Module2–AdvancedPHP Excercises

OOPs Concepts

1. Define Object-Oriented Programming(OOP) and its four main principles :Encapsulation, Inheritance ,Polymorphism, and Abstraction.

➢ Object-Oriented Programming is a programming paradigm based on the concept of **"objects"**, which can contain **data** (attributes) and **code** (methods). It helps structure software in a modular, reusable, and maintainable way by modeling real-world entities.

1.Encapsulation:

Definition: Bundling data and methods that operate on that data within a single unit (class), and restricting direct access to some components.

Purpose: Protects internal object state and promotes controlled interaction via public methods.

2.Inheritance:

Definition: Allows a class (child) to inherit properties and methods from another class (parent).

Purpose: Promotes code reuse and establishes a natural hierarchy.

3. Polymorphism:

Definition: The ability of different classes to be treated as instances of the same class through a common interface, often by overriding methods.

Purpose: Enables flexibility and dynamic behavior.

4. Abstraction:

Definition: Hiding complex implementation details and showing only the essential features of an object.

Purpose: Simplifies usage and focuses on what an object does rather than how it does it.

Class:

2. Explain the structure of a class in PHP, including properties and methods.

➢ A **class** in PHP is a blueprint for creating objects. It defines **properties** (variables) and **methods** (functions) that describe the behavior and state of the object.

➢ Making Group of data member(variable) and member function that called class.A Class is defined by using the class keyword, followed by the name of the class and a pair or curly braces({}).All its properties and method go inside the bracest.

➢ properties and methods:

1. **Class Declaration**
   - Starts with the class keyword followed by the class name.
   - Naming convention: PascalCase (e.g., UserProfile, OrderManager).
2. **Properties**
   - Variables that hold data related to the object.
   - Visibility modifiers:
     1. public: accessible from anywhere.
     2. private: accessible only within the class.
     3. protected: accessible within the class and its subclasses.
3. **Constructor**
   - Special method __construct() that runs when an object is created.
   - Used to initialize properties.
4. **Methods**
   - Functions that define the behavior of the object.
   - Can manipulate properties or perform actions.
5. **Accessing Properties and Methods**

- Use $this->propertyName or $this->methodName() inside the class.
- Use -> operator to access members from an object.

Object:

3. What is an object in OOP? Discuss how objects are instantiated from classes in PHP.
   - An object is a specific instance of a class it represents a real-world entity with state (data) and behavior (functions). While a class is just a blueprint, an object is the actual usable version of that blueprint.
     - Class = Blueprint (e.g., a template for a user)
     - Object = Real thing built from the blueprint (e.g., a specific user like "SVAYAM")

     Each object has: Properties (variables) that hold data, Methods (functions) that define behavior
   - Instantiating Objects from Classes in PHP, To create an object from a class, you use the new keyword followed
   - by the class name and parentheses.
     - Syntax

     $objectName = new ClassName();
     If the class has a constructor, you can pass arguments:
     $objectName = new ClassName($arg1, $arg2);

Extends:

4. Explain the concept of inheritance in OOP and how it is implemented in PHP.

- **Inheritance** is a mechanism that allows one class (called the **child** or **subclass**) to inherit properties and methods from another class (called the **parent** or **superclass**).
- **Types of inheritance** in Object-Oriented Programming:

  1. **Single Inheritance**
  2. **Multilevel Inheritance**
  3. **Hierarchical Inheritance**
  4. **Multiple Inheritance** (via Traits in PHP)

- Why Use Inheritance?
  - Promotes **code reuse**
  - Supports **hierarchical relationships**
  - Makes code **easier to maintain and extend**
- PHP uses the extends keyword to implement inheritance.

```
class ParentClass {
   // Properties and methods
}

class ChildClass extends ParentClass {
   // Inherits everything from ParentClass
   // Can add or override methods
}
```

Overloading :

5. scuss method overloading and how it is implemented in PHP.

