**BAB V**

**CLASSDAN *CONSTRUCTOR***

* 1. Tujuan

1. Praktikan dapat membedakan mana yang disebut *class* dan mana yang disebut *constructor*.
2. Praktikan mampu memahami konsep *class* dan mampu mengaplikasikanya.
3. Praktikan mampu memahami konsep konstruktor dan mampu mengaplikasikanya.
4. Praktikan dapat memahami kegunaan dan fungsi dari pemakaian *class* serta *constructor.*
5. Praktikan dapat membuat sebuah program dari pemakaian *class* serta *constructor.*
   1. Dasar Teori
      1. **Pemrograman OOP**

OOP (*Object Oriented Programming*) adalah suatu metode pemrograman yang berorientasi kepada objek. Tujuan dari OOP diciptakan adalah untuk mempermudah pengembangan program dengan cara mengikuti model yang telah ada di kehidupan sehari-hari. Jadi setiap bagian dari suatu permasalahan adalah objek, nah objek itu sendiri merupakan gabungan dari beberapa objek yang lebih kecil lagi. Saya ambil contoh Pesawat, Pesawat adalah sebuah objek. Pesawat itu sendiri terbentuk dari beberapa objek yang lebih kecil lagi seperti mesin, roda, baling-baling, kursi, dll. Pesawat sebagai objek yang terbentuk dari objek-objek yang lebih kecil saling berhubungan, berinteraksi, berkomunikasi dan saling mengirim pesan kepada objek-objek yang lainnya. Begitu juga dengan program, sebuah objek yang besar dibentuk dari beberapa objek yang lebih kecil, objek-objek itu saling berkomunikasi, dan saling berkirim pesan kepada objek yang lain.

Konsep OOP

* Abstrak *Class*
  1. Kelas merupakan deskripsi abstrak informasi dan tingkah laku dari sekumpulan data.
  2. Kelas dapat diilustrasikan sebagai suatu cetak biru (*blueprint*) atau prototipe yang digunakan untuk menciptakan objek.
  3. Kelas merupakan tipe data bagi objek yang mengenkapsulasi data dan operasi pada data dalam suatu unit tunggal.
  4. Kelas mendefinisikan suatu struktur yang terdiri atas data kelas (*data field*), prosedur atau fungsi (*method*), dan sifat kelas (*property*).
* *Encapsulation*

1. Istilah enkapsulasi sebenarnya adalah kombinasi data dan fungsionalitas dalam sebuah unit tunggal sebagai bentuk untuk menyembunyikan detail informasi.
2. Proses enkapsulasi memudahkan kita untuk menggunakan sebuah objek dari suatu kelas karena kita tidak perlu mengetahui segala hal secara rinci.
3. Enkapsulasi menekankan pada antarmuka suatu kelas, atau dengan kata lain bagaimana menggunakan objek kelas tertentu.
4. Contoh: kelas mobil menyediakan antarmuka fungsi untuk menjalankan mobil tersebut, tanpa kita perlu tahu komposisi bahan bakar, udara dan kalor yang diperlukan untuk proses tersebut.

* *Inheritance*

1. Kita dapat mendefinisikan suatu kelas baru dengan mewarisi sifat dari kelas lain yang sudah ada.
2. Penurunan sifat ini bisa dilakukan secara bertingkattingkat, sehingga semakin ke bawah kelas tersebut menjadi semakin spesifik.
3. Sub kelas memungkinkan kita untuk melakukan spesifikasi detail dan perilaku khusus dari kelas supernya.
4. Dengan konsep pewarisan, seorang programmer dapat menggunakan kode yang telah ditulisnya pada kelas super berulang kali pada kelas-kelas turunannya tanpa harus menulis ulang semua kodekode itu.

* *Polymorphism*

1. Polimorfisme merupakan kemampuan objek-objek yang berbeda kelas namun terkait dalam pewarisan untuk merespon secara berbeda terhadap suatu pesan yang sama.
2. Polimorfisme juga dapat dikatakan kemampuan sebuah objek untuk memutuskan *method* mana yang akan diterapkan padanya, tergantung letak objek tersebut pada jenjang pewarisan.
3. *Method overriding*.
4. *Method name overloading*.

