**BAB IV**

**LANDASAN TEORI**

* 1. **Konsep dan Prinsip Analisa**

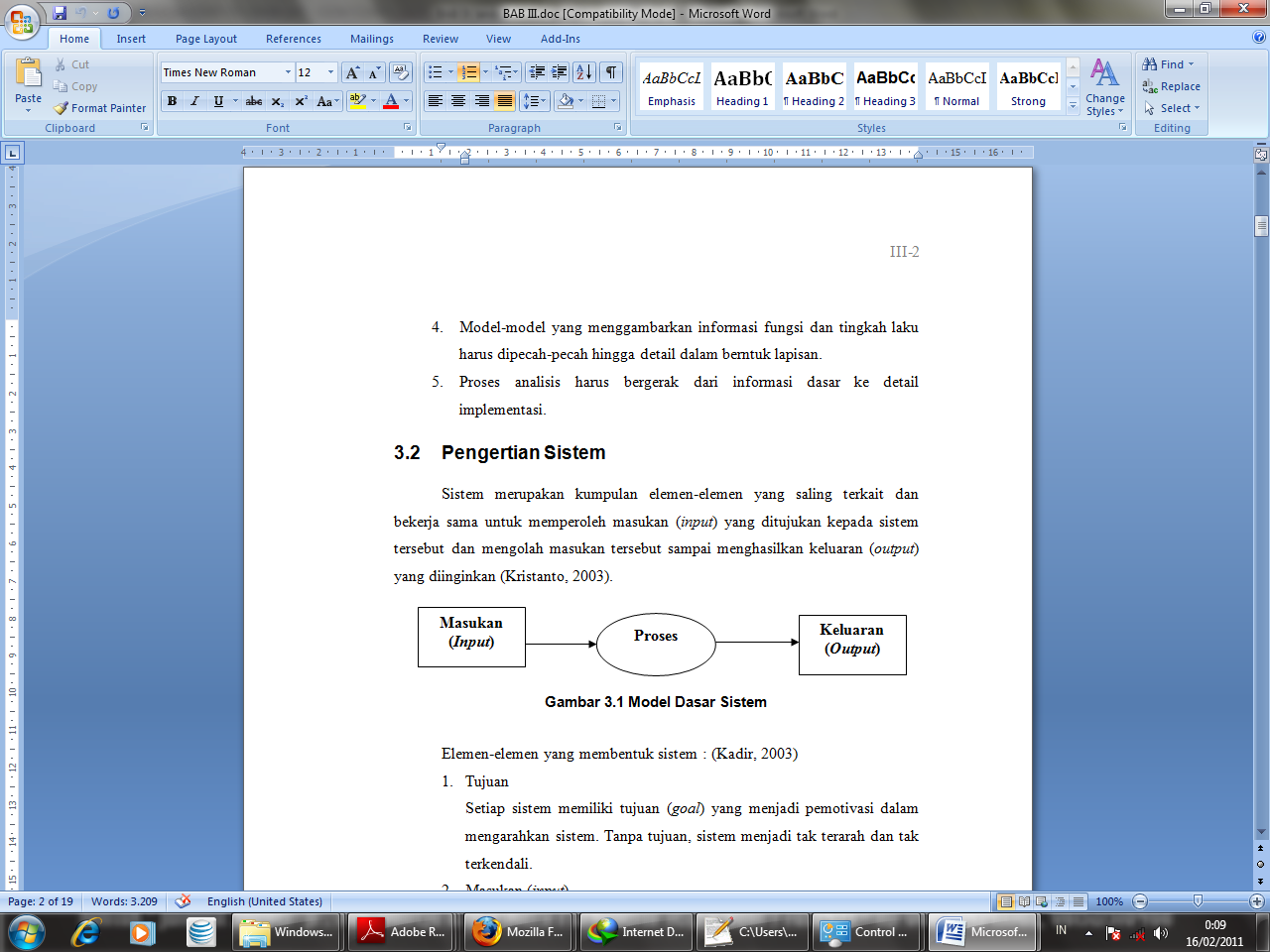
Budgeting adalah suatu proses pengajuan anggaran dana kebutuhan yang diperlukan oleh kantor dari perusahaan berupa laporan yang telah disepakati dari rapat penentuan anggaran disetiap kantornya. Pada era teknologi yang semakin meningkat, pengajuan budgeting yang pada awalnya membuat laporan untuk mengajukan serta mengirim laporan tersebut melalui email ke bagian atasan, sekarang dapat menggunakan sistem yang dapat memudahkan pengajuan hanya dengan menginputkan dana dari kebutuhan yang diperlukan dan akan langsung terkirim kebagian atasan untuk langsung diterima dan di koreksi tanpa mesti mengirim ulang laporan balasan hasil koreksi.

FKTP adalah singkatan dari Fasilitas Kesehatan Tingkat Pertama, fasilitas yang dibutuhkan oleh setiap kantor di PT. Nusa Lima Medika berupa fasilitas yang menyangkut keperluan bagian kesehatan serta bagian pendukung kesehatan lainnya. Data budgeting merupakan data laporan pengajuan dana untuk FKTP (Fasilitas Kesehatan Tingkat Pertama), antara lain: data laporan pengajuan anggaran, laporan penerimaan dana bulanan, laporan selisih dana. Tahap analisis kebutuhan perangkat lunak dalam perancangan sistem informasi budgeting ini merupakan tahapan untuk pembuatan proses kerja baru yang memudahkan proses pengajuan dan pengoreksian data tersebut. Tahapan ini menjadi langkah awal untuk mengetahui bagaimana proses kerja untuk melakukan sistem budged yang dilakukan oleh perusahaan.

Analisa sistem merupakan tahapan awal dalam proses pembangunan sistem, sehingga tahapan ini menjadi acuan pelaksanaan padda proses pengembangan sistem. Penyelesaian sistem menghasilkan solusi dari banyak permasalahaan yang ada, tugas analisis sistem menjadi proses penemuan, perbaikan, pemodelan dan spesifikasi. Setiap model analisa harus dapat mencapai tiga sasaran utama yaitu:

1. Menggambarkan apa yang dibutuhkan oleh pemakai.
2. Membangun dasar bagi pembuatan desain perangkat lunak.
3. Membatasai serangkai persyaratan yang dapat divalidasi begitu perangkat lunak diimplementasikan.
   1. **Pengertian Sistem**

Sistem merupakan komponen atau elemen yang saling terkait dan saling bekerja sama untuk mengolah msukan (*Input*) yang diproses dan menghasilkan keluaran (*Uotput*) yang diinginkan.



**Gambar 4.1 Model Dasar Sistem**

Elemen-elemen yang membentuk sistem : (Kadir, 2003)

1. Tujuan

Setiap sistem memiliki tujuan (*goal*) yang menjadi pemotivasi dalam mengarahkan sistem. Tanpa tujuan, sistem menjadi tak terarah dan tak terkendali.

1. Masukan (*Input*)

Masukan sistem adalah segala sesuatu yang masuk kedalam sistem dan selanjutnya menjadi bahan untuk diproses. Misalnya berupa data transaksi.

1. Proses

Proses merupakan bagian yang melakukan perubahan atau transformasi dari masukan menjadi keluaran yang berguna.

1. Keluaran (*Output*)

Keluaran merupakan hasil dari pemrosesan. Keluaran bisa berupa suatu informasi, saran, cetakan laporan, dan sebagainya.

1. Mekanisme Pengendalian (*Control Mechanism*)

Tujuannya adalah untuk mengatur agar sistem berjalan sesuai dengan tujuan. Dalam bentuk yang sederhana, dilakukan perbandingan antara keluaran sistem dan keluaran yang dikehendaki (standar). Jika terdapat penyimpangan, maka akan dilakukan pengiriman masukan untuk melakukan penyesuaian terhadap proses supaya keluaran berikutnya mendekati standar.

1. Umpan Balik (*Feedback*)

Umpan balik digunakan untuk mengendalikan baik masukan maupun proses.

* 1. **Pengertian Sistem Informasi**

Sistem informasi merupakan sekumpulan subsistem yang saling berhubungan, berkumpul bersama-sama dan membentuk satu kesatuan, saling berinteraksi dan bekerjasama antara bagian satu dengan yang lainnya dengan cara – cara tertentu untuk melakukan fungsi pengolahan data. Salah satu contoh sistem informasi adalah Sistem Informasi Budgeting FKTP (Fasilitas Kesehatan Tingkat Pertama).

Sistem Informasi Budgeting FKTP merupakan sebuah perangkat lunak yang terpadu dan terintegrasi meliputi pengajuan anggaran pertahunnya, pengolahan anggaran yang diajukan, penginputan laporan dana yang di kirim atau di terima, laporan selisih anggaran dari yang diajukan dengan yang telah diberikan.

* 1. **Komponen Sistem**

Didalam komponen sistem terdapat beberapa elemen yang saling mendukung diantaranya:

1. Masukan sistem : Unsur yang dimasukkan ke dalam sistem.
2. Proses sistem : Kegiatan yang dibutuhkan didalam sistem

berjalan dan bertujuan untuk mengubah

masukan menjadi keluaran.