- **Method overloading** means defining **multiple methods with the same name but different parameters** (number or type). It allows a class to perform different tasks based on how the method is called.
- PHP Does **Not** Support Traditional Method Overloading

- In PHP, you **cannot define multiple methods with the same name** in a class. If you try, the last one will override the previous ones.

- ```
  class Calculator {
      public function add($a, $b = 0) {
          return $a + $b;
      }
  }

  $calc = new Calculator();
  echo $calc->add(5);      // 5
  echo $calc->add(5, 10);  // 15
  ```

Abstraction Interface :

6. Explain the concept of abstraction and the use of interfaces in PHP.

- **Abstraction** is the process of hiding complex implementation details and exposing only the essential features of an object. It allows developers to focus on **what** an object does rather than **how** it does it.
- Purpose:
  - Simplifies code interaction
  - Promotes modular design
  - Enhances flexibility and maintainability
- How Abstraction is Implemented in PHP:

  1. **Abstract Classes**
  - Cannot be instantiated directly
  - Can contain both abstract methods (no body) and regular methods
  - Child classes must implement all abstract methods

  Example:
  ```
  abstract class Shape {
      abstract public function area();

      public function describe() {
          echo "This is a shape.";
      }
  }
  class Circle extends Shape {
      private $radius;

      public function __construct($radius) {
          $this->radius = $radius;
      }

      public function area() {
          return pi() * $this->radius * $this->radius;
      }
  }
  ```
  ```

Constructor :

7. What Is a constructor in PHP? Discuss its purpose and how it is used.

➢ A **constructor** is a special method in a class that **automatically runs when an object is created**. It's used to **initialize properties** or perform setup tasks.

- The method name is always __construct()
- It can accept parameters to set initial values

➢ Syntax : public function __construct(parameters) {
// Initialization code }

➢ Purpose of a Constructor:

1. **Initialize object properties**
2. **Set default values**
3. **Run setup logic when an object is created**
4. **Avoid repetitive code**

➢ Example: class User {

```
    public $name;
    public $email;

    public function __construct($name, $email) {
        $this->name = $name;
        $this->email = $email;
    }

    public function display() {
        echo "Name: $this->name, Email: $this->email";
    }
}

// Creating an object
$user1 = new User("SVAYAM", "svayam@example.com");
$user1->display();  // Output: Name: SVAYAM, Email: svayam@example.com
```

Destructor :

8. Explain the role of a destructor in PHP and when it is called.

➢ A **destructor** is a special method in a class that is **automatically called when an object is destroyed** or goes out of scope. It's used to **clean up resources**, such as closing files, releasing memory, or disconnecting from a database.

➢ Syntax: public function __destruct() {
// Cleanup code }

- The method name is always __destruct()
- It does **not** accept parameters
- Each class can have **only one destructor**

➢ Purpose of a Destructor

- Free up memory or resources
- Close open connections (e.g., database, file)
- Log or notify when an object is no longer needed
- Perform final actions before object deletion

Example: class FileHandler {

```
    private $file;
    public function __construct($filename) {
        $this->file = fopen($filename, "w");
        echo "File opened.\n";
    }
    public function write($text) {
        fwrite($this->file, $text);
    }

    public function __destruct() {
       fclose($this->file);
        echo "File closed.\n";
    }
}

// Create object
$handler = new FileHandler("example.txt");
$handler->write("Hello, SVAYAM!");

// Destructor is called automatically when script ends or object is unset
```

- When Is the Destructor Called?
- When the object is **explicitly destroyed** using unset()
- When the object **goes out of scope**
- When the **script ends.**

Magic Methods :

9. Define magic methods in PHP. Discuss commonly used magic methods like __get(), __set(), and __construct().

- **Magic methods** are special predefined methods in PHP that start with double underscores (__). They are automatically triggered by specific actions or behaviors in your code —like creating an object, accessing a property, or converting an object to a string.
- These methods help you customize how objects behave in special situations.

1. **__construct()**
   - **Purpose**: Automatically called when an object is created.
   - **Use**: Initialize properties or run setup logic.

```
class User {
    public $name;

    public function __construct($name) {
        $this->name = $name;
    }
}
$user = new User("SVAYAM"); // __construct is triggered
```

2. **__get($property)**
   - **Purpose**: Called when accessing a **non-accessible** or **undefined** property.
   - **Use**: Dynamically handle property access.

