISO-8583 terdiri dari 3 bagian utama, yakni Message Type Indicator, Bitmap dan rangkaian Data Element. Masing-masing bagian ini merupakan bagian yang tidak dapat terpisahkan, namun minimal sebuah message ISO-8583 terdiri dari Message Type Indicator dan Bitmap.

Penjelasan lebih detil mengenai masing-masing bagian di atas akan dijelaskan pada sub-bab berikutnya.

### 2.1.3 Message Type Indicator (MTI)

Message Type adalah empat digit field berjenis numerik, yang menjelaskan tipe message dan fungsinya. Setiap message dimulai dengan MTI. Komposisi MTI sendiri sebenarnya terdiri dari 2 bagian yaitu Version Number (satu digit) dan Message Type sendiri (tiga digit).

Version Number menjelaskan versi ISO-8583 yang dipergunakan. Saat ini ISO telah merilis 3 versi dari spesifikasi ISO-8583 yaitu:

**Tabel 2.1** Tabel Version Number

| Kode | Standar | Tahun Rilis | Keterangan |
|---|---|---|---|
| 0 | ISO-8583 | 1987 | - |
| 1 | ISO-8583 | 1993 | - |
| 2 | ISO-8583 | 2003 | - |
| 3-9 | - | - | Reservasi oleh ISO |

Message Type Identifier merupakan tiga digit angka yang mengidentifikasi Message Class, Message Function dan Inisiator Transaksi. Berikut merupakan tabel nilai MTI serta artinya.



**Tabel 2.2** Message Type Identifier

| MTI | Kode | Deskripsi |
|---|---|---|
| Message Class | 0 | Reserved for ISO use |
| | 1 | Authorization |
| | 2 | Financial presentment |
| | 3 | File action |
| | 4 | Reversal/chargeback |
| | 5 | Reconciliation |
| | 6 | Administrative |
| | 7 | Fee collection |
| | 8 | Network management |
| | 9 | Reserved for ISO use |

**Tabel 2.2** Message Type Identifier (lanjutan)

| MTI | Kode | Deskripsi |
|---|---|---|
| Message Function | 0 | Request |
| | 1 | Request response |
| | 2 | Advice |
| | 3 | Advice response |
| | 4 | Notification |
| | 5 | Notification acknowledgement |
| | 6 | Instruction |
| | 7 | Instruction acknowledgement |
| | 8 | Reserved for ISO use |
| | 9 | Reserved for ISO use |
| Inisiator Transaksi | 0 | Acquirer |
| | 1 | Acquirer repeat |
| | 2 | Card issuer |
| | 3 | Card issuer repeat |
| | 4 | Other originator |
| | 5 | Other originator repeat |
| | 6 | Reserved for ISO use |
| | 7 | Reserved for ISO use |
| | 8 | Reserved for ISO use |
| | 9 | Reserved for ISO use |

### 2.1.4 Bitmap Data

Komponen message setelah MTI merupakan bitmap yang dapat terdiri dari satu atau dua bitmap. Masing-masing bitmap berisi 64 bit. Setiap bit menunjukan Data Element bersangkutan ada (bit bernilai 1) atau tidak ada (0).

Primary bitmap (bit 1 - 64) akan selalu ada, dan Data Element yang paling sering dipergunakan akan dipetakan oleh bitmap ini. Sedangkan Data Element yang jarang dipergunakan akan menggunakan Secondary bitmap (bit 65 – 128).

Keberadaan secondary bitmap dapat dilihat dari nilai bit ke-1 dari Primary bitmap. Jika bernilai 1 makan Secondary bitmap dipergunakan dan jika bernilai 0 makan Bitmap data hanya berisi Primary Bitmap.

### 2.1.5 Data Element

Elemen Data adalah satuan data terkecil dalam format ISO-8583, yang merupakan representasi informasi dari transaksi. Pada standar ISO-8583:1987 terdapat maksimum 128 elemen data. Tiap elemen memiliki fungsi dan format tertentu. ISO8583 memiliki elemen data umum dan juga elemen dengan fungsi spesifik tertentu, yang dapat digunakan berbeda-beda oleh berbagai standar sistem.