1. Keluaran sistem : Hasil akhir sistem.
2. Umpan balik : Sebagai pengendali.
3. Lingkungan sistem : Elemen di luar sistem.
4. Batas sistem : Batas pemisah sistem dan lingkungannya.
   1. **Model Pengembangan Sistem**

Siklus hidup pengembangan sistem (*System Development Life Cycle*) adalah proses evolusioner yang diikuti dalam menerapkan sistem atau sub sistem informasi berbasis komputer. *SDLC* terdiri atas serangkaian tugas yang erat mengikuti langkah-langkah pendekatan sistem. Karena tugas-tugas tersebut mengikuti suatu pola yang teratur dan dilakukan secara *top-down*, *SDLC* sering disebut sebagai pendekatan air terjun (*waterfall approach*) bagi pengembangan dan penggunaan sistem. Tahapan siklus hidup pengembangan sistem adalah kebijakan dan perencanaan, analisis sistem, perancangan, seleksi, implementasi, dan pemeliharaan.

Model ini bersifat linier karena prosesnya mengalir secara sekuensial mulai dari awal hingga akhir. Model ini mensyaratkan penyelesaian suatu tahap secara tuntas sebelum beranjak pada tahap selanjutnya. Hasil-hasilnya harus didokumentasikan dengan baik. Secara umum kerangka kerja model *Waterfall* adalah sebagai berikut : (Pressman, 1997)



**Gambar 4 .2 Kerangka Kerja Pengembangan Sistem**

Keterangan :

1. Perencanaan

Menyangkut studi kebutuhan pengguna, studi kelayakan baik secara teknis maupun secara teknologi serta penjadwalan pengembangan perangkat lunak. Dapat juga dikatakan sebagai defenisi kebutuhan sistem.

1. Analisa

Tahap dimana kita berusaha mengenali seluruh permasalahan yang muncul pada pengguna (*user*), mengenali komponen-komponen sistem, objek-objek, hubungan antar objek dan sebagainya. Merupakan analisa keadaan internal dan eksternal.

1. Perancangan

Merupakan tahap pencarian solusi dari permasalahan yang didapat dari tahap analisa.

1. Implementasi

Tahap pengimplementasian rancangan sistem kesituasi nyata. Pada tahap ini dimulai proses pemilihan perangkat keras, penyusunan perangkat lunak aplikasi (*coding*) dan pengujian (*testing*) apakah sistem sudah sesuai dengan kebutuhan. Jika belum, dilakukan proses iteratif, yaitu kembali ke tahap-tahap sebelumnya.

1. Pemeliharaan

Mulai melakukan pengoperasian sistem dan melakukan perbaikan -perbaikan kecil jika diperlukan. Jika masa penggunaan sistem habis, maka akan kembali ke tahap pertama, yaitu perencanaan.

* 1. **Elemen Model Analisa**

Perancangan sistem merupakan tahapan yang dilakukan setelah analisa sistem. Perancangan digambarkan sebagain proses multilangkah dimana representasi struktur data, struktur program, karakteristik *interface* dan *detail* *procedure disintesis* dari persyaratan informasi. Perancangan sistem merupakan teknik pemecahan masalah yang saling melengkapi (dengan analisa sistem) yang merangkai kembali bagian – bagian menjadi sebuah sistem yang lengkap, hal ini melibatkan penambahan, penghapusan dan perubahan – perubahan bagian relatif pada sistem.

Pada prinsipnya tahapan desain sistem menekankan pada tahapan teknis untuk mewujudkan sistem dengan lebih nyata. Metode desain sistem yang digunakan pada penellitian ini menggunakan metode berorientasi obyek.

* 1. **Teknik *Object Oriented***

Teknologi *Object Oriented* merupakan paradigma baru dalam rekayasa *software* yang didasarkan pada obyek dan *class* . *Object Oreinted* memandang *software* bagian per bagian, dan menggambarkan satuan bagian tersebut dalam satu obyek. Satu objek dalam satu mode merupakan satu fokus selama proses analisis, desain, dan implementasi dengan menekankan pada *state*, prilaku (*behavior*), dan interaksi objek – objek dalam model tersebut.

*Object Oriented* mencakup bidang aplikasi yang sangat luas. Efek yang ditimbulkan oleh *Object Oriented* dapat dalam bentuk meningkatnya aplikasi *software* yang mudah digunakan dan servis yang lebih fleksibel. Pemodelan dengan menggunakan *Unified Modelling Language* merupakan konsep *object oriented* yang berhubungan dengan *Object Oriented Analysis and Design* (OOAD). Kedua konsep ini merupakan bagian yag sangat penting dalam proses pengembangan sistem yang menggunakan pendekatan *Object Oriented*.(Suhendar dan Gunadi, 2002).

* + 1. **Object Oriented Analysis (OOA)**

Merupakan metode analisis yang memeriksa *requirements* (syarat/ keperluan yang harus dipenuhi oleh sistem) dari sudut pandang *class* – *class* dan objek – objek yang ditemui dalan ruang lingkup permasalahan. Sasaran OOA adalah pengembangan sederetan yag menggambarkan perangkat lunak komputer pada saat perangkat itu bekerja untuk memenuhi serangkai persyaratan yang ditentukan oleh pemakai.

Proses OOA tidak dimulai dengan suatu pemikiran mengenai obyek, melainkan dengan memahami cara sistem akan digunakan oleh manusia atau sistem yang lain.

## **Object Oriented Design (OOD)**

OOD mentranformasi model analisis yang dibuat dengan menggunakan OOA kedalam suatu model desain yang berfungsi sebagai *blue print* perangkat lunak. OOD menghasilkan desain yang mencapai sejumlah atau tingkat yang berbeda dari modularitas. OOD mempunyai sifat unik untuk membangun empat konsep desain perangkat lunak yang penting yaitu: abstraksi, penyembunyian informasi, indepedensi fungsional dan modularitas.

OOD merupakan metode untuk mengarahakan arsitektur *software* yang didasarkan pada manipulasi objek – objek sistem atau subsistem.

## **Komponen – komponen OOAD**

Dalam konsep *Object Oriented* terdapat banyak komponen yang harus diketahui dan dipahami sebelum melakukan proses analisis dan desain. Komponen – komponen yang ada pada OOAD antara lain :

* 1. *Object*

Objek (*Object*) adalah “benda”, secara fisik atau konseptual, yang terdapat disekeliling kita. Sebuah obyek memiliki keadaan sesaat (*state*) dan prilaku (*behavior*). State dari sebuah obyek adalah kondisi objek tersebut atau himpunan dari keadaan yang menggambarkan objek tersebut. Sebagai contoh, salah satu state dari obyek jam adalah waktu saat ini. *State* dinyatakan dengan nilai dari atribut (*atribute*) objeknya*. Atribute* adalah nilai internal suatu objek, kondisi sesaat, koneksi dengan objek yang lain dan identitas. Perubahan *state* dicerminkan oleh perilaku (*behavior*) objek tersebut. Perilaku (*behavior*) suatu obyek mendefinisikan bagaimana suatu objek bertindak dan memberi reaksi. *Behavior* ditentukan oleh himpunan semua atau beberapa operasi yang dapat dilakukan dalam objek sendiri. *Behavior* dari sebuah objek dicerminkan oleh :

* 1. *interface*, adalah pintu untuk mengakses service objek.
  2. *Service*, adalah fungsi yang bisa diemban oleh objek.
  3. *Method*, adalah mekanisme internal objek yang mencerminkan perilaku (*behavior*) objek tersebut.
  4. *Class (Class)*

*Class*  (*class*) adalah definisi umum (pola, tamplate atau cetak biru) untuk himpunan objek sejenis. *Class*  menetapkan spesifikasi perilaku (*behavior*) atau atribut objek – objek tersebut. *Class*  adalah keniskalan atau abterasi dari entitas dalam dunia nyata. Objek adalah “contoh” (*instace*) dari sebuah *class* .

* 1. *Encapsulation*

*Encapsulation* adalah proses menyembunyikan detail implementasi sebuah objek. Untuk dapat berkomunikasi dengan objek diperlukan pesan (*message*). *Message* adalah permintaan untuk obyek penerimaan (*receiver object*) untuk membawa metode yang ditunjukkan atau perilaku dan mengembalikan result dari aksi tersebut kepada obyek pengirim (*sender object*).

* 1. *Association dan Aggregation*

*Association* adalah hubungan antara objek yang saling membutuhkan. *Aggregation* adalah bentuk khusus dari asosiasi yang menggambarkan seluruh bagian dari suatu objek dimana objek tersebut merupakan bagian dari objek lainnya.

## **Unified Modelling Language (UML)**

*UML* merupakan suatu bahasa semi formal dengan spesifikasi semantik yang meliputi sintak secara abstrak, aturan – aturan dasar dan semantik dinamis. *UML* dapat digunakan untuk menerangkan sistem yang berorientasi pada obyek secara lebih jelas dan detail jika dibandingkan dengan menggunakan kode bahasa pemrograman. *UML* biasanya disajikan dalam bentuk diagram atau gambar yang meliputi *class* beserta atribut dan operasinya, serta hubungan antar *class*  yang meliputi *inherintance, association*, dan komposisi.

Pada dasarnya *UML* dikembangkan dengan tujuan – tujuan sebagai berikut:

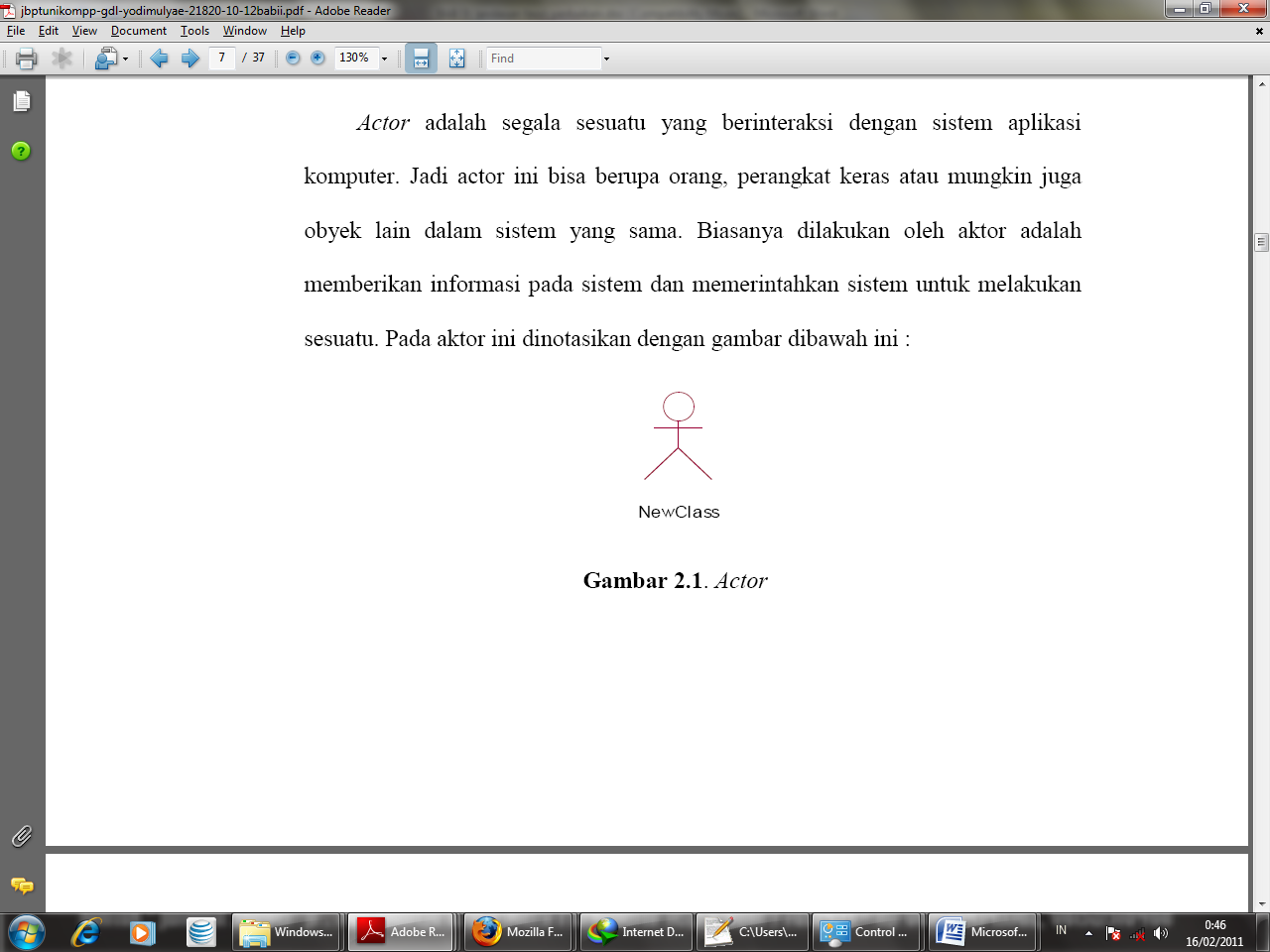
* + - 1. Untuk menyediakan pemodelan visual bagi pengguna, dimana model ini siap digunakan dan dapat saling bertukar model obyek yang dapat dimengerti.
      2. Agar konsep dasarnya dapat dikembangkan dengan mudah dan dibuat lebih spesifik.
      3. Tidak tergantung pada bahasa pemrograman dan proses pengembangan.
      4. Menyediakan dasar yang formal dengan aturan – aturan dasar didalam membuat diagram guna memodelkan suatu objek.
      5. Untuk mendorong pertumbuhan alat bantu yang berorientasi objek sehingga diharapkan akan muncul cukup banyak alat bantu yang memudahkan pengembangan software berorientasi obyek untuk menerapkan konsep *object oriented* dengan mudah.
      6. Memudahkan pengembangan konsep dasar tanpa harus terpaku pada kode dalam bahasa pemrograman.
      7. Memudahkan penggabungan beberapa abstraksi, sudut pandang, teknologi untuk membuat suatu sistem dari permasalahan yang dihadapi menjadi suatu sistem yang lebih kompleks.
      8. Dengan *UML* diharapkan dapat menggambarkan keseluruhan sistem yang hendak dikembangkan dengan benar dan tepat tanpa harus mengurangi maksud dan informasi yang terkandung didalamnya.
      9. Memudahkan untuk memodelkan suatu masalah dari berbagai sudut pandang.
      10. Memudahkan jika nantinya hendak melakukan pembuatan atau pengembangan dari obyek yang lebih selesai diimplemantasikan.
      11. Dapat digunakan sebagai alat untuk mendokumentasikan obyek – obyek yang telah dikembangkan sehingga mudah untuk dibaca dan dimengerti oleh pihak lainnya.

## **Notasi UML**

Adapun notasi yang dapat digunakan dalam *Unifief Modelling Language* (*UML*) antara lain :

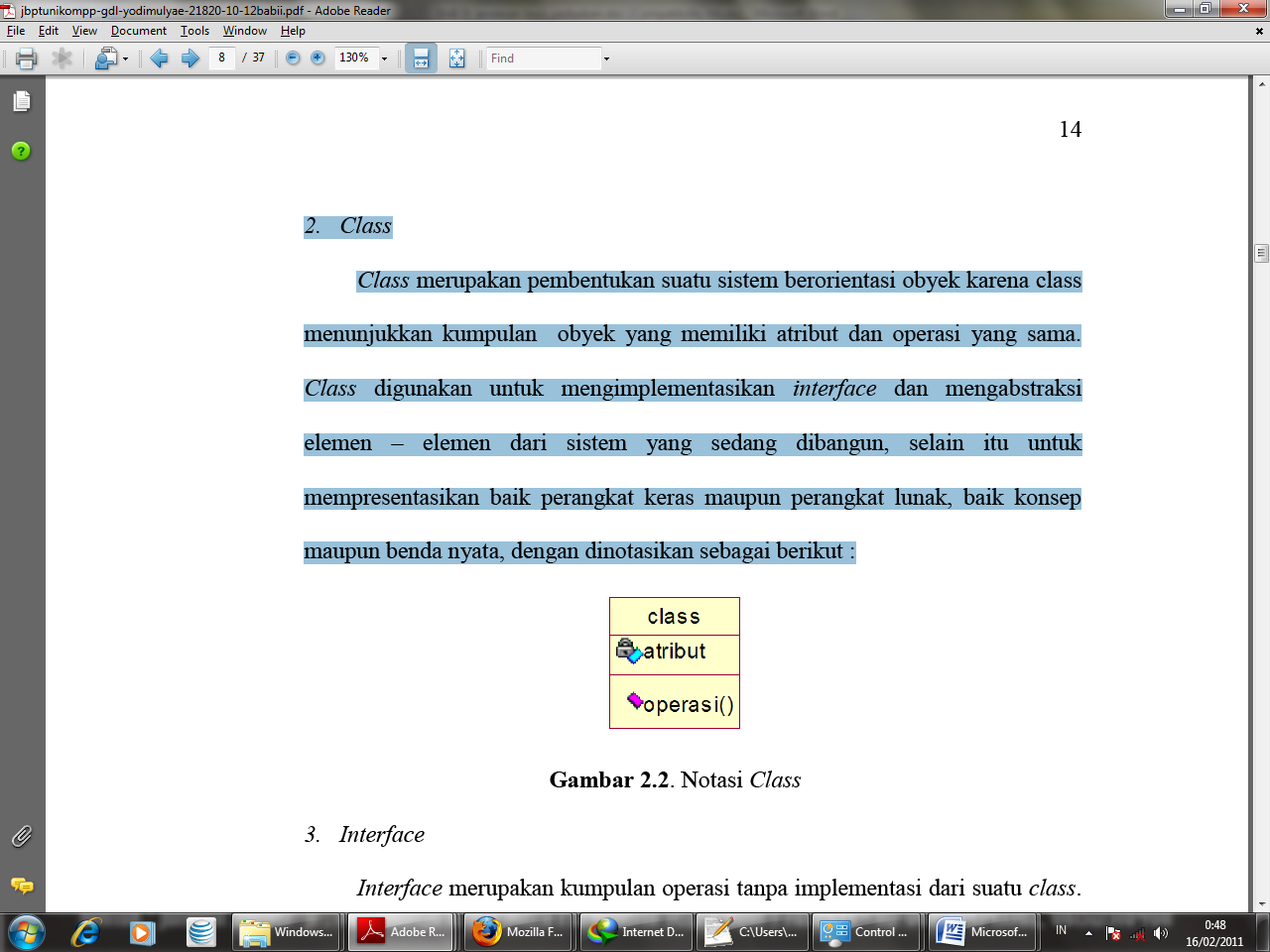
* 1. *Actor*

*Actor* adalah segala sesuatu yang berinteraksi dengan sistem aplikasi komputer. Jadi *actor* ini bisa berupa orang, perangkat keras atau mungkin juga obyek lain dalam sistem yang sama. Biasanya dilakukan oleh *actor* adalah memberikan informasi pada sistem dan memerintahkan sistem untuk melakukan sesuatu. Pada *actor* ini dinotasikan dengan gambar dibawah ini :



**Gambar 4.3. *Actor***

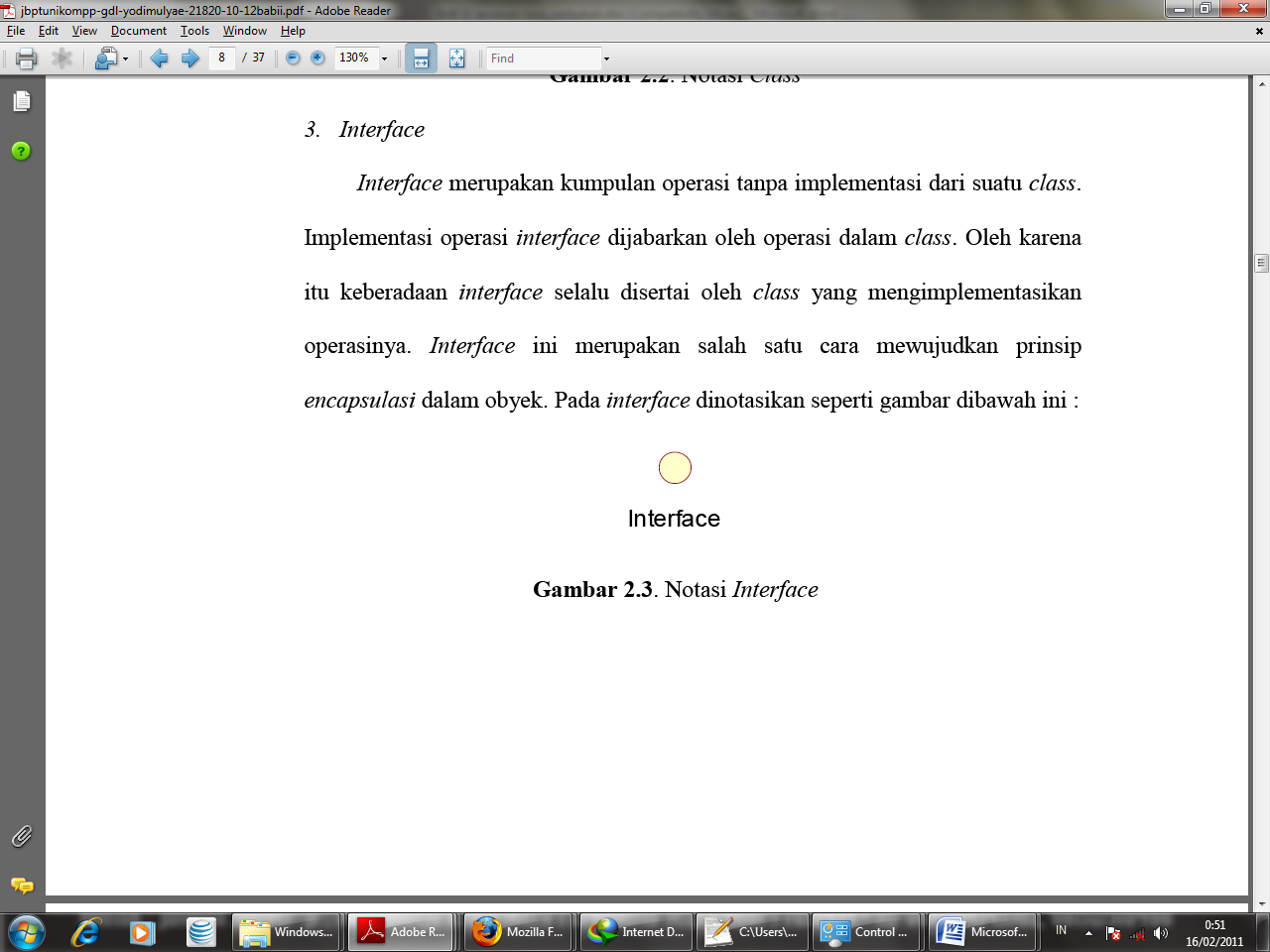
* 1. *Class*

*Class* merupakan pembentukan suatu sistem berorientasi obyek karena *class* menunjukkan kumpulan obyek yang memiliki atribut dan operasi yang sama. *Class* digunakan untuk mengimplementasikan *interface* dan mengabstraksi elemen – elemen dari sistem yang sedang dibangun, selain itu untuk mempresentasikan baik perangkat keras maupun perangkat lunak, baik konsep maupun benda nyata, dengan dinotasikan sebagai berikut :

**Gambar 4.4. Notasi** *Class*

* 1. *Interface*

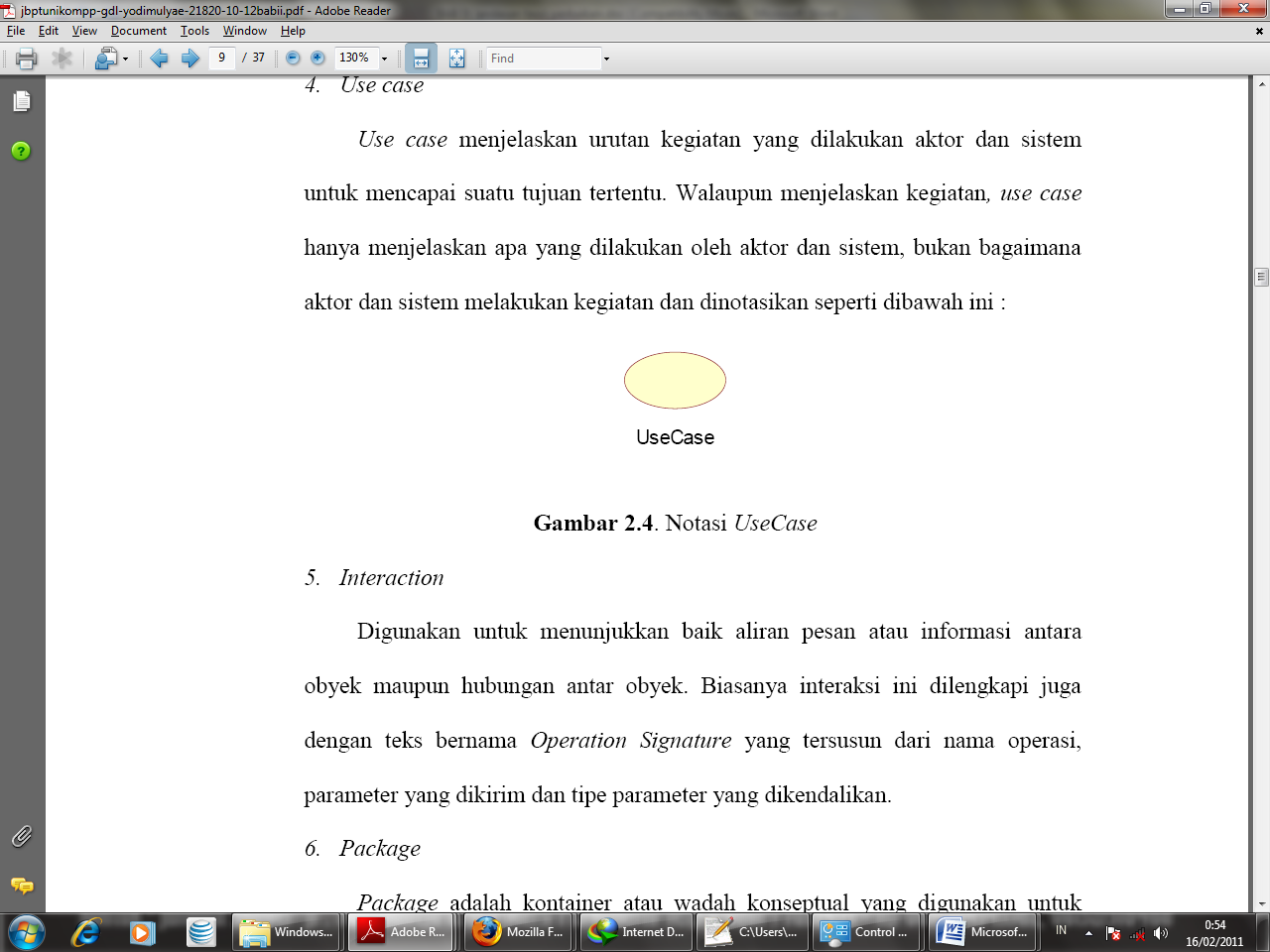
*Interface* merupakan kumpulan operasi tanpa implementasi dari suatu *class*. Implementasi operasi *interface* dijabarkan oleh operasi dalam *class*. Oleh karena itu keberadaan *interface* selalu disertai oleh *class* yang mengimplementasikan operasinya. *Interface* ini merupakan salah satu cara mewujudkan prinsip *encapsulasi* dalam obyek. Pada *interface* dinotasikan seperti gambar dibawah ini :



**Gambar 4.5. Notasi *Interface***

* 1. *Use case*

*Use case* menjelaskan urutan kegiatan yang dilakukan aktor dan sistem untuk mencapai suatu tujuan tertentu. Walaupun menjelaskan kegiatan*, use case* hanya menjelaskan apa yang dilakukan oleh aktor dan sistem, bukan bagaimana aktor dan sistem melakukan kegiatan dan dinotasikan seperti dibawah ini :



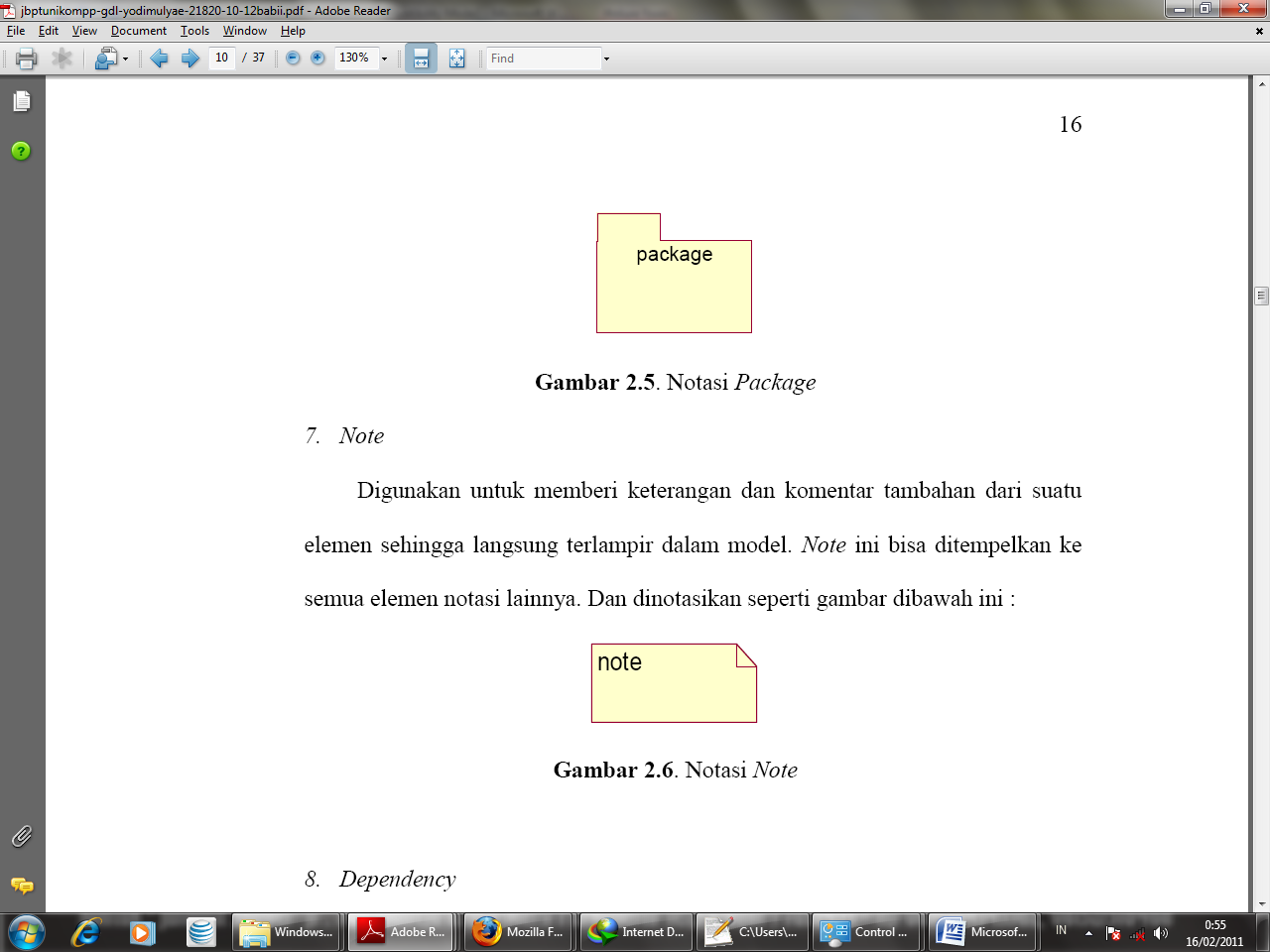
**Gambar 4.6. Notasi *UseCase***

* 1. *Interaction*

Digunakan untuk menunjukkan baik aliran pesan atau informasi antara obyek maupun hubungan antar obyek. Biasanya interaksi ini dilengkapi juga dengan teks bernama *Operation Signature* yang tersusun dari nama operasi, parameter yang dikirim dan tipe parameter yang dikendalikan.

* 1. *Package*

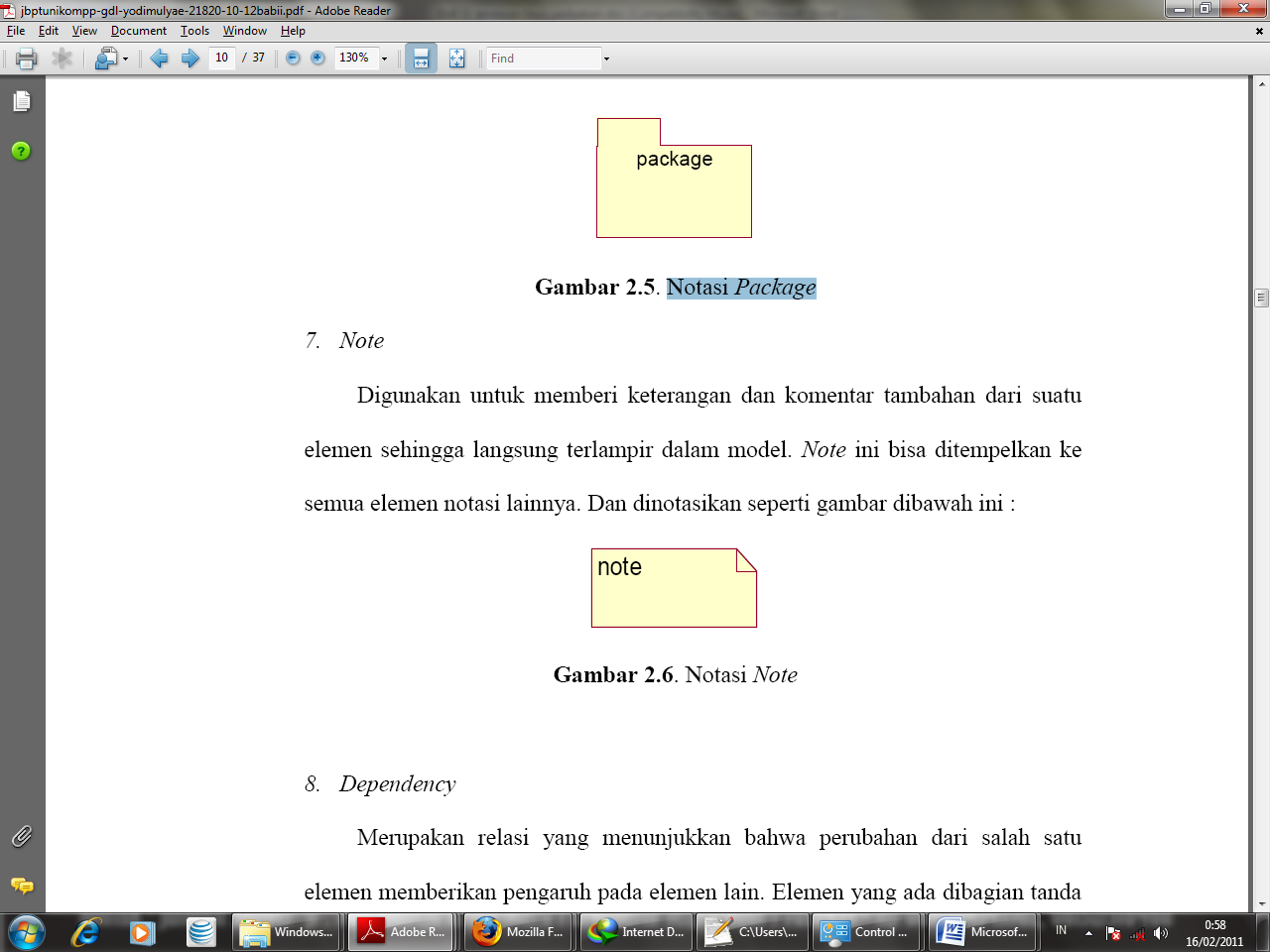
*Package* adalah kontainer atau wadah konseptual yang digunakan untuk mengelompokkan elemen – elemen dari sistem yang sedang dibangun, sehingga bisa dibuat model yang lebih sederhana. Tujuannya adalah untuk mempermudahkan penglihatan (*visibility*) dari model yang sedang dibangun, dinotasikan sebagai berikut :



