**Practical-9**

**OOP with PHP - 3**

**Interface, Exception Handling and Practice of OOP fundamental**

1. **Create a class stack having a property called value(array), as well as having a push ,pop and display method. Display method displays the element of the array. [Note : for deleting elements replace the respective value with ‘0’]. Create an object of stack and perform push, pop, and display operation on the object.**

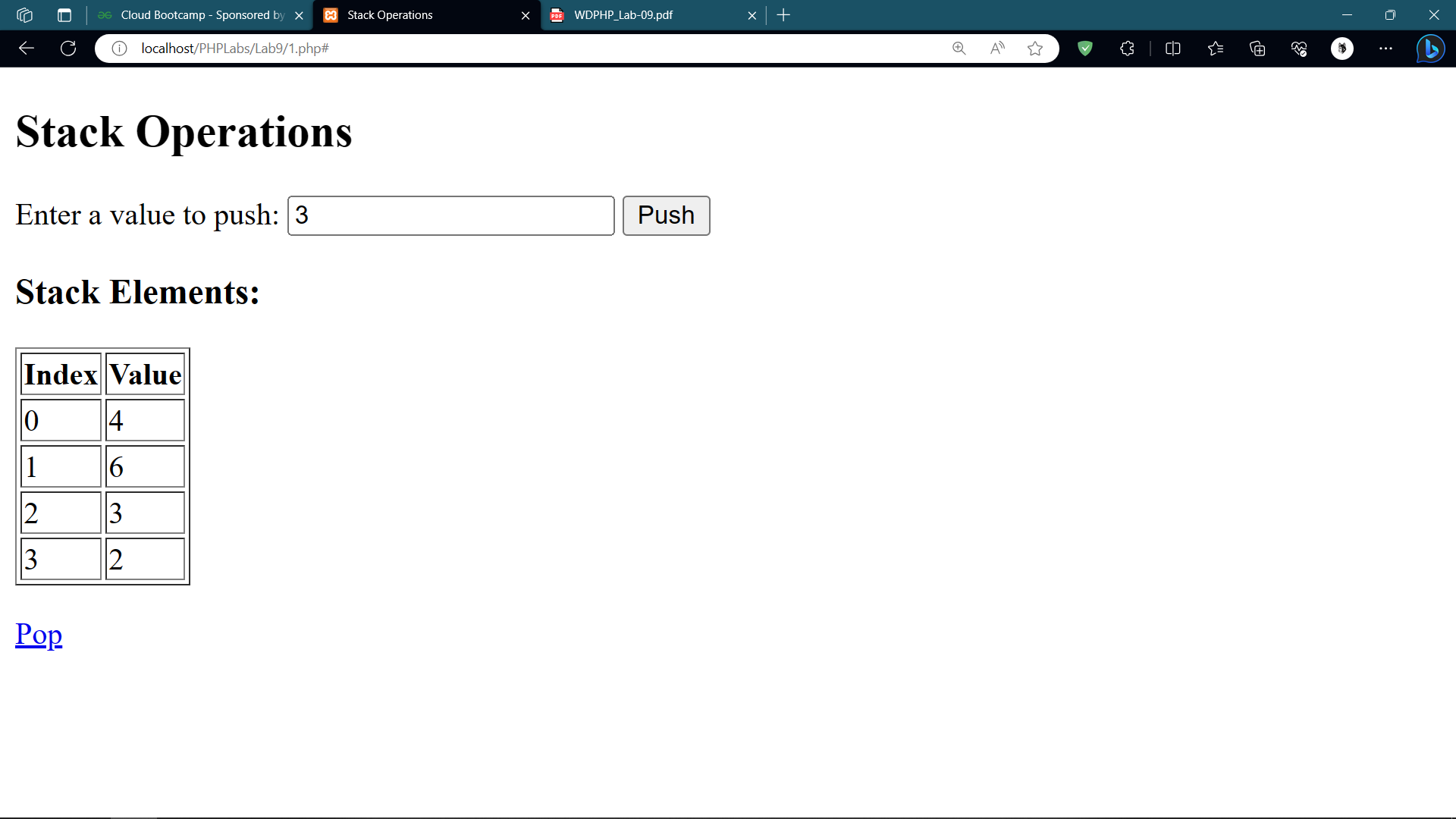
**Create proper HTML form which input the value to be pushed. Display stack element in table format. Take necessary variables required to perform operation. Throw an exception when trying to push the element from an empty array. Use Appropriate technique to show multiple operations via session or object serialization etc.**

