ANALISA KONSEP *OBJECT ORIENTED PROGRAMMING* PADA BAHASA PEMROGRAMAN PHP

Kadek Wibowo
AMIK Bina Sarana Informatika
Jl. Rs Fatmawati No. 24, Pondok Labu, Jakarta Selatan
Email : kadek.kwo@bsi.ac.id

ABSTRACT

In making computer application an important role is the language of programming, without any programming language not be called off computer application desired. Of all languages programming computers that is , programming language based object be a language very popular of creation program computer application. A programming language this is pretty flexible and easy to is modified into an a display that is interesting windows based. Programming language oriented object or object-oriented programming (oop) is an approach programming use object and class. Php at first the script is just a collection of simple. With progress in , then added a variety of programming features oriented object. This was initiated since php 4. With the emergence of php 5 , programming features oriented object the more steadily and the sooner. With php 5 , script using the concept object-oriented will be quickly and more efficient.

Key Word : programming languages, object-oriented programming (OOP)

1. PENDAHULUAN

Secara garis besar, bahasa pemrograman komputer adalah sebuah alat yang dipakai oleh para programmer komputer untuk menciptakan program aplikasi yang digunakan untuk berbagai macam keperluan. Pada tahap awal dikenal beberapa jenis bahasa pemrograman, bahasa ini berbasis teks dan berorientasi linear contohnya : Bahasa BASIC, Bahasa Clipper, Bahasa Pascal, Bahasa cobol.

Pemrograman berorientasi objek atau *object-oriented programming* merupakan suatu pendekatan pemrograman yang menggunakan *object* dan *class*. Saat ini konsep OOP sudah semakin berkembang. Hampir semua programmer maupun pengengmbang aplikasi menerapkan konsep OOP. OOP bukanlah sekedar cara penulisan sintaks program yang berbeda, namun lebih dari itu, OOP merupakan cara pandang dalam menganalisa sistem dan permasalahan pemrograman. Dalam OOP, setiap bagian dari program adalah *object*. Sebuah *object* mewakili suatu bagian program yang akan diselesaikan.

Beberapa konsep OOP dasar, antara lain:

a. *Encapsulation (Class dan Object)*
b. *Inheritance* (Penurunan sifat)
c. *Polymorphisme*

PHP khususnya PHP 5 sudah mendukung beberapa konsep OOP. Akan tetapi PHP 5 tidak mendukung konsep *Multiple-inheritance* dikarenakan konsep Multiple-inheritance terdapat di bahasa pemrograman bahasa C.

Tujuan diadakan penelitian ini adalah Untuk menggambarkan konsep pemrograman berorientasi objek terhadap bahasa pemrograman lain khususnya PHP. Dengan adanya konsep pemrograman berorientasi objek, pada bahasa pemrograman PHP bisa mempermudah para programmer PHP diseluruh dunia dapat lebih mudah berbagi teknik programing. Kita bisa membuat suatu class dan programmer lain dapat dengan mudah menggunakannya tanpa perlu mengetahui proses jalannya class tersebut.

Ruang lingkup pada penelitian ini lebih mengarah pada penjelasan konsep

pemrograman OOP apa saja yang bisa dan belum bisa berjalan pada bahasa pemrograman PHP.

