2. LANDASAN TEORI

2.1. Sistem Informasi

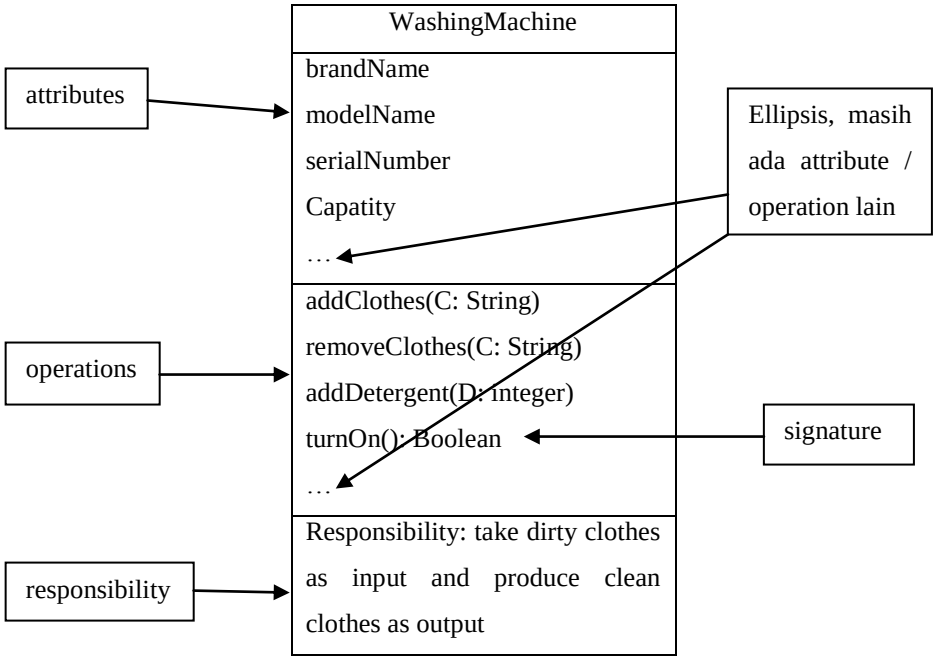
Menurut McLeod (2001), sistem adalah kelompok elemen yang terintegrasi dengan maksud yang sama untuk mencapati suatu tujuan. Menurut Laudon *and* Laudon (2002), informasi adalah data yang telah diproses atau dapat dikatakan sebagai data yang memiliki sebuah arti. Sedangkan data adalah angka dan fakta yang menggambarkan peristiwa yang terjadi dalam suatu organisasi atau lingkungan fisik yang belum diatur atau diproses. Jadi, sistem informasi dapat didefinisikan sebagai seperangkat elemen yang bakerja sama dalam mengumpulkan, memproses, menyimpan dan menyebarkan informasi untuk mendukung pengambilan keputusan, koordinasi, pengawasan, analisis, dan visualisasi dalam organisasi

Menurut Romney *and* Steimbart (2003), pengertian secara umum, sistem informasi merupakan kumpulan komponen atau elemen yang saling bekerja sama untuk mencapai tujuan tertentu. Sistem memerlukan sumber daya yang akan mengubah *input* menjadi *output*. Informasi merupakan data yang sudah diolah sehingga berguna untuk pengambilan keputusan. Sistem informasi merupakan suatu susunan dari komponen-komponen berhubungan yang saling berinteraksi untuk mendukung kegiatan, manajemen operasi dan pengambilan operasi yang dibutuhkan oleh suatu perusahaan atau organisasi.

2.2. *Class Diagram*

2.2.1. *Class* dan *Object*

Menurut Rostianingsih dan Yulia (2009), *Object* merupakan instans dari sebuah *class* dan *class* merupakan *blue print* dari *real object.* Sebuah *class* memiliki atribut dan *operation* (dalam *java*: *method*). *Object* sebagai instans memiliki harga dari setiap atribut yang didefinisikan dalam *class*. Berikut ini merupakan struktur dari sebuah *class* yang dapat dilihat pada Gambar 2.1.



Gambar 2.1. Struktur *Class*

Sumber : Rostianingsih dan Yulia, (2009, hal 42)

2.2.2. *Relationship*

Menurut Rostianingsih dan Yulia (2009), suatu *class* dapat berelasi dengan *class* yang lainnya melalui hubungan :

- *Association*
- *Multiplicity*
- *Qualified Association*
- *Reflexive Association*
- *Inheritance* and Generalisasi
- Agregasi
- *Dependency*

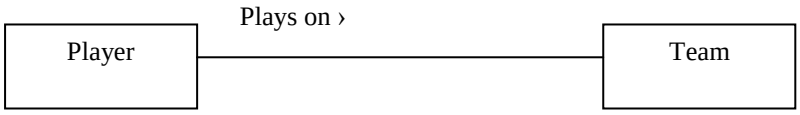
Berikut ini penjelasan dari masing-masing relasi tersebut.

2.2.2.1.*Association*

Menurut Rostianingsih dan Yulia (2009), jika dua kelas berhubungan secara konseptual maka hubungan tersebut disebut asosiasi. Asosiasi selain dinyatakan dengan garis yang menghubungkan kedua kelas, juga dispesifikasikan

dengan label hubungan asosiasi tersebut serta mata panah sebagai arah dari hubungan tersebut.

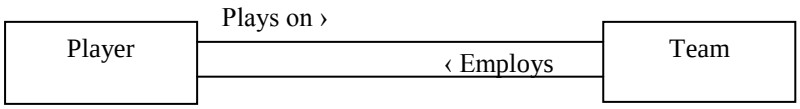
Dalam hubungan tersebut masing-masing kelas dapat memiliki peranan (*role*). Peranan dituliskan dekat pertemuan garis dan segiempat kelas yang bersangkutan. Contoh asosiasi dapat dilihat pada Gambar 2.2.



Gambar 2.2. Contoh Asosiasi

Sumber : Rostianingsih dan Yulia, (2009, hal 43)

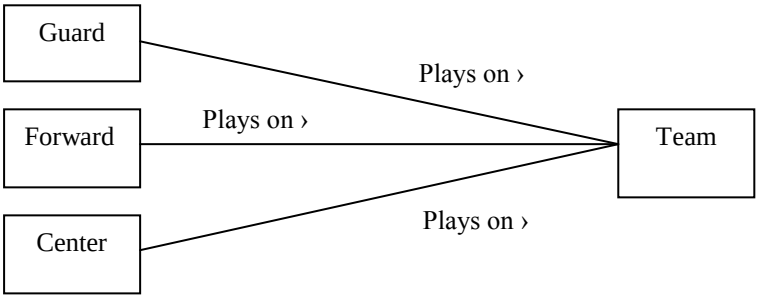
Antar dua kelas terdefinisi dapat terjadi dua jenis asosiasi secara bolak-balik yang dapat dilihat pada Gambar 2.3.



Gambar 2.3. Asosiasi Bolak Balik

Sumber : Rostianingsih dan Yulia, (2009, hal 43)

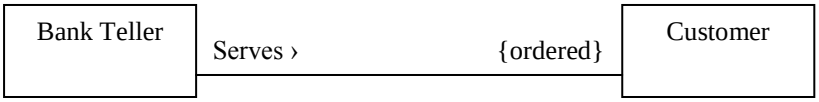
Asosiasi bisa lebih kompleks, dimana beberapa kelas berkoneksi dengan suatu kelas yang dapat dilihat pada Gambar 2.4.



Gambar 2.4. Asosiasi 1 *Class* dengan Beberapa *Class*

Sumber : Rostianingsih dan Yulia, (2009, hal 43)

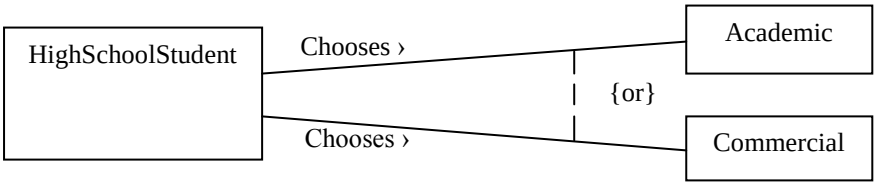
Asosiasi antar dua kelas mungkin harus mengikuti suatu aturan (constraint). Misalnya "{ordered}" yang dapat dilihat pada Gambar 2.5.