**Code:**

**1.php**

|  |
| --- |
| <?php  class Stack  {  private $value = [];  private $top = -1;  public function push($item)  {  $this->value[++$this->top] = $item;  }  public function pop()  {  if ($this->isEmpty()) {  throw new Exception("Stack is empty. Cannot pop.");  }  unset($this->value[$this->top--]);  }  public function display()  {  return array\_values($this->value);  }  public function isEmpty()  {  return ($this->top == -1);  }  }  ?>  <!DOCTYPE html>  <html>  <head>  <title>Stack Operations</title>  </head>  <body>  <h2>Stack Operations</h2>  <form method="post" action="#">  <label for="value">Enter a value to push:</label>  <input type="text" id="value" name="value">  <input type="submit" value="Push">  </form>  <h3>Stack Elements:</h3>  <table border="1">  <tr>  <th>Index</th>  <th>Value</th>  </tr>  <?php  session\_start();  if (!isset($\_SESSION['stack'])) {  $\_SESSION['stack'] = new Stack();  }  if (isset($\_POST['value'])) {  $value = $\_POST['value'];  $\_SESSION['stack']->push($value);  }  if (isset($\_GET['pop'])) {  try {  $\_SESSION['stack']->pop();  } catch (Exception $e) {  echo '<p>' . $e->getMessage() . '</p>';  }  }  $stackElements = $\_SESSION['stack']->display();  foreach ($stackElements as $index => $element) {  echo "<tr>";  echo "<td>$index</td>";  echo "<td>$element</td>";  echo "</tr>";  }  ?>  </table>  <p><a href="?pop=true">Pop</a></p>  </body>  </html> |

**Output:**



**2. Create an interface called operation having a method plus(), minus(),**

**div(), mul(). Create a class Arithmetic having property x and y**

**Implement the method of interface to perform arithmetic operations**

**on two values and display answers. Create a class called String having**

**property str1 and str2. Implements the method of operation interface**

**Plus() : concat two strings and display the answer. Minus() : find out**

**the position of str2 in str1. Mul() : find out the number of occurrences**

**of str2 in str1. Div() : find out the last word form str1. Do appropriate**

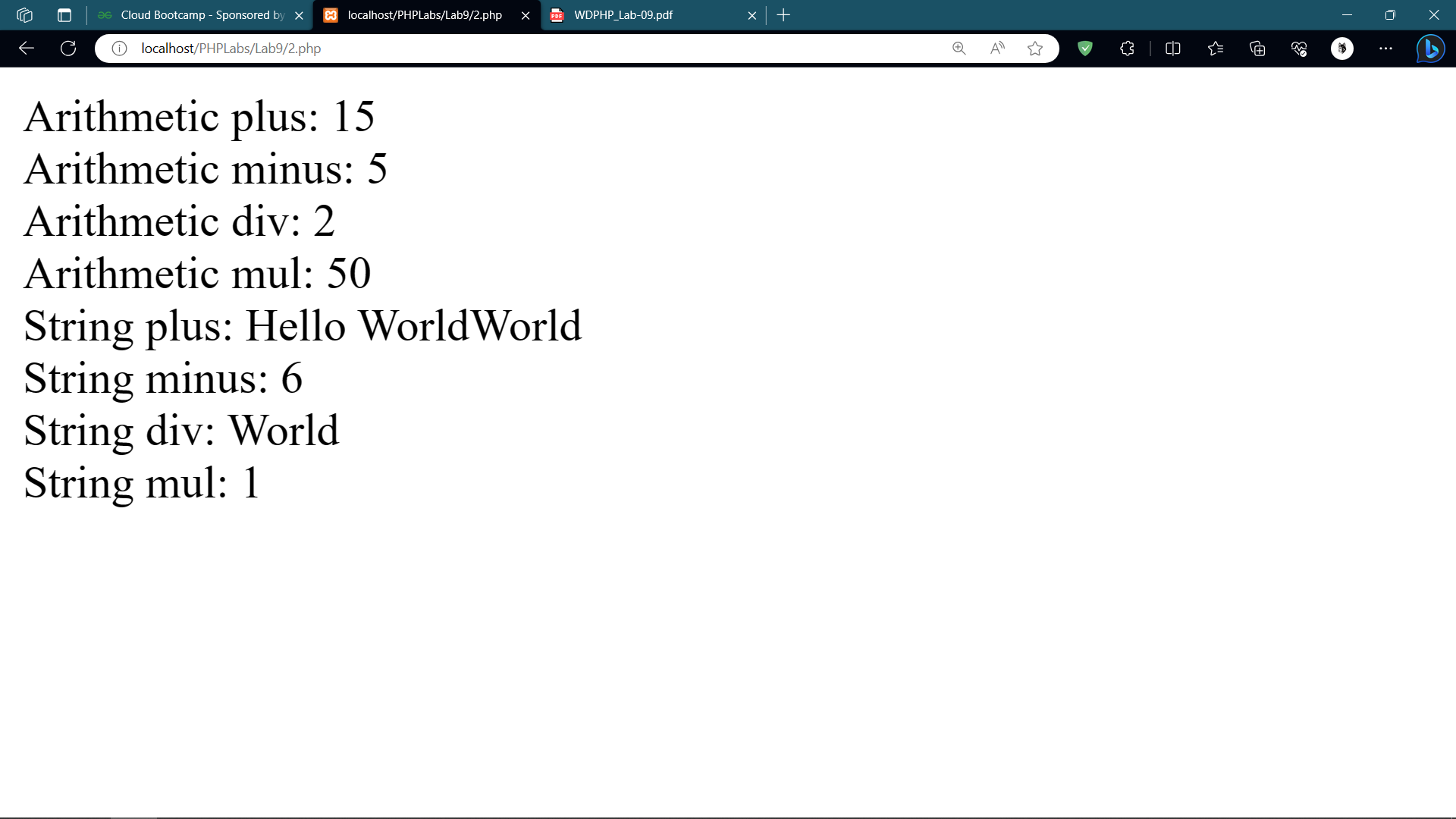
**handling of exceptions**

**Code:**

**2.php**

|  |
| --- |
| <?php  // Define the Operation interface  interface Operation {  public function plus();  public function minus();  public function div();  public function mul();  }  // Create the Arithmetic class  class Arithmetic implements Operation {  private $x;  private $y;  public function \_\_construct($x, $y) {  $this->x = $x;  $this->y = $y;  }  public function plus() {  return $this->x + $this->y;  }  public function minus() {  return $this->x - $this->y;  }  public function div() {  if ($this->y == 0) {  throw new Exception("Division by zero is not allowed.");  }  return $this->x / $this->y;  }  public function mul() {  return $this->x \* $this->y;  }  }  // Create the String class  class StringOperation implements Operation {  private $str1;  private $str2;  public function \_\_construct($str1, $str2) {  $this->str1 = $str1;  $this->str2 = $str2;  }  public function plus() {  return $this->str1 . $this->str2;  }  public function minus() {  $position = strpos($this->str1, $this->str2);  if ($position === false) {  return "Not found";  }  return $position;  }  public function div() {  $words = explode(" ", $this->str1);  return end($words);  }  public function mul() {  return substr\_count($this->str1, $this->str2);  }  }  // Example usage  try {  $arithmetic = new Arithmetic(10, 5);  echo "Arithmetic plus: " . $arithmetic->plus() . "<br>";  echo "Arithmetic minus: " . $arithmetic->minus() . "<br>";  echo "Arithmetic div: " . $arithmetic->div() . "<br>";  echo "Arithmetic mul: " . $arithmetic->mul() . "<br>";  $stringOperation = new StringOperation("Hello World", "World");  echo "String plus: " . $stringOperation->plus() . "<br>";  echo "String minus: " . $stringOperation->minus() . "<br>";  echo "String div: " . $stringOperation->div() . "<br>";  echo "String mul: " . $stringOperation->mul() . "<br>";  } catch (Exception $e) {  echo "Error: " . $e->getMessage() . "<br>";  }  ?> |