**Gambar 4.7. Notasi *Package***

1. *Note*

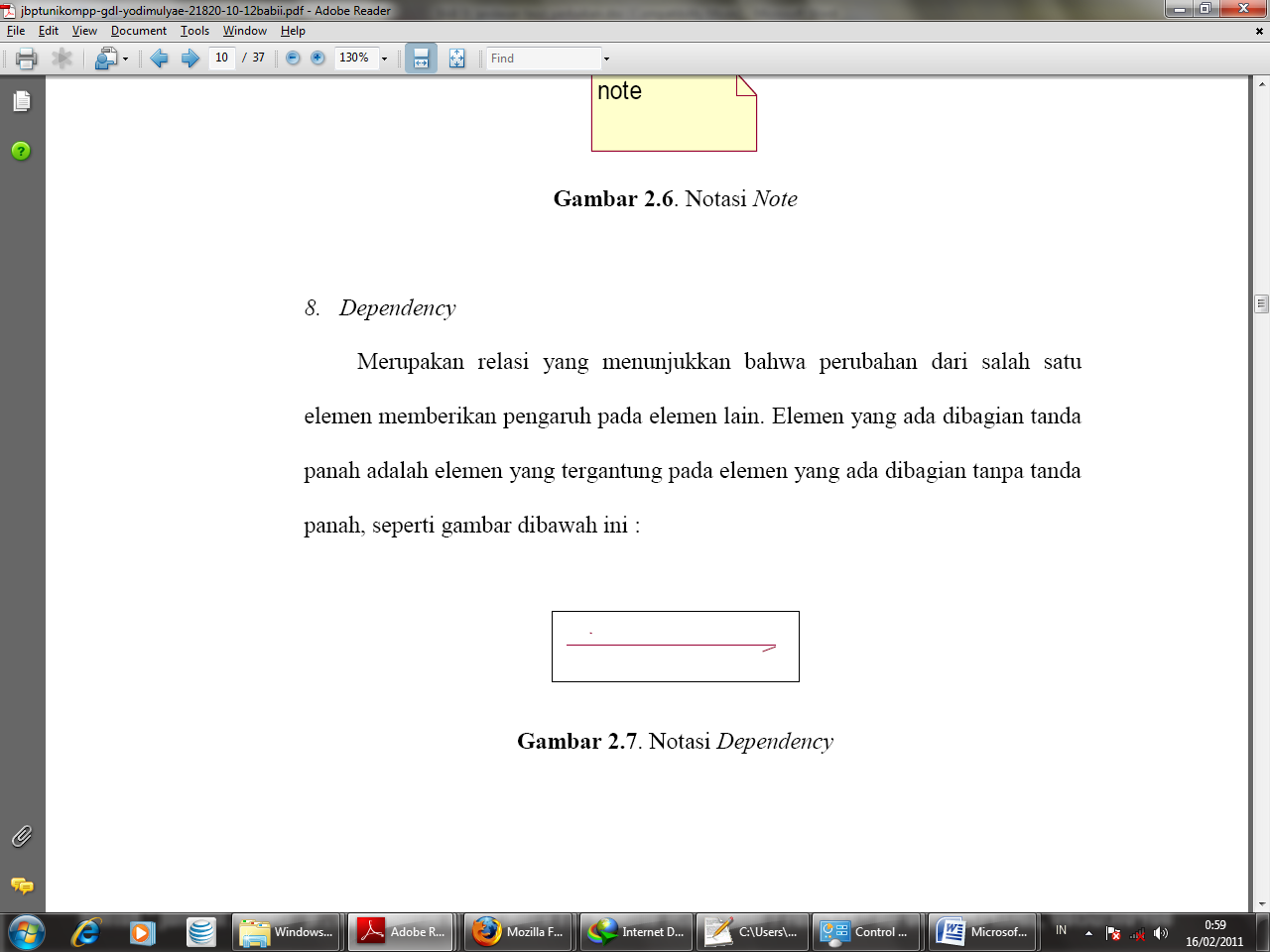
Digunakan untuk memberi keterangan dan komentar tambahan dari suatu elemen sehingga langsung terlampir dalam model. *Note* ini bisa ditempelkan ke semua elemen notasi lainnya. Dan dinotasikan seperti gambar dibawah ini :



**Gambar 4.8. Notasi *Note***

1. *Dependency*

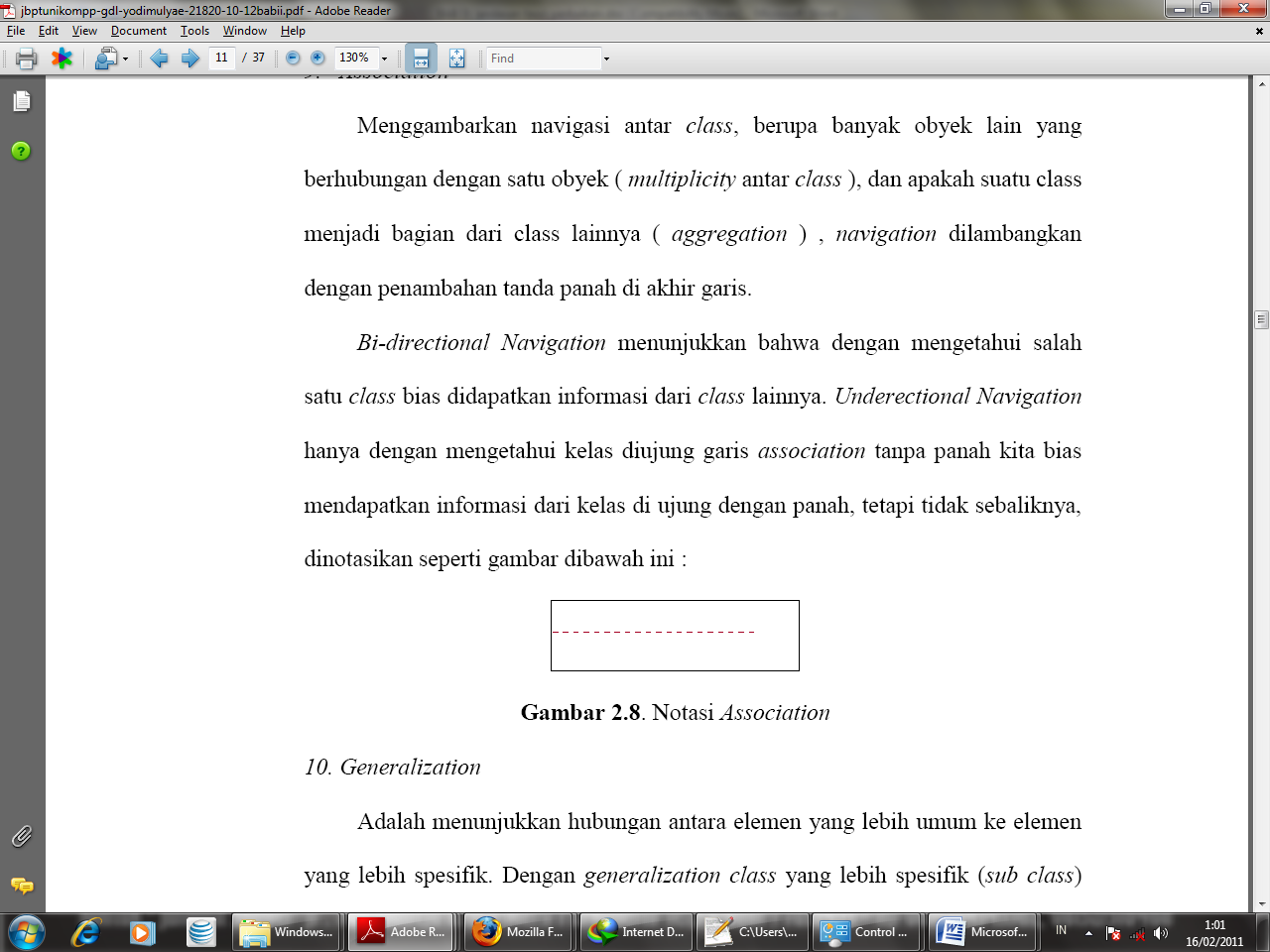
Merupakan relasi yang menunjukkan bahwa perubahan dari salah satu elemen memberikan pengaruh pada elemen lain. Elemen yang ada dibagian tanda panah adalah elemen yang tergantung pada elemen yang ada dibagian tanpa tanda panah, seperti gambar dibawah ini :