*(sumber :* [*https://idcloudhost.com/panduan/mengenal-pengertian-dan-konsep-oop/*](https://idcloudhost.com/panduan/mengenal-pengertian-dan-konsep-oop/)*)*

Kelebihan pemrograman berorientasi objek adalah

1. *Multiplatform*. Pada hal ini yang dimaksud dengan *multiplatform* adalah dapat dijalankan diberbagai sistem operasi sekaligus. Dengan menggunakan prinsip tulis sebuah program kemudian dikompail dapat dijalankan disegala program sesuai *platform* yang ada.
2. OOP (*Object Oriented Programming*) maksudnya adalah semua aspek yang terdapat di java berorientasi objek. Dalam hal ini sangat memudahkan dalam hal membuat program karena semua berdasarkan objek atau class, serta memudahkan dalam hal mendesain, mencari kesalahan dan juga mengembangkan program yang telah kita rencanakan atau kita buat.
3. Bergaya C++. Karena *syntax* dalam pembuatan program yang terdapat di java sama persis dengan C++ maka banyak pemrogram yang bermula dari menggunakan C++ banyak beralih menggunakan pemrograman yang berbasis objek (dalam hal ini adalah java).
4. Perpustakaan kamus bahasa yang lengkap. Artinya dalam java terdapat kamus bahasa yang lengkap untuk memudahkan pemrogram untuk membangun dan menggembangkan program yang telah dibuat.
5. Pengumpulan sampah yang otomatis. Memiliki fasilitas pengaturan memori sehingga para pemrogram tidak perlu lagi repot mengatur memori yang akan digunakannya.

Kekurangan pemrograman berorientasi objek adalah

1. Mudah didekomplikasi. Dekomplikasi adalah proses membalikan kode menjadi kode sumber.
2. Penggunaan memori yang banyak.
3. Tulis sekali - perbaikan dimana saja.

*(sumber :* [*http://mikael23.blogspot.com/2012/04/kelebihan-dan-kekurangan-pemrograman.html*](http://mikael23.blogspot.com/2012/04/kelebihan-dan-kekurangan-pemrograman.html)*)*

* + 1. ***Class***

*Class* adalah ‘cetak biru’ atau ‘*blueprint’* dari objek. *Class* digunakan hanya untuk membuat kerangka dasar. Yang akan kita pakai nantinya adalah hasil cetakan dari *class*, yakni *object*.

Sebagai analogi, *class* bisa diibaratkan dengan laptop atau *notebook*. Kita tahu bahwa laptop memiliki ciri-ciri seperti merk, memiliki *keyboard*, memiliki *processor*, dan beberapa ciri khas lain yang menyatakan sebuah benda tersebut adalah laptop. Selain memiliki ciri-ciri, sebuah laptop juga bisa dikenakan tindakan, seperti: menghidupkanlaptop atau mematikan laptop.

*Class* dalam analogi ini adalah gambaran umum tentang sebuah benda. Di dalam pemrograman nantinya, contoh class seperti: *koneksi\_database* dan *profile\_user*.

Di dalam PHP, penulisan *class* diawali dengan *keyword* ‘*class*’, kemudian diikuti dengan *nama dari class*. Aturan penulisan nama *class* sama seperti aturan penulisan variabeldalam PHP, yakni diawali dengan huruf atau *underscore* untuk karakter pertama, kemudian boleh diikuti dengan huruf, *underscore* atau angka untuk karakter kedua dan selanjutnya. Isi dari *class* berada dalam tanda kurung kurawal.

*(sumber :* [*https://www.duniailkom.com/tutorial-belajar-oop-php-pengertian-class-object-property-dan-method/*](https://www.duniailkom.com/tutorial-belajar-oop-php-pengertian-class-object-property-dan-method/)*)*

* + 1. ***Constructor***

***Constructor*** (bahasa indonesia: konstruktor) adalah method khusus yang akan dijalankan secara otomatis pada saat sebuah objek dibuat (instansiasi), yakni ketika perintah “**new**” dijalankan.

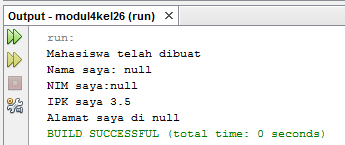
Constructor biasa digunakan untuk membuat proses awal dalam mempersiapkan objek, seperti memberi nilai awal kepada property, memanggil method internal dan beberapa proses lain yang digunakan untuk *‘*mempersiapkan*’* objek. Dalam PHP, constructordibuat menggunakan method : **\_\_construct().**

*(sumber : https://www.duniailkom.com/tutorial-belajar-oop-php-pengertian-constructor-dan-destructor/)*

* 1. Hasil Percobaan dan Analisis
     1. ***Class* pada Java**

|  |
| --- |
| package modul4kel26;  public class Modul4kel26 {  public static void main(String[] args) {  Student dkp = new Student();  System.out.println(dkp.intro());  System.out.println(dkp.tellGPA((float)3.5));  System.out.println(dkp.tellAddress());  }  } |

|  |
| --- |
| package modul4kel26;  public class Student {  public String name;  public String id;  private String address;  private float gpa;    public String intro()  {  return "Nama saya: "+name+ "\nNIM saya:" + id;  }    public String tellGPA(float gpa)  {  if (gpa >= 3) {  return "IPK saya "+ gpa;  }  else {  return "IPK saya tidak boleh di beritahu ke orang lain";  }  }    public String tellAddress()  {  return "Alamat saya di " + address;  }  } |



Gambar 5.1 class pada JAVA

Terdapat dua *class* yang digunakan yaitu *driver class* yang isinya fungsi *main* untuk membuat objek baru dari *class* Student()lalu akan memanggil *method* yang berada pada *class* Student().

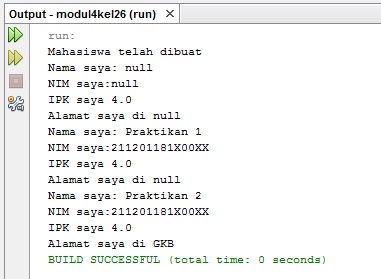
Pada kelas objek dengan nama Student() berisi pendeklarasian variabel yang diperlukan lalu membuat *method* dengan nama intro() dengan tipe *string.* *Method* ini akan mengembalikan nilai berupa "Nama saya: "+name+ "\nNIM saya:" + id *name* dan *id* adalah variabel global. Kemudian membuat *method* dengan nama tellGPA(float gpa) dengan tipe *string* dan parameter bertipe *float.* *Method* ini akan mengeksekusi perkondisian apabila benar maka akan mengembalikan nilai berupa "IPK saya "+ gpa *gpa* adalah variabel lokal dari parameter. Apabila kondisi salah maka akan mengembalikan nilai berupa "IPK saya tidak boleh di beritahu ke orang lain"

*Method* yang selanjutnya adalah tellAddress() yang akan mengembalikan nilai "Alamat saya di " + address *adress* adalah variabel global.

* + 1. ***Constructor* pada Java**

|  |
| --- |
| package modul4kel26;  public class Student {  public String name;  public String id;  private String address;  private float gpa;    public Student()  {  System.out.println("Mahasiswa telah dibuat");  }    public Student(String name, String id)  {  this.name= name;  this.id = id;  }    public Student(String name, String id, String address)  {  this.name = name;  this.id = id;  this.address = address;  }  } |

|  |
| --- |
| package modul4kel26;  public class Modul4kel26 {  public static void main(String[] args) {  Student one = new Student();  System.out.println(one.intro());  System.out.println(one.tellGPA(4));  System.out.println(one.tellAddress());    Student two = new Student("Praktikan 1", "211201181X00XX");  System.out.println(two.intro());  System.out.println(two.tellGPA(4));  System.out.println(two.tellAddress());    Student three = new Student("Praktikan 2", "211201181X00XX", "GKB");  System.out.println(three.intro());  System.out.println(three.tellGPA(4));  System.out.println(three.tellAddress());  }  } |

****

Gambar 5.2 constructor pada JAVA

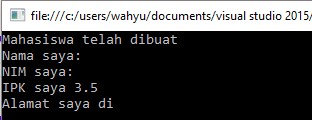
Pada kelas Student() ditambahkan *constructor* yang akan langsung dijalankan saat kelas diinisialisasi, yaitu akan menampilkan pesan "Mahasiswa telah dibuat". Lalu mendefiniskan *constructor* dengan parameter String name, String id yang akan memberikan nilai pada variabel global dengan argumen yang dimasukkan. Kemudian mendefinisakan *overloading constructor* yaitu konstruktor yang memiliki tambahan parameter berbeda, sehingga ketika ada argumen yang ditambahkan ke parameter String address maka konstruktor ini yang akan digunakan.

Fungsi *main* untuk membuat objek baru bernama *one* dari *class* Student()karena disini didefinisikan konstrukor maka akan langsung dijalankan ketika kelas dipanggil, lalu akan menampilkan *output* dengan memanggil *method* yang berada pada *class* Student(). Kemudian membuat objek lagi dengan nama *two* dengan argumen ("Praktikan 1", "211201181X00XX") yang akan mejalankan konstruktor dengan parameter selanjutnya akan menampilkan *output* dengan memanggil *method* yang berada pada *class* Student(), lalu membuat objek lagi dengan nama *three* dengan argument ("Praktikan 2", "211201181X00XX", "GKB") yang akan menjalakan *overloading constructor* karena terdapat argumen untuk parameter *address.* Selanjutnya akan menampilkan *output* dengan memanggil *method* yang berada pada *class* Student().

* + 1. ***Class* pada C#**

|  |
| --- |
| using System;  using System.Collections.Generic;  using System.Linq;  using System.Text;  using System.Threading.Tasks;  namespace modul4kel26  {  class Program  {  static void Main(string[] args)  {  Student dkp = new Student();  Console.WriteLine(dkp.intro());  Console.WriteLine(dkp.tellGPA((float)3.5));  Console.WriteLine(dkp.tellAddress());  Console.ReadKey();  }  }  } |

|  |
| --- |
| using System;  using System.Collections.Generic;  using System.Linq;  using System.Text;  using System.Threading.Tasks;  namespace modul4kel26  {  class Student  {  public String name;  public String id;  private String address;  private float gpa;  public String intro() //Method non parameter//  {  return "Nama saya: " + name + "\nNIM saya:" + id;  }  public String tellGPA(float gpa) //Method berparameter  {  if (gpa >= 3)  {  return "IPK saya " + gpa;  }  else  {  return "IPK saya tidak boleh di beritahu ke orang lain";  }  }  public String tellAddress()  { //Method non parameter  return "Alamat saya di " + address;  }  }  } |



Gambar 5.3 class pada C#

Hampir sama dengan java, terdapat dua *class* yang digunakan yaitu *class program* yang isinya fungsi *main* untuk membuat objek baru dari *class* Student()lalu akan memanggil *method* yang berada pada *class* Student() terdapat perintah Console.ReadKey() untuk mencegah program menutup saat dijalankan.

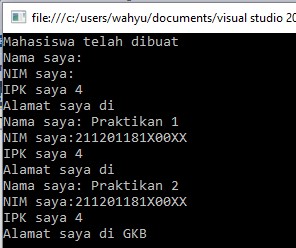
Pada kelas objek dengan nama Student() berisi pendeklarasian variabel yang diperlukan lalu membuat *method* dengan nama intro() dengan tipe *string.* *Method* ini akan mengembalikan nilai berupa "Nama saya: "+name+ "\nNIM saya:" + id *name* dan *id* adalah variabel global. Kemudian membuat *method* dengan nama tellGPA(float gpa) dengan tipe *string* dan parameter bertipe *float.* *Method* ini akan mengeksekusi perkondisian apabila benar maka akan mengembalikan nilai berupa "IPK saya "+ gpa *gpa* adalah variabel lokal dari parameter. Apabila kondisi salah maka akan mengembalikan nilai berupa "IPK saya tidak boleh di beritahu ke orang lain"

*Method* yang selanjutnya adalah tellAddress() yang akan mengembalikan nilai "Alamat saya di " + address *adress* adalah variabel global.

* + 1. ***Constructor* pada C#**

|  |
| --- |
| using System;  using System.Collections.Generic;  using System.Linq;  using System.Text;  using System.Threading.Tasks;  namespace modul4kel26  {  class Student  {  public String name;  public String id;  private String address;  private float gpa;  public Student() //Constructor default//  {  Console.WriteLine("Mahasiswa telah dibuat");  }  public Student(String name, String id) //Constructor dengan parameter//  {  this.name = name;  this.id = id;  }  public Student(String name, String id, String address) //Overloading Construtor//  {  this.name = name;  this.id = id;  this.address = address;  }  }  } |

|  |
| --- |
| using System;  using System.Collections.Generic;  using System.Linq;  using System.Text;  using System.Threading.Tasks;  namespace modul4kel26  {  class Program  {  static void Main(string[] args)  {  Student one = new Student();  Console.WriteLine(one.intro());  Console.WriteLine(one.tellGPA(4));  Console.WriteLine(one.tellAddress());  Student two = new Student("Praktikan 1", "211201181X00XX");  Console.WriteLine(two.intro());  Console.WriteLine(two.tellGPA(4));  Console.WriteLine(two.tellAddress());  Student three = new Student("Praktikan 2", "211201181X00XX", "GKB");  Console.WriteLine(three.intro());  Console.WriteLine(three.tellGPA(4));  Console.WriteLine(three.tellAddress());  Console.ReadLine();  }  }  } |



Gambar 5.4 constructor pada C#

Hampir sama seperi java, pada kelas Student() ditambahkan *default constructor* yang akan langsung dijalankan saat kelas diinisialisasi, yaitu akan menampilkan pesan "Mahasiswa telah dibuat". Lalu mendefiniskan *constructor* dengan parameter String name, String id yang akan memberikan nilai pada variabel global dengan argumen yang dimasukkan. Kemudian mendefinisakan *overloading constructor* yaitu konstruktor yang memiliki tambahan parameter berbeda, sehingga ketika ada argumen yang ditambahkan ke parameter String address maka konstruktor ini yang akan digunakan.

Fungsi *main* untuk membuat objek baru bernama *one* dari *class* Student()karena disini didefinisikan konstrukor maka akan langsung dijalankan ketika kelas dipanggil, lalu akan menampilkan *output* dengan memanggil *method* yang berada pada *class* Student(). Kemudian membuat objek lagi dengan nama *two* dengan argumen ("Praktikan 1", "211201181X00XX") yang akan mejalankan konstruktor dengan parameter selanjutnya akan menampilkan *output* dengan memanggil *method* yang berada pada *class* Student(), lalu membuat objek lagi dengan nama *three* dengan argument ("Praktikan 2", "211201181X00XX", "GKB") yang akan menjalakan *overloading constructor* karena terdapat argumen untuk parameter *address.* Selanjutnya akan menampilkan *output* dengan memanggil *method* yang berada pada *class* Student(). Terdapat perintah Console.ReadLine() untuk mencegah program menutup saat dijalankan.

* + 1. ***Class* pada PHP**
       1. **Percobaan 1**

<?php

class Produk {

public $jenis = "Televisi";

public $merek = "Samsung";

public $stok = 20;

public function pesanProduk(){

return "Produk dipesan...";

}

}

$televisi = new Produk();

echo $televisi->jenis;

echo "<br>";

echo $televisi->merek;

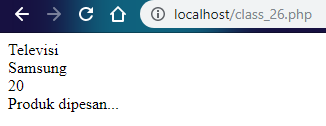
echo "<br>";

echo $televisi->stok;

echo "<br>";

echo $televisi->pesanProduk();

?>

****

Gambar 5.5 Percobaan 1 class pada PHP

Pertama terlebih dahulu membuat class dengan nama objek produk. Lalu mendeklarasikan tiap variabel yang menggunakan enkapsulasi *public* yang terdiri atas variabel jenis, merek, dan stok.

Setelah itu dilanjutkan engan membuat fungsi bernama pesanProduk yang tidak memiliki parameter dan menggunakan *return* yang berisi *output* Produk dipesan.

Dilanjutkan dengan mendeklarasikan *method* dengan mengganti nama objeknya menjadi variabel televisi nantinya akan menggantikan *class* produk tadi.

Dilanjutkan dengan menulis *output*nya untuk jenis, merek dan juga stok dengan menggunakan *method* yang nantinya akan menghasilkan output seperti pada gambar 5.5.