Gambar 2.5. Asosiasi dengan *Constraint*

Sumber : Rostianingsih dan Yulia, (2009, hal 44)

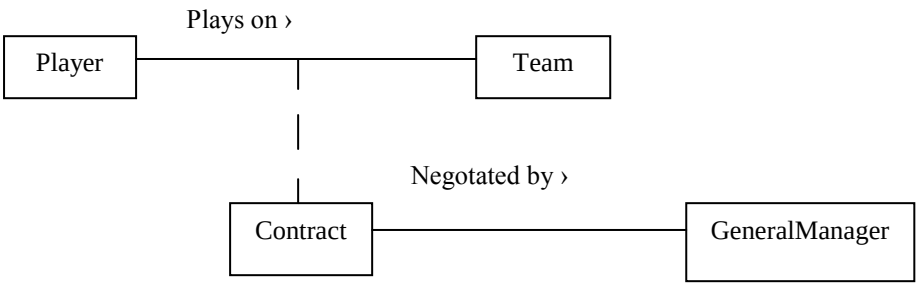
Bentuk asosiasi dengan *constraint* lainnya adalah relasi "Or" yang dinyatakan dengan notasi {or} pada garis putus-putus antar dua garis asosiasi yang dapat dilihat pada Gambar 2.6.



Gambar 2.6. Asosiasi dengan *Constraint-OR*

Sumber : Rostianingsih dan Yulia, (2009, hal 44)

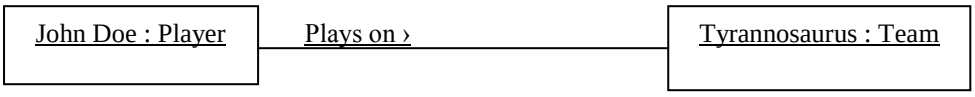
Asosiasi bisa memiliki atribut dan operasi, sebagaimana suatu kelas, maka disebut *class association.* Penggambaran dari asosiasi ke *class* adalah dengan garis putus-putus yang dapat dilihat pada Gambar 2.7.



Gambar 2.7. *Class Association*

Sumber : Rostianingsih dan Yulia, (2009, hal 44)

Jika instan dari suatu kelas adalah objek, maka suatu asosiasi dapat juga memiliki instan yaitu yang disebut *link*. *Link* digambarkan antara dua objek dan labelnya digarisbawahi (sama halnya dengan nama objek). *Link* dapat dilihat pada Gambar 2.8.



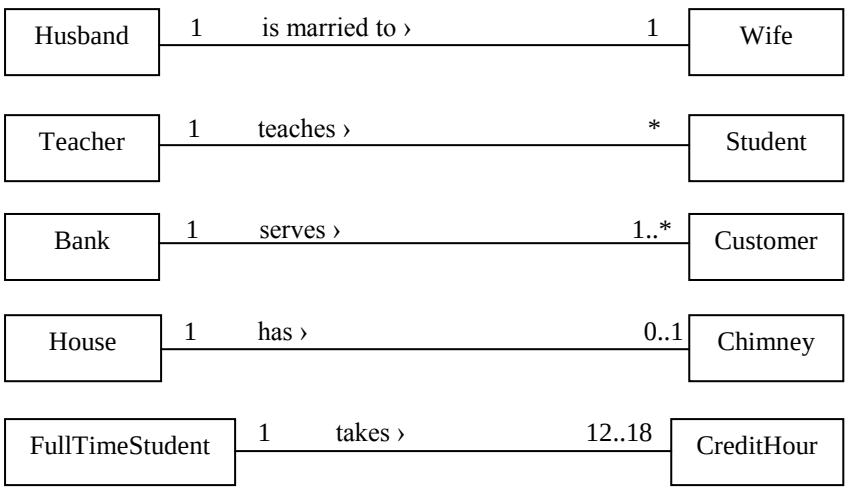
Gambar 2.8. *Link* – Asosiasi antar 2 *Object*

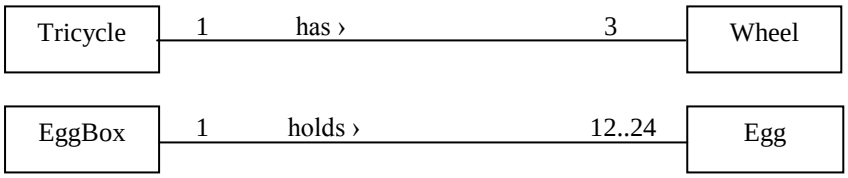
Sumber : Rostianingsih dan Yulia, (2009, hal 44)

2.2.2.2.*Multiplicity*

Menurut Rostianingsih dan Yulia (2009), asosiasi sering kali harus dispesifikasikan dalam jumlah objek yang terkait dalam asosiasi, disebut multiplisitas (*multiplicity*). Untuk menyatakan jumlah, maka digunakan notasi bilangan tertentu, bilangan tak berhingga (*), *range* bilangan (n..m), pilihan (a,b). Contoh-contoh *multiplicity* dapat dilihat pada Gambar 2.9.

- *One-to-one*
- *One-to-many*
- *One-to-one or more*
- *One-to-zero or one*
- *One-to-12 through 18*
- *One-to-three*
- *One-to-12 or 24*



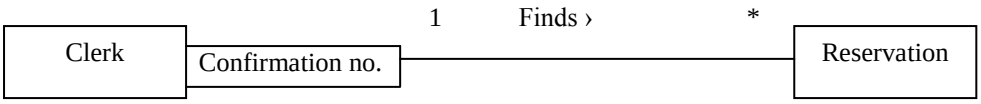


Gambar 2.9. Contoh-Contoh *Multiplicity*

Sumber : Rostianingsih dan Yulia, (2009, hal 45)

2.2.2.3.Qualified Association

Menurut Rostianingsih dan Yulia (2009), 2.2.2.3. *Qualified Association* merupakan asosiasi berdasarkan mekanisme "lookup" (pemilihan objek sesuai dengan role tertentu dari asosiasi tersebut). Contoh : *clerk* memeriksa reservasi anda berdasarkan nomor konfirmasi yang sebelumnya sudah diberikan saat reservasi dilakukan. *Qualified Association* dapat dilihat pada Gambar 2.10.

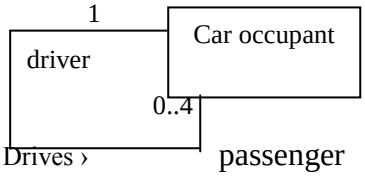


Gambar 2.10. *Qualified Association*

Sumber : Rostianingsih dan Yulia, (2009, hal 46)

2.2.2.4.*Reflexive Association*

Menurut Rostianingsih dan Yulia (2009), *Reflexive Association* merupakan asosiasi dari suatu *class* ke *class* itu sendiri. Biasanya terjadi karena objek-objek dari kelas tersebut memiliki *role* yang bervariasi. Contoh : orang yang ada dalam mobil bisa sopir, bisa penumpang. *Reflexive Association* dapat dilihat pada Gambar 2.11.

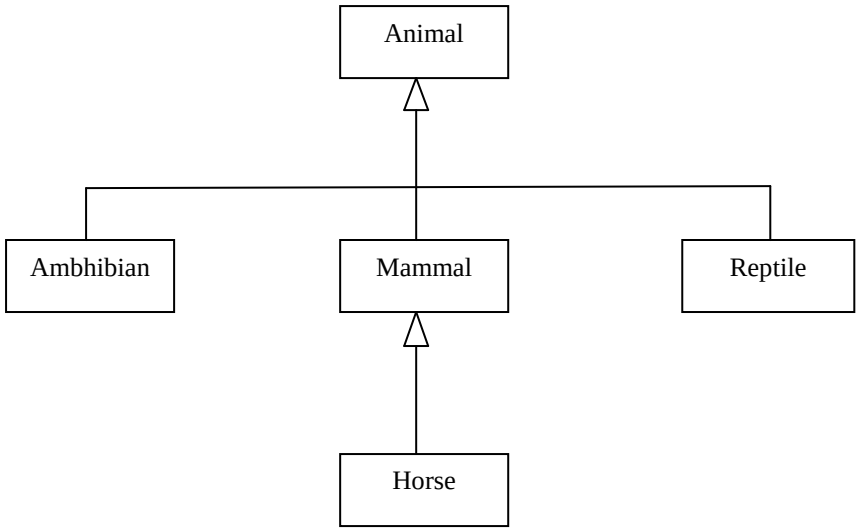


Gambar 2.11. *Reflexive Association*

Sumber : Rostianingsih dan Yulia, (2009, hal 46)

2.2.2.5.*Inheritance* dan *Generalization*

Menurut Rostianingsih dan Yulia (2009), *Inheritance* merupakan terminologi dari *object-orientation*, sedang dalam UML menggunakan istilah *generalization*. Dengan *inheritance*, suatu *class* (*child class* atau *subclass*) dapat meng-inherit atribut-atribut dan operasi-operasi dari kelas lainnya (*parent class* atau *superclass*). *Parent class* lebih *general* dari *child class*. *Parent* dapat disubtitusi oleh *child*, tetapi tidak sebaliknya. Pada notasi UML menggunakan garis berpanah dengan bentuk segitiga kosong mengarah ke *parent.* Penggambaran hirarki dan garis hubungan tersebut meniru struktur organisasi. *Inheritance* dapat dilihat pada Gambar 2.12.

