EXPERIMENT NO. 8

Working with OOPs concepts in PHP

AIM: To develop understanding of implementing OOPs concepts in PHP.

Question (1). Write programs to implement OOPs concepts.

SOURCE CODE:

PHP

```
<?php
echo "<b>Classes and Objects</b><br>";
class User{
    public $firstname;
    public $lastname;

    public function hello($firstname, $lastname){
        echo "hello, ". $firstname. " ". $lastname. "<br>";
    }
}

$user1 = new User();
$user2 = new User();
$user1->hello("John", "Doe");
$user2->hello("Jane", "Doe");
echo "<hr>";
echo "<b>Using <i>\$this</i> Keyword</b><br>";
class User2{
    public $firstname;
    public $lastname;

    public function hello(){
        echo "hello, ". $this->firstname. " ". $this->lastname. "<br>";
    }
}

$user3 = new User2();
$user3->firstname="Jonnie";
$user3->lastname="Roe";
$user3->hello();
echo "<hr>";
echo "<b>Chaining Methods and Properties</b><br>";
class User3{
    public $firstName;

    public function hello()
    {
      echo "hello, " .  $this -> firstName;
      return $this;
    }
```

```
  public function register()
  {
    echo " >> registered";
    return $this;
  }

  public function mail()
  {
    echo " >> email sent";
  }
}

$user4 = new User3();
$user4 -> firstName = "Jane";
$user4 -> hello() -> register() -> mail();
echo "<hr>";
echo "<b>Public v/s Private</b><br>";
class User4{
  private $firstName;

  public function set($str)
  {
    $this -> firstName = $str;
  }

  public function get()
  {
    return $this -> firstName;
  }
}

$user5 = new User4();
$user5 -> set("Joe");
echo $user5 -> get();
echo "<hr>";
echo "<b>Magic Methods and Constants</b><br>";
class User5{
  private $firstName;
  private $lastName;

  public function __construct($firstName,$lastName)
  {
    $this -> firstName = $firstName;
    $this -> lastName = $lastName;
  }

  public function getName()
  {
      echo $this -> firstName . " " . $this -> lastName;
  }
}

$user6 = new User5("John", "Doe");
echo $user6->getName();
```

```
echo "<hr>";
echo "<b>Inheritance</b><br>";
class User6{
  protected $username;

  public function set($name)
  {
    $this -> username = $name;
  }
}

class Admin extends User6{
  public function expressYourRole()
  {
    return strtolower(__CLASS__);
  }

  public function sayHello()
  {
    return "Hello admin, " . $this -> username;
  }
}


$admin1 = new Admin();
$admin1 -> set("Balthazar");
echo $admin1 -> sayHello();
echo "<hr>";
echo "<b>Abstract Classes and Methods</b><br>";
abstract class User7{
  protected $username;

  public function set($name){
      return $this->username=$name;
  }

  public function get(){
      echo $this->username;
  }

  abstract function stateYourRole();
}

class Admin2 extends User7{
  public function stateYourRole(){
      echo "admin";
  }
}

class Viewer extends User7{
  public function stateYourRole(){
      echo "viewer";
  }
}
```

```
  $admin2 = new Admin2();
$admin2 -> set("Balthazar");
echo $admin2 -> stateYourRole();
echo "<hr>";
echo "<b>Interface</b><br>";
class User8{
    protected $username;

    public function set($name){
        return $this->username=$name;
    }

    public function get(){
        return $this->username;
    }
}
interface Author{
    public function setAuthorPrivileges($array);
    public function getAuthorPrivileges();
}
interface Editor{
    public function setEditorPrivileges($array);
    public function getEditorPrivileges();
}

class AuthorEditor extends User8 implements Author, Editor{
    private $authorPrivilegesArray = array();
    private $editorPrivilegesArray = array();

    public function setAuthorPrivileges($array){
        $this->authorPrivilegesArray = $array;
    }
    public function getAuthorPrivileges(){
        return $this->authorPrivilegesArray;
    }
    public function setEditorPrivileges($array){
        $this->editorPrivilegesArray = $array;
    }
    public function getEditorPrivileges(){
        return $this->editorPrivilegesArray;
    }
}
$user9 = new AuthorEditor();
$user9 -> set("Balthazar");
$user9 -> setAuthorPrivileges(array("write text","add punctuation"));
$user9 -> setEditorPrivileges(array("edit text","edit punctuation"));
$userName  = $user9 -> get();
$userPrivileges = array_merge($user9 -> getAuthorPrivileges(),
  $user9 -> getEditorPrivileges());

echo $userName . " has the following privileges: ";
echo implode(", ", $userPrivileges);
echo ".";
echo "<hr>";
echo "<b>Polymorphism</b><br>";
```

```
abstract class User9{
    protected $scores           = 0;
    protected $numberOfArticles = 0;

    public function setNumberOfArticles($int)
    {
      $numberOfArticles = (int)$int;
      $this -> numberOfArticles = $numberOfArticles;
    }

    public function getNumberOfArticles()
    {
      return $this -> numberOfArticles;
    }

    abstract public function calcScores();
  }

  class Author2 extends User9{
    public function calcScores()
    {
      return $this -> scores = $this -> numberOfArticles * 10 + 20;
    }
  }
  class Editor2 extends User9{
    public function calcScores()
    {
      return $this -> scores = $this -> numberOfArticles * 6 + 15;
    }
  }
  $author2 = new Author2();
$author2 -> setNumberOfArticles(8);
echo $author2 -> calcScores();
echo "<br>";
$author3 = new Editor2();
$author3 -> setNumberOfArticles(15);
echo $author3 -> calcScores();
echo "<hr>";
?>
```

OUTPUT SCREENSHOT:

Classes and Objects
hello, John Doe
hello, Jane Doe

Using *$this* Keyword
hello, Jonnie Roe

Chaining Methods and Properties
hello, Jane >> registered >> email sent

Public v/s Private
Joe

Magic Methods and Constants
John Doe

Inheritance
Hello admin, Balthazar

Abstract Classes and Methods
admin

Interface
Balthazar has the following privileges: write text, add punctuation, edit text, edit punctuation.

Polymorphism
100
105