**Gambar 4.9. Notasi *Dependency***

1. *Association*

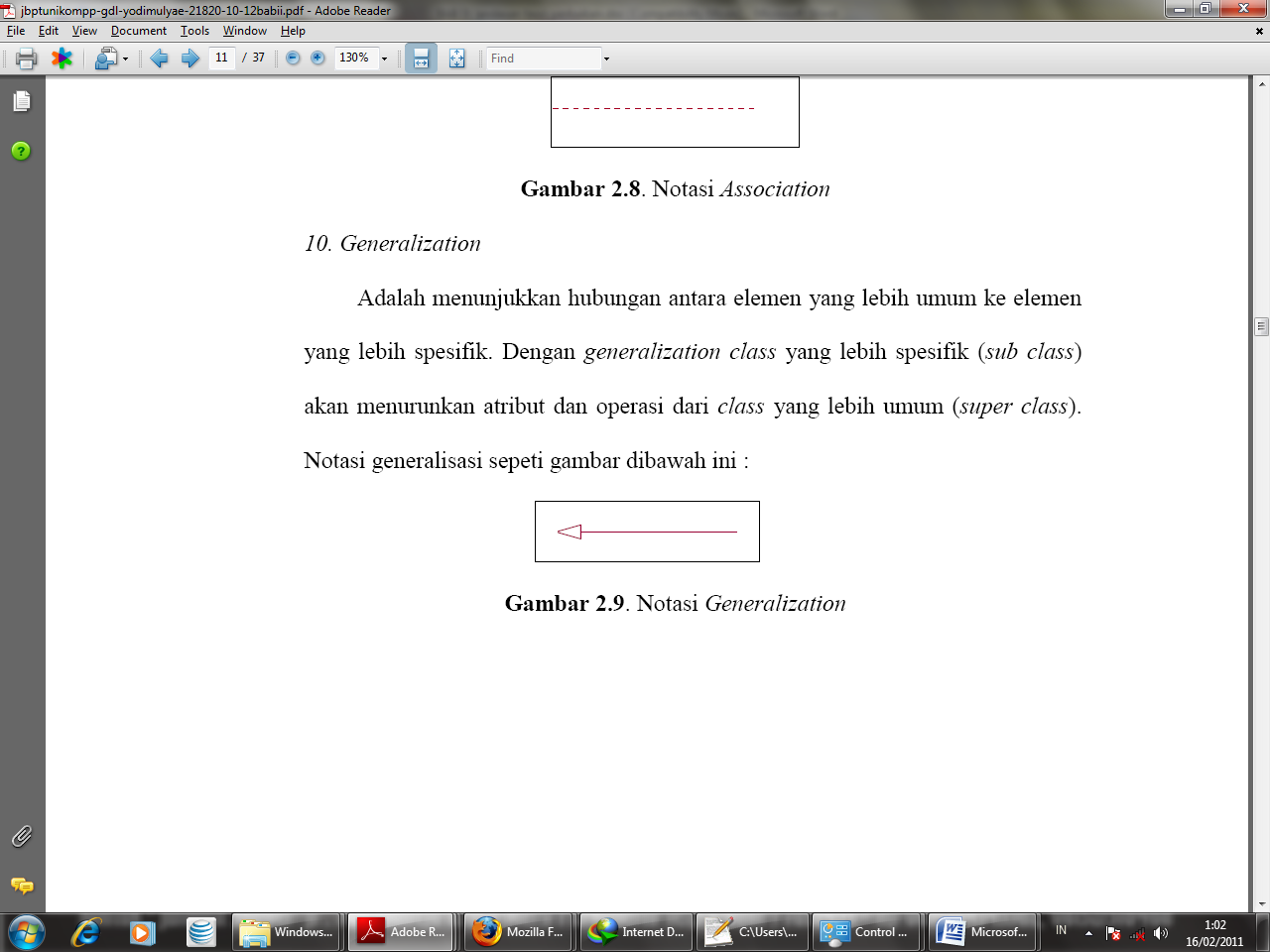
Menggambarkan navigasi antar *class*, berupa banyak obyek lain yang berhubungan dengan satu obyek ( *multiplicity* antar *class* ), dan apakah suatu *class* menjadi bagian dari *class* lainnya (*aggregation*), *navigation* dilambangkan dengan penambahan tanda panah di akhir garis. *Bi-directional Navigation* menunjukkan bahwa dengan mengetahui salah satu *class* bias didapatkan informasi dari *class* lainnya. *Underectional Navigation* hanya dengan mengetahui *class*  diujung garis *association* tanpa panah kita bias mendapatkan informasi dari *class*  di ujung dengan panah, tetapi tidak sebaliknya, dinotasikan seperti gambar dibawah ini:



**Gambar 4.10. Notasi *Association***

1. *Generalization*

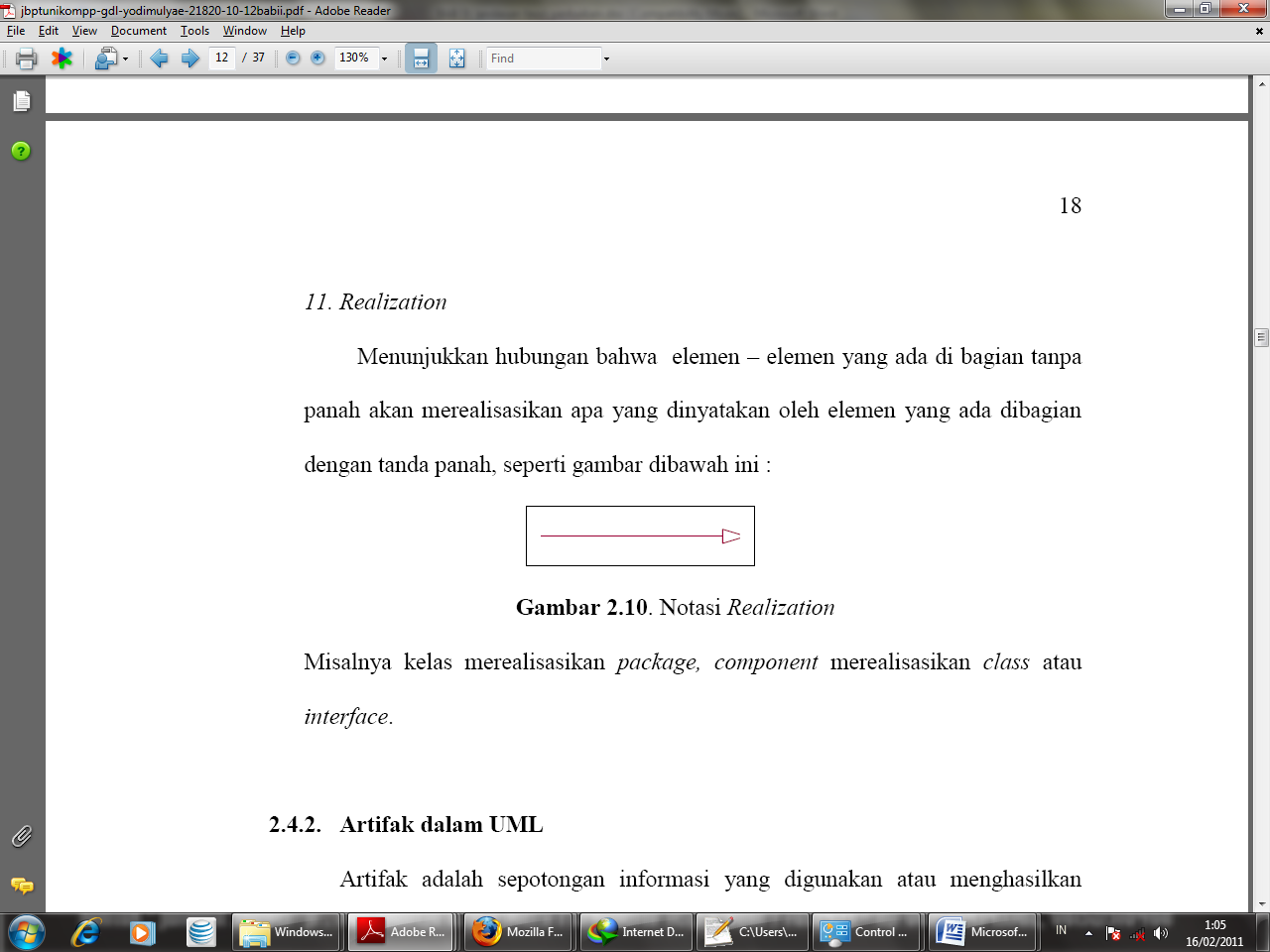
Adalah menunjukkan hubungan antara elemen yang lebih umum ke elemen yang lebih spesifik. Dengan *generalization class* yang lebih spesifik (*sub class*) akan menurunkan atribut dan operasi dari *class* yang lebih umum (*super class*). Notasi generalisasi sepeti gambar dibawah ini :

****

**Gambar 4.11. Notasi *Generalization***

1. *Realization*

Menunjukkan hubungan bahwa elemen – elemen yang ada di bagian tanpa panah akan merealisasikan apa yang dinyatakan oleh elemen yang ada dibagian dengan tanda panah, seperti gambar dibawah ini :



**Gambar 4.12. Notasi *Realization***

## **Artifak dalam UML**

Artifak adalah sepotongan informasi yang digunakan atau menghasilkan dalam suatu proses rekayasa *software*. Artifak dapat berupa model, deskripsi, atau *software*. Untuk membuat suatu model, *UML* memiliki diagram grafis sebagai berikut :

1. *Use Case Diagram*

Diagram ini memperlihatkan himpunan *use case* dan aktor – aktor. Diagram ini sangat penting untuk mengorganisasi dan memodelkan prilaku sesuatu sistem yang dibutuhkan serta diharapkan pengguna.

1. *Class Diagram*

Diagram ini memperlihatkan himpunana *class* , antar muka, kolaburasi, serta relasi – relasi. Diagram ini umumnya dijumpai pada pemodelan sistem berorientasi obyek.

1. *Objek Diagram*

Diagram ini memperlihatkan obyek – obyek serta relasi – relasi antar obyek.

1. *Sequence Diagram*

Pada diagram ini menekankan pada pengiriman pesan ddalam suatu waktu tertentu.

1. *Collaboration Diagram*

Diagram interaksi yang menekankan organisasi struktural dari obyek yang menerima serta mengirim pesan.