2. KAJIAN LITERATUR

a. Menurut Gata (2012:7) OOP (Object Oriented Programing "merupakan suatu cara atau paradigm pemrograman yang berorientasikan kepada objek".
b. PHP (*PHP Hypertext Preprocessor*) menurut Sutaji (2012:2) adalah "kode/skrip yang akan dieksekusi pada server side".
c. Menurut Sidik (2012:518) Class di dalam PHP secara sederhana "adalah kumpulan variabel dan fungsi dalam suatu variabel".
d. Hasil dari pendefinisian suatu class yang lain disebut sebagai subclass. Definisi subclass dapat juga digunakan menjadi subclass yang lain. "Proses pendefinisian suatu class berdasarkan class yang lain disebut dengan pewarisan (inheritance)" menurut Sidik (2012:526).
e. Menurut Aziz (2005:23) Constructor adalah method yang akan dipanggil pertama kali setiap pembuatan sebuah object dari suatu class.
f. Menurut Aziz (2005:23) Destructor adalah method yang akan dipanggil terarkhir kali setiap pembuatan sebuah object dari suatu class.
g. Menurut Thamura dkk (2006:37) Model dalam MVC adalah "sebuah layer yang lebih dekat ke sumber data, baik itu berupa database, webservice, atau file system.
h. Menurut Thamura dkk (2006:19) viewer "merupakan bagian khusus untuk menangani *layer presentation*".
i. Menurut Thamura dkk (2006:65) controller adalah "sebuah layer yang bekerja untuk mengurusi urusan antar layer, yang artinya bertanggung jawab terhadap eksekusi aplikasi.
j. Menurut Indrajani dan Martin konsep OOP mengenai Enkapsulasi adalah "suatu mekanisme untuk menyembunyikan atau memproteksi suatu proses dari kemungkinan interfensi atau penyalahgunaan dari luar sistem sekaligus menyederhanakan penggunaan sistem itu sendiri".
k. Menurut Gata (2012:10) Polymorphism adalah suatu kemampuan sebuah variable reference untuk merubah behavior sesuai dengan apa yang dipunyai object.
l. Menurut Bogdan dan Bikien (1982:12) Studi kasus merupakan pengujian secara rinci terhadap satu latar atau satu orang subjek atau satu tempat penyimpanan dokumen atau suatu peristiwa tertentu.
m. Menurut Creswell (1998:21) penelitian studi kasus adalah penelitian yang menempatkan sesuatu atau obyek yang diteliti sebagai 'kasus'. Tetapi, pandangan tentang batasan obyek yang dapat disebut sebagai 'kasus' itu sendiri masih terus diperdebatkan hingga sekarang.

3. METODE PENELITIAN

3.1. Metode Penelitian Studi Kasus

Menurut Bogdan dan Bikien (1982) studi kasus merupakan pengujian secara rinci terhadap satu latar atau satu orang subjek atau satu tempat penyimpanan dokumen atau suatu peristiwa tertentu.

Selama sekitar lima belas tahun lebih, tepatnya sejak tahun 1993, seiring dengan semakin populernya penelitian studi kasus, banyak pengertian penelitian studi kasus telah dikemukakan oleh para pakar tentang penelitian studi kasus (Creswell, 1998).

Secara umum, pengertian-pengertian tersebut mengarah pada pernyataan bahwa, sesuai dengan namanya, penelitian studi kasus adalah penelitian yang menempatkan sesuatu atau obyek yang diteliti sebagai 'kasus'. Tetapi, pandangan tentang batasan obyek yang dapat disebut sebagai 'kasus' itu sendiri masih terus diperdebatkan hingga sekarang.

Perdebatan ini menyebabkan perbedaan pengertian di antara para ahli tersebut.

3.2. Langkah-Langkah Penelitian Studi Kasus

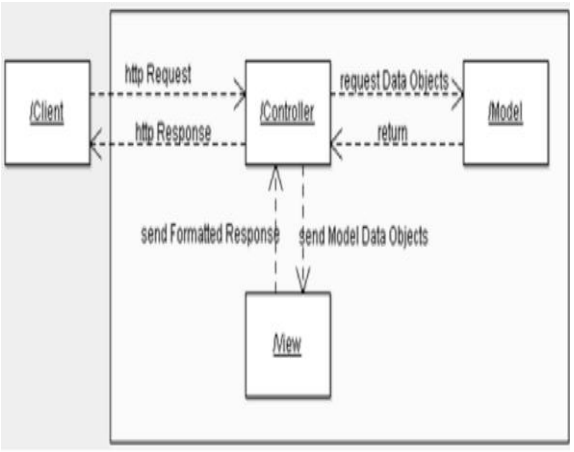
a. Pemilihan Kasus
   Dalam pemilihan kasus hendaknya dilakukan secara (purposive) dan bukan secara rambang. Kasus dapat dipilih oleh peneliti dengan menjadikan objek orang, lingkungan, program, proses dan masyarakat.
b. Analisa data
   setelah data terkumpul peneliti dapat mulai mengagregasi, mengorganisasi, dan mengklasifikasi data menjadi unit-unit yang dapat dikelola. Agregasi merupakan proses mengabstraksi hal-hal khusus menjadi hal-hal umum guna menemukan pola umum data. Data dapat diorganisasi secara kronologis, kategori atau dimasukkan ke dalam tipologi. Analisis data dilakukan sejak peneliti di lapangan, sewaktu pengumpulan data dan setelah semua data terkumpul atau setelah selesai dan lapangan.
c. Perbaikan
   meskipun semua data telah terkumpul, dalam pendekatan studi kasus hendaknya clilakukan penvempurnaan atau penguatan (reinforcement) data baru terhadap kategori yang telah ditemukan. Pengumpulan data baru mengharuskan peneliti untuk kembali ke lapangan dan barangkali harus membuat kategori baru, data baru tidak bisa dikelompokkan ke dalam kategori yang sudah ada.
d. Penulisan Laporan
   laporan hendaknya ditulis secara komunikatif, rnudah dibaca, dan mendeskripsikan suatu gejala atau kesatuan sosial secara jelas, sehingga rnernudahkan pembaca untuk mernahami seluruh informasi penting. Laporan diharapkan dapat membawa pembaca ke dalam situasi kasus kehiclupan seseorang atau kelompok

4. PEMBAHASAN

4.1. MVC *(Model View Controllers)*

MVC merupakan suatu konsep yang memungkinkan pengerjaan *web* antara logika dan presentasi tampilan *web* dilakukan secara terpisah. Metode MVC sudah banyak diterapkan dan digunakan dalam aplikasi yang mendukung sistem, salah satu di antaranya adalah perancangan dan implementasi perangkat lunak dengan menerapkan arsitektur MVC (Model View Controller). MVC adalah sebuah metode pengembangan aplikasi dengan membagi aplikasi menjadi 3 bagian:

a. **Bagian Model** adalah bagian yang bertugas mengolah data atau memanipulasi data sesuai dari bisnis proses yang terjadi pada data tertentu.
b. **Bagian View** adalah bagian yang mempresentasikan data dalam bentuk tampilan dan menuntun alur interaksi user terhadap aplikasi.
c. **Bagian Controller** adalah bagian yang menhubungkan antar bagian model dengan bagian view dan bertanggung jawab mengatur alur transisi antar kedua bagian tersebut. Berikut gambar metode MVC :



Gambar 1. Metode MVC

4.2. Dasar MVC

MVC *(Model View Controller) pattern* adalah sebuah *pattern* yang banyak digunakan untuk membangun aplikasi web saat ini. MVC pattern terbagi menjadi 3 modul, Model, View, dan Cotroller. Contoh

sederhana penerapan MVC pada aplikasi web dengan PHP, pertama yang kita lakukan adalah mendefinisikan model.

a. Model

Model adalah layer yang bertanggung jawab untuk melakukan hubungan dengan database, untuk contoh kali ini tidak menggunakan database, asumsi kita adalah layer model telah berhasil mendapatkan data dari database. Contoh:

```
<?
include_once'book.php';

class model {

	public function getData()
	{
	return array(
			new book('Pemrograman PHP & MySql','Bayu
Priyambadha','UB Press','2011'),
			new book('Pemrograman Framework MVC dengan
PHP','Widhy','UB Press','2011'),
			new book('Membangun Aplikasi Web dengan PHP dan
AJAX','Achmad Arwan','UB Press','2012'),
			new book('Kolaborasi PHP, MVC dan AJAX','Bayu
Priyambadha','UB Press','2012')
			);
	}
}

?>
```

Gambar 2. Model

b. *View*

Untuk layer *view*, kita hanya akan mendefinisikan sebuah template HTML sebagai tempat untuk menampilkan data. Berikut adalah skrip HTML untuk layer *view:*

```
<html>
	<head>
		<title>MVC dengan PHP</title>
	</head>
	<body>
		<center>

			<?=$data?>

		</center>
	</body>
</html>
```

Gambar 3. *View*

c. *Controller*

Sebagai layer yang berfungsi sebagai *"play maker"*, *controller* harus mempunyai akses ke model dan view. Berikut adalah kode untuk controller:

```
<?
include_once'model.php';

class controller {

	function invoke()
	{
		$model_data = new model();
		$row_data = $model_data->getData();

		$data ="<table border=1>

	<tr><th>JUDUL</th><th>PENGARANG</th><th>PENERBIT</th><th>TAHUN</th
></tr>";
		foreach ($row_data as $key => $value) {
			$data .="<tr><td>".$value->judul."</td><td>".$value-
>pengarang."</td><td>".$value->penerbit."</td><td>".$value-
>tahun."</td></tr>";
		}
		$data .="</table>";
		include('view.php');
	}
}

?>
```

Gambar 4. *Controller*

Controller mempunyai metode invoke, dimana di metode tersebut proses penyataan data dari model dan *view* digabungkan.

Setelah ketiga layer selesai dibuat, maka tahap akhir adalah membuat file index.php, dimana file ini adalah sebagai file penghubung yang diakses pertama kali user melakukan request. Berikut adalah file index.php:

```
<?
	include_once'controller.php';

	$main_ctrl = new controller();
	$main_ctrl->invoke();
?>
```

Gambar 5. *File* index.php

JUDUL	PENGARANG	PENERBIT	TAHUN
Pemrograman PHP & MySql	Bayu Priyambadha	UB Press	2011
Pemrograman Framework MVC dengan PHP	Widhy	UB Press	2011
Membangun Aplikasi Web dengan PHP dan AJAX	Achmad Arwan	UB Press	2012
Kolaborasi PHP, MVC dan AJAX	Bayu Priyambadha	UB Press	2012

Done | Internet | Protected Mode: On | 100%

Gambar 6. Hasil akhir

4.3. Konsep OOP yang bisa berjalan di PHP

a. *Class*

Class merupakan gambaran dari sebuah object atau bisa dikatan output dari sebuah object. Pada bahasa pemrograman class merupakan sekumpulan kode yang dituliskan untuk mendefinisikan property dan metod yang ada pada sebuah object. Berikut adalah contoh script PHP untuk membuat class:

```
1  <?php
2      class coba
3      {
4
5      }
6  ?>
```

Gambar 7. Pembuatan class di PHP

Class didefinisikan dengan menampung nilai property dan metod, dimana properti adalah sebuah data yang menjelaskan tentang class dan metode adalah tingkah laku yang bisa dilakukan oleh *object*. Berikut adalah contoh script PHP pembuatan class beserta properti dan metodnya:

```
1  <?php
2      class coba
3      {
4      $nama; //ini properti
5      function bilangHallo() //ini metod
6      {
7          echo"Hallo $this->nama";
8      }
9      }
10 ?>
```

Gambar 8. Pembuatan class serta properti dan metodnya

b. *object*

object adalah hasil instaNsiasi dari class, dan mengandung seluruh *resource* yang telah didefinisikan pada class. Berikut cara meng-instansiasi object dari class yang sudah didefinisikan.

```
1 <?php
2 require_once('file-class.php'); //file yang memuat class
3 $ob=new coba(); //proses instansiasi object
4 ?>
```

Gambar 9. Proses instansiasi *object* dari *class*

Dikarenakan class merupakan cetakan dari *object*, maka object hasil instansiasi juga mempunyai *resource* seperti *class*. Berikut contoh kode memanggil properti dan metodenya:

```
1 <?php
2 require_once('file-class.php'); //file yang memuat class
3 $ob=new coba(); //proses instansiasi object
4 $ob->nama="Kadek Wibowo"; //mendefinisikan nilai properti
5 echo $ob->nama; //memanggil properti
6 $ob->bilangHallo();//menjalankan metode
7 ?>
```

Gambar 10. Perintah memanggil properti dan metode

c. *Encapsulation*

Encapsulation adalah mekanisme "membungkus" sebuah data pada sebuah object. Dalam istilah lain seringkali disebut "Information Hiding". Pada PHP terdapat 3 modifier yang dapat diimplementasikan untuk melakukan "pembungkusan" data yaitu *private*, *protacted* dan *public*. Modifier tersebut digunakan untuk mendefinisikan tingkat visibilitas sebuah data(properti) atau fungsi (metode) yang ada di dalam class.

```php
<?php
class orang
{
private $_nama
private $_umur
    function __construction($name, $age=0)
    {
        if(!is_int($age))
        {
            throw new Exception("Cannot assign non integer value to integer field, 'Age'");
        }
        $this->_umur=$umur;
        $this->_nama=$nama;
    }
    public function setUmur($umur)
    {
        if(!is_int($umur))
        {
            throw new Exception("Cannot assign non integer value to integer field, 'Age'");
        }
        $this->_umur=$umur;
    }
    public function yearsToRetire()
    {
        return 67 - $this->_umur;
    }
}
$orang= new Orang ("Kadek");
$orang->setUmur(24);
echo $orang->yearsToRetire()
?>
```

Gambar 11. Kode encapsulation

d. *Polymorphisme*

Polymorphism membuat objek-objek yang berasal dari subclass yang berbeda, diperlakukan sebagai objek-objek dari satu superclass. Hal ini terjadi ketika memilih method yang sesuai untuk diimplementasikan ke objek tertentu berdasarkan pada subclass yang memiliki method bersangkutan.

