

**10111026 – KARISMA AINUN NISSA**

**SI 2B**

JOB SHEET   
PEMROGRAMAN BERORIENTASI OBJEK

PERTEMUAN 7

Dosen Pengampu

Sari Azhariyah, M.Pd.T.  
Usep Abdul Rosid, S.T., M.Kom

|  |  |
| --- | --- |
| POLITEKNIK NEGERI SUBANG |  |
| Jurusan : TIK | Waktu : 240 Menit |
| Program Studi : Sistem Informasi | Topik : Encapsulation dan Polymorphism |
| Mata Kuliah : PBO | Judul : Encapsulation dan Polymorphism |
| Tanggal : 1 Oktober 2025 | Pertemuan ke : 7 |

# Tujuan : Mahasiswa mampu memahami dan menjelaskan konsep dasar dan mengimplementasi serta berlatih penggunaan elemen pada HTML.

# Alat dan Bahan : XAMPP, Web Browser, Text Editor

# Link Github : https://github.com/karisss26/karis\_2B\_PBO.git

**Langkah Kerja :**

1. **Siapkan dan jalankan xampp, web browser, dan text editor**
2. **Buat folder praktikum pemrograman web pada folder htdocs**
3. **Lakukan penulisan sintak program pada text editor**
4. **Simpan file hasil penulisan sintak tersebut kedalam folder praktikum**
5. **Jalankan web browser**
6. **Jalankan web pada mode local dengan mengetikan localhost/folder praktikum/file sintak pada web browser**
7. **Output program akan tampil pada web browser**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **No** | **Praktik** | **Output Program** | **Sintak Kode Program** | **Keterangan** |
| 1. | **Latihan Soal 7.1** |  | <?php  class **manusia**{  *//properties (encapsulation)*      public $nama="Ardi";      var $kelas = "SI 2";  *//method manusia*      function **tampilkan\_nama**(){          return $this->nama;      }      public function **tampilkan\_kelas**(){          return $this->kelas;      }  }  *//instansiasi class manusia*  $manusia = new **manusia**();  *//memanggil method tampilkan\_nama dari class manusia*  **echo** "Nama : ".$manusia->**tampilkan\_nama**()."<br />";  **echo** "Kelas : ".$manusia->**tampilkan\_kelas**();  ?> | Latihan hak akses public |
| 2. | **Latihan Soal 7.2** |  | <?php  class **manusia**{  *//prop*      private $nama="Ardi";      private $kelas = "SI 2";  *//method manusia*      private function **m\_nama**(){          return $this->nama;      }      public function **tampilkan\_nama**(){          return $this->**m\_nama**();      }      function **tampilkan\_kelas**(){          return $this->kelas;      }  }  *//instansiasi class manusia*  $manusia = new **manusia**();  *//memanggil method tampilkan\_nama dari class manusia*  **echo** "Nama : ".$manusia->**tampilkan\_nama**()."<br />";  **echo** "Kelas : ".$manusia->**tampilkan\_kelas**();  ?> | Latihan hak akses private |
| 3. | **Latihan Soal 7.3** |  | <?php  class **manusia**{  *// property dengan protected*      protected $nama = "Ardi";      var $kelas = "SI 2";  *// method protected*      protected function **nama**(){          return "Nama : " . $this->nama;      }      public function **tampilkan\_nama**(){          return $this->**nama**();      }  *// ubah jadi public biar bisa dipanggil dari luar*      public function **tampilkan\_kelas**(){          return "Kelas : " . $this->kelas;      }  }  *// instansiasi class manusia*  $manusia = new **manusia**();  *// memanggil method public*  **echo** $manusia->**tampilkan\_nama**()."<br />";  **echo** $manusia->**tampilkan\_kelas**();  ?> | Latihan hak akses protected (ada error, jadinya dijadiin public) |
| 4. | **Latihan Soal 7.4** |  | <?php  *// buat class komputer*  class **komputer** {  *// property diubah ke protected supaya bisa diakses oleh subclass*      protected $jenis\_processor = "Intel Core i7-4790 3.6Ghz";      protected $jenis\_RAM = "DDR 4";      public $jenis\_VGA = "PCI Express";      public function **tampilkan\_processor**() {          return $this->jenis\_processor;      }      public function **tampilkan\_jenisprocessor**() {          return $this->jenis\_processor;      }  *// diubah ke protected supaya subclass bisa memanggilnya*      protected function **tampilkan\_ram**() {          return $this->jenis\_RAM;      }      protected function **tampilkan\_vga**() {          return $this->jenis\_VGA;      }      public function **tampilkan\_vga2**() {          return $this->jenis\_VGA;      }  }  *// buat class laptop*  class **laptop** extends **komputer**{      public function **display\_processor**() {  *// sekarang bisa akses karena $jenis\_processor protected*          return $this->jenis\_processor;      }      public function **display\_processor2**() {          return $this->**tampilkan\_processor**();      }      public function **display\_ram**() {          return $this->jenis\_RAM;      }      public function **display\_ram2**() {  *// tampilkan\_ram sekarang protected, jadi bisa dipanggil*          return $this->**tampilkan\_ram**();      }      public function **display\_vga**() {          return $this->**tampilkan\_vga2**();      }  *// ubah jadi public supaya bisa dipanggil dari luar*      public function **display\_processorkomputer**() {  *// lebih aman panggil method parent yg public*          return $this->**tampilkan\_processor**();      }  }  *// buat objek dari class komputer (instansiasi)*  $komputer = new **komputer**();  $laptop = new **laptop**();  *// jalankan method dari class komputer*  **echo** "Line 61 = ".