Format Data Element memiliki 3 tipe format, yaitu:

1. Fixed length

   Data element dengan format ini akan memiliki panjang data yang tetap. Misalnya pada Data Element No 3 (Processing Code), panjang data akan selalu 6 digit.

2. LLVAR

   Data Element dengan tipe ini memiliki kemungkinan panjang data yang bervariasi bergantung kebutuhan yang ada. Namun panjang data maksimum yang memungkinkan adalah 99 digit. Walaupun demikian ISO-8583 membatasi nilai panjang digit Data Element dengan tipe format ini. Sebagai contoh Data Element no 2 (PAN), nilai Data Element ini memungkinkan sampai dengan 19 digit.

   Notasi yang dipergunakan adalah: 2 digit informasi panjang + data.

3. LLLVAR

   Sama halnya dengan LLVAR, Data Element dengan tipe ini memiliki kemungkinan panjang data yang bervariasi bergantung kebutuhan yang ada. Namun panjang data maksimum yang memungkinkan adalah 999 digit. Sebagai contoh Data Element no 48 (Additional Data), nilai Data Element ini memungkinkan sampai dengan 999 digit.

   Notasi yang dipergunakan adalah: 3 digit informasi panjang + data.

Variasi dalam penggunaan elemen ini yang menyebabkan untuk setiap institusi atau bahkan setiap layanan dalam satu institusi finansial dapat memiliki beberapa variasi ISO-8583.

Setiap versi ISO-8583 dapat memiliki spesifikasi panjang data element yang berbeda. Misalnya ISO-8583:1987 dengan ISO-8583:1993 memiliki beberapa perbedaan panjang data element (daftar tabel data element ISO-8583:1987 dapat dilihat pada Lampiran 1).

## 2.2 Konsep Basis Data

Pada bagian ini akan diberikan penjelasan mengenai teori dasar basis data yang meliputi Database Management System (DBMS) dan Structured Query Language (SQL).

### 2.2.1 Database Management System

Basis data adalah sebuah koleksi data yang diorganisir sedemikian rupa sehingga dapat diakses, dikelola, dan dirubah dengan mudah. Sedangkan perangkat lunak yang didesain untuk mengelola basis data disebut Database Management System (DBMS). Alternatif lain untuk menyimpan data adalah dengan menggunakan file system. (Bambang 2004)

DBMS dibangun untuk mengatasi masalah-masalah di atas dalam penyimpanan data. Sebuah DBMS meliputi:

1. Sebuah bahasa pemodelan untuk mendefinisikan skema dari setiap basis data sesuai dengan model data dari DBMS. Model data adalah model abstrak yang menjelaskan bagaimana data direpresentasikan dan digunakan. Istilah model data secara umum dapat diartikan sebagai teori model data atau instance model data. Teori model data atau model basis data adalah deskripsi formal mengenai bagaimana data terstruktur dan digunakan. Sedangkan instance model data adalah pengaplikasian sebuah teori model data untuk membuat sebuah instance model data praktikal untuk aplikasi tertentu. Model data yang biasa digunakan adalah model hierarki, jaringan, dan relasional. Sebuah DBMS dapat mengimplementasi-kan satu atau lebih model data.

2. Struktur data yang dioptimalkan untuk menangani penyimpanan data dalam jumlah yang sangat banyak pada sebuah perangkat penyimpanan data permanen.
3. Sebuah bahasa query basis data yang memungkinkan pengguna untuk berinteraksi, menganalisis, dan melakukan perubahan pada data di dalam basis data sesuai dengan hak aksesnya.
4. Sebuah mekanisme transaksi yang menangani pengaksesan sistem oleh banyak pengguna secara konkuren sehingga integritas data dapat terjaga.

Model basis data yang sering digunakan saat ini adalah model relasional. Model relasional adalah model basis data yang berdasarkan pada logika predikat dan teori himpunan. Pertama kali diformulasikan dan diajukan pada tahun 1969 oleh Edgar F. Codd dengan tujuan menghindari kebutuhan menulis program untuk mengekspresikan query basis data dan menerapkan integrity constraint basis data .

DBMS yang berdasarkan pada model relasional disebut dengan Relational Database Management System (RDBMS). RDBMS menyimpan data dan relasinya dalam bentuk tabel. Hampir semua RDBMS menggunakan Structured Query Language sebagai bahasa query-nya.



### 2.2.2 Structured Query Language

Structured Query Language atau SQL adalah sebuah bahasa komputer yang didesain untuk melakukan pengambilan dan manajemen data, pembuatan dan perubahan skema basis data, serta manajemen pengontrolan akses objek basidata dalam RDBMS (Mike Chapple 2007).

Versi pertama dari SQL dikembangkan di IBM oleh Donald D. Chamberlin dan Raymond F. Boyce pada awal tahun 1970-an. Versi ini, yang awalnya disebut dengan SEQUEL, didesain untuk memanipulasi dan memperoleh data yang tersimpan di dalam produk basis data relasional IBM, System R. Selanjutnya bahasa SQL distandarisasi oleh American National Standards Institute (ANSI) dan International Organization for Standardization (ISO).

Pada awalnya SQL didesain sebagai sebuah declarative query dan data manipulation language. Tapi beberapa vendor DBMS menambahkan procedural constructs, control-of-flow statements, tipe data user-defined, dan ekstensi bahasa lainnya. Dengan ditetapkannya standar SQL:1999, banyak ekstensi yang secara formal diadopsi sebagai bagian dari bahasa SQL melalui bagian SQL Persistent Stored Modules (SQL/PSM) dari standar.

Bahasa SQL terdiri atas beberapa bagian, meliputi:

1. Data Definition Language (DDL). Bagian ini menyediakan perintah untuk mendefinisikan skema relasi, menghapus relasi, dan memodifikasi skema relasi.
2. Data Manipulation Language (DML). Bagian ini menyediakan perintah untuk menambahkan, menghapus, dan memodifikasi data dalam basidata.
3. View definition. Bagian ini menyediakan perintah untuk mendefinisikan view.
4. Transaction control. Bagian ini menyediakan perintah untuk menspesifikasikan awal dan akhir dari transaksi.
5. Embedded SQL dan dynamic SQL. Bagian ini mendefinisikan bagaimana statement SQL dapat digunakan di dalam bahasa pemrograman seperti C, C++, Java, PL/I, Cobol, Pascal, atau Fortran.
6. Integrity. Bagian ini menyediakan perintah untuk menspesifikasikan integrity constraints yang harus dipenuhi oleh data yang disimpan.
7. Authorization atau Data Control Language (DCL). Bagian ini menyediakan perintah untuk menspesifikasikan hak akses pengguna terhadap basis data.

## 2.3 MySQL

MySQL adalah RDBMS yang menggunakan Structured Query Language (SQL) sebagai bahasa query nya dan merupakan perangkat lunak open source yang menggunakan lisensi GNU General Public License (GPL) (Paul 2005).

MySQL terdiri atas MySQL Server, MySQL Client, tools administratif, dan programming interface (Paul 2005). Lahirnya MySQL berawal pada tahun 1979 saat tool basis data UNIREG dibuat oleh Michael "Monty" Widenius untuk perusahaan Swedia, TcX. Pada tahun 1994, TcX mulai mencari sebuah RDBMS dengan sebuah interface SQL untuk digunakan dalam pengembangkan aplikasi Web. Mereka melakukan pengetesan beberapa server basis data komersial, tapi semuanya masih terlalu lambat untuk tabel TcX yang besar. Akhirnya Monty mulai mengembangkan sebuah server baru. Programming interface nya didesain secara eksplisit agar mirip dengan yang digunakan oleh mSQL.

Pada tahun 1995, David Axmark dari Detron HB mulai mendorong TcX untuk meluncurkan MySQL. David juga mengerjakan dokumentasinya dan mengusahakan agar MySQL dibangun dengan GNU configure utility. MySQL 3.11.1 diluncurkan di dunia pada tahun 1996 dalam bentuk binary distribution untuk Linux dan Solaris.

Saat ini MySQL dimiliki dan disponsori oleh perusahaan Swedia, MySQL AB, yang memiliki copyright dari sebagian besar codebase-nya. MySQL AB didirikan oleh David Axmark, Allan Larsson, dan Michael "Monty" Widenius. MySQL telah tersedia untuk berbagai macam sistem operasi dan pada berbagai arsitektur komputer. Hingga saat ini MySQL mendukung sistem operasi Linux, Windows 95/98/NT/2000/XP/Vista, Solaris, FreeBSD, MacOS X, HP-UX, AIX, SCO, SCI Irix, Dec OSF, dan BSDi. Versi Linux dapat berjalan pada berbagai arsitektur termasuk Intel libc6, Alpha, IA64, SPARC, dan S/390.

Awalnya, MySQL menjadi populer karena kecepatan dan kesederhanaannya. Tetapi ada yang mengkritik MySQL karena tidak memiliki fitur-fitur tertentu seperti transactions dan dukungan foreign key. Kemudian MySQL terus berkembang, menambahkan berbagai macam fitur baru seperti row-level locking, replication, subqueries, stored procedures, views, dan triggers.

MySQL adalah proyek open source yang dapat digunakan secara bebas dalam berbagai keadaan, dan ini menjadikannya populer di antara komunitas open source.Kepopuleran MySQL tidak hanya terbatas pada kalangan open source, tetapi menyentuh hingga kalangan pengguna komputer pribadi. Ini disebabkan oleh kemampuan MySQL yang dapat berjalan dengan mudah pada komputer yang memiliki spesifikasi rendah sekalipun.

MySQL menawarkan banyak fitur menarik yang diantaranya adalah:

1. Kecepatan. MySQL menawarkan sistem basis data yang tercepat.
2. Mudah dalam penggunaan. MySQL secara relatif merupakan sistem basis data yang sederhana dan tidak terlalu kompleks dalam pengaturannya.
3. Query language support. MySQL menggunakan SQL, bahasa standar yang digunakan sistem basis data modern.
4. Kapabilitas. MySQL Server adalah multi-threaded, sehingga memungkinkan banyak koneksi dari client dalam satu waktu. Setiap client dapat menggunakan beberapa basis data secara simultan.
5. Konektifitas dan keamanan. MySQL adalah sistem yang dirancang untuk jaringan, sehingga basis data dapat diakses dari manapun melalui jaringan Internet. MySQL memiliki access control sehingga kemampuan akses user nya dapat diatur. Selain itu MySQL juga mendukung koneksi terenkripsi menggunakan protokol Secure Sockets Layer (SSL).
6. Portabilitas. MySQL dapat berjalan pada berbagai macam sistem operasi dan arsitektur komputer.
7. Ukuran kecil. MySQL memiliki ukuran distribusi yang relatif kecil dibanding sistem basis data komersial yang ada.
8. Availability dan biaya. MySQL memiliki dua lisensi. Lisensi pertama adalah GPL yang berarti MySQL tersedia dengan tanpa biaya. Dan lisensi kedua adalah lisensi komersial yang diperuntukkan bagi organisasi yang tidak ingin terikat oleh GPL.
9. Open distribution dan open source code.

## 2.4 MySQL User-Defined Functions (UDF)

Basis data MySQL memungkinkan pengembang atau *user* membuat sebuah fungsi *native* yang dapat diintegrasikan ke dalam MySQL dan menjadikannya sebagai sebuah kesatuan fungsi dalam MySQL seperti halnya sebuah fungsi bawaan dalam MySQL. Contohnya adalah fungsi password() atau md5(). Hal ini disebut sebagai User-Defined Functions (UDF) dalam MySQL.

MySQL memberikan akses kepada fungsi yang telah dikembangkan terhadap proses internal MySQL dan memberikan kemampuan untuk dapat mengakses dan memanipulasi data.

Umumnya pengembangan UDF dilakukan menggunakan bahasa C. Namun dikarenakan pada dasarnya MySQL mengenali fungsi tambahan ini sebagai sebuah fungsi dalam *shared library*, maka pembuatan UDF tidak terbatas hanya menggunakan bahasa C. Seluruh bahasa pemrograman yang dapat melakukan kompilasi kode menjadi *shared library* dapat dipergunakan.

MySQL mendukung 2 jenis UDF yaitu:

1. *Standard*, untuk jenis UDF ini MySQL akan mengaplikasikan fungsi pada sebuah hasil query (result set). Sebagai contoh:

```
SELECT UPPER(Nama) FROM User;
```

   Contoh di atas sebuah fungsi standar `UPPER` akan melakukan konversi terhadap seluruh field `Nama` dalam table `User`.

2. *Aggregate*, untuk jenis UDF ini MySQL akan mengaplikasikan fungsi pada sejumlah grup dari data hasil sebuah query `GROUP BY`.

```
SELECT AVG(Nilai) FROM Mahasiswa GROUP BY Angkatan;
```

   Contoh di atas sebuah fungsi aggregate `AVG` akan melakukan perhitungan rata-rata terhadap field `Nilai` dalam tabel `Mahasiswa` untuk `Angkatan` yang sama.

Selanjutnya akan dijelaskan lebih dalam mengenai fungsi standar yang akan mendukung penyusunan Tugas Akhir ini.

### 2.4.1 Standard Function

Shared library dengan fungsi standar memiliki struktur sebagai berikut:

1. Init

   Pada saat query SQL memanggil sebuah UDF library, sub rutin init akan dipanggil terlebih dahulu jika ada. Sub rutin ini berfungsi untuk alokasi memori internal library UDF pada saat pemrosesan data berlangsung.

   Atau dapat pula dilakukan pengecekan dan setting terhadap parameter-parameter yang dibutuhkan oleh sebuah fungsi UDF.

2. Main

   Sub rutin main merupakan inti dari pemrosesan oleh sebuah UDF. Artinya seluruh pemrosesan data harus dilakukan pada sub rutin ini, serta mengembalikan hasilnya setelah selesai dilakukan pemrosesan. Sub rutin ini merupakan sub rutin yang harus dimiliki oleh sebuah UDF.

3. Deinit

   Setelah sub rutin main dipanggil dan mengembalikan hasil pemrosesannya, maka MySQL akan memanggil sub rutin deinit.

   Jika pada sub rutin init melakukan alokasi memori terhadap sebuah variable, maka pada sub rutin ini dilakukan dealokasi memori, sehingga tidak terjadi *memory leak*.



Urutan eksekusi MySQL untuk fungsi *standard* dapat digambarkan dalam urutan sebagai berikut:

```
Init
  Main
  Main
  Main
  ...
Deinit
```

Setiap kali MySQL melakukan *retrieval* terhadap sebuah baris data, maka fungsi Main akan dipanggil dan nilai kembalian dari fungsi tersebut akan menjadi data akhir yang akan diberikan MySQL kepada client/aplikasi sebagai hasil dari sebuah query dengan menggunakan fungsi UDF (George Reese 2002).

## 2.5 Pemrograman Delphi

Delphi merupakan salah satu bahasa pemrograman tingkat tinggi yang mendukung pemrograman secara struktural maupun object oriented. Bahasa pemrograman Delphi merupakan pengembangan programan Object Pascal.

Delphi menyediakan lingkungan pengembangan aplikasi atau Integrated Development Environment (IDE) sebagai sarana perancangan dan penulisan kode program. Baik untuk antarmuka maupun program bersifat non visual.

Lingkungan kerja IDE ini menyediakan sarana yang diperlukan untuk merancang, membangun, mencoba, mencari atau melacak kesalahan, serta mendistribusikan aplikasi. Sarana-sarana inilah yang memungkinkan pembuatan prototipe aplikasi menjadi lebih mudah dan waktu yang diperlukan untuk mengembangkan aplikasi menjadi lebih singkat.

### 2.5.1 Struktur Standar Project Delphi

Sebuah project dalam Delphi selalu tersimpan dalam file berekstensi DPR (Delphi Project). Minimal sebuah file ini harus ada agar sebuah project dapat dikompilasi.

Sebagai salah satu development tools pada lingkungan Windows, Delphi menyediakan sarana bagi programmer untuk dapat mengembangkan aplikasi maupun sebuah modul dalam shared library (Dynamic Linking Library).



#### 2.5.1.1 Struktur Project Aplikasi

Struktrur standar sebuah file DPR untuk sebuah aplikasi Delphi adalah:

```
program Project1;

uses
  ...;

begin

  ...

end.
```

#### 2.5.1.2 Struktur Project Shared Library

Untuk sebuah project berupaa modul library, file project Delphi memiliki struktur sebagai berikut:

```
library Project1;

uses
  ...;

begin
 ...
end.
```

### 2.5.2 Object Oriented Dengan Delphi

Sebagai bahasa pemrograman yang telah mendukung pengembangan aplikasi berorientasi object, Delphi menyediakan syntax khusus dalam implementasi pemrograman berorientasi object.

Berikut akan dijelaskan beberapa konsep pemrograman berbasis object pada Delphi.



#### 2.5.2.1 Kelas dan Object

Seperti halnya bahasa berbasis OOP lainnya (termasuk Java dan C#), di dalam Delphi sebuah tipe kelas tidak serta merta diasumsikan sebagai object, namun hanya sebagai referensi terhadap sebuah object dalam memori. Sebelum dapat dipergunakan, sebuah object harus terlebih dahulu dialokasikan dalam memory. Hal ini dilakukan dengan membuat instansiasi dari kelas bersangkutan.

Definisi tipe kelas dalam Delphi:

```
NamaKelas = class
    Var1, Var2, Var3: Integer;
    procedure NamaProsedur (Param1, Param2, Param3: Integer);
    function NamaFungsi: Boolean;
  end;
```

Contoh instansiasi Object dari sebuah Kelas:

```
var
  Obj1 : TMyClass;
begin
  Obj1 := TMyClass.Create;
```

### 2.5.2.2 Enkapsulasi

Dalam konsep OOP enkapsulasi dilakukan untuk memproteksi variable dalam sebuah kelas agar tidak dapat diakses secara langsung. Delphi mengimplementasi konsep enkapsulasi dengan memberikan 3 tipe akses terhadap atribut sebuah kelas, yaitu: Private, Protected, dan Public.

Contoh:

```
type
  TDate = class
  private
    Month, Day, Year: Integer;
  public
    procedure SetValue (y, m, d: Integer); overload;
    procedure SetValue (NewDate: TDateTime); overload;
    function LeapYear: Boolean;
    function GetText: string;
    procedure Increase;
  end;
```



### 2.5.2.3 Konstruktor dan Destruktor

Sebuah kelas akan diinstansiasi menjadi sebuah object dengan menggunakan konstruktor kelas bersangkutan, serta sebuah object dihapus dari daftar alokasi memori dengan destruktor.

Delphi mengimplementasikan konstruktor dan destruktor sebuah kelas dengan menggunakan keyword contructor Create dan destructor Destroy.

Sebagai contoh constructor dan destructor pada Delphi:

```
type
  TDate = class
  private
    Month, Day, Year: Integer;
  public
    constructor Create;
```

```
      destructor Destroy;
      procedure SetValue (y, m, d: Integer); overload;
    procedure SetValue (NewDate: TDateTime); overload;
    function LeapYear: Boolean;
    function GetText: string;
    procedure Increase;
  end;
```

### 2.5.2.4 Inheritance

Delphi mengimplementasikan konsep inheritance sebuah kelas dengan cara sebagai berikut:

```
type
  TDate = class (NamaKelasParent)
  private
    Month, Day, Year: Integer;
  public
    constructor Create;
      destructor Destroy;
      procedure SetValue (y, m, d: Integer); overload;
    procedure SetValue (NewDate: TDateTime); overload;
    function LeapYear: Boolean;
    function GetText: string;
    procedure Increase;
  end;
```