Kondisi yang harus dipenuhi supaya polimorfisme dapat diimplementasikan adalah:

1) *Method* yang dipanggil harus melalui *variable* dari basis *class* atau *superclass*.
2) *Method* yang dipanggil harus juga menjadi *method* dari basis *class*.
3) *Signature* (argument/parameter) *method* harus sama baik pada *superclass* maupun *subclass*.
4) *Method acces attribute* pada *subclass* tidak boleh lebih terbatas dari basic *class*.

```php
<?php
interface halointf
{
    public function halloOOP();
}
class halo implements halointf {

    public function halloOOP()
    {
        return"Kadek";
    }

}
class halo2 extends halo implements halointf
{
    public function halloOOP()
    {
        return"Windu Gata";
    }
    public function halloOOP2()
    {
        return"Bakalan Eror";
    }
}
function ngetest (halointf $test)
{
    echo $test->halloOOP();
}
$biodata= new halo();
$biodata->halloOOP();
ngetest($biodata);
?>
```

Gambar 12. Kode *polymorphisme*

e. *Constructor* dan *Destructor*

PHP memungkinkan pengembangan untuk menyatakan metode konstuktor untuk sebuah class. Class yang memiliki metode konstruktor memanggil metode ini pada setiap objek yang baru dibentuk (diinstansiasi), diperlukan iniliasisasi seblum objek digunakan. Berikut contoh kode konstruktor:

```php
<?php
class orang {
    function_construct() {
    print "Konstruktor Person \n";
    }
}
$ob=new orang();
?>
```

Gambar 13. Kode konstruktor

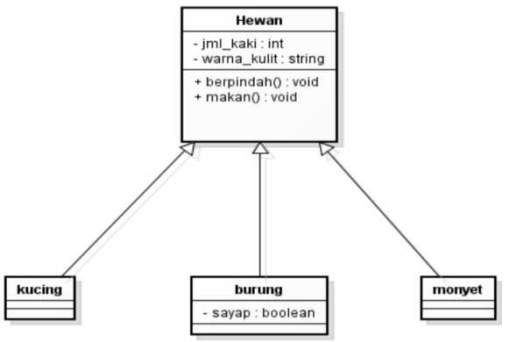
PHP sama konsepnya untuk *destructor* dengan bahasa berorientasi objek yang lain seperti C++. *Destructor* akan segera dipanggil setelah tidak ada referensi lain. untuk objek tertentu. Berikut contoh kode *destructor* pada PHP:

```php
<?php
class orang {
    function_destruct() {
    print "Destruktor Orang \n";
    }
}
$ob=new orang();
?>
```

Gambar 14. Kode konstruktor

f. *Inheritance*

Dalam pemrograman berorientasi objek (OOP), *Inheritance* (pewarisan) adalah cara untuk menggunakan kembali kode objek yang ada, atau untuk mendirikan *subtype* dari objek yang sudah ada, atau keduanya, tergantung pada dukungan bahasa pemrograman. Berikut contoh kasus *inheritance*:



Gambar 15. *Class diagram*

g. Final *Keyword*

PHP memperkenalkan "final" *keyword*, dimana ini akan mencegah proses *overidding method* pada *class* anak (sub-class) hal ini dapat kita terapkan pada metode dan *class*. Apabila metode kita beri status final, maka metode tersebut tidak akan bisa dioverride, begitu juga pada class, apabila kita berikan status "final" pada deklarasi *class* maka *class* tersebut tidak bisa diwariskan.

h. *Class Abstraction*

PHP memperkenalkan *abstract class* dan *abstract* metod. *Class* yang mendefinisikan sebagai *abstract* tidak bisa diintasiasi, dan *class* yang terdiri paling tidak satu metod abstract harus didefinisikan sebagai *abstract class*. *Class abstract* hanya bisa mewariskan *resourcenya.*

pada *class* anaknya. Setiap *class* yang mewarisi *class* abstract, wajib menuliskan seluruh metod abstract yang dipunyai oleh *parent class*-nya (*superclass*).

i. *Object Interfaces*

Object Interface memungkinkan kita untuk membuat kode yang menentukan metod mana yang akan diimplementasikan tanpa harus mendefinisikan bagaimana metod tersebut akan bekerja (hanya nama metod saja). *Interface* didefinisikan dengan *"Interface" keyword*, mirip dengan deklarasi class biasa, hanya saja definisi atau detail metod tidak dituliskan. Seluruh metod yang dideklarasikan pada interface harus memiliki modifier *"public"*.

Untuk mengimplementasikan sebuah i*nterface*, kita dapat menggunakan *"implement" keyword*. Seluruh metod yang ada pada *interface* harus diimplementasikan seluruhnya. Sebuah *class* bisa mengimplementasikan lebih dari satu *interface.* Catatan :

1. *Class* tidak bisa mengimplementasikan dua *interface* yang mempunyai nama metod yang sama.

2. *Interface* bisa diwariskan seperti *class* menggunakan "*extends*".

Class yang mengimplementasikan interface harus menggunakan metod-metod yang ada pada *interface* tersebut dengan nama dan spesifikasi yang sama persis.

4.4. Konsep OOP yang belum bisa berjalan di PHP

a. *Multiple-inheritance*.

Pada konsep OOP *multiple-inheritance* sebuah *class* turunan yang mewarisi lebih dari satu *class* induk (*join*). Adapun code PHP dengan konsep OOP *multiple-inheritance* sebagai berikut :

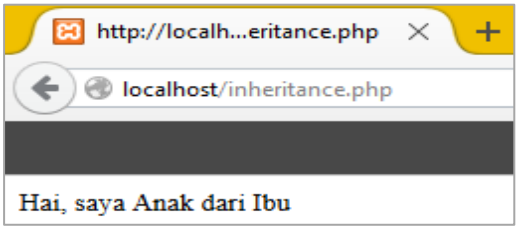
```php
<?php

class Ibu {
var $nama="Ibu";

function Ibu ($n){
    $this->nama=$n;
}
function Hai(){
    echo "Hai, saya $this->nama <br>";
 }
}
class Anak extends Ibu {
}
$test=new Anak ("Anak dari Ibu");
$test->Hai();
?>
```

Gambar 16. Code PHP dengan konsep *multiple-inheritance*

Hasil tampilan dari listing program di atas adalah "Hai, Saya Anak dari Ibu" dan bukannya "Hai, Saya Ibu" di dalam *class* Ibu didefinisikan variabel nama dengan nilai Ibu, selanjutnya kita membuat objek dari *class* Anak yang merupakan turunan dari *class* Ibu. Lihat bahwa instantiasi sekaligus mengisikan parameter baru "Anak dari Ibu", sehingga ketika dipanggil maka mengisikan $this->nama dengan parameter tersebut.



Gambar 17. *Running* konsep *multiple-inheritance*

5. KESIMPULAN

a. Perkembangan perangkat lunak komputer khususnya bahasa pemrograman berorientasi berbasis objek adalah pilihan untuk membangun sebuah aplikasi saat ini.
b. PHP khususnya PHP 5 sudah mendukung beberapa konsep OOP. *Class* turunan selalu bergantung pada base *class* tunggal, dan sampai dengan hari ini PHP belum mendukung konsep OOP *Multiple-inheritance*.
c. MVC (Model *View Controller)* sebuah konsep yang saat ini sudah banyak digunakan untuk mendasain suatu web. MVC memungkinkan pengerjaan *web* antara logika dan presentasi tampilan *web* dilakukan secara terpisah.

DAFTAR PUSTAKA

Aziz, M. Farid. 2005. Object Oriented Programing dengan PHP5, Jakarta: PT Elex Media Komputindo.

Bogdan, R. & Biklem, S. 1982. *Qualitative research for education*: An indtroduction to theory and methods. Boston, MA:Allyn and Bacon.

Creswell, J. 1998. *Qualitative inquiry and research design*: choosing among five traditions. Thousand Oaks, CA: Sage Publications, Inc.

Gata, Windu. 2012. Asiknya Mengenal Java, Jakarta: PT Elex Media Komputindo.

Indrajani dan Martin. 2007. Pemrograman Berbasis Objek Dengan Bahasa Java, Jakarta: PT Elex Media Komputindo.

Sidik, Betha. 2012. Pemrograman Web dengan PHP, Bandung: Informatika.

Sutaji, Deni. 2012. Sistem Inventory Mini Market dengan PHP & JQUERY, Yogtakarta: Lokomedia.

Thamura, Frans., Heryanto, Leo., Dan Muhardin Endy. 2006. Cara Cepat Mengembangkan Aplikasi Java dengan Metode MVC, Seri Enterprise Opensource, Jakarta: Bambumas.