```
class Product {
    private $data = ["price" => 100];
```

```
    public function __get($name) {
        return $this->data[$name] ?? "Property not found";
    }
}
$p = new Product();
echo $p->price; // Triggers __get(), outputs 100
```

3. **__set($property, $value)**
   - **Purpose**: Called when assigning a value to a **non-accessible** or **undefined** property.
   - **Use**: Dynamically handle property assignment.

```
class Product {
    private $data = [];

    public function __set($name, $value) {
        $this->data[$name] = $value;
    }

    public function __get($name) {
        return $this->data[$name] ?? null;
    }
}

$p = new Product();
$p->price = 150; // Triggers __set()
echo $p->price;  // Triggers __get(), outputs 150
```

Scope Resolution:

10.Explain the scope resolution operator (::) and its use in PHP.

- Properties and methods can have access modifiers which control where they can be accessed.
- The (::) operator is used to **reference class-level elements** — things that belong to the class itself rather than to an object.

```
public $a=10; //availabale in class and outside class

private $b=20;// available only in own class

protected $c=30; // available only in own class and inherit class
```

- Example: class abc

```
                {
public $a=10; //availabale own class and inherit class and outside class
private $b=20;// available only in own class
protected $c=30; // available only in own class and inherit class

function sum()
        {
        echo $this->a."<br>";
        echo $this->b."<br>";
        echo $this->c."<br>";
        }
}
```

```
class xyz extends abc
{
    function multi()
    {
            echo $this->a."<br>";// work
            echo $this->b."<br>";// error
            echo $this->c."<br>";// work
    }
}

$obj=new xyz;
echo $obj->a; // run
echo $obj->b;// error
echo $obj->c;// error
```

Traits

11.Define traits in PHP and their purpose in code reuse.

- Traits are declared with the trait keyword: as class To use a trait in a class, use the use keyword: // for inheritance Traits are used to declare methods that can be used in multiple classes.
- A **trait** is a mechanism for **code reuse** in single inheritance languages like PHP. Traits allow you to define methods that can be **shared across multiple classes**, without requiring inheritance.
- Syntax of a Trait: trait Logger {

```
  public function log($message) {
    echo "Log: $message";
  }
}
class User {
   use Logger;
}
class Admin {
   use Logger;
}
```

12. Discuss the visibility of properties and methods in PHP (public, private, protected).

- Visibility controls **how and where** class properties and methods can be accessed. It's a key part of **encapsulation** in OOP.
- Properties and methods can have access modifiers which control where they can be accessed.

1. public:
   - **Access Level**: Accessible **from anywhere** — inside the class, outside the class, and by child classes.
   - **Use Case**: For methods or properties meant to be used freely.
   - Eg.

```
class User {
  public $name = "SVAYAM";

  public function greet() {
     echo "Hello, " . $this->name;
  }
 }
```

2. private
   - **Access Level**: Accessible **only within the class** where it's defined.

- **Use Case**: For sensitive data or internal logic that should not be exposed.
- **Eg**.

```
class Account {
   private $balance = 1000;
   private function getBalance() {
      return $this->balance;
   }
}
```

13. Explain type hinting in PHP and its benefits.

- **Type hinting** means specifying the **expected data type** for function arguments, return values, and class properties. It ensures that the values passed or returned match the declared types.
- Benefits of Type Hinting
  - **Improves code clarity**: Makes it obvious what type of data is expected.
  - **Reduces bugs**: Prevents unexpected behavior from wrong data types.
  - **Enables better auto-completion**: IDEs can suggest methods and properties more accurately.
  - **Supports strict typing**: When enabled, PHP throws errors for type mismatches.

14. Discuss the purpose of the final keyword in PHP and how it affects classes and methods.

- The final keyword is used to **restrict inheritance and method overriding** in object-oriented programming.
- PHP introduces the final keyword, which prevents child classes from overriding a method by prefixing the definition with final. If the class itself is being defined final then it cannot be extended.
- Eg.

```
final class a{
        function test()
        {
           echo "This is final method";
        }
    }
class b extends a  // extend not possible for final class
{
        function test()
        {
         echo "This is final method of b class";
   }
}
```

15. