**LAPORAN PRAKTIKUM**

**PEMROGRAMAN WEB**

****

**oleh:**

**FARKHAN 20081010060**

**PROGRAM STUDI INFORMATIKA**

**FAKULTAS ILMU KOMPUTER**

**UNIVERSITAS PEMBANGUNAN NASIONAL “VETERAN”**

**JAWA TIMUR**

**2022**

# JAVASCRIPT & JQUERY

## SOAL dan JAWABAN

1. Soal nomor 1
2. Implementasi pembuatan dan pengaksesan Class.
3. Implementasi pembuatan dan pengaksesan Objek.
4. Implementasi Enkapsulasi Objek (Public, Protected dan Private).
5. Implementasi Variabel $this dalam Pemrograman Objek.
6. Penggunaan Pseudo-Variable $this dalam Objek.
7. Cara Membuat Method dalam Pemrograman Objek.
8. Implementasi Constructor dan Destructor.
9. Implementasi Inheritance (Pewarisan).
10. Cara Mengakses Property dan Method Parent Class.
11. Implementasi Constructor dan Destructor padaa Parent Class.
12. Implementasi Static Property dan Static Method.
13. Implementasi Konstanta Class dalam Pemrograman Objek.
14. Implementasi Final Method dan Final Class.
15. Implementasi Abstract Class dan Abstract Method PHP.
16. Implementasi Object Interface.
17. Implementasi Polimorfisme dalam Pemrograman Objek.

# PENULISAN KODE

**KODE NOMOR 1**

|  |
| --- |
| class buku {  // isi dari class buku  }  $buku1 = new buku(); |

Pada potongan kode di atas, saya membuat class buku kemudian mengakses class tersebut untuk saya buat menjadi objek dengan nama $buku1.

|  |
| --- |
| $buku1 = new buku();  echo get\_class($buku1); |

Pada potongan kode di atas, saya membuat objek dari class buku dengan nama $buku1. Kemudian saya mengakses objek tersebut agar nama class dari objek tersebut dapat ditampilkan pada layar.

|  |
| --- |
| class buku {  public $judul;  protected $penulis;  private $penerbit;  } |

Pada potongan kode di atas merupakan contoh implementasi dari enkapsulasi objek (public, protected, private).

|  |
| --- |
| class buku {  var $judul;  var $penulis;  var $penerbit;  var $tahun\_terbit;  function \_\_construct($judul, $penulis, $penerbit, $tahun\_terbit) {  $this->judul = $judul;  $this->penulis = $penulis;  $this->penerbit = $penerbit;  $this->tahun\_terbit = $tahun\_terbit;  }  } |

Pada potongan kode di atas merupakan contoh implementasi variabel $this.

|  |
| --- |
| <?php  class buku {  public $judul;  protected $penulis;  private $penerbit;  var $tahun\_terbit;  function \_\_construct($judul, $penulis, $penerbit, $tahun\_terbit) {  $this->judul = $judul;  $this->penulis = $penulis;  $this->penerbit = $penerbit;  $this->tahun\_terbit = $tahun\_terbit;  }  function getLabel() {  return "$this->penulis, $this->penerbit";  }  function getInfoLengkap() {  $str = "{$this->judul} | {$this->getLabel()} ({$this->tahun\_terbit})";  return $str;  }  // contoh pseudo-variabel  function getInfo() {  $str = "{$this->judul} | {$this->getLabel()}";  return $str;  }  }  $buku1 = new buku("Belajar PHP", "Dicky Susanto", "Gramedia", "2015");  echo $buku1->getInfo();  ?> |

Pada potongan kode di atas merupakan contoh implementasi pseudo-variabel $this.

|  |
| --- |
| <?php  class buku {  var $judul;  var $penulis;  var $penerbit;  var $tahun\_terbit;  function \_\_construct($judul, $penulis, $penerbit, $tahun\_terbit) {  $this->judul = $judul;  $this->penulis = $penulis;  $this->penerbit = $penerbit;  $this->tahun\_terbit = $tahun\_terbit;  }  function getLabel() {  return "$this->penulis, $this->penerbit";  }  }  $buku1 = new buku("Belajar PHP", "Dicky Susanto", "Gramedia", "2015");  echo $buku1->getLabel();  ?> |