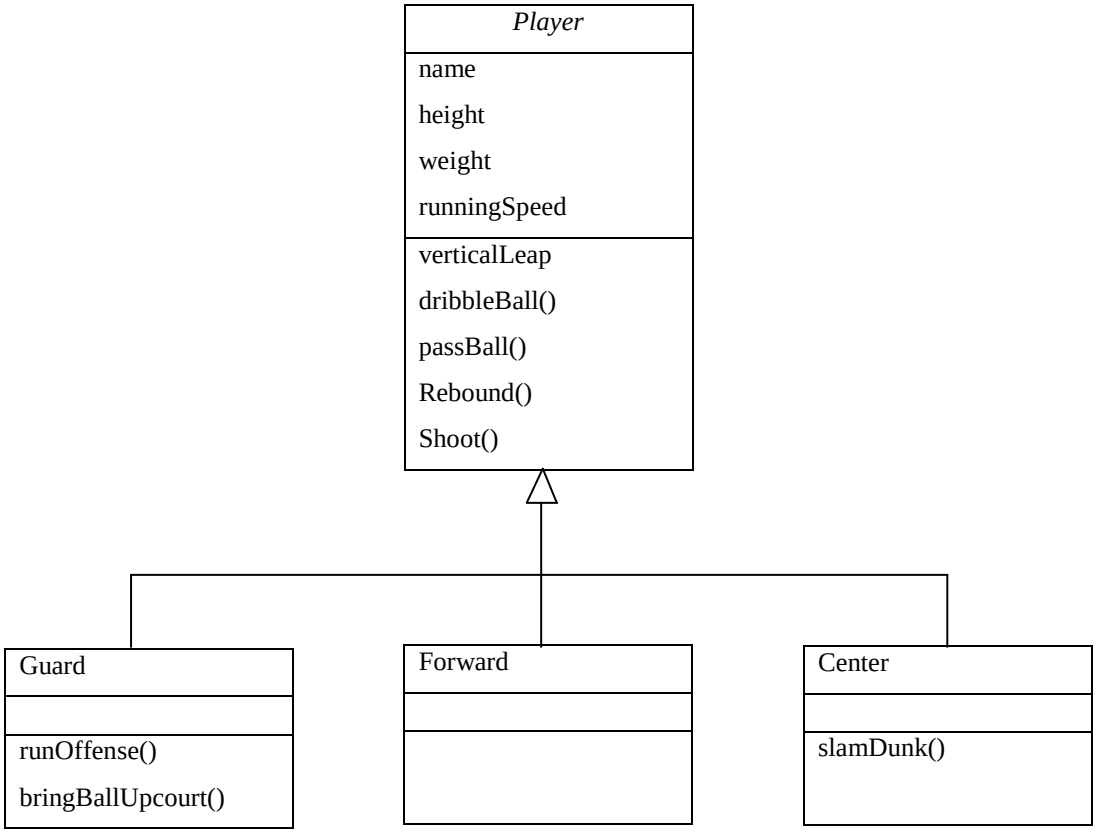


Gambar 2.12. *Inheritance*

Sumber : Rostianingsih dan Yulia, (2009, hal 46)

2.2.2.6.*Abstract Class*

Menurut Rostianingsih dan Yulia (2009), *Abstract Class* : jika satu *class* hanya diperlukan sebagai *template* untuk *class-class* yang lebih spesifik (dalam sistem tidak aka nada *object* dari *class* tersebut). Dalam notasi namanya dituliskan huruf miring (*italic*). Suatu *abstract class* tidak mempunyai *object/instant*. *Abstract Class* dapat dilihat pada Gambar 2.13.

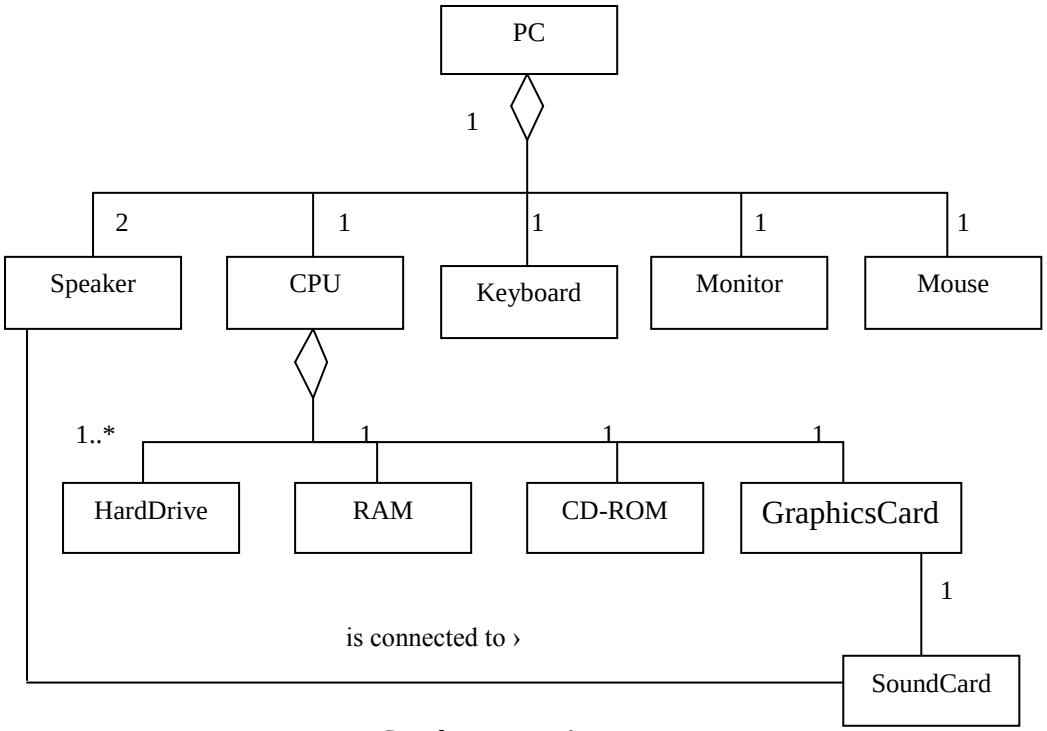


Gambar 2.13. *Abstract Class*

Sumber : Rostianingsih dan Yulia, (2009, hal 47)

2.2.2.7.*Agregasi*

Menurut Rostianingsih dan Yulia (2009), relasi dimana suatu *class* berisikan sejumlah *component class*; asosiasi "Part Whole". Biasa digambarkan sebagai hirarki "Whole" di atas dan "part" di bawah. Garis yang menghubungkan *part* ke *whole* berujungkan dekat *whole* lambang *diamond* kosong. Suatu *part* komponen bisa dimiliki oleh lebih dari satu *class whole* (memiliki relasi *part whole* dengan dari lebih satu *whole*). Agregasi dapat dilihat pada Gambar 2.14.

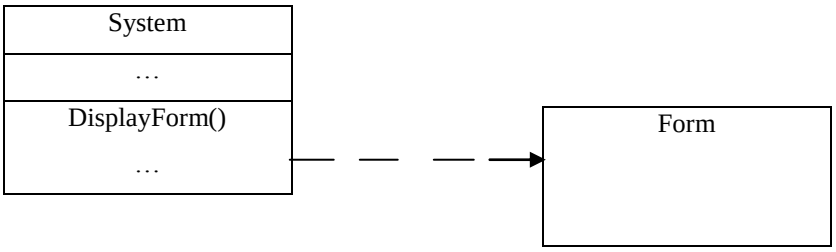


Gambar 2.14. Agregasi

Sumber : Rostianingsih dan Yulia, (2009, hal 47)

2.2.2.8.*Dependency*

Menurut Rostianingsih dan Yulia (2009), *Dependency* terjadi jika suatu *signature* (*output function*) suatu *class* digunakan oleh *class* lain. *Dependency* digambarkan dengan garis putus-putus yang dapat dilihat pada Gambar 2.15.