* + - 1. **Percobaan 2**

<?php

class Produk {

public $jenis;

public $merek;

public $stok;

public function pesanProduk(){

return "Produk telah dipesan...";

}

}

$produk01 = new Produk();

echo $produk01->jenis = "Televisi";

echo "<br>";

echo $produk01->merek = "Samsung";

echo "<br>";

echo $produk01->stok = 20;

echo "<br>";

echo $produk01->pesanProduk();

echo "<br>";echo "<br>";

$produk02 = new Produk();

echo $produk02->jenis = "Mesin cuci";

echo "<br>";

echo $produk02->merek = "LG";

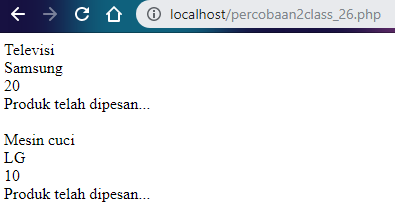
echo "<br>";

echo $produk02->stok = 10;

echo "<br>";

echo $produk02->pesanProduk();

?>



Gambar 5.6 Percobaan 2 class pada PHP

Pertama terlebih dahulu membuat *class* dengan nama objek produk. Lalu mendeklarasikan tiap variabel yang menggunakan enkapsulasi *public* yang terdiri atas variabel jenis, merek, dan stok.

Setelah itu dilanjutkan engan membuat fungsi bernama pesanProduk yang tidak memiliki parameter dan menggunakan *return* yang berisi output Produk dipesan.

Lalu dilanjutkan dengan membuat objek baru, dan pada percobaan ke2 ini menggunakan 2 objek baru yaitu pertama televisi dan juga yang kedua mesin cuci. Lalu memunculkan *output*nya yang nantinya akan tertampil seperti pada gambar 5.6.

* + - 1. **Percobaan 3**

<?php

class Produk {

public $jenis;

public $merek;

public $stok;

public function pesanProduk(){

return "tipe ". $this->jenis ." sudah dipesan";

//return $this->jenis." ".$this->merek." dipesan..."." Dan stok nya adalah" .$this->stok;

}

}

$produk01 = new Produk();

$produk01->jenis = "Televisi";

$produk01->merek = "Samsung";

$produk01->stok = 20;

$produk02 = new Produk();

$produk02->jenis = "Mesin cuci";

$produk02->merek = "LG";

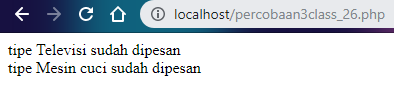
$produk02->stok = 10;

echo $produk01->pesanProduk();

echo "<br>";

echo $produk02->pesanProduk();

?>



Gambar 5.7 Percobaan 3 class pada PHP

Pertama terlebih dahulu membuat *class* dengan nama objek produk. Lalu mendeklarasikan tiap variabel yang menggunakan enkapsulasi *public* yang terdiri atas variabel jenis, merek, dan stok.

Setelah itu dilanjutkan dengan membuat fungsi bernama pesanProduk yang tidak memiliki parameter dan menggunakan *return* yang berisi *output* Produk dipesan.

* + 1. ***Consructor* pada php**

<?php

class Produk {

public $jenis;

public $merek;

public $stok;

public function \_\_construct($a, $b, $c){

$this->jenis = $a;

$this->merek = $b;

$this->stok = $c;

}

}

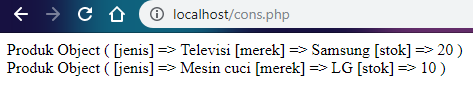
$produk01 = new Produk("Televisi","Samsung",20);

$produk02 = new Produk("Mesin cuci","LG", 10);

print\_r ($produk01);

echo "<br>";

print\_r ($produk02);



Gambar 5.8 Constructor pada PHP

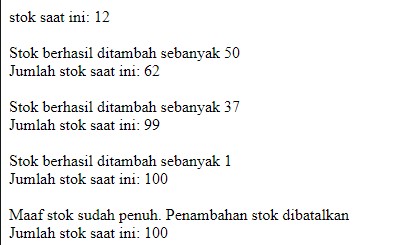
Pertama terlebih dahulu membuat *class* dengan nama produk. Lalu mendeklarasikan tiap variabel yang menggunakan enkapsulasi *public* yang terdiri atas variabel jenis, merek, dan stok.