Pada potongan kode di atas merupakan contoh pembuatan method.

|  |
| --- |
| <?php  class buku {  var $judul;  var $penulis;  var $penerbit;  var $tahun\_terbit;  function \_\_construct($judul, $penulis, $penerbit, $tahun\_terbit) {  $this->judul = $judul;  $this->penulis = $penulis;  $this->penerbit = $penerbit;  $this->tahun\_terbit = $tahun\_terbit;  }  function getLabel() {  return "$this->penulis, $this->penerbit";  }  function \_\_destruct() {  echo "Destructor buku";  }  }  $buku1 = new buku("Belajar PHP", "Dicky Susanto", "Gramedia", "2015");  echo $buku1->getLabel();  echo "<br>";  ?> |

Pada potongan kode di atas merupakan contoh pembuatan constructor dan destructor.

|  |
| --- |
| <?php  class buku {  var $judul;  var $penulis;  var $penerbit;  var $tahun\_terbit;  function \_\_construct($judul, $penulis, $penerbit, $tahun\_terbit) {  $this->judul = $judul;  $this->penulis = $penulis;  $this->penerbit = $penerbit;  $this->tahun\_terbit = $tahun\_terbit;  }  function getLabel() {  return "$this->penulis, $this->penerbit";  }  }  class komik extends buku {  var $jml\_halaman;  function \_\_construct($judul, $penulis, $penerbit, $tahun\_terbit, $jml\_halaman) {  parent::\_\_construct($judul, $penulis, $penerbit, $tahun\_terbit);  $this->jml\_halaman = $jml\_halaman;  }  function getInfoLengkap() {  $str = "{$this->judul} | {$this->getLabel()} (Rp. {$this->harga}) | {$this->jml\_halaman} Halaman.";  return $str;  }  }  $komik\_baru = new komik("Naruto", "Masashi Kishimoto", "Shonen Jump", "1998", "100");  echo $komik\_baru->getInfoLengkap();  ?> |

Pada potongan kode di atas merupakan contoh penerapan pewarisan class.

|  |
| --- |
| class komik extends buku {  var $jml\_halaman;  function \_\_construct($judul, $penulis, $penerbit, $tahun\_terbit, $jml\_halaman) {  parent::\_\_construct($judul, $penulis, $penerbit, $tahun\_terbit);  $this->jml\_halaman = $jml\_halaman;  }  function getInfoLengkap() {  $str = "{$this->judul} | {$this->getLabel()} (Rp. {$this->harga}) | {$this->jml\_halaman} Halaman.";  return $str;  }  } |

Pada potongan kode di atas merupakan contoh pengaksesan properti dan method dari parent class yang menggunakan keyword “parent::”.

|  |
| --- |
| <?php  class buku {  var $judul;  var $penulis;  var $penerbit;  var $tahun\_terbit;  function \_\_construct($judul, $penulis, $penerbit, $tahun\_terbit) {  $this->judul = $judul;  $this->penulis = $penulis;  $this->penerbit = $penerbit;  $this->tahun\_terbit = $tahun\_terbit;  echo "constructor buku <br>";  }  function getLabel() {  return "$this->penulis, $this->penerbit";  }  function \_\_destruct() {  echo "Destructor buku";  }  }  class komik extends buku {  var $jml\_halaman;  function \_\_construct($judul, $penulis, $penerbit, $tahun\_terbit, $jml\_halaman) {  parent::\_\_construct($judul, $penulis, $penerbit, $tahun\_terbit);  echo "construct komik <br>";  $this->jml\_halaman = $jml\_halaman;  }  function getInfoLengkap() {  $str = "{$this->judul} | {$this->getLabel()} (Rp. {$this->harga}) | {$this->jml\_halaman} Halaman.";  return $str;  }  function \_\_destruct() {  echo "Destructor komik <br>";  parent::\_\_destruct();  }  }  $komik\_baru = new komik("Naruto", "Masashi Kishimoto", "Shonen Jump", "1998", "100");  echo $komik\_baru->getInfoLengkap();  echo "<br>";  ?> |