**Output:**



**3. Create an HTML form which takes the input of Employee name, age.**

**Form should have a drop down list to select the Employee type, as**

**Developer /worker. Create php script for following:**

**● Class Employee**

**● Property : name,age**

**● Method : constructor() , display()**

**● Class Developer inherits Employee**

**● Property: skill[] array, salary, degree**

**● Method: constructor, disp\_salary() , disp\_skill() .**

**● Class Worker inherits Employee**

**● Property : woring\_hr , per\_hr\_sprice**

**● Methods : constructor , calsalary() dispsalary()**

**At the end based on drop down value it create the object of**

**employee/manager / worker and display the detailCode:**

**3.php**

|  |
| --- |
| <?php  // Define the base Employee class  class Employee  {  protected $name;  protected $age;  public function \_\_construct($name, $age)  {  $this->name = $name;  $this->age = $age;  }  public function display()  {  echo "Name: {$this->name}<br>";  echo "Age: {$this->age}<br>";  }  }  // Define the Developer class, inheriting from Employee  class Developer extends Employee  {  private $skills = [];  private $salary;  private $degree;  public function \_\_construct($name, $age, $skills, $salary, $degree)  {  parent::\_\_construct($name, $age);  $this->skills = $skills;  $this->salary = $salary;  $this->degree = $degree;  }  public function disp\_salary()  {  echo "Salary: {$this->salary}<br>";  }  public function disp\_skill()  {  echo "Skills: " . implode(', ', $this->skills) . "<br>";  echo "Degree: {$this->degree}<br>";  }  }  // Define the Worker class, inheriting from Employee  class Worker extends Employee  {  private $working\_hr;  private $per\_hr\_price;  public function \_\_construct($name, $age, $working\_hr, $per\_hr\_price)  {  parent::\_\_construct($name, $age);  $this->working\_hr = $working\_hr;  $this->per\_hr\_price = $per\_hr\_price;  }  public function calsalary()  {  return $this->working\_hr \* $this->per\_hr\_price;  }  public function dispsalary()  {  echo "Salary: " . $this->calsalary() . "<br>";  }  }  // Get form input  if ($\_SERVER["REQUEST\_METHOD"] == "POST") {  $name = $\_POST["name"];  $age = $\_POST["age"];  $employee\_type = $\_POST["employee\_type"];  // Create objects based on employee type  if ($employee\_type == "Developer") {  $skills = ["Programming", "Problem Solving"];  $salary = 60000;  $degree = "Bachelor's Degree";  $employee = new Developer($name, $age, $skills, $salary, $degree);  } elseif ($employee\_type == "Worker") {  $working\_hr = 40; // Assuming 40 working hours  $per\_hr\_price = 10; // Assuming $10 per hour rate  $employee = new Worker($name, $age, $working\_hr, $per\_hr\_price);  }  // Display employee details  echo "<h2>Employee Details</h2>";  $employee->display();  if ($employee\_type == "Developer") {  $employee->disp\_skill();  $employee->disp\_salary();  } elseif ($employee\_type == "Worker") {  $employee->dispsalary();  }  }  ?>  <!DOCTYPE html>  <html>  <head>  <title>Employee Form</title>  </head>  <body>  <h2>Employee Details</h2>  <form action="#" method="post">  <label for="name">Name:</label>  <input type="text" id="name" name="name" required><br><br>  <label for="age">Age:</label>  <input type="text" id="age" name="age" required><br><br>  <label for="employee\_type">Employee Type:</label>  <select id="employee\_type" name="employee\_type" required>  <option value="Developer">Developer</option>  <option value="Worker">Worker</option>  </select><br><br>  <input type="submit" value="Submit">  </form>  </body>  </html> |

**Output:**