1. *Statechart Diagram*

Diagram ini memperlihatkan *state – state* pada sistem, terutama untuk memperlihatkan sifat dinamis dari antar muka, *class* , kolaburasi dan terutama penting pada pemodelan sistem – sistem reaktif.

1. *Activity Diagram*

Pada diagram ini memperlihatkan aliran dari suatu aktivitas ke aktivitas lainnya dalam suatu sistem.

1. *Component Diagram*

Diagram ini memperlihatkan organisasi serta kebergantungan sistem atau perangkat lunak pada komponen yang sudah ada sebelumnya.

1. *Deployment Diagram*

Diagram ini memperlihatkan konfigurasi saat aplikasi dijalankan. Digram ini memuat simpul – simpul beserta komponen – komponen yang ada didalamnya. Diagram ini sangat erat dengan diagram komponen, diagram ini sangat berguna saat aplikasi yang kita buat digunakan banyak mesin.

* 1. **Program Berbasis *Web***

Pembuatan aplikasi berbasis *web* berbeda dengan pembuatan aplikasi berbasis windows (*visual* *programming*), misalnya *Visual Basic, Delphi*, atau *KDevelop*. Dalam *visual* *programming*, kita meningkatkan kecepatan dan kinerja aplikasi dengan mengoptimasi penggunaan memori, manajemen proses, dan pengaturan *Input* - *Output*. Pada pemrograman berbasis *web*, faktor yang menentukan kinerja aplikasi adalah kecepatan akses *database* dan kecepatan akses jaringan dan internet.

Perbedaan kedua adalah cara aplikasi berjalan. Pada aplikasi visual, aplikasi dibangun dengan menggunakan *tool* tertentu, kemudian dikompilasi. Hasilnya dapat langsung digunakan dalam komputer. Aplikasi berbasis *web* tidak dapat dijalankan langsung di komputer. Untuk menjalankannya, dibutuhkan engine tertentu, dalam hal ini *web* *server*.

* + 1. ***Apache***

*Apache* merupakan *web* *server* yang paling sering digunakan sebagai *server* internet dibandingkan *web server* lainnya. *Web server* merupakan *server* internetyang mampu melayani koneksi transfer di dalam protokol *HTTP* (*Hypertext Transfer Protocol*) saat ini *web server* merupakan inti dari *server-server* internetselain *e-mail server*, *ftp server*, dan *news server*.

*Web server* sendiri dirancang untuk dapat melayani berbagai jenis data, diantaranya *text, hypertext*, gambar, suara dan bentuk *file* *HTML*. Dari *file* *HTML* kemudian dapat dikaitkan ke *file* *HTML* lainnya yang hendak dipublikasikan lewat internet.

* + 1. ***PHP***

*PHP* (*PHP Hypertext Preprocessor*) yang merupakan bahasa pemrograman berbasis *web* yang memiliki kemampuan untuk memproses data dinamis. *PHP* sebagai sebuah *server*-side embedded script language merupakan sintaks-sintaks dan perintah yang sepenuhnya dijalankan oleh *server* tetapi disertakan pada halaman *HTML* biasa. Aplikasi-aplikasi yang dibangun oleh *PHP* pada umumnya akan memberikan hasil pada *web* *browser*, tetapi prosesnya secara keseluruhan dijalankan di *server*.

* + - 1. **Konsep *PHP***

*PHP* digunakan oleh programmer berlatar belakang C/C++ karena kemiripan *syntax*nya. *Open source*, gratis dan bebas. Untuk *database* biasanya menggunaka *MySQL*, dijalankan bersama *webserver* Apache di atas *operating* *system Linux* maupun *Windows*.

Script *PHP* dapat disisipkan di bagian manapun dari *HTML*, asalkan ditulis di antara tag <? dan ?> atau <?*PHP* dan ?>. Tag tersebutlah yang memberi tanda bahwa di dalamnya terdapat kode-kode *PHP* yang sebagian besar mirip dengan bahasa yang digunakan pada pemrograman bahasa C. Setiap akhir perintah diakhiri dengan tanda titik koma *(semicolons)*.

Pada prinsipnya *server* akan bekerja apabila ada permintaan dari *client*. Dalam hal ini *client* menggunakan kode-kode *PHP* untuk mengirimkan permintaan ke *server* (dapat dilihat pada gambar dibawah). Ketika menggunakan *PHP* sebagai *server*-side *embedded script language* maka *server* akan melakukan hal-hal sebagai berikut :

* Membaca permintaan dari *client*/*browser*
* Mencari halaman/*page* di *server*
* Melakukan instruksi yang diberikan oleh *PHP* untuk melakukan modifikasi pada halaman/*page*.
* Mengirim kembali halaman tersebut kepada *client* melalui internet atau intranet.
  + - 1. **Keunggulan *PHP***

Beberapa keunggulan yang dimiliki program *PHP* ialah:

* Tingkat akses yang lebih cepat dan memiliki tingkat keamanan yang tinggi.
* *PHP* memiliki tingkat *Life Cycle* yang singkat, sehingga selalu mengikuti perkembangan teknologi internet.
* *PHP* mampu berjalan di hampir semua *server* yang ada, misalnya *Apache, Microsoft IIS, PWS, AOLserver, phttpd, fhttpd* dan lainnya.
* *PHP* mampu berjalan di berbagai platform sistem operasi, seperti *Windows, Linux, Solaris*, dan lain sebagainya.
  + 1. ***Database* *MySQL***

Bahasa *PHP* mempunyai kelebihan yaitu kompabilitasnya dengan berbagai macam jenis *database*, dukungan dengan berbagai macam jenis sistem operasi. *PHP* lebih cocok dan umum digunakan jika di gabungkan dengan *database* *MySQL*. *MySQL* dengan *PHP* seakan-akan dua hal yang tidak dapat dipisahkan. Tentunya untuk dapat menggunakan keduanya dibutuhkan tingkat kemampuan *programming* tertentu. Syarat yang harus dikuasai terlebih dahulu ialah harus memahai benar-benar *HTML*.

Bahasa SQL pada umumnya informasi tersimpan dalam tabel-tabel yang secara logika merupakan struktur dua dimensi terdiri dari baris (*row* atau *record*) dan kolom (*column* atau *field*). Sedangkan dalam sebuah *database* dapat terdiri dari beberapa *table*.

**Tabel 4.1 Tipe data dalam MySQL**

|  |  |
| --- | --- |
| **Tipe Data** | **Keterangan** |
| INT() | Angka  -2147483648 s/d 2147483647 |
| FIOAT() | Angka pecahan |
| DATE | Tanggal  Format : YYYY-MM-DD |
| DATETIME | Tanggal dan waktu  Format : YYYY-MM-DD HH:MM:SS |
| CHAR() | String dengan panjang tetap sesuai dengan yang ditentukan  Panjangnya 1 – 255 karakter. |
| VARCHAR() | String dengan panjang yang berubah – ubah sesuai dengan yang disimpan saat itu.  Panjangnya 1 – 255 karakter. |

* + 1. **Framework**

*Framework* adalah sekumpulan *class* dan *library* yang terpadu sehingga memudahkan menyelesaikan permasalahan seraca menyeluruh, efeknya waktu untuk membuat program menjadi lebih singkat, beberapa contoh *framework* di *PHP* adalah *CakePHP, Zend Framework, CodeIgniter* dan *Symfony.*

* + 1. ***Codeigniter* (CI)**

*CodeIginter* adalah salah satu jenis dari *PHP* *framework* yang sedang berkembang sekarang ini. Struktur *CI* lebih banyak meniru *Ruby on Rail (RoR)*, sebuah *framework* pembuatan aplikasi *web* dengan menggunakan bahasa pemrograman Ruby. Selain itu, dokumentasinya lebih lengkap dan tentunya mudah dipahami. Kelebihan lain dari *CodeIgniter* adalah *Search Engine* dan *Human Friendly*.

Secara umum, *framework* *CodeIgniter* menggunakan struktur *MVC* (*Model, View, Controller*).

1. *Model*

*Model* mencangkup semua proses yang terkait dengan pemanggilan struktur data baik berupa pemanggilan fungsi, *Input* processing atau mencetak *Output* ke dalam *browser*.

1. *View*

*View* mencangkup semua proses yang terkait *layout* *Output*. Bisa dibilang untuk menaruh template interface website atau aplikasi.

1. *Control*

*Controller* mencangkup semua proses yang terkait dengan pemanggilan *database* dan kapsulisasi proses-proses utama. Jadi semisal di bagian ini ada *file* bernama member *PHP*, maka semua proses yang terkait dengan member akan dikelompokan dalam *file* ini.

* 1. ***User Acceptance Test***

Teknik pengujian *User Acceptance Test* sebuah proses untuk mendapatkan konfirmasi dari seorang *SME – Subject Matter Expert* (ahli dibidangnya), terutama pemilik atau *klien* yang mengerti tentang objek yang sedang dalam fase pengetesan, melalui *trial* atau *review* yang mofikasi dan tambahannya sesuai dengan *requirement* yang sudah disetujui sebelumnya.

* 1. ***Black Box***

Teknik pengujian *black box* befokus pada domain informasi dari perangkat lunak, dengan melakukan *test case* dengan mempartisi domain *input* dari suatu program dengan cara memberikan cakupan pengujian yang mendalam.