Pada potongan kode di atas merupakan contoh pengaksesan constructor dan destructor dari parent class.

|  |
| --- |
| <?php  class buku {  var $judul;  var $penulis;  var $penerbit;    // static property  public static $harga;  public static function getLabel() {  return "Buku";  }  }  buku::$harga = 5000;  echo "harga buku : " . buku::$harga;  echo "<br>";  echo buku::getLabel();  ?> |

Pada potongan kode di atas merupakan contoh penerapan static property dan static method.

|  |
| --- |
| <?php  class buku {  const HARGA = 10000;  }  echo "harga buku : " . buku::HARGA;  ?> |

Pada potongan kode di atas merupakan contoh penerapan konstanta class

|  |
| --- |
| <?php  class buku {  final public function get\_buku() {  return "Buku";  }  }  $buku = new buku();  echo $buku->get\_buku();  ?> |

Pada potongan kode di atas merupakan contoh penerapan final method. Final method tidak akan bisa dioverride pada anak class.

|  |
| --- |
| <?php  abstract class buku {  protected $judul;  protected $penulis;  protected $penerbit;  protected $harga;  public function \_\_construct($judul = "judul", $penulis = "penulis", $penerbit = "penerbit", $harga = 0) {  $this->judul = $judul;  $this->penulis = $penulis;  $this->penerbit = $penerbit;  $this->harga = $harga;  }  public function getLabel() {  return "$this->penulis, $this->penerbit";  }  abstract public function getInfo();  }  ?> |

Pada potongan kode di atas merupakan contoh pembuatan abstract class dan abstract method.

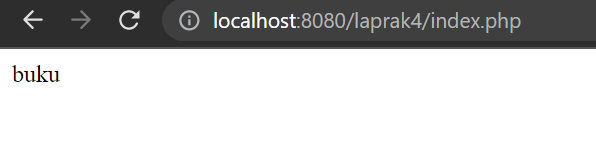
|  |
| --- |
| <?php  interface Buku {  public function getJudul();  public function getPengarang();  public function getPenerbit();  public function getTahunTerbit();  }  class BukuTulis implements Buku {  private $judul;  private $pengarang;  private $penerbit;  private $tahunTerbit;  public function \_\_construct($judul, $pengarang, $penerbit, $tahunTerbit) {  $this->judul = $judul;  $this->pengarang = $pengarang;  $this->penerbit = $penerbit;  $this->tahunTerbit = $tahunTerbit;  }  public function getJudul() {  return $this->judul;  }  public function getPengarang() {  return $this->pengarang;  }  public function getPenerbit() {  return $this->penerbit;  }  public function getTahunTerbit() {  return $this->tahunTerbit;  }  }  $buku = new BukuTulis('Cinta', 'Ridwan', 'Gramedia', '2015');  echo $buku->getJudul();  ?> |

Pada potongan kode di atas merupakan implementasi dari object interface.

|  |
| --- |
| <?php  abstract class buku {  abstract public function getInfo();  }  class novel extends buku {  public $judul, $penerbit, $pengarang;  public function \_\_construct($judul = "judul", $penerbit = "penerbit", $pengarang = "pengarang") {  $this->judul = $judul;  $this->penerbit = $penerbit;  $this->pengarang = $pengarang;  }  public function getInfo() {  return "{$this->judul} | {$this->penerbit}";  }  }  // buat objek dari class novel  $novel = new novel("Cinta", "Gramedia", "Dewi");  echo $novel->getInfo();  ?> |

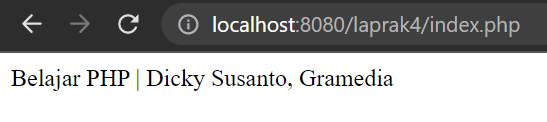
Pada potongan kode di atas merupakan contoh penerapan polymorphism dengan studi kasus buku.

