**BAB II**

**2.1. Sistem**

Kata sistem berasal dari bahasa Yunani yaitu “Systema” yang mengandung arti kesatuan atau keseluruhan dari bagian-bagian yang saling berhubungan satu sam lainnya dan memiliki tujuan yang sama. Bahwa sistem adalah sekelompok komponen-komponen sistem yang saling bersatu untuk mencapai tujuan yang sama.



Gambar 2.1. Gambar symbol system operasi

(sumber : ….)

Sistem yang penulis buat dan dirancang ini bertujuan untuk memudahkan dalam menyajikan data yang akurat dan efisien sehingga nyaman dan puas dalam memperoleh informasi dan mempercepat kerja. Sistem ini dibuat untuk membantu mempermudah mendata Penyewa Kamar Hotel.

Dengan menerapkankonsep pengolahan data menjadi informasi melalui sistem yang akan dirancang dan dibuat yaitu untuk membantu operasional data di Penyewaan Kamar Hotel dengan merancang Dfd, Ert, dan Flowchart.

(contoh aplikasi dan sistem penjelasannya)

**2.1.1. Flowchart**

Flowchart adalah suatu bagan dengan simbol-simbol tertentu yang menggambarkan urutan proses secara mendetail dan hubungan antara suatu proses (instruksi) dengan proses lainnya dalam suatu program.

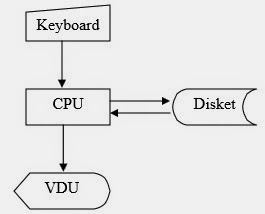
-Tujuan Membuat Flowchart :

* Menggambarkan suatu tahapan penyelesaian masalah
* Secara sederhana, terurai, rapi dan jelas
* Menggunakan simbol-simbol standar

#### **Model Flowchart**

#### Dalam penulisan Flowchart dikenal dua model, yaitu **Sistem Flowchart**dan **Program Flowchart :**

***1. System Flowchart***

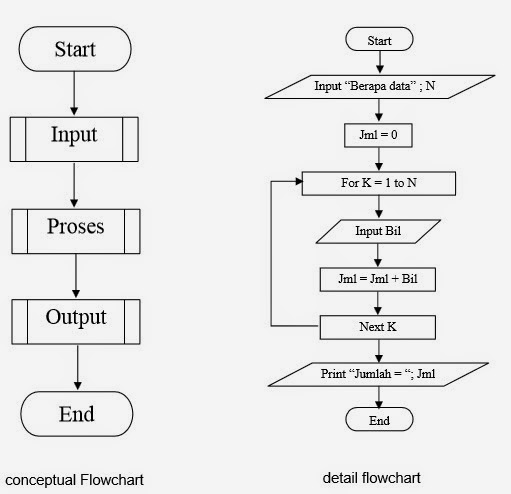
Yaitu bagan yang memperlihatkan urutan prosedure dan proses dari beberapa file di dalam media tertentu.Melalui flowchart ini terlihat jenis media penyimpanan yang dipakai dalam  pengolahan data.

Selain itu juga menggambarkan file yang dipakai sebagai input dan output.Tidak digunakan untuk menggambarkan urutan langkah untuk memecahkan masalah namun hanya untuk menggambarkan prosedur dalam sistem yang dibentuk.

***2. Program Flowchart***

Yaitu bagan yang memperlihatkan  urutan dan hubungan proses dalam suatu  program. Dua jenis metode penggambaran program flowchart :

* **Conceptual flowchart**, menggambarkan alur pemecahan masalah secara global.
* **Detail flowchart**, menggambarkan alur pemecahan masalah secara rinci .

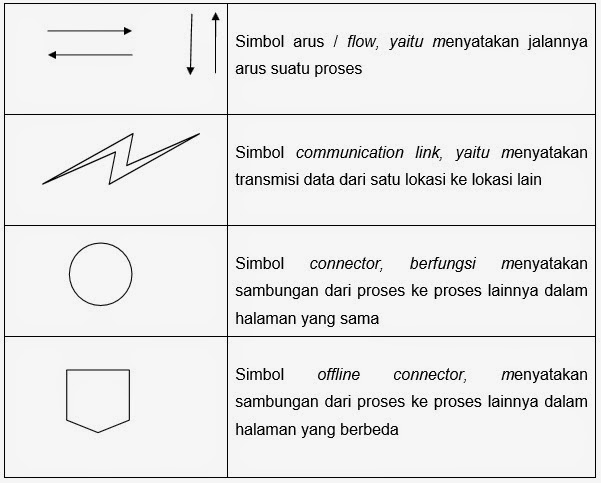
[](http://2.bp.blogspot.com/-lZ3OaiQxAl8/VR5NqLNRzTI/AAAAAAAAAe8/yJzsBffUjN4/s1600/Program+Flowchart.jpg)

#### **Simbol Flowchart**

Simbol-simbol yang di pakai dalam flowchart dibagi menjadi3 kelompok :

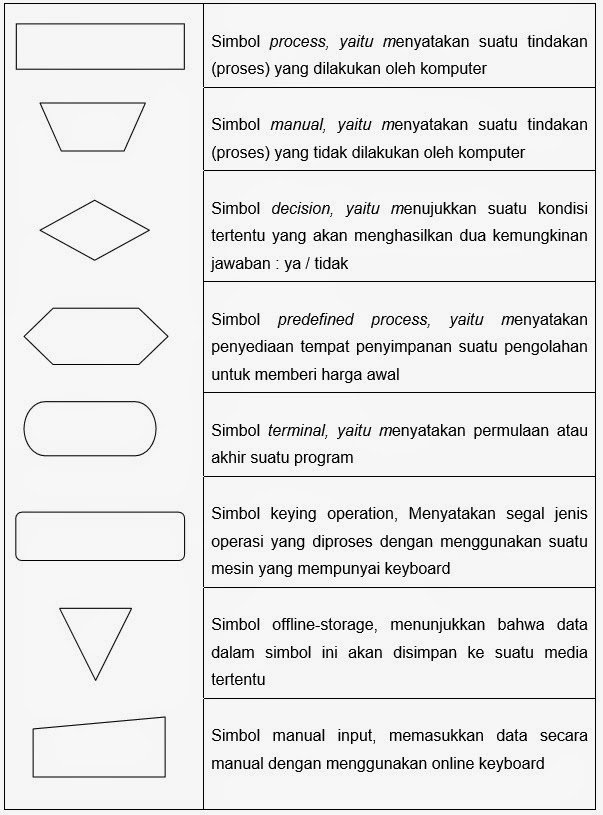
Tabel 2.1. Flow Direction Symbols

(sumber : ….)

[](http://2.bp.blogspot.com/-JHyO3OWi6ac/VR5OT7mY6qI/AAAAAAAAAfc/luXpnrjEyGA/s1600/Flow+Direction+Symbol.jpg)

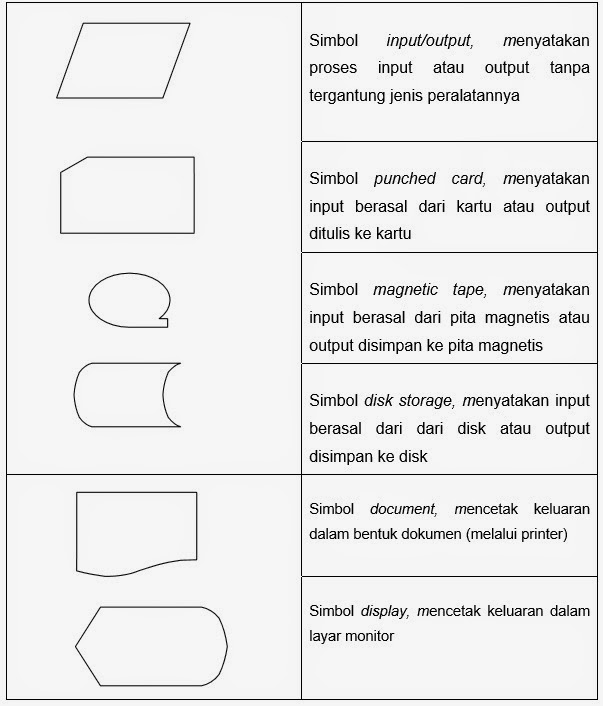
Digunakan untuk menghubungkan simbol satu dengan yang lain – Disebut juga connecting line

**2. Processing symbols**



Menunjukan jenis operasi pengolahan dalam suatu  proses / prosedur Input / Output.

**3. symbols**



Menunjukkan jenis peralatan yang digunakan sebagai media input    atau output.

-*Flowchart terbagi atas lima jenis,yaitu* :

* Flowchart Sistem (System Flowchart)
* Flowchart Flowchart Dokument (Document Flowchart)
* Flowchart Skematik (Schematic Flowchart)
* Flowchart Progam (progam Flowchart)
* Flowchart Proses (process Flowchart)

*-Flowchart Sistem*

Flowchart Sistem merupakan bagan yang menunjukkan alur kerja atau apa yang sedang dikerjakan di dalam sistem secara keseluruhan dan menjelaskan ururtan dari prosedur-prosedur yang ada di dalam sistem.Dengan kata lain,flowchart ini merupakan deskripsi secara grafis dari urutan prosedur-prosedur yang terkombinasi yang membentuk suatu sistem.

-Flowchart Dokumen

Bagan alir dokumen atau disebut juga bagan alir formulir atau paperwork flowchart merupakan bagan alir yang menunjukkan arus dari laporan dan formulir termasuk tembusan-tembusannya.Bagan alir dokument ini menggunkan simbol-simbol yang sama dengan yang digunkan di dalam bagan alir sistem.

-Flowchart Skematik

Bagan alir skematik merupakan bagan alir yang mirip dengan bagan alir sistem,yaitu untuk menggambarkan prosedur di dalam sistem.Perbedaanya adalah ,bagian alir skematik selain menggunakan simbol-simbol bagan alir sistem,juga menggunakan gambar-gambar komputer dan peralatan lainnya yang digunakan.Maksud penggunakan gambar-gambar ini adalah untuk memudahkan komunikasi kepada orang yang kurang paham dengan simbol-simbol bagan alir.Penggunaan gambar-gambar inimemudahkan untuk dipahami,tetapi sulit dan lama menggambarnya.

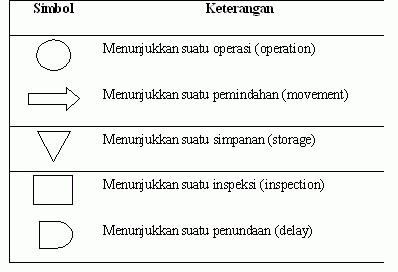
-Flowchart Program

Bagan alir program (program flowchart) merupakan bagan yang menjelaskan secara rinci langkah-langkah dari proses program. Bagan alir program dibuat dari derivikasi bagan alir sistem.

Bagan alir program dapat terdiri dari dua macam, yaitu bagan alir logika program (program logic flowchart) dan bagan alir program komputer terinci (detailed computer program flowchart). Bagan alir logika program digunakan untuk menggambarkan tiap-tiap langkah di dalam program komputer secara logika. Bagan alir logika program ini dipersiapkan oleh analis sistem.

**Flowchart Proses**

Flowchart Proses merupakan teknik penggambaran rekayasa industrial yang memecah dan menganalisis langkah-langkah selanjutnya dalam suatu prosedur atau sistem. Bagan alir proses menggunakan lima buah simbol tersendiri seperti terlihat pada tabel di bawah ini.

[](https://cybernur.files.wordpress.com/2010/12/flowchart-proses.gif)

Flowchart Proses digunakan oleh perekayasa industrial dalam mempelajari dan mengembangkan proses-proses manufacturing. Dalam analisis sistem, flowchart ini digunakan secara efektif untuk menelusuri alur suatu laporan atau form.

### [Jenis flowchart dan perbedaanya](http://redoneelectro.blogspot.com/2014/11/jenis-flowchart-dan-perbedaanya.html)

 Flowchart adalah sekumpulan gambar – gambar tertentu untuk menyatakan alur dari suatu program yang akan diterjemahkan ke salah satu bahasa pemrograman. Kegunaan flowchart sama seperti halnya algoritma yaitu untuk menuliskan alur program tetapi dalam bentuk gambar atau symbol.

Flowchart dibagi menjadi dua (2) bagian, yaitu :  
1. Flowchart yang menggambarkan alur suatu sistem  
2. Flowchart yang menggambarkan alur dari suatu program.

**2.1.2. User interface**

Dilansir dari Wikipedia.com, user interface memiliki pengertian sebagai sebuah bentuk tampilan yang berhubungan langsung dengan pengguna, dan user interface berfungsi untuk menghubungkan antara pengguna [dengan sistem operasi](https://www.nesabamedia.com/pengertian-dan-fungsi-sistem-operasi/) sehingga komputer dapat dioperasikan.

Sementara menurut PCMag.com, user interface dapat diartikan sebagai cara seorang pengguna berinteraksi dengan komputer, tablet, smartphone maupun perangkat elektronik lainnya.

UI atau user interface atau tampilan antarmuka adalah bagaimana cara program dan user (pengguna) berinteraksi. Istilah user interface ini terkadang juga digunakan untuk pengganti istilah HCI (Human Computer Interaction).

User interface adalah semua aspek yang berkaitan erat dengan interaksi / komunikasi antara pengguna dan komputer, khususnya [dalam bab software](https://www.nesabamedia.com/pengertian-software/). Semua aspek yang tampil di layar, dapat dilihat, dibaca, dipahami, dan dimanipulasi dengan keyboard dan tetikus (mouse) merupakan bagian dari user interface.

User interface yang baik adalah user interface yang dapat memberikan pengalaman interaksi yang dengan mudah dapat dipahami oleh user (pengguna), user interface semacam ini disebut sebagai user friendly.

Secara teknis, user interface berguna untuk menjembatani atau menerjemahkan informasi antara user dengen sistem operasi, sehingga komputer dapat didayagunakan oleh user. User interface juga dapat diartikan sebagai mekanisme inter-relasi dari hardware dan software yang membentuk pengalaman menggunakan sebuah set komputer. User inferface dari sisi perangkat lunak biasanya berbentuk GUI (Graphic User Interface) dan CLI (Command Line Interface), sedangkan [dari sisi perangkat keras](https://www.nesabamedia.com/pengertian-dan-macam-macam-perangkat-keras-komputer/) biasanya berbentuk ADB (Apple Desktop Bus), USB, dan Firewire.

**2.1.3. Power designer**

Power Designer adalah salah satuTools yang dapat dipergunakan untuk membangun atau merancang sebuah basisdata melalui ER-diagram, merancang sistem melalui Data Flow Diagram (DFD) serta mampu membuat program aplikasi. Dalam Power Designer, ada beberapa macam pemodelan data yang digunakan untuk perancangan basis data diantaranya pertama, Conceptual Data Model (CDM) adalah model yang dibuat berdasarkan anggapan bahwa dunia nyata terdiri dari koleksi obyek-obyek dasar yang dinamakan entitas (entity) serta hubungan (relationship) antara entitas-entitas itu. Sementara itu, kedua, Physical Data Model (PDM) yaitu model yang menggunakan sejumlah tabel untuk menggambarkan data serta hubungan antara data-data tersebut.

Power designer 6.0 32 bit adalah sebuah paket program desain tools yang digunakan untuk membuat Visualisasi, dokumentasi dan mendesain suatu sistem perangkat lunak. Paket ini meliputi  :

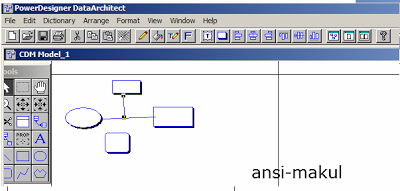
1) Process analyst Model (PAM)

2) Data architect tools

3) Aplication modeler tools

4) Metawork tools

**Gambar 2.3.** gambar contoh (Power Dsigner).



Sumber(<https://halfcoder82.wordpress.com/category/projects/powerdesigner/>)

**A. Process Analyst Model**

Process analyst adalah suatu *Powerdesigner tool* yang memungkinkan untuk menggambarkan aliran informasi dari suatu proses di dalam sistem informasi. Suatu PAM menunjukkan bagaimana data diubah dan menunjukkan proses fungsional yang berlaku untuk data itu.

Produk ini adalah merupakan suatu alat (tools) yang dapat digunakan untuk membuat pemodelan aliran informasi suatu sistem perangkat. Dengan menggunakan produk software ini diharapkan dapat :

* Membangun model fungsional ( OMT) dan diagram arus data
* Menggunakan lambang sesuai dengan berbagai metoda didukung: OMT, Yourdon/Demarco, Gane& Sarson, dan SSADM
* Menciptakan suatu hirarki proses
* Customize dan mencetak laporan model
* Menggunakan data item dari suatu Data Konseptual Model ( CDM)
* Menggunakan OLE teknologi untuk menghubungkan rancangan model  ke aplikasi lain

Suatu PAM memungkinkan untuk:

* Meneliti proses dari suatu sistem
* Menghadirkan proses ini dalam wujud suatu model
* Menciptakan suatu struktur pohon hirarkis [yang] mewakili proses ini

Suatu PAM menggambarkan interaksi dari beberapa object berikut:

**Tabel 2.2.** interaksi dari beberapa objek (PAM).

|  |  |
| --- | --- |
| **Object** | **Keterangan** |
| Data Item | potongan informasi Dasar |
| External Entity | Object aktif yang mengirim atau menerima data dari suatu sistem |
| Process | mentransfer data informasi |
| Data flow | Memindahkan data (transfer data antara proses eksternal entity dan data stores |
| Data store | Tempat storage dimana data disimpan |

**B. Data architect**

Data architect adalah sebuah tools yang digunakan untuk mendesain suatu data base. Tools ini akan memberikan keuntungan yaitu akan dapat memberikan dua tingkat pendekatan dalam desain pemodelan data  yaitu tingkat konseptual atau logikal (CDM ; Conseptual data Model) dan tingkat fisikal (PDM : Physical data Model)

Dengan menggunakan data architect kita akan dapat :

* Membuat Model suatu sistem informasi dengan menggunakan suatu entity- relationship diagram. Model ini disebut  Conceptual Data Model (CDM)
* Menghasilkan suatu Physical Data Model ( PDM) untuk suatu target database management sistem ( DBMS), mempertimbangkan pokok-pokok dari  DBMS
* Mengcustumize PDM untuk menyesuaikan performa tingkah laku dan fisik
* Menghasilkan script database untuk target DBMS
* Menghasilkan suatu referensi integritas jika mereka didukung oleh target database
* meng-customize  dan mencetak laporan model
* Merekayasa balik aplikasi dan database dan aplikasi
* Menggambarkan atribut yang diperluas untuk PDM object

**1. CDM (Conceptual Data Model)**

Suatu CDM menghadirkan keseluruhan struktur logis dari suatu database, yang mana  tidak terikat pada jenis apapun perangkat lunak atau penyimpanan data struktur. Suatu model konseptual sering berisi object data yang belum diterapkan didalam fisik database [Itu] memberi suatu penyajian yang formal menyangkut data yang diperlukan untuk menjalankan perusahaan atau suatu aktivitas bisnis.

CDM mempunyai beberapa peran sebagai berikut :

     Menghadirkan organisasi data di (dalam) suatu format grafis

     Memverifikasi kebenaran disain data

     Menghasilkan PDM yang (mana)  menetapkan implementasi phisik (menyangkut) database

**Suatu CDM menggambarkan interaksi dari beberapa object berikut:**

**Tabel 2.3.** interaksi dari beberapa objek(CDM).

|  |  |
| --- | --- |
| Object | Keterangan |
| Domain | Satuan nilai-nilai Daerah di mana suatu data item sah |
| Data item | potongan informasi Dasar |
| Entity atribute | potongan informasi Dasar yang terberkait dengan suatu entity |
| Entity | Orang Kesatuan, tempat, hal, atau konsep yang mempunyai karakteristik [bunga/minat] kepada perusahaan dan sekitar yang anda inginkan untuk menyimpan informasi |
| Relationship | Nama asosiasi atau koneksi antar entity |
| Inheritance link | hubungan Khusus yang menggambarkan suatu kesatuan sebagai kasus khusus dari suatu kesatuan yang] lebih umum |

**2 PDM (Physical data Model)**

PDM adalah merupakan suatu implementasi phisik dari database. Dengan PDM, kita dapat mempertimbangkan secara detil tentang implementasi phisik nyata. PDM mmemasukkan kedalam laporan perangkat lunak atau penyimpanan data struktur. Kamu dapat memodifikasi PDM untuk menyesuaikan  batasan phisik (physical constrain) atau hasil rancangan

 PDM mempunyai beberapa peran sebagai berikut :

* Menghadirkan organisasi phisik data di (dalam) suatu format grafis
* Menghasilkan catatan untuk modifikasi dan pembuatan database
* Menggambarkan batasan (constrain) dan referensi integritas
* Menghasilkan extended atribut
* Merekayasa balik database yang ada
* Memperbaharui suatu CDM

Ada beberapa jalan untuk membuat suatu PDM:

* Membuat suatu PDM dari suatu CDM
* Membuat suatu PDM sejak dari awal mula
* Mengkonversi dari suatu database ke dalam suatu PDM

Suatu PDM menggambarkan interaksi dari beberapa object berikut :

**Tabel 2.4.** interaksi dari beberapa objek (PDM).

|  |  |
| --- | --- |
| Object | Keterangan |
| Table | Kumpulan dari baris-baris (record)  dan kolom-kolom (fields) |
| colomn | Struktur data yang berisi data item atau attribute didalam suatu baris (record), suatu model persamaan dari kolom database (fileds) |
| Primery Key | colomn yang memiliki nilai unik mengidentifikasikan suatu baris dalam tabel |
| Foriegn key | colomn yang memiliki nilai tergantung dari primary key dalam tabel yang lain (relation tabel) |
| Index | Struktur data yang didasarkan pada suatu kunci (key), didasarkan pada kecepatan akses dan nilai control tertentu |
| Reference | hubungan antara primary key dan foreign key dari tabel yang berbeda |
| View | Struktur data yang merupakan hasil dari query sql dan dibangun dari data di dalam satu atau banyak tabel |

**C. Metaworks**

Metaworks merupakan kamus adalah suatu alat untuk me-manage pengembangan aplikasi di dalam suatu kelompok lingkungan perangkat lunak

Dengan menggunakan Metaworks kamus kita dapat:

      Menciptakan profil dan rekening/tg-jawab untuk para pemakai kamus

      Menciptakan proyek, model dan submodels, dan menugaskan [mereka/nya] ke para pemakai

      Mengunci dan membuka kunci model

      Menyuling/Menyadap dan memperkuat model dan submodels

      Membandingkan definisi obyek dan mengatur konflik data

      Membuat kamus query

      Memodifikasi object di (dalam) kamus

Apa yang merupakan tujuan dari suatu kamus?Bagian ini menjelaskan bagaimana  kamus Metaworks dapat membantu kamu untuk mengembangkan aplikasi.

**1.**    **Memungkinkan kerjasama sekelompok**

Manakala mengembang;kan aplikasi yang besar diperlukan pembagian dan organisasi tenaga kerja. Hal ini dimungkinkan untuk menyimpan atau memelihara aplikasi yang besar seghingga  dapat dikendalikan dalam  dunia waktu nyata. Para pengembang dapat  Berbagi informasi, membuat aplikasi  secara bersama-sama.

**2.**    **Memastikan integritas data**

Team perancang memberikan permasalahan yang serupa kepada user lain yang jumpai di (dalam) multiuser database. Para perancang mungkin punya daerah tanggung jawab yang tumpang-tindih dan boleh mengakses data yang sama secara serempak, hal ini akan dapatmembahayakan integritas data. Oleh karena itu perlu komunikasi dan kerjasama secara saksama dan kontinyu antar anggota.

Kamus Metaworks membantu pengembang  memastikan integritas data selagi sejumlah besar para pemakai mengijinkan untuk mengakses data yang sama.

**3.**    **Memastikan konsistensi**

Yang tidak kalah penting untuk team perancang aplikasi adalah pemusatan data. Dengan ituteam perancang dapat menggunakan kembali bagian-bagian dari model, adalah mungkin untuk membangun aplikasi tambahan selagi pemeliharaan model tetap (konsisten). Sehingga keseluruhan proses pengembangan akan menjadi  lebih efisien. Kamus  Metaworks memusatkan data disain untuk semua sistem informasi aplikasi dan dapat memastikan kecocokan dan konsistensi dari  aplikasi yang berbeda .

**4.**    **Memungkinkan data untuk bersama**

Di dalam banyak organisasi, aplikasi jarang beroperasi dengan bebas dari satu/ orang ke orang lain. Operasi mandiri akan bisa dipastikan di dalam data inconsistencies dan akan mendorong kearah suatu gangguan di  dalam sistem informasi. Kamus  Metaworks membantu team perancang mengatur definisi daerah dan data. [kamus metaworks akan menyimpan semua data di dalam sistem informasi, dan mungkin dapat digunakan  untuk proyek yang berbeda. Hal ini memungkinkan semua para perancang, mengikuti hak akses yang sudah dikenal mereka

**2.1.4. Bahasa Pemograman JAVA**

Bahasa Pemrograman atau disebut juga dengan bahasa komputer merupakan sebuah intruksi standart untuk memerintah komputer agar bisa menjalankan fungsi tertentu. Seperti yang sudah disinggung oleh penulis di atas, bahasa pemrograman merupakan suatu himpunan dari aturan sintaks dan semantic yang dipakai untuk mendefinisikan program komputer. seseorang yang bisa memahami bahasa pemrograman dapat menentukan data yang akan disimpan atau diteruskan,data mana yang akan diolah, dan langkah-langkah yang diambil dalam berbagai situasi. Bahasa pemrograman yang akan dipakai penulis adalah *Java*.

 Gambar 2.2 Logo Java

*Java* merupakan bahasa bahasa pemrograman tingkat tinggi yang berorientasi pada *object* dan program *java* tersusun dari bagian yang disebut dengan *Class*. *Class* terdiri dari metode-metode yang melakukan pekerjaan dan mengembalikan informasi setelah melakukan tugasnya. Para pengguna *Java* banyak mengambil keuntungan dari kumpulan *class* di pustaka *class Java* yang disebut dengan *Java Application Programming Interface (API)*. Class- class ini diorganisasikan menjadi sekelompok yang disebut dengan paket (*package*). *Java API* telah menciptakan applet dan aplikasi canggih dengan menyediakan fungsionalitas yang memadai.

Bahasa ini awalnya dibuat oleh James Gosling saat masih bergabung di *Sun Microsystem* yang saat ini merupakan bagian dari *Oracle* dan dirilis tahun 1995. Aplikasi-aplikasi berbasis *java* umumnya dikompilasi ke dalam p-code (*bytecode*) dan dapat dijalankan pada berbagai Mesin Virtual Java (JVM). Bersifat umum/non-spesifik(*general purpose*), dan secara khusus didesain untuk memanfaatkan dependensi implementasi seminimal mungkin. Karena fungsionalitasnya yang memungkinkan aplikasi *java* mampu berjalan di beberapa platform sistem operasi yang berbeda. Pada Java program *javac* untuk mengkompilasi file kode sumber *Java* menjadi class-class *bytecode*. File kode sumber mempunyai ekstensi \*.java. Kompilator *javac* menghasilkan file *bytecode* class dengan ekstensi \*.class. *Interpreter* adalah modul utama pada sistem *Java* yang digunakan aplikasi *Java* dan menjalankan program *bytecode Java*.

Dengan kata lain *Java* adalah bahasa pemrograman yang dapat membuat seluruh bentuk aplikasi tidak hanya *desktop* dan *web* namun juga bisa membuat aplikasi *mobile* dan lainnya, sebagaimana dibuat dengan menggunakan bahasa pemrograman konvensional yang lain. Bahasa Pemrograman *Java* berorientasi object (*OOP-Object Oriented Programming*), dan dapat dijalankan pada berbagai platform sistem operasi. Pada *OOP*, program komputer sebagai kelompok *object* yang saling berinteraksi. Perkembangan *Java* tidak hanya terfokus pada satu sistem operasi, namun dikembangkan untuk berbagai sistem operasi dan bersifat *open source*. Bahasa pemrograman *java* banyak mengadopsi sintaksis yang terdapat pada *C* dan *C++* namun dengan sintaksis model object yang lebih sederhana.

Pengertian OOP secara singkat adalah mengorganisasikan program sebagai kumpulan komponen yang disebut *object*. Object- object ini ada secara independen, mempunyai aturan-aturan berkomunikasi dengan *object* lain dan untuk memerintahkan *object* lain gunanya untuk meminta informasi tertentu atau meminta *object* lain mengerjakan sesuatu. Class bertindak sebagai modul sekaligus tipe. Sebagai tipe maka pada saat dijalankan, program menciptakan object-object yang merupakan instan- instan Class. Class dapat mewarisi Class lain. *Java* tidak mengijinkan pewarisan jamak akan tetapi menyelesaikan kebutuhan pewarisan jamak dengan fasilitas antarmuka yang lebih elegan. Beberapa fungsi dari bahasa pemrograman ini adalah menggunakan sistem exception-handling dan daftar Library yang lengkap.

**Struktur Program Java**

Dalam pembuatan program Java yang harus diperhatikan dalam pembuatan program java adalah penulisan huruf besar dan kecil karena java memiliki sifat Case Sensitive. Berikut adalah bentuk umum dari penulisan program Java :

**Pertama** dalam program Java minimal terdapat sebuah class, dimana nama dari class tersebut diusahakan sama dengan nama file Java, dan setiap class harus dibuka dengan tanda „{„ dan ditutup dengan tanda „}‟.

Contoh: class coba {

(isi dari class)

}

**Kedua** dalam program Java terdapat fungsi main() adalah dijadikan sebagai awal pengeksekusian aplikasi Java, kode (code) yang terdapat pada metode inilah yang akan dieksekusi pertama kali.

Contoh: class coba

{

public static void main(String[] args)

{

(tulis code/ program disini)

}

}

Metode main () didefinisikan sebagai public static void, berikut penjelasannya :

public, berarti metode ini dapat dipanggil dari luar class

static, menunjukkan metode ini bersifat sama untuk semua class

void, berarti metode ini tidak mengembalikan nilai.

Argument args [] adalah array objek string argument baris-baris perintah yang dilewatkan ke kelas yang di eksekusi.

**Contoh Program Sederhana JAVA**

// nama file : Hello.java

class Hello {

public static void main(String[] args)

{

System.out.println(“Hello, World!”);

}

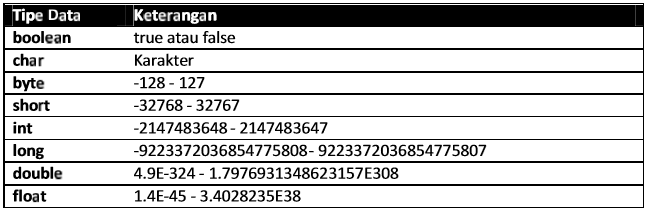
}

Pada program di atas, langkah pertama yang kita lakukan yaitu membuat class Java dengan nama ProgramPertama. Class tersebut memiliki fungsi main() dengan parameter variabel args dengan tipe array dari String. Fungsi main ini yang akan mengeksekusi statement System.out.println (“Hello World!”); untuk menampilkan kata Hello World!

**1. Tipe data**

Terdapat beberapa tipe data primitive yang ada Di Java yaitu :

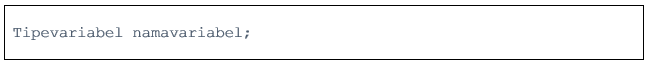
**Tabel 2.5.** contoh tipe data.



**2. Variabel**

Variabel merupakan sesuatu yang digunakan untuk menampung, sebuah data. Sebuah variabel harus ada dalam sebuah kelas atau metode.Pembuatan sebuah variabel di java terlihat pada kode dibawah ini.

**Tabel 2.6.** contoh Variabel.



Syarat-syarat penamaan variabel adalah :

1). Harus diawali dengan huruf.

2). Tidak boleh terdapat karakter unik seperti @,#,% dan lain-lain.

3). Tidak boleh mengandung karakter putih (spasi,enter,tab).

**3. Pemrograman Berorientasi Objek**

Pemrograman berorientasi objek merupakan pemrograman yang menjadikan objek sebagai komponen utama dalam sistem. Objek merupakan gabungan data dan fungsi, dimana sebuah objek dibuat dari sebuah kelas.

**a. Object**

Objek merupakan hasil dari sebuah kelas. Untuk membuat objek dalam java diperlukan sebuah perintah new, dimana cara pembuatannya sama dengan pembuatan variabel.

**Tabel 2.7.** contoh Object



**b). Class**

Dalam Java, kelas didefinisikan menggunakan kata kunci class. Contoh kelas sederhana adalah sebagai berikut :

**Tabel 2.8.** contoh Class.



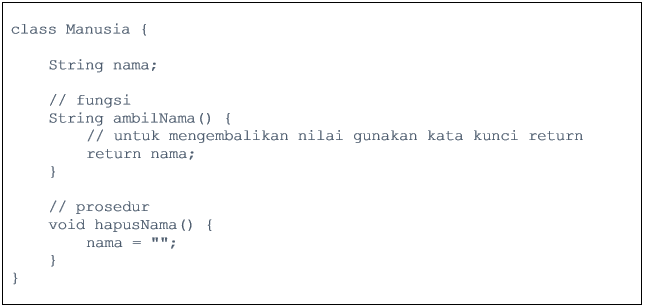
**c). Metode**

Dlam Java terdapat dua buah metode

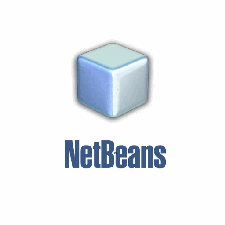
1. Fungsi, merupakan metode yang memiliki nilai balik jika metode tersebut dipanggil,cara pembuatan sebuah fungsi adalah dengan cara menentukan nilai baliknya, lalu membuat nama metodenya.

2. Prosedur, merupakan metode yang tidak memiliki nilai balik, cara pembuatan prosedur sama dengan fungsi namun bedanya, nilai baliknya menggunakan kata kunci void.

**Tabel 2.9.** contoh metode.



**2.1.4.1. Netbeans**

****

Sumber(https:// <https://www.pramudito.com/netbeans>)

NetBeans adalah suatu serambi pengembangan perangkat lunak yang ditulis dalam bahasa pemrograman Java. Serambi Pada NetBeans, pengembangan suatu aplikasi dapat dilakukan dimulai dari setelan perangkat lunak modular bernama modules. Semula, aplikasi NetBeans IDE ini diperuntukkan bagi pengembangan dalam Java. Namun, aplikasi ini juga mendukung program-program pembuatan bahasa lain secara khusus seperti PHP, C/C++ dan HTML5.

NetBeans adalah alat lintas serambi serta penerapannya dijalankan pada Microsoft Windows, Mac OS X, Linux, Solaris dan serambi-serambi lainnya yang mendukung JVM yang sepadan.

Awal kemunculan aplikasi NetBeans telah ada sejak tahun 1997 yaitu sebagai sebuah proyek kuliah. Pada tahun tersebut, suatu perusahaan dibangun oleh Roman Stanek di sekitar proyek kuliah tersebut lalu perusahaan tersebut memulai memproduksi versi NetBeans IDE komersial hingga akhirnya dibeli oleh Sun Microsystems pada tahun 1999 lalu menjadikan NetBeans IDE sebagai serambi bersifat sumber terbuka pada bulan Juni 1999.

Cara menjalankan NetBeans:

1. Buka Aplikasi NetBeans
2. Pilih Projek yang akan anda Run dengan cara buka menu ‘File’ kemudian Open Project atau klik CTRL+SHIFT+O, pilih penyimpanan file, dan pilih file.
3. Kemudian pada menu disibelah kiri klik menu ‘Project’ dan klik project anda, lalu pilih source package dan klik package anda, kemudian klik kanan dan pilih run, atau anda bisa dengan klik CTRL+SHIFT+F6 atau klik CTRL+F6 saja.

**Fitur fitur yang terdapat dalam netbeans antara lain:**

Smart Code Completion: untuk mengusulkan nama variabel dari suatu tipe, melengkapi keyword dan mengusulkan tipe parameter dari sebuah method.

**Bookmarking:** fitur yang digunakan untuk menandai baris yang suatu saat hendak kita modifikasi.

**Go to commands:** fitur yang digunakan untuk jump ke deklarasi variabel, source code atau file yang ada pada project yang sama.

**Code generator:** jika kita menggunakan fitur ini kita dapat meng-generate constructor, setter and getter method dan yang lainnya.

**Error stripe:** fitur yang akan menandai baris yang eror dengan memberi highlight merah.

**I. Pengertian**

Pada dasarnya, *Exception* merupakan subkelas dari kelas *java*.*lang*.*Throwable*.  “*Bukalah dokumentasi java untuk lebih menyakinkan anda*”. Karena *Exception* adalah sebuah kelas maka hakikatnya ketika program berjalan dan muncul sebuah bug atau kesalahan maka *bug* tersebut dapat dianggap sebuah *object*.  Sehingga ketika *object* ini di tampilkan di layar maka java akan secara otomatis memanggil method *toString* yang terdapat dalam *object* bertipe *Exception* ini. Java memberikan akses kepada *developer* untuk mengambil *object* bug yang terjadi ini dengan mekanisme yang dikenal *Exception* *Handling*. *Exception* *handling* merupakan fasilitas di java yang memberikan flexibilitas kepada *developer* untuk menangkap *bug* atau kesalahan yang terjadi ketika program berjalan. Contoh *Exception* *Handling* akan dibahas pada bagian berikutnya.

**II. Perbedaan antara *Class* *Error* dan *Class* *Exception* di java**

Seperti yang telah dijelaskan diatas bahwa kelas *Exception* merupakan kelas turunan dari kelas *Throwable* di *package* *Java*.*Lang*. Selain *Exception*, java.*lang*.*Throwable* juga memiliki *subclass* yaitu *class* *Error*. Tentu, kita bertanya-tanya, sebetulnya apa sih perbedaan antara *class* *Error* dengan *class* *Exception*.

**Penjelasan dari *Class* *Error* :**

*“An Error is a subclass of Throwable that indicates serious problems that a reasonable application should not try to catch. Most such errors are abnormal conditions”* (JDK 5.0 *Documentation*)

*Error* dapat saja terjadi dalam pemrograman. Secara garis besarnya, error dalam pemrograman Java ini ada beberapa jenis, yaitu :

* *Syntax error*
* *Runtime error*
* *Logic error*

Untuk masing-masing penjelasannya bisa di lihat di bawah ini.  
  
**A. *Syntax error***

Apa itu *syntax*?  
  
*Syntax* merupakan suatu aturan penulisan yang sudah ditetapkan pada struktur elemen - elemen dalam bahasa pemrograman.  
  
Jadi *syntax error*ini adalah kesalahan dalam *coding* karena aturan penulisan yang tidak sesuai atau kesalahan pada konstruksi kode, misalnya :

* Salah dalam menuliskan *keyword* Java
* Tidak menggunakan tanda kurung kurawal untuk pernyataan
* Tidak menggunakan tanda atau karakter yang sesuai, misalnya tidak menggunakan tanda " ", untuk tipe data String.

Syntax error ini mudah ditelusuri atau ditemukan karena *compiler* akan memberi tahu kita di mana letak kesalahan dalam penulisan kode program.  
  
Sebagai contoh :

[?](http://www.bahasajava.com/2015/10/kenali-jenis-error-pada-pemrograman-java.html)

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9 | public class SelamatDatang {        public static void main(String args []){          //Seharusnya "Selamat datang di pemrograman Java"          System.out.println(Selamat datang di pemrograman Java");        }    } |

Kode di atas memiliki *error*, yaitu seharusnya teks Selamat datang di pemrograman Java berada dalam tanda " ".  
  
Oleh karena itu, setelah anda melakukan *compile*, maka *compiler* akan memberi tahu *error* tersebut seperti gambar di bawah ini :  
  
a) Syntax error pada *command prompt*

|  |
| --- |
| [Kenali 3 Macam Error Pada Pemrograman Java](https://2.bp.blogspot.com/-E2hPo5yB28o/V4MPzH5sizI/AAAAAAAAAPk/Z60PZxmu2q4GeBZX8pt6f3P7OBysvuVRQCLcB/s1600/Java+Syntax+Error.jpg) |
| Syntax Error Pada Command Prompt |

b). Syntax error pada IDE Netbeans

|  |
| --- |
| [Syntax error pada Netbeans](https://1.bp.blogspot.com/-YAQ9EqeH02w/V4MkQXSUbNI/AAAAAAAAAQA/S7j2EKYq46YCSvANUYdxpYwCbyNJgAkpQCLcB/s1600/Syntax+error+netbeans.jpg) |
| Syntax error pada IDE Netbeans |

c). Syntax error pada IDE Eclipse

|  |
| --- |
| [Syntax error pada Eclipse](https://4.bp.blogspot.com/-1iHmp-CFOOM/V4MmOov0lmI/AAAAAAAAAQQ/nrgWo-5SCEEB0lQg2EPkeZZ75HJNV5pbgCLcB/s1600/Syntax+error+eclipse.jpg) |
| Syntax error pada IDE Eclipse |

**B. *Runtime Error***

*Runtime error* merupakan salah satu jenis *error* yang terjadi karena program secara tiba-tiba dihentikan ketika program tersebut sedang berjalan.  
  
*Runtime error* ini dapat terjadi bila program tidak bisa menangani operasi yang tidak mungkin untuk dilakukan.  
  
Contohnya *runtime error*ini dapat terjadi karena :

* Ketika program meminta *user* untuk menginputkan angka, namun *user* malah menginputkan huruf
* Ketika melakukan perhitungan pembagian angka 1 dengan 0

Sebagai contoh:

[?](http://www.bahasajava.com/2015/10/kenali-jenis-error-pada-pemrograman-java.html)

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8 | public class CekError {        public static void main(String args []){          //membagi angka 1 dengan angka 0          System.out.println( 1 / 0 );        }    } |

Pada program di atas akan menampilkan pesan *error* karena program di minta untuk membagi angka 1 dengan angka nol. Pada *command prompt*akan menampilkan pesan *error* seperti di bawah ini :

|  |
| --- |
| [Runtime error pada pemrograman Java](https://1.bp.blogspot.com/-qMl7KOp1dGE/V4M7N-meJ0I/AAAAAAAAAQg/4xBoSmc7gm0TBOufRDM7Pd-aNWIBlBwZgCLcB/s1600/Run+time+error+cmd+prompt.jpg) |
| *Runtime error* pada *command prompt* pemrograman Java |

Demikian pula jika kode diatas di jalankan pada IDE Netbeans ataupun Eclipse, maka *compiler* akan memberikan pesan dan tanda *runtime error*yang terjadi.  
  
**C. Logic error**

Logic error terjadi bila program berjalan dan tidak memberikan hasil seperti yang diinginkan.  
  
Misalnya anda membuat program konversi dari nilai centimeter ke nilai milimeter.  
  
Seperti sudah diketahui bahwa :  
  
1 cm = 10 mm  
  
Contoh program  sederhananya :

[?](http://www.bahasajava.com/2015/10/kenali-jenis-error-pada-pemrograman-java.html)

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11 | public class Konversi {        public static void main(String args []){            System.out.println("1 cm itu sama dengan ");          //Konversi cm ke mm          System.out.println( 1.0 / 10.0 + "mm");        }    } |

Setelah mengeksekusi program di atas, maka akan terlihat pada command prompt seperti di bawah ini:

|  |
| --- |
| [Logic error pada pemrograman Java](https://2.bp.blogspot.com/-CToKhe__Gg4/V4M_Do5ilaI/AAAAAAAAAQs/0BXaS72n6FIdXeu9c6BA8qDKTWr55i8LACLcB/s1600/Logic+error.jpg) |
| *Logic error* pada *command prompt* pemrograman Java |

Program di atas berjalan dengan mulus ketika dieksekusi baik ketika menggunakan IDE Netbeans ataupun Eclipse, namun bila diperhatikan terlihat nilai bahwa 1 cm itu sama dengan 0.1 mm, tentu saja ini adalah hal yang salah, dan inilah yang disebut dengan logic error.  
  
**Kesimpulan dari *error* pada pemrograman Java**

Jadi kesimpulan dari *error* yang mungkin dapat pada pemrograman Java terjadi adalah bahwa *logic error*adalah *error* yang paling menantang bagi para programmer untuk di atasi, sementara itu *syntax error* dan *runtime error* relatif lebih mudah ditangani karena  *compiler* akan memberitahu pada bagian mana *error*tersebut terjadi.  
Semoga pembahasan mengenai jenis *error* yang mungkin terjadi pada pemrograman Java ini dapat membantu para pemula dalam mempelajari Java.

**Penjelasan dari *class* *Exception* :**

“The class *Exception* and its subclasses are a form of *Throwable* that indicates conditions that a reasonable application might want to catch. “ (JDK 5.0 *Documentation*)

Seperti dari penjelasan yang diberikan oleh *JDK* *Documentation*, maka dapat kita lihat bahwa *error* dan *exception* pada dasarnya berbeda. *Error* merupakan masalah yang muncul tapi tidak ada alasan yang kuat untuk menangkapnya. Sedangkan *Exception* merupakan kesalahan kecil yang muncul dan ingin diperlakukan sesuai keinginan *developer*.

**III. *Keyword* penting pada *exception* *handling***

Ada 5 *keyword* penting dalam java dalam hal *exception* *handling* :

**1. *try***

*Keyword* ini biasanya digunakan dalam suatu *block* *program*. *keyword* ini digunakan untuk mencoba menjalankan *block* *program* kemudian mengenai dimana munculnya kesalahan yang ingin diproses. *Keyword* ini juga harus dipasangkan dengan *keyword* *catch* atau *keyword* *finally* yang akan dibahas pada point kedua dan ketiga.

Contoh penggunaan :

|  |
| --- |
| **public class A**  **{**  **public static void main(String[] args) {**  **try**  **{**  **int a = 1 / 0; // berpotensi untuk menimbulkan kesalahan yaitu**  **// pembagian dengan bilangan 0**  **System.out.println("perintah selanjutnya");**  **}**  **catch (Exception kesalahan)**  **{**  **System.err.println(kesalahan);**  **}**  **}**  **}** |

**Output :**

**java.lang.ArithmeticException: / by zero**

Perhatikan contoh diatas, ada beberapa hal penting yang perlu dilihat. Pertama, *block* *program* yag diyakini menimbulkan kesalahan maka ada di dalam *block* *try* and *catch*.Kedua, kesalahan yang muncul akan dianggap sebagai *object* dan ditangkap *catch* kemudian di *assign* ke *variable* kesalahan dengan tipe *Exception*. Ketiga, perintah setelah munculnya kesalahan pada *block* *try* tidak akan dieksekusi.

**2. catch**

Jika anda sudah melihat contoh *try* maka secara tidak langsung anda sudah memahami kegunaan dari *keyword* ini. Dalam java, keyword *catch* harus dipasangkan dengan *try*. Kegunaan keyword ini adalah menangkap kesalahan atau *bug* yang terjadi dalam *block* *try.* Setelah menangkap kesalahan yang terjadi maka *developer* dapat melakukan hal apapun pada *block* *catch* sesuai keinginan *developer*. Contoh Program :

|  |
| --- |
| **catch(Exception kesalahan)**  **{**  **System.out.println(“mohon maaf, terdapat kesalahan pada program”);**  **//lakukan hal lainnya disini**  **}** |

***Keyword* *catch* juga dapat diletakan berulang-ulang sesuai dengan kebutuhan. Contoh :**

|  |
| --- |
| **public class A**  **{**  **public static void main(String[] args) {**  **try**  **{**  **int a = 1/0;  //berpotensi untuk menimbulkan kesalahan yaitu pembagian dengan bilangan 0**  **System.out.println("perintah selanjutnya");**  **}**  **catch(NullPointerException e)**  **{**  **}**  **catch(ArrayIndexOutOfBoundsException e)**  **{**  **}**  **catch(Exception e)**  **{**  **}**  **}**  **}** |

**3. finally**

*Keyword* ini merupakan *keyword* yang menunjukan bahwa *block* *program* tersebut akan selalu dieksekusi meskipun adanya kesalahan yang muncul atau pun tidak ada. Contoh implementasinya pada program :

|  |
| --- |
| **public class A**  **{**  **public static void main(String[] args) {**  **try**  **{**  **//int a = 1/0;**  **}**  **finally**  **{**  **System.out.println("terima kasih telah menjalankan program");**  **}**  **}**  **}** |

**Output Program diatas:**

**terima kasih telah menjalankan program**

Jika saya lakukan modifikasi program diatas menjadi  :

|  |
| --- |
| **public class A**  **{**  **public static void main(String[] args) {**  **try**  **{**  **int a = 1/0;**  **}**  **catch (Exception e)**  **{**  **System.out.println(“ada kesalahan yang muncul”);**  **}**  **finally**  **{**  **System.out.println(“terima kasih telah menjalankan program”);**  **}**  **}**  **}** |

**Output Program :**

**ada kesalahan yang muncul**

**terima kasih telah menjalankan program**

Perhatikan kedua contoh diatas, *block* *finally* akan selalu dieksekusi meskipun adanya kesalahan atau tidak pada *block* *try*. Berbeda dengan *keyword* ***catch*** *keyword* ***finally*** hanya dapat diletakan 1 kali setelah *keyword* ***try****.*

**4. throw**

*Keyword* ini digunakan untuk melemparkan suatu *bug* yang dibuat secara manual. Contoh program :

|  |
| --- |
| **public class A**  **{**  **public static void main(String[] args) {**  **try**  **{**  **throw new Exception("kesalahan terjadi");**  **}**  **catch(Exception e)**  **{**  **System.out.println(e);**  **}**  **}**  **}** |

**Output Program :**

**java.lang.Exception: kesalahan terjadi**

Seperti yang anda lihat pada program diatas, pada *keyword* ***throw new Exception(“kesalahan terjadi”);*** akan melempar *object* bertipe ***exception*** yang merupakan *subclass* dari *class* ***Exception*** sehingga akan dianggap sebagai suatu kesalahan yang harus ditangkap oleh *keyword* ***catch*.**

Perhatikan contoh berikut ini :

|  |
| --- |
| **public class A**  **{**  **public static void main(String[] args) {**  **try**  **{**  **throw new B(); //cobalah  ganti baris ini dengan à new B();**  **}**  **catch(Exception e)**  **{**  **System.out.println(e);**  **}**  **}**  **}**  **class B extends Exception**  **{**  **B()**  **{**  **}**  **public String toString()**  **{**  **return “object dengan tipe kelas B”;**  **}**  **}** |

**Output Program :**

**object dengan tipe kelas B**

Program diatas telah mendefinisikan suatu kelas B *mengextends* dari kelas *Exception*. Ketika kita melakukan *throw* *new* *B();* maka object dari kelas bertipe B ini akan dianggap kesalahan dan ditangkap oleh *block* *catch*. Sekarang jika anda menghilangkan *keyword* *throw* apa yang terjadi?.

**5. throws**

*Keyword* *throws* digunakan dalam suatu method ~~atau kelas~~ yang mungkin menghasilkan suatu kesalahan sehingga perlu ditangkap errornya. Cara mendefinisikannya dalam method adalah sebagai berikut : *<method modifier> type method-name throws* ***exception-list1, exceptio-list2, …*** *{}.*

Contoh Program :

|  |
| --- |
| **public class A**  **{**  **public static void main(String[] args) {**  **try**  **{**  **f();**  **}**  **catch(Exception e)**  **{**  **System.out.println(e);**  **}**  **}**  **public static void f() throws NullPointerException, ArrayIndexOutOfBoundsException**  **{**  **//implementasi method**  **throw new NullPointerException();**  **//throw new ArrayIndexOutOfBoundsException();**  **}**  **}** |

**Output Program :**

**java.lang.NullPointerException**

Contoh program lainnya :

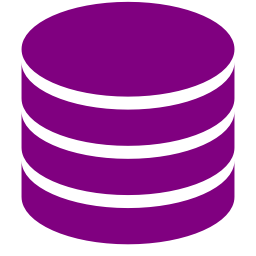
|  |
| --- |
| **public class A**  **{**  **public static void main(String[] args) {**  **try**  **{**  **f();**  **}**  **catch(Exception e)**  **{**  **System.out.println(e);**  **}**  **}**  **public static void f() throws NullPointerException, ArrayIndexOutOfBoundsException**  **{**  **//implementasi method**  **//throw new NullPointerException();**  **throw new ArrayIndexOutOfBoundsException();**  **}**  **}** |

**Output Program :**

**java.lang.ArrayIndexOutOfBoundsException**

Perhatikan kedua contoh pengguaan *keyword* *throws* pada *method*. Ketika *method* tersebut dipanggil dalam *block* *try*. Maka *method* tersebut akan membuat *object* yang merupakan *subclass* dari *class* *Throwable* dan *method* tersebut akan melemparkan kesalahan yang ada dalam *block* *method* kedalam *block* *try*. Di dalam *block* *try*, kesalahan tersebut kemudian ditangkap kedalam *block* *catch*.

**2.2. Sistem Databases**

 Gambar 2.3 Logo Database

Basis data atau yang dikenal dengan *Database* adalah kumpulan data yang disimpan secara sistematis (berurut) di dalam komputer yang dapat mengelola dan permintaan panggilan (*query*) basis data yang disebut sistem manajemen database (*database management system, DBMS*). Sistem perangkat lunak ini memungkinkan pengguna basis data untuk memelihara, mengontrol dan mengakses data secara praktis dan efisien. Konsep dasar dari database adalah kumpulan dari catatan, atau potongan dari pengetahuan.

Tujuan utamanya adalah untuk memberikan tinjauan abstrak data kepada pengguna. Jadi sistem menyembunyikan bagaimana data disimpan, dipelihara dan dapat diakses secara mudah dan efisien. Maksud efisien disini adalah rancangan stuktur data yang kompleks tetapi masih bisa digunakan oleh kebanyakan orang tanpa mengetahui kompleksitas struktur.

1. **Manfaat dari database sendiri adalah:**
2. Kecepatan dan Kemudahan

Database memiliki kemampuan untuk memilih data sehingga menjadi kelompok diurutkan dengan cepat. Inilah yang ahirnya dapat menghasilkan informasi yang dibutuhkan dengan cepat pula. seberapa cepat diolah oleh database juga tergantung pada desain database.

1. **Dapat Digunakan Bersamaan**

Database dapat digunakan oleh siapa saja dalam sebuah perusahaan. Misalnya dalam database siswa perguruan tinggi diperlukan oleh beberapa bagian, seperti admin, keuangan, bagian akademik. Semua bidang ini memerlukan database mahasiswa, tetapi tidak perlu setiap bagian dibuat database itu sendiri, cukup dari database mahasiswa disimpan pada server pusat. Kemudian aplikasi masing-masing bagian dapat dihubungkan ke database siswa.

1. **Kontrol data terpusat**

Terkait dengan menunjuk ke dua, meskipun pada sebuah perusahaan memiliki banyak bagian atau divisi tapi database yang diperlukan tetap menjadi salah satu saja. Ini memfasilitasi data kontrol seperti ketika Anda ingin memperbarui data siswa, maka kita perlu memperbarui semua data dalam setiap bagian atau divisi, tapi cukup dalam satu database yang ada di server pusat.

1. **Perangkat hemat biaya**

Dengan memiliki database terpusat maka dalam setiap divisi tidak memerlukan perangkat untuk menyimpan database karena database hanya diperlukan satu yang disimpan di server pusat, ini akan memotong biaya pembelian perangkat.

1. **Keamanan Data**

Hampir semua sekarang memiliki aplikasi manajemen database fasilitas manajemen pengguna. Manajemen pengguna ini mampu menciptakan hak akses yang berbeda tergantung disesuaikan dengan kepentingan dan posisi pengguna. selain itu data yang disimpan dalam database diperlukan password untuk mengaksesnya.

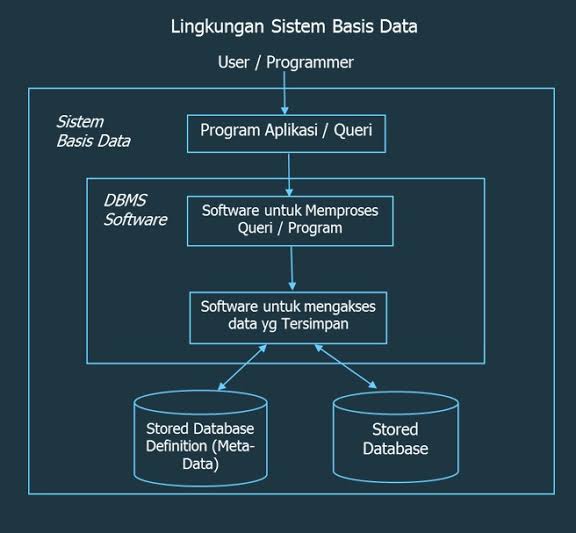
1. **Memfasilitasi pembuatan Aplikasi baru**

Pada titik ini database dirancang dengan sangat baik, sehingga perusahaan membutuhkan aplikasi baru tidak perlu membuat database baru juga, atau tidak perlu mengubah struktur database yang sudah ada. Sehingga pengembang aplikasi atau programmer Si hanya cukup untuk membuat atau antarmuka aplikasi regulasi saja.

Dengan segudang manfaat dan kegunaan yang dimiliki oleh database maka seharusnya semua perusahaan yang baik Para pengusaha kecil terutama perusahaan besar memiliki database dibangun dengan desain yang baik. Ditambah dengan penggunaan teknologi jaringan komputer, manfaat dari database ini akan semakin besar. Penggunaan database di teknologi jaringan komputer yang sama telah banyak digunakan oleh berbagai Perusahaan, misalnya, hanya bank-bank yang memiliki cabang di setiap kota. Bank Perusahaan hanya memiliki database yang disimpan pada server pusat, sedangkan cabang terhubung melalui jaringan komputer untuk mengakses database yang terletak di pusat.

1. **Fungsi Dasar Dari Database**
2. Pengelompokan data, database bertujuan untuk mengklasifikasikan data agar mudah dipahami. Misalnya dalam sistem perpustakaan, ada kelompok data buku, penerbit, transaksi peminjaman, dan mahasiswa.
3. Menghindari duplikasi dan inkonsistensi dalam data.
4. Membuatnya mudah untuk menyimpan, mengakses, dan update, dan menghapus data.
5. Menjamin kualitas data dan informasi yang dapat diakses sesuai dengan dimasukkan (integritas data)
6. Sebuah solusi dalam proses penyimpanan data, terutama data.
7. Mendukung kinerja aplikasi yang memerlukan penyimpanan data.
8. **Jenis-jenis software database**
9. Microsoft Access
10. Oracle
11. Ms SQL Server
12. MySQL
13. Firebird
14. Postgre SQL

**Contoh Sistem database:**



1. **Pengertian dan Perintah-perintah pada database**

Query adalah bahasa standar dalam database untuk mengelola database secara langsung. Dengan query database dapat diakses secara cepat dan mudah dimengerti oleh pengguna atau end – user, query memiliki banyak fungsi yang berkaitan dengan database dan ada beberapa contoh kode yang berbeda – beda dalam tiap kasus yang ada. Query memiliki fungsi-fungsi seperti Melakukan akses langsung pada basis data, query dalam bentuk bahasa SQL (Subquery language) memiliki fungsi membuat basis data baru maupun menghapus yang sudah ada,ketiga query memiliki fungsi menentukan struktur dari database yang ada, baik itu jumlah tabel, kolom, dan lainnya, keempat dapat melakukan manipulasi data dari database yang sudah dapat dibuat, kelima untuk mengeksekusi perintah – perintah untuk mengambil, menambah, menghapus, data serta operasi lainnya di dalam basis data, dan membuat fungsi – fungsi tertentu (store procedure) yang digunakan dalam basisdata.

**2.2.1. DFD**

*Data Flow Diagram* atau disebut *DFD* atau nama lainnya *Context Diagram* merupakan model yang digunakan untuk menjawab sejumlah pertanyaan yang muncul dalam pembuatan *statement of purpose* (deskripsi tekstual fungsi sistem) dan dipresentasikan dengan lingkaran tunggal yang mewakili keseluruhan sistem. *DFD* juga Diagram yang menggunakan notasi-notasi untuk menggambarkan arus dari data sistem atau dikenal dengan nama diagram arus data.

*DFD* digunakan untuk menggambarkan suatu sistem yang telah ada atau baru untuk dikembangkan secara logika tanpa mempertimbangkan lingkungan fisik dimana data tersebut mengalir (seperti lewat telepon, surat dan sebagainya) atau lingkungan fisik dimana data tersebut akan disimpan (misalnya file kartu, *microfiche*, *hard disk*, *tape*, *diskette*).

Tabel 2.1 Simbol DFD

|  |  |
| --- | --- |
| **GAMBAR** | **KETERANGAN** |
| Terminator | Kesatuan Luar (*external entity)* atau batas sistem *(boundary)*  Untuk Entitas. |
|  | Alur data (*data flow)* |
|  | Proses (*Process)* |
|  | Simpanan Data (*Data Store*)  Untuk menyimpan data. |
|  | Pemisah atau Penggabungan Arus Data (*Split/Merge*) |

**2.2.2. ERD**

Tabel 2.2 Simbol ERD

|  |  |
| --- | --- |
| **SIMBOL** | **KETERANGAN** |
|  | Entitas (*Entity)* |
|  | Atribut (*Attribute)* |
|  | Hubungan,Relasi (*Relationship*) |
|  | Garis(Penghubung) |

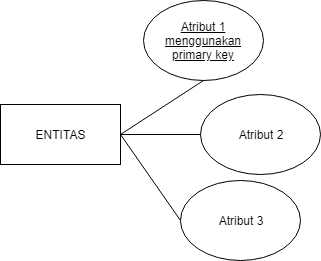
1. Entitas(*Entity*)

Entitas adalah obyek yang dibedakan dan dapat diidentifikasi dalam luingkungan pengguna, dan merupakan suatu yang penting bagi pengguna dalam konteks sistem yang akan dibuat. Simbol *Entity* adalah persegi panjang atau persegi.

1. Atribut

Atribut merupakan atribut dari entitas atau data elemen (*data item*), *data field* yang menggambarkan suatu *entity*. Jenis-jenis atribut:

1. Atribut *Key*, digunakan untuk membedakan semua basis data. Di simbolkan dengan oval dan di bawahnya diberi garis bawah yang menunjukkan *Key*. Misal NPM Mahasiswa, NISN, dll.
2. Atribut *Simple*, atribut yang tidak bisa di pecah. Misal Alamat, penerbit, tahun terbit, judul buku.
3. Atribut *Multivalue*, mempunyai kurang lebih satu *multivalue* nilai dari atribut yang bersangkutan. Misal pengarang buku.
4. Atribut Komposit, terdiri dari atribut yang lebih kecil dan mempunyai arti tertentu yang masih bisa dipecah lagi. Misal dari entitas nama yaitu nama depan,nama tengah, dan nama belakang.
5. Atribut *Derivatif*, atribut yang tidak harus disimpan dalam basis data,dilambangkan dengan bentuk oval dan garis putus-putus. Misal Total.

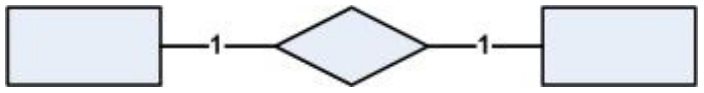


Gambar 2.6 Contoh ERD

Ada tiga jenis hubungan antar *entity* yaitu :

1. *One to One Relationship* (1:1)

Menggambarkan hubungan satu ke satu, yaitu satu *record* pada *entity* pertama berhubungan dengan satu *record* pada *entity* kedua atau sebaliknya.



Gambar 2.7 Entitas one to one

1. *One to Many Relationship* (1:M)

Menggambarkan hubungan satu ke banyak, yaitu satu *record*  pada *entity* pertama berhubungan dengan banyak *record* pada *entity* kedua atau sebaliknya.



Gambar 2.8 Entitas one to many

1. *Many to Many Relationship* (M:N)

Menggambarkan hubungan banyak ke banyak, yaitu lebih dari satu *record* pada *entity* pertama berhubungan dengan lebih dari satu *record* pada *entity* kedua atau sebaliknya.



Gambar 2.9 Entitas many to many

Konsekwensi dari penggunaan hubungan *Many to Many* antar tabel adalah dibutuhkan suatu *associative entity* atau *entity* perantara yang berisi *atribut* atau *field* kunci dari masing-masing *entity.*



Gambar 2.10 Entitas Relasi

**2.2.3. Mysql**

****

Gambar 2.4 Logo MySQL

*MySQL* adalah sebuah perangkat lunak sistem manajemen basis data *SQL* (*database management system*) atau *DBMS* yang *multithread, multi-user*, dengan sekitar 6 juta instalasi di seluruh dunia. *MySQL AB* membuat *MySQL* tersedia sebagai perangkat lunak gratis dibawah lisensi *GNU General Public License (GPL),* tetapi mereka juga menjual dibawah lisensi komersial untuk kasus-kasus dimana penggunaannya tidak cocok dengan penggunaan *GPL. MySQL* adalah *Relational Database Management System (RDBMS)* yang didistribusikan secara gratis dibawah lisensi *GPL (General Public License).* Dimana setiap orang bebas untuk menggunakan *MySQL*, namun tidak boleh dijadikan produk turunan yang bersifat komersial. *MySQL* sebenarnya merupakan turunan salah satu konsep utama dalam *database* sejak lama, yaitu  *SQL (Structured Query Language). SQL* adalah sebuah konsep pengoperasian *database,* terutama untuk pemilihan atau seleksi dan pemasukan data, yang memungkinkan pengoperasian data dikerjakan dengan mudah secara otomatis. Keandalan suatu sistem *database (DBMS)* dapat diketahui dari cara kerja *optimizer*-nya dalam melakukan proses perintahperintah *SQL,* yang dibuat oleh *user* maupun program-program aplikasinya. Sebagai *database server, MySQL* dapat dikatakan lebih unggul dibandingkan *database server* lainnya dalam *query* data. Hal ini terbukti untuk  *query* yang dilakukan oleh *single user*, kecepatan *query MySQL* bisa sepuluh kali lebih cepat dari *PostgreSQL* dan lima kali lebih cepat dibandingkan *Interbase.*

**2.2.3.1. Query**

Query adalah bahasa standar dalam database untuk mengelola database secara langsung. Dengan query database dapat diakses secara cepat dan mudah dimengerti oleh pengguna atau end – user, query memiliki banyak fungsi yang berkaitan dengan database dan ada beberapa contoh kode yang berbeda – beda dalam tiap kasus yang ada. Query memiliki fungsi-fungsi seperti Melakukan akses langsung pada basis data, query dalam bentuk bahasa SQL (Subquery language) memiliki fungsi membuat basis data baru maupun menghapus yang sudah ada,ketiga query memiliki fungsi menentukan struktur dari database yang ada, baik itu jumlah tabel, kolom, dan lainnya, keempat dapat melakukan manipulasi data dari database yang sudah dapat dibuat, kelima untuk mengeksekusi perintah – perintah untuk mengambil, menambah, menghapus, data serta operasi lainnya di dalam basis data, dan membuat fungsi – fungsi tertentu (store procedure) yang digunakan dalam basisdata. Jenis-jenis query seperti DDL(Data Definition Language)contoh: create database, create table, alter table. Kemudian DML (Database Manipulation Language) contoh: SELECT, UPDATE, INSERT INTO.

1. Contoh DDL Query / Query untuk menentukan dan mengubah struktur sebuah database :

* Create Database – membuat sebuah database

CREATE DATABASE db\_name

* Alter Database – mengubah struktur database

**ALTER DATABASE db\_name ADD table\_name**

* Drop database – menghapus sebuah database

**(DROP DATABASE db\_name)**

* Create db – membuat table dalam database

**CREATE TABLE table\_name**

**(**

**Column\_name1 data\_type,**

**Column\_name2 data\_type.**

**---**

**PRIMARY KEY(column\_name))**

* Alter table – memodifikasi table

**ALTER TABLE table\_name ADD column\_name data\_type**

* Drop table – menghapus table

**DROP TABLE table\_name**

1. Contoh DML Query, query untuk mengubah data dari database

* SELECT – mengekstrak/ mengambil data dari sebuah basisdata

**SELECT \* from table\_name**

* UPDATE – mengubah data dari sebuah basisdata

**UPDATE table\_name SET column1=value,column2=value2,.. WHERE some\_column = some\_value**

* DELETE – menghapus data dari database

**Delete From table\_name WHERE some\_column=some\_value**

* INSERT INTO – menambahkan data ke database

**INSERT INTO table\_name VALUES (value1,value2,…).**