$komputer->**tampilkan\_processor**()."<br />";  **echo** "Line 62 = ".$laptop->**display\_processor**()."<br />";  **echo** "Line 63 = ".$laptop->**display\_processor2**()."<br />";  **echo** "Line 64 = ".$laptop->**tampilkan\_jenisprocessor**()."<br />";  **echo** "Line 65 = ".$laptop->**display\_ram**()."<br />";  **echo** "Line 66 = ".$laptop->**display\_vga**()."<br />";  **echo** "Line 67 = ".$laptop->**display\_processorkomputer**()."<br />";  ?> | Latihan analisis, error karena tipenya private harusnya public biar bisa diakses, jadinya diubah ke public |
| 5. | **Latihan Soal 7.5** |  | <?php  *// Class abstrak (abstract class) didefinisikan sebagai class yang tidak bisa diinstansiasi menjadi object.*  *// Sebuah class abstrak setidaknya memiliki satu method abstrak (abstract method).*  *// Untuk menggunakan sebuah class abstrak, kita harus membuat class turunan dari class abstrak tersebut.*  *// Tujuan dari dibuat abstrak class adalah agar setiap class turunan menggunakan method abstrak pada class induk secara fleksibel.*  *// Jadi pada method abstrak di class induk hanya terdapat signature saja tanpa isi yang konkrit.*  abstract class **Hewan** {      abstract public function **Makan**();      abstract public function **Bergerak**();      abstract public function **Beranak**();  }  class **Burung** extends **Hewan** {      public function **Makan**() {          return "Burung makan biji-bijian<br>";      }      public function **Bergerak**() {          return "Burung bergerak dengan berjalan, terbang dan melompat<br>";      }      public function **Beranak**() {          return "Burung beranak dengan bertelur<br>";      }  }  class **Kambing** extends **Hewan** {      public function **Makan**() {          return "Kambing makan rumput<br>";      }      public function **Bergerak**() {          return "Kambing bergerak dengan berjalan dan berlari<br>";      }      public function **Beranak**() {          return "Kambing beranak dengan melahirkan<br>";      }  }  $burung = new **Burung**;  $kambing = new **Kambing**;  **echo** "<b>Perilaku Burung : </b><br>";  **echo** $burung->**Makan**();  **echo** $burung->**Bergerak**();  **echo** $burung->**Beranak**();  **echo** "<br><br>";  **echo** "<b>Perilaku Kambing : </b><br>";  **echo** $kambing->**Makan**();  **echo** $kambing->**Bergerak**();  **echo** $kambing->**Beranak**();  ?> | Latihan polymorphism abstract |
| 6. | **Latihan Soal 7.6** |  | <?php  interface **Hewan** {  *// membuat interface : hanya definisi fungsi tanpa ada implementasi*      public function **Makan**();      public function **Bergerak**();      public function **Beranak**();  }  class **Burung** implements **Hewan** {      public function **Makan**() {          return "Burung makan biji-bijian<br>";      }      public function **Bergerak**() {          return "Burung bergerak dengan berjalan, terbang dan melompat<br>";      }      public function **Beranak**() {          return "Burung beranak dengan bertelur<br>";      }  }  class **Kambing** implements **Hewan** {      public function **Makan**() {          return "Kambing makan rumput<br>";      }      public function **Bergerak**() {          return "Kambing bergerak dengan berjalan dan berlari<br>";      }      public function **Beranak**() {          return "Kambing beranak dengan melahirkan<br>";      }  }  $burung = new **Burung**;  $kambing = new **Kambing**;  **echo** "<b>Perilaku Burung : </b><br>";  **echo** $burung->**Makan**();  **echo** $burung->**Bergerak**();  **echo** $burung->**Beranak**();  **echo** "<br/>";  **echo** "<b>Perilaku Kambing : </b><br>";  **echo** $kambing->**Makan**();  **echo** $kambing->**Bergerak**();  **echo** $kambing->**Beranak**();  ?> | Latihan polymorphism interface |
| 7. | **Latihan Soal 7.7** |  | <?php  class **Employee**  {      private $first\_name;      private $last\_name;      private $age;      public function **\_\_construct**($first\_name, $last\_name, $age)      {          $this->first\_name = $first\_name;          $this->last\_name = $last\_name;          $this->age = $age;      }      public function **getFirstName**()      {          return $this->first\_name;      }      public function **getLastName**()      {          return $this->last\_name;      }      public function **getAge**()      {          return $this->age;      }  }  ?>  <?php  $objEmployeeOne = new **Employee**('Bob', 'Smith', 30);  **echo** $objEmployeeOne->**getFirstName**(); *// prints 'Bob'*  **echo** "<br>";  **echo** $objEmployeeOne->**getLastName**();  *// prints 'Smith'*  **echo** "<br>";  **echo** $objEmployeeOne->**getAge**();       *// prints '30'*  **echo** "<br><br>";  $objEmployeeTwo = new **Employee**('John', 'Smith', 34);  **echo** $objEmployeeTwo->**getFirstName**(); *// prints 'John'*  **echo** "<br>";  **echo** $objEmployeeTwo->**getLastName**();  *// prints 'Smith'*  **echo** "<br>";  **echo** $objEmployeeTwo->**getAge**();       *// prints '34'*  ?> | Latihan polymorphism |