Setelah itu dilanjutkan dengan membuat fungsi konstruktor dengan sintaks \_\_construct($a, $b, $c) yang memiliki parameter a untuk mengisi variabel jenis parameter b untuk mengisi variabel merek dan parameter c untuk mengisi variabel stok. Kemudian membuat objek baru dengan nama produk1 yang menginstanisasi dari *class* produk dengan argumen (“Televisi”, “Samsung”, “20”) dan membuat objek baru lagi dengan nama produk2 yang menginstanisasi dari *class* produk dengan argumen (“Mesin cuci”, “LG”, “10”)kemudian menampilkan informasi yang dapat dibaca dengan perintah print\_r () yang akan menampilkan *output* seperti pada gambar 5.8.

* 1. Tugas

|  |
| --- |
| <?php  include "tambah.php";  $produk1 = new Produk();  $produk1->stok = 12;  echo $produk1->cekStok();  echo "<br>";  $produk1->tambahStok(50);  $produk1->tambahStok(37);  $produk1->tambahStok(1);  $produk1->tambahStok(19);  ?> |

|  |
| --- |
| <?php  class Produk {  public $stok;  public function tambahStok($jumlah){  if($this->stok+$jumlah<=100){  $this->stok = $this->stok + $jumlah;    echo "Stok berhasil ditambah sebanyak ".$jumlah."<br>";  echo "Jumlah".$this->cekStok()."<br>";  return $this->stok;  } else {  echo "Maaf stok sudah penuh. Penambahan stok dibatalkan"."<br>";  echo "Jumlah".$this->cekStok()."<br>";  }    }  public function cekStok(){  return " stok saat ini: ".$this->stok. "<br>";  }  }  ?> |



Gambar 5.9 Tampilan output program

Terdapat dua *file* yang digunakan yaitu tugas.php yang berisi *main program* dan tambah.php yang berisi *method* yang dibutuhkan.

Dalam tugas.php dibutuhkan *file* tambah.php untuk dapat berjalan, karena itu digunakan sintaks include untuk menyisipkan *file* yang diperlukan. Lalu membuat objek baru bernama $produk1 = new Produk(), Produk() merupakan kelas pada *file* tambah.php. Kemudian variabel stok diberi nilai awal 12. Lalu ada pemanggilan *method* $produk1->cekStok() untuk menampilkan jumlah stok. Kemudian memanggil *method* $produk1->tambahStok() yang masing-masing memiliki argument berbeda sebagai parameter.

Lalu pada *file* tambah.php kita membuat kelas baru bernama Produk yang isinya terdapat deklarasi variabel stok dan dua *method,* yang pertama *method* tambahStok($jumlah) dengan parameter $jumlah sebagai variabel untuk mengambil nilai dari argumen yang dikirimkan dari *main program.*

Dalam *method* tersebut digunakan kondisi *if* untuk memeriksa apakah nilai varibel stok yang dijumlah dengan nilai dari argument bernilai kurang dari 100. Jika iya maka akan dieksekusi perintah untuk menambahkan stok dengan nilai yang ada pada variabel jumlah. Lalu menampilkan pesan yang isinya stok berhasil ditambah dan menampilkan nilai yang ditambahkan dan mengecek stok setelah ditambahkan.

Jika kondisi salah maka akan menampilkan pesan yang isinya maaf stok sudah penuh. Penambahan stok dibatalkan dan menampilkan jumlah stok yang ada. Lalu *method* yang kedua cekStok() digunakan untuk menampilkan nilai dari stok dengan me-*return* varibel global $this->stok.

* 1. Kesimpulan

1. Pemrograman berorientasi objek dapat membuat banyak objek dari kelas yang sama.
2. Terdapat dua istilah *class* yang digunakan yaitu *driver class* yang berisi *main program* dan *object class* yang akan dibuat objek*.*
3. *Default constructor* akan langsung dijalankan ketika sebuah *class* digunakan untuk membuat objek (instanisasi).
4. Objek yang dipanggil dengan mengisi argumen akan menjalankan *constructor* dengan parameter bila ada.
5. Overloading Constructor digunakan saat ada tambahan argumen yang sesuai dengan parameter pada *constructor* tersebut.
6. Pada PHP penulisan konstruktor menggunakan dua buah *underscore* sebelum sintaks *construct.*
7. Pada C# dan Java penulisan konstruktor sesuai dengan nama *class.*
8. Penggunaan sintaks *this* merujuk pada variable global dalam *class* itu sendiri.