LAPORAN PRAKTIKUM

PEMROGRAMAN WEB

oleh:

FARKHAN 20081010060

PROGRAM STUDI INFORMATIKA

FAKULTAS ILMU KOMPUTER

UNIVERSITAS PEMBANGUNAN NASIONAL "VETERAN"

JAWA TIMUR

2022

1. JAVASCRIPT & JQUERY

1.1. SOAL dan JAWABAN

1) Soal nomor 1
   a. Implementasi pembuatan dan pengaksesan Class.
   b. Implementasi pembuatan dan pengaksesan Objek.
   c. Implementasi Enkapsulasi Objek (Public, Protected dan Private).
   d. Implementasi Variabel $this dalam Pemrograman Objek.
   e. Penggunaan Pseudo-Variable $this dalam Objek.
   f. Cara Membuat Method dalam Pemrograman Objek.
   g. Implementasi Constructor dan Destructor.
   h. Implementasi Inheritance (Pewarisan).
   i. Cara Mengakses Property dan Method Parent Class.
   j. Implementasi Constructor dan Destructor padaa Parent Class.
   k. Implementasi Static Property dan Static Method.
   l. Implementasi Konstanta Class dalam Pemrograman Objek.
   m. Implementasi Final Method dan Final Class.
   n. Implementasi Abstract Class dan Abstract Method PHP.
   o. Implementasi Object Interface.
   p. Implementasi Polimorfisme dalam Pemrograman Objek.

2. 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
<?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
<?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;
    }
}
```

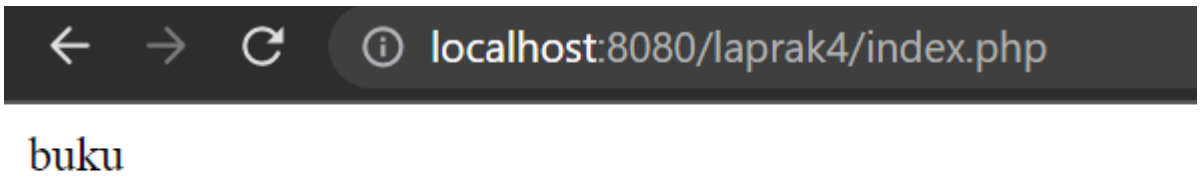
```
$buku = new BukuTulis('Cinta', 'Ridwan', 'Gramedia', '2015');
echo $buku->getJudul();

