

**10111026 – KARISMA AINUN NISSA**

**SI 2B**

JOB SHEET   
PEMROGRAMAN BERORIENTASI OBJEK

PERTEMUAN 8

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 : Magic Methods dan Exception Handling |
| Mata Kuliah : PBO | Judul : Magic Methods dan Exception Handling |
| Tanggal : 8 Oktober 2025 | Pertemuan ke : 8 |

# 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 8.1** |  | <?php  class **HtmlElement**  {      private $attributes = [];      private $tag;      public function **\_\_construct**($tag)      {          $this->tag = $tag;      }      public function **\_\_set**($name, $value)      {          $this->attributes[$name] = $value;      }      public function **\_\_get**($name)      {          if (**array\_key\_exists**($name, $this->attributes)) {              return $this->attributes[$name];          }      }      public function **html**($innerHTML = '')      {          $html = "<{$this->tag}";          foreach ($this->attributes as $key => $value) {              $html .= ' ' . $key . '="' . $value . '"';          }          $html .= '>';          $html .= $innerHTML;          $html .= "</{$this->tag}>";          return $html;      }  }  $article = new **HtmlElement**('article');  $article->id = 'main';  $article->class = 'light';  *// show the attributes*  **echo** $article->class . "<br />"; *// light*  **echo** $article->id; *// main*  ?> | Setting Getting Method |
| 2. | **Latihan Soal 8.2** |  | <?php  class **BankAccount**  {      private $accountNumber;      private $balance;      public function **\_\_construct**($accountNumber, $balance)      {          $this->accountNumber = $accountNumber;          $this->balance = $balance;      }      public function **\_\_toString**()      {          return "Bank Account: {$this->accountNumber}. <br /> Balance: {$this->balance}";      }  }  $account = new **BankAccount**('123456789', 100);  **echo** $account;  ?> | To String Method |
| 3. | **Latihan Soal 8.3** |  | <?php  class **Str**  {      private $s = '';      private $functions = [          'length' => 'strlen',          'upper' => 'strtoupper',          'lower' => 'strtolower'  *// map more method to functions*      ];      public function **\_\_construct**(string $s)      {          $this->s = $s;      }      public function **\_\_call**($method, $args)      {          if (!**in\_array**($method, **array\_keys**($this->functions))) {              throw new **BadMethodCallException**();          }  **array\_unshift**($args, $this->s);          return **call\_user\_func\_array**($this->functions[$method], $args);      }  }  $s = new **Str**('Hello, World!');  **echo** $s->**upper**() . '<br>';  **echo** $s->**lower**() . '<br>';  **echo** $s->**length**() . '<br>';  ?> | Call Method |
| 4. | **Latihan Soal 8.4** |  | <?php  class **Str**  {      private static $methods = [          'upper' => 'strtoupper',          'lower' => 'strtolower',          'len' => 'strlen'      ];      public static function **\_\_callStatic**(string $method, array $parameters)      {          if (!**array\_key\_exists**($method, self::$methods)) {              throw new **Exception**('The ' . $method . ' is not supported.');          }          return **call\_user\_func\_array**(self::$methods[$method], $parameters);      }  }  **echo** **Str**::**lower**('Hello') . '<br>';  **echo** **Str**::**upper**('Hello') . '<br>';  **echo** **Str**::**len**('Hello') . '<br>';  ?> | Call Static Method |
| 5. | **Latihan Soal 8.5** |  | <?php  class **Comparator**  {      private $key;      public function **\_\_construct**(string $key)      {          $this->key = $key;      }      public function **\_\_invoke**($a, $b)      {          return $a[$this->key] <=> $b[$this->key];      }  }  $customers = [      ['id' => 1, 'name' => 'John', 'credit' => 20000],      ['id' => 3, 'name' => 'Alice', 'credit' => 10000],      ['id' => 2, 'name' => 'Bob', 'credit' => 15000]  ];  *// sort customers by names*  **usort**($customers, new **Comparator**('name'));  **print\_r**($customers);  *// sort customers by credit*  **usort**($customers, new **Comparator**('credit'));  **print\_r**($customers);  ?> | Invoke Method |
| 6. | **Latihan Soal 8.6** |  | <?php  class **Address**  {      public $street;      public $city;  }  class **Person**  {      public $name;      public $address;      public function **\_\_construct**($name)      {          $this->name = $name;          $this->address = new **Address**();      }      public function **\_\_clone**()      {          $this->address = clone $this->address;      }  }  $bob = new **Person**('Bob');  $bob->address->street = 'North 1st Street';  $bob->address->city = 'San Jose';  $alex = clone $bob;  $alex->name = 'Alex';  $alex->address->street = '1 Apple Park Way';  $alex->address->city = 'Cupertino';  **var\_dump**($bob);  **var\_dump**($alex);  ?> | Clone Method |
| 7. | **Latihan Soal 8.7** |  | <?php  *//create function with an exception*  function **checkNum**($number) {      if ($number > 1) {          throw new **Exception**("Value must be 1 or below");      }      return true;  }  *//trigger exception in a "try" block*  try {  **checkNum**(2);  }  *//catch exception*  catch(**Exception** $e) {  **echo** 'Error: ' . $e->**getMessage**();  }  ?> | Exception Handling. kode ini mengalami *error* karena tidak ada blok catch untuk menangani Exception yang dilempar oleh fungsi checkNum(). Untuk memperbaikinya, perlu mengemas panggilan fungsi checkNum(2) di dalam blok **try...catch** agar *exception* bisa ditangkap dan diatasi dengan baik, bukan membuat program berhenti. |