# SCREENSHOT HASIL PERCOBAAN



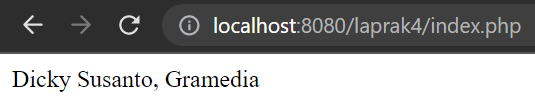
Gambar 3.1. Tampilan Nomor Dua

Pada gambar 3.1. menunjukkan hasil dari soal nomor dua



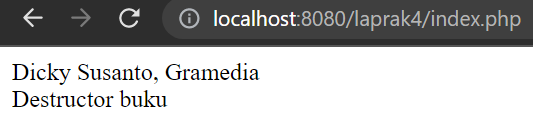
Gambar 3.2. Tampilan Nomor Lima

Pada gambar 3.2. menunjukkan hasil dari soal nomor lima



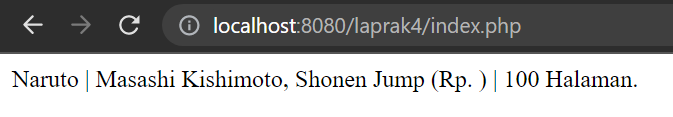
Gambar 3.3. Tampilan Nomor Enam

Pada gambar 3.3. menunjukkan hasil dari soal nomor enam



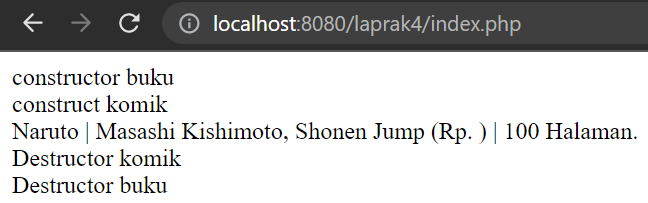
Gambar 3.4. Tampilan Nomor Tujuh

Pada gambar 3.4. menunjukkan hasil dari penerapan constructor dan destructor



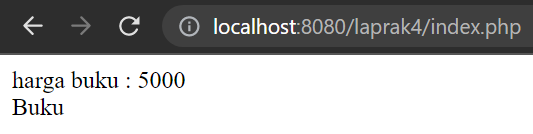
Gambar 3.5. Tampilan Nomor Delapan

Pada gambar 3.5. menunjukkan hasil dari penerapan pewarisan class komik yang mewarisi class buku.



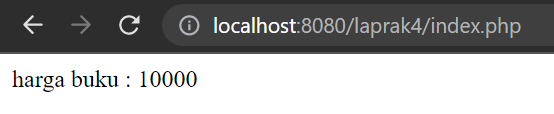
Gambar 3.6. Tampilan Nomor Sembilan

Pada gambar 3.6. menunjukkan hasil pengaksesan constructor dan destructor dari parent class



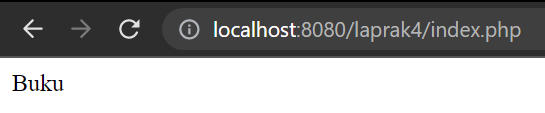
Gambar 3.7. Tampilan Nomor Sepuluh

Pada gambar 3.7. menunjukkan hasil dari penerapan static property dan static method



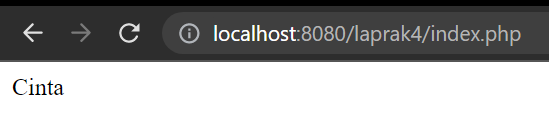
Gambar 3.8. Tampilan Nomor Duabelas

Pada gambar 3.8. menunjukkan hasil penerapan konstanta class



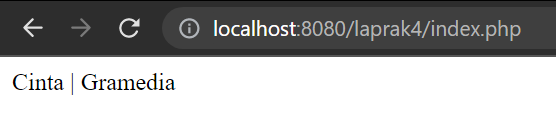
Gambar 3.9. Tampilan Nomor Tigabelas

Pada gambar 3.9. menunjukkan hasil dari penerapan final method.



Gambar 3.10. Tampilan Nomor Limabelas

Pada gambar 3.10. menunjukkan hasil dari penerapan object interface.



Gambar 3.11. Tampilan Nomor Enambelas

Pada gambar 3.11. menunjukkan hasil dari penerapan polymorphism.