?>
```

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.

3. 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

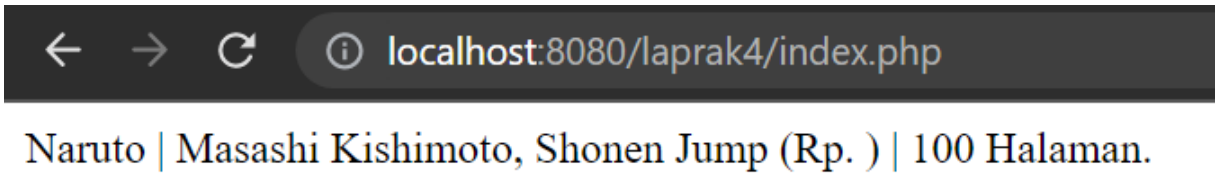
Pada gambar 3.2. menunjukkan hasil dari soal nomor lima

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

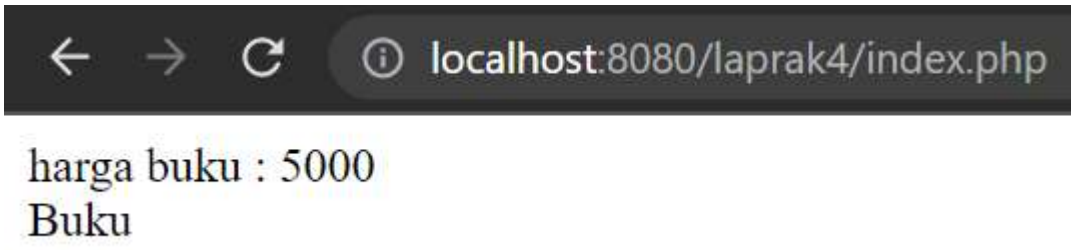


Gambar 3.5. Tampilan Nomor Delapan

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

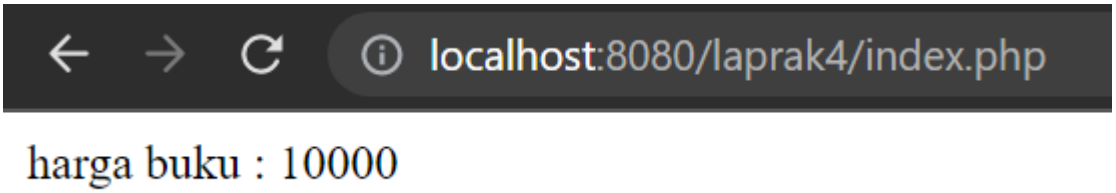
Gambar 3.6. Tampilan Nomor Sembilan

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



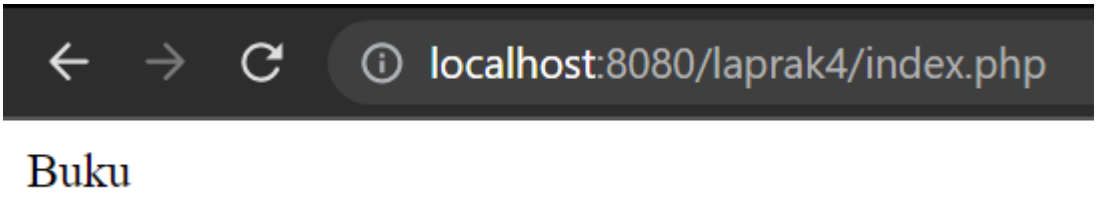
Gambar 3.7. Tampilan Nomor Sepuluh

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



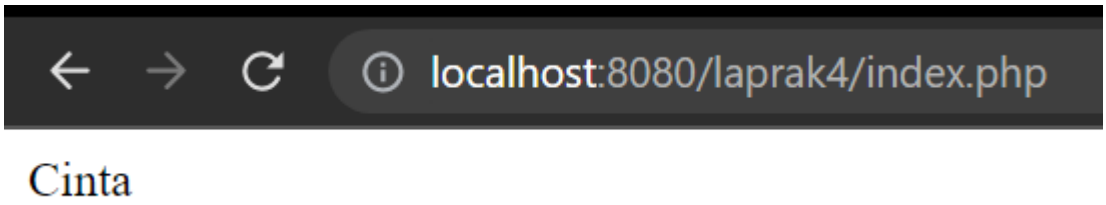
Gambar 3.8. Tampilan Nomor Duabelas

Pada gambar 3.8. menunjukkan hasil penerapan konstanta class



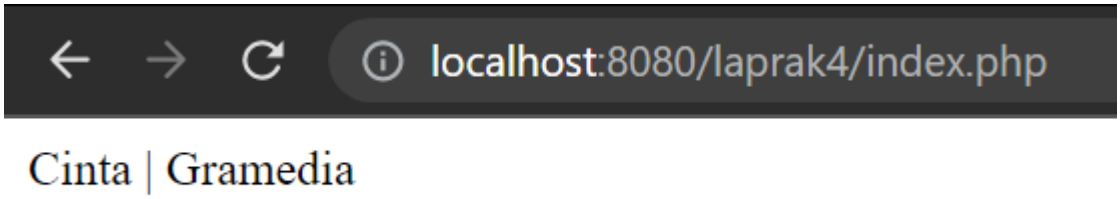
Gambar 3.9. Tampilan Nomor Tigabelas

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.