|  | **Soal Tes Praktikum** |  | <?php  *// Parent class Tabungan (Savings)*  class **Tabungan** {      protected $saldo;      public function **\_\_construct**($saldo\_awal) {          $this->saldo = $saldo\_awal;      }      public function **getSaldo**() {          return $this->saldo;      }      public function **setorTunai**($jumlah) {          if ($jumlah > 0) {              $this->saldo += $jumlah;          }      }      public function **tarikTunai**($jumlah) {          if ($jumlah > 0 && $jumlah <= $this->saldo) {              $this->saldo -= $jumlah;          }      }  }  *// Child classes for each student*  class **Siswa1** extends **Tabungan** {      private $nama = "Siswa 1";      public function **getNama**() {          return $this->nama;      }  }  class **Siswa2** extends **Tabungan** {      private $nama = "Siswa 2";      public function **getNama**() {          return $this->nama;      }  }  class **Siswa3** extends **Tabungan** {      private $nama = "Siswa 3";      public function **getNama**() {          return $this->nama;      }  }  *// Create student objects with initial saldo*  $siswa1 = new **Siswa1**(100000);  $siswa2 = new **Siswa2**(150000);  $siswa3 = new **Siswa3**(200000);  *// Store students in an array*  $siswaArray = [$siswa1, $siswa2, $siswa3];  *// Display initial saldo*  **echo** "\n==========Saldo Awal Masing-masing Siswa:==========\n";  foreach ($siswaArray as $siswa) {  **echo** $siswa->**getNama**() . ": Rp " . $siswa->**getSaldo**() . "\n";  }  *// Open input stream for command prompt interaction*  $input = **fopen**("php://stdin", "r");  do {  **echo** "\nPilih siswa untuk transaksi:\n";      foreach ($siswaArray as $index => $siswa) {  **echo** ($index + 1) . ". " . $siswa->**getNama**() . "\n";      }  **echo** "0. Keluar\n";  **echo** "Masukkan pilihan: ";      $pilihan = **trim**(**fgets**($input));      if ($pilihan == "0") {          break;      }      if (!**is\_numeric**($pilihan) || $pilihan < 1 || $pilihan > **count**($siswaArray)) {  **echo** "Pilihan tidak valid. Silakan coba lagi.\n";          continue;      }      $siswaDipilih = $siswaArray[$pilihan - 1];      do {  **echo** "\nTransaksi untuk " . $siswaDipilih->**getNama**() . ":\n";  **echo** "1. Setor Tunai\n";  **echo** "2. Tarik Tunai\n";  **echo** "Masukkan pilihan transaksi: ";          $transaksi = **trim**(**fgets**($input));          switch ($transaksi) {              case "1":  **echo** "Masukkan jumlah setor tunai: ";                  $jumlahSetor = **trim**(**fgets**($input));                  if (**is\_numeric**($jumlahSetor) && $jumlahSetor > 0) {                      $siswaDipilih->**setorTunai**((int)$jumlahSetor);  **echo** "Setor tunai berhasil.\n";                  } else {  **echo** "Jumlah tidak valid.\n";                  }                  break;              case "2":  **echo** "Masukkan jumlah tarik tunai: ";                  $jumlahTarik = **trim**(**fgets**($input));                  if (**is\_numeric**($jumlahTarik) && $jumlahTarik > 0) {                      if ($jumlahTarik <= $siswaDipilih->**getSaldo**()) {                          $siswaDipilih->**tarikTunai**((int)$jumlahTarik);  **echo** "Tarik tunai berhasil.\n";                      } else {  **echo** "Saldo tidak cukup.\n";                      }                  } else {  **echo** "Jumlah tidak valid.\n";                  }                  break;              case "3":  *// Ganti siswa, keluar dari transaksi loop*                  break 2;              case "0":  *// Keluar program*  **fclose**($input);  **echo** "\n==========Saldo Akhir Masing-masing Siswa:==========\n";                  foreach ($siswaArray as $siswa) {  **echo** $siswa->**getNama**() . ": Rp " . $siswa->**getSaldo**() . "\n";                  }                  exit;              default:  **echo** "Pilihan transaksi tidak valid.\n";          }  **echo** "Saldo saat ini: Rp " . $siswaDipilih->**getSaldo**() . "\n";  **echo** "Apakah ingin melanjutkan transaksi untuk siswa ini? (y/n): ";          $lanjut = **trim**(**fgets**($input));      } while (**strtolower**($lanjut) == 'y');  } while (true);  **fclose**($input);  **echo** "\n==========Saldo Akhir Masing-masing Siswa:==========\n";  foreach ($siswaArray as $siswa) {  **echo** $siswa->**getNama**() . ": Rp " . $siswa->**getSaldo**() . "\n";  }  ?> | Praktikum membuat tabungan siswa 1, 2 dan 3 dengan ourput di terminal dan input di terminal |