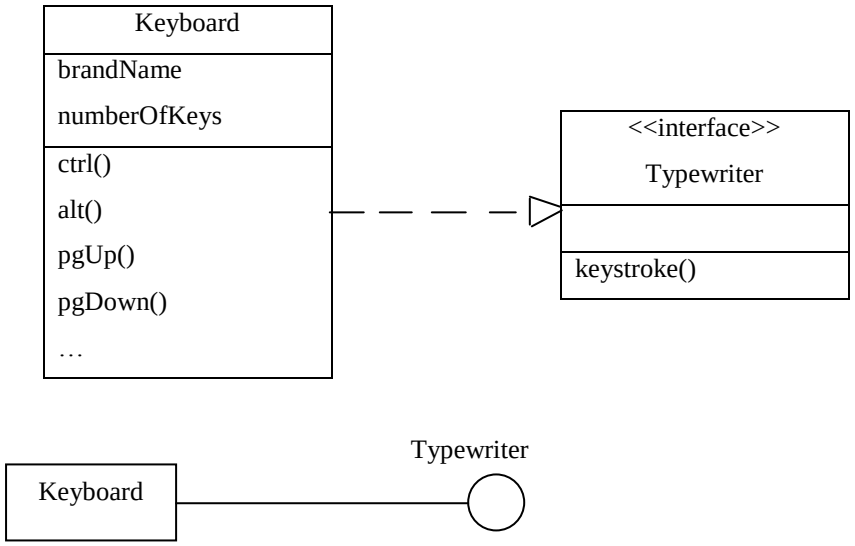
Gambar 2.15. *Dependency*

Sumber : Rostianingsih dan Yulia, (2009, hal 48)

2.2.2.9.*Interface* dan *Realization*

Menurut Rostianingsih dan Yulia (2009), jika sejumlah *class* yang tidak berelasi dengan *parent* tertentu tapi memiliki operasi dengan *signature* yang serupa. Mendefinisikan himpunan operasi-operasi tersebut dan menggunakan

kembali di *class-clas* tersebut. Himpunan operasi tersebut adalah *interface* melalui relasi "realization". Satu *class* bisa merealisasi beberapa *interface* dan satu *interface* bisa direalisasi oleh beberapa *class*. Digambarkan dengan garis putus-putus dengan mata panah kosong diujung garis mengarah *interface*. *Note* pada *inheritance* relasi digambarkan dengan garis solid. *Interface* digambarkan sebagai *class* dengan notasi "<<interface>>" di atas nama *interface*-nya. Cara alternatif penggambaran *interface* adalah dengan lingkaran kecil yang dihubungkan dengan *class* melalui garis solid yang dapat dilihat pada Gambar 2.16.



Gambar 2.16. *Interface*

Sumber : Rostianingsih dan Yulia, (2009, hal 48)

2.2.2.10. *Visibilitas*

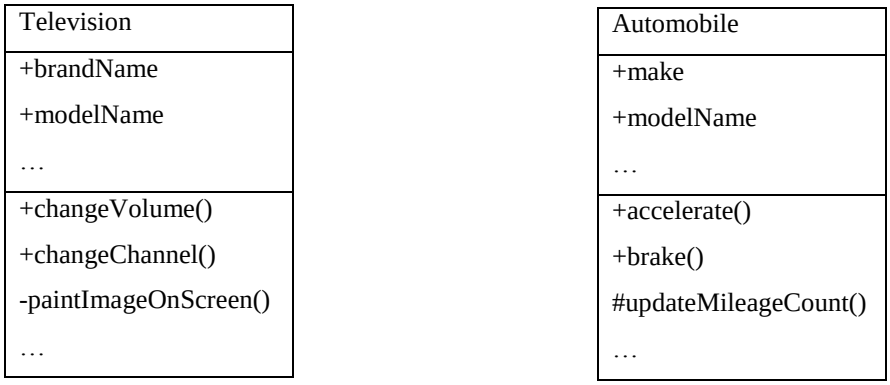
Menurut Rostianingsih dan Yulia (2009), Visibilitas suatu *class* : menentukan atribut dan operasi yang dapat digunakan oleh *class* lain. Terdapat empat tingkatan visibilitas :

- *Public* : *visible* untuk *class* lain manapun.
- *Protected* : *visible* untuk *class* lain yang diinherit dari *class* yang bersangkutan.
- *Private* : *visible* hanya untuk *class* yang bersangkutan.
- *Package* : *visible* hanya untuk *class* yang berada dalam *package* yang sama.

Dalam diagram, dinyatakan dengan menambahkan :

- "+" untuk *public*,
- "#" untuk *private*,
- "-" untuk *protected,*
- "~" untuk *package.*

Di depan masing-masing nama atribut/*operation*. Notasi tersebut diperlukan untuk *code generation*. Visibilitas dapat dilihat pada Gambar 2.17.



Gambar 2.17. Visibilitas dalam *Class*

Sumber : Rostianingsih dan Yulia, (2009, hal 49)

2.2.3. Membangun Sebuah *Class Diagram*

Menurut Rostianingsih dan Yulia (2009), cara membangun sebuah *Class Diagram* :

- Identifikasi semua *class*, beri nama dan definisikan mengapa menjadi bagian dari model.
- Identifikasi, beri nama dan definisikan *asosiasi* antara pasangan *class* termasuk *reflexive* dan *composition.*

2.2.4. Pentingnya *Class Diagram*

Menurut Rostianingsih dan Yulia (2009) :

- *Class Diagram* merupakan yang paling digunakan.
- Lebih dikenal sebagai diagram berorientasi ojek.

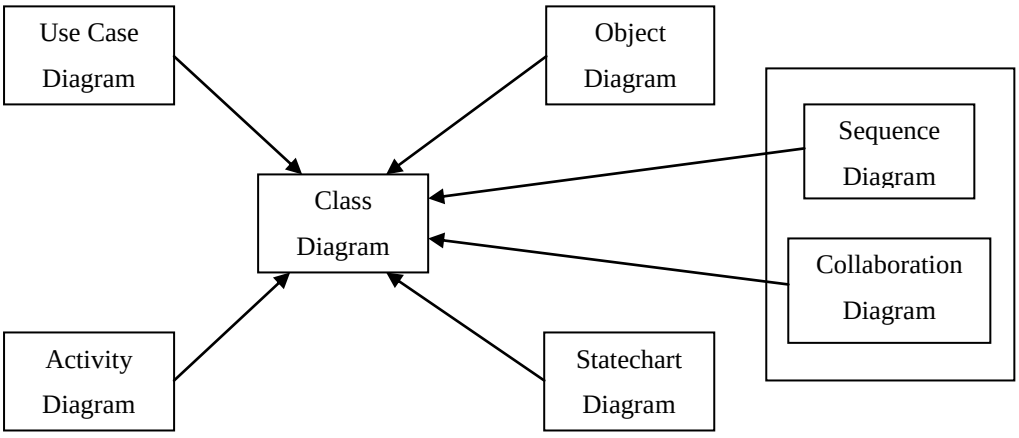
- Merupakan sumber untuk *code generation* dan target dari *reverse engineering code.*
- *Diagram* lain sebagai *tool* untuk memberi arti tambahan dari *class diagram*.

2.2.4.1.Seluruh *Diagram* Mendukung *Class Diagram*

Menurut Rostianingsih dan Yulia (2009) :

- *Use Case Diagram* untuk mengidentifikasi kebutuhan objek sebagai *resource* yang digunakan sistem untuk mencapai tujuannya.
- *Sequence* dan *Collaboration Diagram* untuk mengetahui interaksi antar objek dan mendefinisikan *interface*.
- *Activity Diagram* untuk menemukan sifat yang diimplementasikan oleh objek-objek dan membantu untuk mendefinisikan logika dari operasi-operasi objek.

Seluruh *diagram* yang mendukung *Class Diagram* dapat dilihat pada Gambar 2.18.



Gambar 2.18. Seluruh *Diagram* Mendukung *Class Diagram*

Sumber : Rostianingsih dan Yulia, (2009, hal 50)

2.3. *Use Case Diagram*

Menurut Rostianingsih dan Yulia (2009), *Use Case Diagram* menggambarkan tentang fungsionalitas yang disediakan oleh sistem. Tujuan utama dari *Use Case Diagram* adalah membentuk tim pengembang proyek untuk dapat melakukan visualisasi terhadap kebutuhan fungsional sistem, termasuk relasi dari *actor* (yang berinteraksi dengan sistem) terhadap proses dan relasi antar *use case*. Berikut ini adalah beberapa simbol pada *Use Case Diagram* yang digunakan untuk menggambarkan sistem.

2.3.1. Relasi

Menurut Rostianingsih dan Yulia (2009), *Use Case* menggambarkan kumpulan fungsi sistem yang disediakan untuk *actor*. Setiap *use case* dapat berinteraksi dengan satu atau lebih *actor*. Dalam sebuah *use case* terdapat urutan pekerjaan antara sistem dan *actor* tersebut. Simbol *Use Case* dapat dilihat pada Gambar 2.19.

Gambar 2.19. *Use Case*

Sumber : Rostianingsih dan Yulia, (2009, hal 31)

2.3.2. Actor

Menurut Rostianingsih dan Yulia (2009), *Actor* menggambarkan seseorang yang dapat menginisialisasi *use case*. *Actor* dapat berupa orang atau benda, yang artinya tidak mungkin dijadikan dalam bentuk *coding*. Simbol *Actor* dapat dilihat pada Gambar 2.20.

Gambar 2.20. *Actor*

Sumber : Rostianingsih dan Yulia, (2009, hal 31)

2.3.3. Relasi

Menurut Rostianingsih dan Yulia (2009), relasi menggambarkan bagaimana *actor* dan *use case* saling bersosialisasi. Terdapat tiga jenis garis relasi yang digunakan yaitu : (1) garis lurus, (2) garis putus-putus, dan (3) garis lurus dengan anak panah berbentuk segitiga kosong. Relasi dibagi menjadi tiga bagian, yaitu :

1. Relasi antara *actor* dengan *use case*, menggunakan relasi jenis pertama, disebut juga relasi asosiasi. Relasi ini berarti actor berpartisipasi pada *use case*. Simbol relasi asosiasi dapat dilihat pada Gambar 2.21.

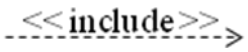
Gambar 2.21. Relasi Asosiasi

Sumber : Rostianingsih dan Yulia, (2009, hal 31)

2. Relasi antara *actor* dengan *actor*, menggunakan relasi jenis ketiga, disebut juga relasi generalisasi. Tujuan dari relasi ini agar kemampuan pada suatu *actor* dapat digeneralisasi menjadi *actor* lain.

3. Relasi antara *use case* dengan *use case*, disebut juga dependensi antar *use case*. Relasi ini dibagi menjadi tiga macam, yaitu :
   - *Include*

     Relasi Include digunakan untuk menyatakan bahwa suatu *use case* mempunyai urutan sifat dari *use case* lain. Simbol Relasi *Include* dapat dilihat pada Gambar 2.22.

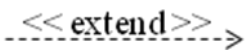


Gambar 2.22. Relasi *Include*

Sumber : Rostianingsih dan Yulia, (2009, hal 31)

- *Extend*

  Relasi *extend* menggambarkan penambahan suatu kemampuan tertentu pada situasi tertentu. Simbol relasi *extend* dapat dilihat pada Gambar 2.23.



Gambar 2.23. Relasi *Extend*

Sumber : Rostianingsih dan Yulia, (2009, hal 31)

- Generalisasi

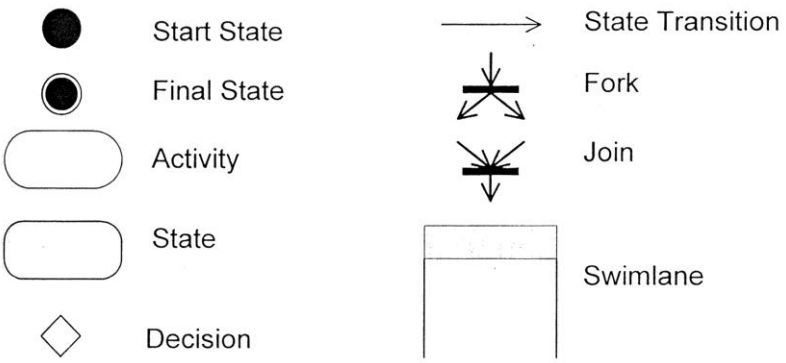
  Relasi Generalisasi menggambarkan variasi dari suatu *use case* yang *general*. *Use case parent* dapat berupa abstrak atau konkrit. Simbol Relasi Generalisasi dapat dilihat pada Gambar 2.24.

Gambar 2.24. Relasi Generalisasi

Sumber : Rostianingsih dan Yulia, (2009, hal 31)

2.4. *Activity Diagram*

Menurut Rostianingsih dan Yulia (2009), *Activity diagram* menunjukkan urutan suatu proses yang kompleks, seperti algoritma atau *workflow*. Notasi yang digunakan untuk menggambar activity diagram dapat dilihat pada Gambar 2.25.



Gambar 2.25. Notasi *Activity Diagram*

Sumber : Rostianingsih dan Yulia, (2009, hal 38)

2.4.1. *Start State*

Menurut Rostianingsih dan Yulia (2009), *Start state* digunakan untuk menunjukkan awal dari suatu *workflow*. Dalam sebuah *activity diagram* hanya diperbolehkan ada satu *start state.* Notasi yang digunakan adalah lingkaran *solid* yang selanjutnya diikuti dengan garis transisi dengan arah panah menjauhi *start state* sebagai tanda bahwa alur dimulai.

2.4.2. *Start Transition*

Menurut Rostianingsih dan Yulia (2009), *State transition* menggambarkan aktivitas apa yang dilakukan selanjutnya setelah suatu aktivitas.

2.4.3. *Activity* dan *State*

Menurut Rostianingsih dan Yulia (2009), *Activity* atau *state* menggambarkan suatu *performance* dari suatu pekerjaan dalam *workflow*. *Activity* dipresentasikan dengan persegi panjang yang mempunyai ujung bulat. *Activity* dapat di-*decompose* lebih jauh dengan *activity diagram* yang lain dan dapat diinterupsi. *State* adalah *activity state* yang tidak dapat di-*decompose* lebih jauh.

Nama *action* sebaiknya berupa gabungan kata kerja dan kata benda untuk menjelaskan apa yang terjadi dan apa yang terlibat di dalamnya, seperti misalnya *find customer, cancel listing, delete resume.*

2.4.4. *Decision*

Menurut Rostianingsih dan Yulia (2009), *Decision* adalah sebuah titik dimana terdapat suatu kondisi yang menyebabkan terjadi beberapa pilihan transisi. *Decision* direpresentasikan dengan gambar wajik.

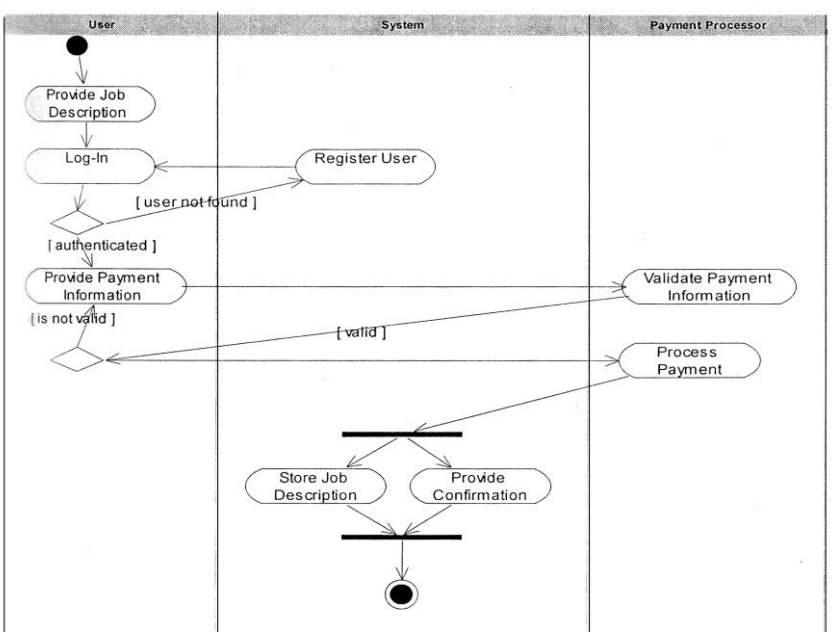
2.4.5. *Fork* dan *Join*

Menurut Rostianingsih dan Yulia (2009), *Fork* dan *join* merupakan pasangan yang menggunakan *synchronization bar* untuk menunjukkan konkurensi suatu kegiatan pada work flow suatu *use case. Synchronization bar* direpresentasikan dengan garis tebal horinsontal atau vertikal. Untuk *fork* tidak dispesifikasikan apakah *behaviour* yang ada di dalamnya berjalan simultan atau

berurutan. Sedangkan untuk *join*, artinya semua *activity* atau *state* yang masuk dalam *join* telah mencapai aktivitasnya.

2.4.6. *Swimlane*

Menurut Rostianingsih dan Yulia (2009), untuk dapat menunjukkan siapa atau apa yang bertanggung jawab dalam suatu aktivitas, digunakan *swimlane*. *Swimlane* adalah kotak pembatas dengan nama yang diletakkan di atasnya. Hal ini dilakukan untuk menunjukkan bahwa *state* atau *activity* merupakan bagian dari *swimlane* tersebut. Contoh gambar activity diagram dapat dilihat pada Gambar 2.26.



Gambar 2.26. Contoh *Activity Diagram*

Sumber : Rostianingsih dan Yulia, (2009, hal 41)

2.5. *Hypertext Markup Language* (HTML)

Menurut Peranginangin (2006), *Hypertext Markup Language* (HTML) merupakan *standard* bahasa yang digunakan untuk menampilkan dokumen *web*. Setiap dokumen HTML harus diawali dengan *tag*: <HTML> dan diakhiri tag: </HTML>. Tag tidak bersifat *case sensitive* sehingga dapat digunakan <HTML> atau <html>.

Dokumen HTML dibagi menjadi 3 bagian utama, yaitu HTML itu sendiri, kemudian HEAD dan BODY:

- HEAD

Bagian *header* ini diapit oleh *tag* <HEAD> dan </HEAD>. Pada bagian ini biasanya memuat *tag* TITLE yang menampilkan judul dari halaman aplikasi *web* tersebut. *Title* juga berguna apabila *user* ingin mem-*bookmark* halaman *web* tersebut dan berguna juga sebagai *keyword* untuk keperluan *searching*. Contoh:

```
<TITLE> Pasticceria Bakery </TITLE>
```

- BODY

Bagian *Body* digunakan untuk menampilkan isi dari web seperti *text*, tabel, gambar, dan lain-lain. Contoh:

```
<HTML>
<HEAD>
       <TITLE> Pasticceria Bakery </TITLE>
</HEAD>
<BODY bgcolor="white">
       <p> Content Here... </p>
</BODY>
</HTML>
```

2.6. *Cascading Style Sheet* (CSS)

Menurut Peranginangin (2006), *Cascading Style Sheet* (CSS) merupakan fitur yang sangat penting dalam membuat *dynamic* HTML, tetapi CSS bukanlah suatu keharusan dalam membuat suatu halaman *web*. Teknologi ini telah di-*support* pada hampir semua *web browser* karena telah di-*standart*-kan oleh *World Wide Web Consortium* (W3C) untuk digunakan di *browser*. CSS mempunyai kelebihan tersendiri yaitu sebagai *template* dari dokumen-dokumen yang digunakan, jadi apabila ada perubahan maka tidak perlu mengganti setiap dokumen akan tetapi cukup mengganti dari dokumen CSS-nya. Hal ini membuat desain pada halaman *web* kita tetap terjaga konsistensinya. Selain itu CSS dapat memberikan efek-efek spesial di dalam halaman *web* seperti pada saat suatu *text* yang akan di *link* ke halaman lainnya.

Contoh:

```
/* Isi comments untuk CSS */
/* Mengatur format text pada body */
body { font-family:Verdana, Arial; }
/* Membuat font-color H1 menjadi biru */
H1 { color:blue; }
/* Membuat efek pada text yang di-link */
/* Untuk warna link ketika posisi mouse berada diatasnya */
:hover {text-decoration: none; color: #FF0000}
/* Untuk warna link ketika link ditekan */
:active  {text-decoration: none; color: #FF0000}
/* Untuk warna link yang belum diakses */
:link {text-decoration: none; color: #FFFF00; font-family:Verdana;
 font-size:15px; font-weight:bold}
/* Untuk warna link yang telah diakses */
:visited {text-decoration: none; color: #FFFF00; font-family:
 Arial; font-size:15px; font-weight:bold}
```

2.7. *PHP Hypertext Preprocessor* (PHP)

Menurut Peranginangin (2006), *PHP Hypertext Preprocessor* (PHP) merupakan bahasa *script server-side* dalam pengembangan *web* yang disisipkan pada dokumen HTML. Penggunaan PHP memungkinkan halaman *web* bersifat dinamis sehingga *maintenance* situs *web* akan menjadi lebih mudah dan efisien.

PHP memiliki banyak kelebihan yang tidak dimiliki oleh bahasa *script* sejenis. PHP difokuskan pada pembuatan *script server-side*, yang bisa melakukan apa saja yang dapat dilakukan oleh *Common Gateway Interface* (CGI). CGI merupakan definisi *standard* untuk *interface* antara *web server* dan *external program* yang mengijinkan *external program* untuk me-*request service* dari *web server*. PHP dapat digunakan pada semua sistem operasi, antara lain: Linux, Unix (termasuk variannya HP-UX, Solaris, dan OpenBSD), Microsoft Windows, Max OS X, RISC OS. PHP juga mendukung banyak *web server*, seperti: Apache, Microsoft Internet Information Server (MIIS), Personal Web Server (PWS), Netscape and iPlanet Servers, Oreilly Website Pro Server, audium, Xitami, OmniHTTPd, dan masih banyak lagi lainnya, bahkan PHP dapat bekerja sebagai suatu CGI *processor*.

PHP tidak terbatas pada hasil keluaran HTML. PHP juga mempunyai keunggulan dalam mengolah keluaran gambar, *file* PDF, dan *movies* Flash. PHP

juga dapat menghasilkan teks seperti XHTML dan *file* XML lainnya. Salah satu fitur yang dapat diandalkan dalam PHP adalah kompatibilitasnya terhadap banyak *database*, seperti: Adabas D, dBase, Direct MS-SQL, Empress, FilePro(read only), FrontBase, Hyperwave, IBM DB2, Informix, Ingres, Interbase, MSQL, MySQL, PostgrSQL, Oracle (OCI7 dan OCI8), Unix DBM, ODBC, dan masih banyak lainnya

Contoh *syntax* sederhana PHP:

```
<?php
 Echo "test.. test..";
 Echo"<br>";
 Echo "contoh sederhana dari bahasa PHP";
?>
```

2.7.1. Keuntungan PHP

Menurut Valade (2004), saat ini, PHP sangat berkembang pesat karena memiliki banyak keuntungan. Diantaranya adalah :

- **PHP cepat**. karena PHP tertanam di dalam kode HTML dan membutuhkan waktu respon yang singkat.
- **PHP murah dan gratis**. PHP adalah bukti bahwa makan siang gratis itu memang ada dan bisa mendapatkan lebih dari yang dibayar.
- **PHP sangat mudah untuk digunakan**. PHP berisi banyak fitur khusus dan fungsi yang diperlukan untuk membuat halaman *web* dinamis. Bahasa PHP dirancang untuk dimasukkan dengan mudah di dalam *file* HTML.
- **PHP dapat berjalan di banyak sistem operasi**. PHP berjalan di berbagai macam sistem operasi, seperti : Windows, Linux, Mac OS dan sistem operasi lainnya.
- **Dukungan teknis PHP tersedia secara luas**. Sebuah basis besar pengguna menyediakan dukungan gratis melalui daftar diskusi *e-mail*.
- **PHP aman**. Pengguna tidak dapat melihat kode PHP.

- **PHP dirancang untuk mendukung *database***. PHP termasuk fungsional yang dirancang untuk berinteraksi dengan *database* tertentu. PHP dapat mengurangi kebutuhan untuk mengetahui rincian teknis yang diperlukan untuk berkomunikasi dengan *database*.
- **PHP dapat disesuaikan**. Lisensi *open source* memungkinkan programmer untuk memodifikasi perangkat lunak PHP, menambahkan atau memodifikasi fitur-fitur yang diperlukan untuk disesuaikan dengan lingkungan yang spesifik.

2.7.2. *Session*

Menurut Peranginangin (2006), *Session* adalah sebuah *identifier* dari koneksi yang terjadi dimana data yang diperlukan disimpan pada *server*. *User* tidak dapat mengaksesnya secara langsung. *Session* akan diawali ketika *user* mulai masuk suatu situs dan akan berakhir ketika *user* menutup situs yang dikunjungi tersebut. *Session* bekerja dengan cara menciptakan id unik (UID) untuk setiap *user* dan menyimpan variabel-variabel berdasarkan UID tersebut. *Session* bukanlah syarat mutlak bagi pemrograman PHP karena tidak semua aplikasi PHP memerlukan *Session*. Di bawah ini adalah dasar-dasar *Session*:

Inisialisasi *Session*:

```
<?php
  Session_start();
?>
```

Session_start ini mutlak diletakkan di bagian paling atas program sebelum coding HTML/PHP lainnya.