| 8. | **Latihan Soal 8.8** |  | <?php  *//create function with an exception*  function **checkNum**($number) {      if ($number > 1) {          throw new **Exception**("Value must be 1 or below");      }      return true;  }  *//trigger exception in a "try" block*  try {  **checkNum**(2);  *//If the exception is thrown, this text will not be shown*  **echo** 'If you see this, the number is 1 or below';  }  *//catch exception*  catch(**Exception** $e) {  **echo** 'Message: ' . $e->**getMessage**();  }  ?> | Menangani Exception Dengan Try-Catch |
| 9. | **Latihan Soal 8.9** |  | <?php  class **customException** extends **Exception** {      public function **errorMessage**() {  *//error message*          $errorMsg = 'Error on line ' . $this->**getLine**() . ' in ' . $this->**getFile**()          . ': <b>' . $this->**getMessage**() . '</b> is not a valid E-Mail address';          return $errorMsg;      }  }  $email = "someone@example...com";  try {  *//check if*      if (**filter\_var**($email, FILTER\_VALIDATE\_EMAIL) === FALSE) {  *//throw exception if email is not valid*          throw new **customException**($email);      }  }  catch (**customException** $e) {  *//display custom message*  **echo** $e->**errorMessage**();  }  ?> | Custom Exception Class |
| 10. | **Latihan Soal 8.10** |  | <?php  class **customException** extends **Exception** {      public function **errorMessage**() {  *//error message*          $errorMsg = 'Error on line ' . $this->**getLine**() . ' in ' . $this->**getFile**()          . ': <b>' . $this->**getMessage**() . '</b> is not a valid E-Mail address';          return $errorMsg;      }  }  $email = "someone@example.com";  try {  *//check if*      if (**filter\_var**($email, FILTER\_VALIDATE\_EMAIL) === FALSE) {  *//throw exception if email is not valid*          throw new **customException**($email);      }  *//check for "example" in mail address*      if (**strpos**($email, "example") !== FALSE) {          throw new **Exception**("$email is an example e-mail");      }  }  catch (**customException** $e) {  **echo** $e->**errorMessage**();  }  catch (**Exception** $e) {  **echo** $e->**getMessage**();  }  ?> | Multiple Exception Class |
|  | **Soal Tes Praktikum** |  | <?php  class **customException** extends **Exception** {      private $email;      public function **\_\_construct**($message, $email = '') {          parent::**\_\_construct**($message);          $this->email = $email;      }      public function **errorMessage**() {          return "Error: Email '" . $this->email . "' tidak mengandung kata 'lab4' dan E-mail tidak valid";      }  }  *// Data array email*  $emails = [      'lab4a@polsub.ac.id',      'lab4b@polsub.ac.id',      'lab4d@polsub.ac.id',      'lab4f@polsub.ac.id',      'lab5a@polsub.ac.id',      'lab5b@polsub.ac.id',      'lab5c@polsub.ac.id',      'someone@example...com', *// Contoh email tidak valid*      'lab4-contoh@polsub.ac.id' *// Contoh email valid*  ];  $validCount = 0;  $invalidCount = 0;  **echo** "<h3>Hasil Validasi Email</h3>";  foreach ($emails as $email) {      try {  *// Validasi format email*          if (**filter\_var**($email, FILTER\_VALIDATE\_EMAIL) === FALSE) {              throw new **Exception**("Email tidak valid");          }  *// Validasi kata kunci 'lab4'*          if (**strpos**($email, 'lab4') === FALSE) {              throw new **customException**("Email tidak mengandung 'lab4'", $email);          }  *// Jika lolos semua validasi*  **echo** "Email \*\*{$email}\*\* valid.<br>";          $validCount++;      } catch (**customException** $e) {  *// Tangani custom exception (email tidak mengandung 'lab4')*  **echo** $e->**errorMessage**() . "<br>";          $invalidCount++;      } catch (**Exception** $e) {  *// Tangani exception umum (format email tidak valid)*  **echo** "Error: Email '" . $email . "' tidak valid. " . $e->**getMessage**() . "<br>";          $invalidCount++;      }  }  **echo** "<br>";  **echo** "<h4>Rekapitulasi</h4>";  **echo** "Jumlah email valid: {$validCount}<br>";  **echo** "Jumlah email tidak valid: {$invalidCount}<br>";  ?> | Praktikum 8 dengan menggunakan customException, try and catch dan FILTER VALIDATE EMAIL kemudian masukan data email tersebut ke dalam array. Lakukan pengecekan dan validasi email menggunakan perulangan dalam array. Kemudian terakhir hitung counting data email yang valid dan tidak valid tersebut. |

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

Kode ini adalah program PHP untuk **validasi email** menggunakan **OOP** (Object-Oriented Programming) dan **exception handling**. Program ini membuat kelas customException yang khusus menangani kesalahan validasi, seperti email yang tidak mengandung kata "lab4", sementara Exception bawaan PHP digunakan untuk kesalahan format email. Dengan menggunakan perulangan foreach, kode akan memeriksa setiap email dalam sebuah array. Di dalam blok try, program mencoba memvalidasi format dan konten email. Jika salah satu validasi gagal, ia akan **melemparkan** sebuah *exception*, yang kemudian akan **ditangkap** oleh blok catch yang sesuai untuk menampilkan pesan *error* yang lebih informatif. Terakhir, program menghitung dan mencetak jumlah email yang berhasil divalidasi dan yang gagal, memberikan rekapitulasi dari seluruh proses.