**Analisis dan Penjelasan Kode Program dalam Soal Tes Praktikum :**

Kode PHP ini merupakan simulasi tabungan sederhana berbasis OOP yang dijalankan di CLI, dimana terdapat kelas induk Tabungan dengan properti protected $saldo untuk menyimpan jumlah saldo serta method getSaldo() untuk mengambil nilai saldo, setorTunai($jumlah) untuk menambah saldo jika jumlah yang dimasukkan lebih dari nol, dan tarikTunai($jumlah) untuk mengurangi saldo jika jumlah yang dimasukkan valid dan tidak melebihi saldo yang tersedia; kemudian ada tiga kelas turunan yaitu Siswa1, Siswa2, dan Siswa3 yang masing-masing mewarisi konstruktor dari Tabungan (karena tidak mendefinisikan konstruktor sendiri) sekaligus memiliki properti private $nama dengan nilai berbeda sesuai nama siswa dan method getNama() untuk mengembalikan namanya; setelah kelas didefinisikan, dibuatlah tiga objek siswa dengan saldo awal berbeda (100000, 150000, dan 200000) lalu disimpan dalam array $siswaArray, kemudian program mencetak saldo awal setiap siswa, membuka input stream php://stdin untuk membaca input user melalui terminal, meminta masukan jumlah setor untuk siswa yang dipilih dan menambahkan saldo jika input numeric dan lebih besar dari nol, dilanjutkan dengan meminta input jumlah tarik untuk siswa yang dipilih dan mengurangi saldo jika input valid serta tidak melebihi saldo yang ada, lalu memberikan opsi untuk lanjut transaksi atau ganti siswa, dan terakhir menampilkan saldo akhir siswa yang dipilih.