Membuat variabel *Session*:

```
<?php
  Session_start();
  $_SESSION['var_session']='session_content';
  /* var_session adalah nama variabel session yang dibuat */
  /*  session_content adalah isi dari variabel session yang dibuat
*/
?>
```

Menghapus *Session*:

```
<?php
  Session_start();
  /* Menghapus seluruh Session */
  Session_destroy();

  /* Menghapus Session tertentu */
  Unset($_SESSION['var_session'])
?>
```

2.8. MySQL *Database*

Menurut Valade (2004), MySQL cepat dan mudah digunakan *Relational Database Management System* (RDBMS) yang digunakan untuk *database* pada banyak situs *web*. Kecepatan adalah fokus utama dari pengembangan awal. Untuk kepentingan kecepatan, MySQL membuat keputusan untuk menawarkan fitur lebih sedikit dari pesaing utama mereka (misalnya : Oracle dan Sybase). Namun, meskipun MySQL kurang fitur penuh dari pesaing komersial, MySQL memiliki semua fitur yang dibutuhkan oleh sebagian besar pengembang *database*. Lebih mudah untuk menginstal dan digunakan daripada pesaing komersial dan selisih harga sangat mendukung MySQL.

MySQL dikembangkan, dipasarkan dan didukung oleh MySQL AB, yang merupakan perusahaan Swedia. Lisensi perusahaan itu ada dua :

- ***Open Source Software***

  MySQL tersedia melalui GNU GPL (General Public License) tanpa biaya. Siapa pun yang dapat memenuhi syarat-syarat dari GPL dapat menggunakan perangkat lunak secara gratis. Jika menggunakan MySQL sebagai *database* di situs *web* (pokok bahasan buku ini), dapat menggunakan MySQL secara gratis, bahkan jika menghasilkan uang dengan situs *web*.

- ***Commercial License***

  MySQL tersedia dengan lisensi komersial untuk mereka yang lebih memilih ke GPL. Jika pengembang ingin menggunakan MySQL sebagai bagian dari produk perangkat lunak baru dan ingin menjual

produk baru, bukan melepaskannya di bawah GPL, pengembang perlu membeli lisensi komersial dan biayanya sangat wajar.

Mencari dukungan teknis untuk MySQL tidaklah masalah, dengan cara bergabung dengan salah satu diskusi beberapa daftar *e-mail* yang ditawarkan di situs *web* MySQL di *www.mysql.com*. Bahkan dapat mencari daftar arsip *e-mail*, yang berisi basis pengetahuan besar pertanyaan dan jawaban dari MySQL. Jika lebih nyaman mendapatkan tingkat dukungan komersial, mulai dari dukungan *e-mail* langsung ke dukungan telepon, di lima tingkat harga.

2.8.1. Keuntungan dari MySQL

Menurut Valade (2004), MySQL adalah *database* populer di kalangan pengembang *web*. Kecepatan dan ukuran yang kecil membuatnya ideal untuk sebuah situs *web*. Menambah fakta bahwa itu *open source*, yang berarti gratis dan memiliki dasar popularitasnya. Di bawah ini adalah beberapa keuntungan dari MySQL :

- **MySQL cepat**. Tujuan utama dari orang-orang yang mengembangkan MySQL adalah kecepatan. Akibatnya, perangkat lunak dirancang dari awal dengan kecepatan dalam pikiran.
- **MySQL murah**. MySQL gratis di bawah lisensi GPL *open source* dan biaya untuk lisensi komersial sangat wajar.
- **MySQL sangat mudah untuk digunakan**. *User* dapat membangun dan berinteraksi dengan database MySQL dengan menggunakan laporan sederhana dalam bahasa SQL, yang merupakan bahasa standar untuk berkomunikasi dengan RDBMS.
- **MySQL dapat berjalan pada berbagai macam sistem operasi**. MySQL berjalan pada berbagai sistem operasi, contoh : Windows, Linux, Mac OS, varietas dari Unix (termasuk Solaris, AIX, dan DEC UNIX), FreeBSD, OS/2, Irix dan lain-lain.
- **Dukungan teknis MySQL tersedia secara luas**. Sebuah basis besar pengguna memberikan dukungan gratis melalui *mailing list*. Para pengembang MySQL juga berpartisipasi dalam daftar *e-mail*,

juga dapat membeli dukungan teknis dari MySQL AB untuk biaya yang sangat ringan.

- **MySQL aman**. Sistem otorisasi fleksibel MySQL memungkinkan beberapa atau semua hak *database* (misalnya hak istimewa untuk membuat *database* atau menghapus data) untuk pengguna tertentu atau kelompok pengguna. Password akan dienkripsi.
- **MySQL mendukung *database* yang besar**. *Database* MySQL menangani hingga 50 juta baris atau lebih. Batas ukuran *file default*-nya untuk sebuah tabel adalah 4 GigaBytes (GB), tetapi dapat ditingkatkan (jika sistem operasi dapat menangani itu) dengan batas teoritis 8 juta TeraBytes (TB).
- **MySQL dapat disesuaikan**. Lisensi *open source* GPL memungkinkan para programmer untuk memodifikasi perangkat lunak MySQL agar sesuai dengan lingkungan mereka sendiri yang spesifik.

2.9. Koneksi PHP ke MySQL

Menurut Valade (2004), MySQL dan PHP sering digunakan bersama. Kombinasi antara MySQL dan PHP sering disebut duo dinamis. MySQL menyediakan bagian *database* dan PHP menyediakan bagian penerapan aplikasi *database web*.

Untuk menghubungkan antara PHP dengan MySQL diperlukan inisialisasi awal nama *host*, nama *user*, *user password* dan nama *database* yang akan digunakan.

Contoh *syntax* PHP untuk koneksi dengan MySQL:

```
<?php
  $db['host'] = 'localhost';
  $db['usr'] = 'root';
  $db['pwd'] = '';
  $db['db'] = 'admin_pasticceria';
  $conn =
  mysqli_connect($db['host'],$db['usr'],$db['pwd'],$db['db'])or
  die('koneksi gagal');
  /* Die digunakan untuk menghentikan program apabila request yang
```

```
diminta gagal kemudian menampilkan pesan sesuai isi dari Die
tersebut sehingga programmer akan lebih mudah mengetahui terdapat
error di bagian mana. Hal ini sangat membantu saat aplikasi PHP
yang dibuat cukup besar. */
?>
```

Setelah inisialisasi dijalankan maka fungsi-fungsi koneksi PHP dengan MySQL dapat dijalankan. Fungsi-fungsi yang ada dapat dilihat pada Tabel 2.1.

Tabel 2.1. Fungsi-Fungsi Koneksi PHP dengan MySQL

Fungsi	Keterangan
Mysql_connect	Membuka / membuat koneksi ke suatu *server* MySQL.
Mysql_pconnect	Membuka / membuat koneksi ke suatu *server* MySQL secara *persisten*.
Mysql_close	Menutup koneksi ke *server* MySQL.
Mysql_select_db	Memilih *database* yang digunakan.
Mysql_query	Melakukan *query* ke *server* MySQL.
Mysql_fetch_array	Mengambil *record* dari *database* dan memasukkannya ke dalam *array* assosatif, array numerik, atau keduanya.
Mysql_fetch_row	Mengambil *record* dari *database* dan memasukkannya ke dalam array numerik
Mysql_fetch_field	Memperoleh informasi suatu kolom.
Mysql_num_rows	Memperoleh informasi jumlah *record* / baris data dari suatu *query*.
Mysql_num_fields	Memperoleh informasi jumlah kolom dari suatu *query*.
Mysql_create_db	Membuat database, selain menggunakan *query*.
Mysql_list_dbs	Memperoleh daftar *database*.
Mysql_list_tables	Memperoleh daftar nama tabel dari suatu *database* MySQL.
Mysql_list_fields	Memperpoleh informasi nama field dari suatu tabel pada suatu *database* MySQL.

2.9.1. Keuntungan Hubungan MySQL dan PHP

Menurut Valade (2004), MySQL dan PHP sebagai sepasang yang memiliki beberapa keunggulan, diantaranya adalah :

- **MySQL dan PHP gratis**. Sulit untuk mengalahkan efektivitas biaya gratis keduanya.
- **MySQL dan PHP berorientasi *web***. Keduanya dirancang khusus untuk digunakan di situs *web*. Keduanya memiliki seperangkat fitur yang difokuskan pada membangun situs *web* dinamis.
- **MySQL dan PHP mudah digunakan**. Keduanya dirancang untuk mendapatkan situs *web* dengan cepat.
- **MySQL dan PHP cepat**. Keduanya dirancang dengan kecepatan sebagai tujuan utama. Bersama-sama mereka menyediakan salah satu cara tercepat untuk mengirimkan halaman *web* dinamis kepada pengguna.
- **MySQL dan PHP berkomunikasi dengan baik satu sama lain**. PHP telah dibangun dalam fitur untuk berkomunikasi dengan MySQL, tidak perlu mengetahui rincian teknis, serahkan saja ke PHP.
- **Basis dukungan luas tersedia untuk MySQL dan PHP**. Keduanya memiliki basis pengguna yang besar. Karena mereka sering digunakan sebagai pasangan, mereka sering memiliki basis pengguna yang sama. Banyak orang yang bersedia untuk membantu, termasuk pada daftar diskusi *e-mail* yang berpengalaman menggunakan MySQL dan PHP bersama-sama.
- **MySQL dan PHP dapat disesuaikan**. Keduanya merupakan *open source*, sehingga memungkinkan pemrogram untuk memodifikasi perangkat lunak PHP dan MySQL agar sesuai dengan lingkungan mereka sendiri yang spesifik.

2.10. Joomla Sebagai Content Management System (CMS)

2.10.1 Content Management System (CMS)

Menurut Hakim dan Indra (2009), *Content Management System* (CMS) merupakan aplikasi untuk manajemen dokumen yang mengatur mulai dari pembuatan, pengubahan, pencarian, hingga pengarsipan. Joomla adalah CMS berbasis *web* yang digunakan untuk keperluan publikasi artikel dalam bentuk *website.*

Kelebihan CMS terletak pada kemudahannya umtuk mengatur artikel dan media, terpisah dari desain dan fungsi situs. Bahkan, pengguna CMS tidak harus mengerti banyak tentang pemrograman dan *database.*

2.10.2. Membedah Struktur Situs Joomla

Menurut Hakim dan Indra (2009), pada dasarnya, situs yang menggunakan Joomla dapat dipisahkan menjadi tiga bagian utama :

- **Menu**, merupakan bagian untuk melakukan navigasi. Menu dapat terdiri atas menu standar (*default*-nya terletak di bagian kiri), top menu dan *breadcrumbs.*
- ***Content***, merupakan isi situs yang berada di tengah-tengah halaman *web*. *Content* dalam Joomla dapat dikonfigurasikan melalui komponen.
- ***Fungsionalitas***, merupakan fitur tambahan yang dapat dibongkar-pasang (di-*install* dan di-*uninstall*). Fungsionalitas biasanya terletak di bagian pinggir halaman *web*. Fungsionalitas dalam Joomla dapat dikonfigurasikan melalui modul.

2.10.3. *Extensions*

Menurut Hakim dan Indra (2009), dalam Joomla dapat membongkar-pasang (*install* dan *uninstall*) fitur tertentu. Fitur yang dapat dibongkar-pasang pada Joomla disebut juga dengan ekstensi (*extensions*). Ada empat jenis ekstensi pada Joomla, yaitu :

- ***Module***, merupakan fitur tambahan yang biasanya diletakkan di bagian pinggir halaman *web*. Dengan modul inilah anda dapat

menambahkan fasilitas unik pada situs, seperti iklan, polling, *online*, kalender, *image slide*, dan lain-lain.

- ***Plugin***, merupakan fungsionalitas tambahan yang cenderung bersifat minor dan tidak berdiri sendiri. Plugin dapat terletak di bagian mana saja, bias di tengah maupun di pinggir halaman *web*. Contoh plugin, antara lain *pagebreak*, *rating*, *search*, dan lain-lain.
- ***Template***, merupakan desain dan *layout* pada *website* Joomla dan dapat diganti-ganti.
- ***Language***, merupakan bahasa yang digunakan pada antarmuka Joomla. *Default* adalah *English* (bahasa Inggris).

2.11. Metode FIFO (*First In First Out*)

Menurut Weigandt, Kieso, Kimmel (2005), metode FIFO mengasumsikan bahwa barang yang dibeli lebih awal adalah barang pertama yang dijual. Dengan metode FIFO biaya untuk pembelian barang yang dijual lebih dahulu ditetapkan disebut sebagai harga pokok penjualan. Contoh FIFO dapat dilihat dibawah ini.

Cost of goods available for sale

Date	*Explanation*	*Units*	*Unit Cost*	*Total Cost*
01/01	*Beginning inventory*	100	$10	$1.000
04/15	*Purchase*	200	$11	$2.200
08/24	*Purchase*	300	$12	$3.600
11/27	*Purchase*	400	$13	$5.200 +
	Total	1000		$12.000

Ending Inventory

Date	*Units*	*Unit Cost*	*Total Cost*
11/27	400	$13	$5,200
08/24	50	$12	$ 600 +
	450		$5,800

Sumber : Weigandt, Kieso, Kimmel, (2005, hal. 237)

2.12. Harga Pokok Produksi

Menurut Mulyadi (2005), dalam produksi suatu barang terdapat 2 jenis biaya yaitu biaya produksi dan biaya non produksi. Biaya produksi merupakan biaya-biaya yang dikeluarkan dalam pengolahan bahan baku menjadi barang jadi. Sedangkan biaya non produksi merupakan biaya-biaya yang dikeluarkan untuk kegiatan non produksi, yang meliputi biaya pemasaran dan biaya administrasi. Harga pokok produksi hanya terdiri dari biaya produksi saja. Harga pokok produksi terdiri dari unsur-unsur biaya produksi berikut ini :

- Biaya bahan baku langsung *(direct material costs)*

  Biaya yang dikeluarkan untuk membeli bahan baku yang berhubungan langsung dengan produk yang dihasilkan oleh pabrik. Bahan baku merupakan bahan dasar yang dipakai untuk membentuk barang jadi, yang diolah dalam pabrik, dapat diperoleh dari pembelian atau pengolahan sendiri.

- Biaya tenaga kerja langsung *(direct labor costs)*

  Biaya yang dikeluarkan untuk membayar tenaga kerja yang berhubungan langsung dari pengolahan bahan baku menjadi produk jadi selama proses produksi.

- Biaya *overhead* pabrik *(manufacture overhead costs)*

  Biaya *overhead* pabrik adalah semua biaya produksi pabrik selain bahan baku dan biaya tenaga kerja langsung. Yang termasuk dalam biaya *overhead* pabrik antara lain: biaya bahan penolong, biaya tenaga kerja tidak langsung, biaya listrik pabrik, maupun biaya-biaya lain yang telah ditentukan perusahaan sebagai biaya *overhead*.

- *Cost driver*

  Cost driver adalah sebuah faktor, misalnya aktifitas atau *volume*, yang mempengaruhi biaya-biaya, terutama untuk biaya-biaya yang berubah-ubah (*variable costs*) karena adanya pertambahan aktifitas atau *volume*, misalnya: biaya listrik akan semakin membengkak bila mesin digunakan dalam waktu yang lama. Maka *cost driver*-nya adalah jam pemakaian mesin.

2.13. Laporan Laba Rugi

Menurut Dunia (2005), laporan laba rugi (*income statement*) merupakan ikthisar dari pendapatan (*revenue*) dan beban-beban (*expenses*) untuk suatu periode waktu atau masa tertentu, misalnya sebulan atau setahun. Dengan kata lain, laporan ini menunjukkan hasil usaha atau kinerja perusahaan dalam kurun waktu tertentu.

2.14. Interaksi Manusia Komputer

Menurut Jauari (2007), Pada awal tahun 70-an, mulai muncul isu teknik antarmuka pemakai (*user interface*) yang diketahui sebagai Man Machine Interaction (MMI) atau Interaksi Manusia Mesin. Pada Man Machine Interaction sudah diterapkan sistem yang "*user friendly*". Namun, sifat user friendly pada MMI ini diartikan secara terbatas. Sistem tersebut hanya menitik beratkan pada aspek rancangan antarmukanya saja, sedangkan faktor-faktor atau aspek-aspek yang berhubungan dengan pemakai baik secara organisasi atau individu belum diperhatikan. Pada pertengahan tahun 80-an diperkenalkanlah istilah Human Computer Interaction (HCI) atau Interaksi Manusia Komputer. Pada HCI ini cakupan atau fokus perhatiannya lebih luas, tidak hanya berfokus pada rancangan antarmuka saja, tetapi juga memperhatikan semua aspek yang berhubungan dengan interaksi antara manusia dan komputer. Dengan demikian terlihat jelas bahwa fokus perhatian HCI tidak hanya pada keindahan tampilannya saja atau hanya tertuju pada tampilan antarmukanya saja, tetapi juga memperhatikan aspek-aspek pamakai, implementasi sistem rancangannya dan fenomena lingkungannya, dan lainnya.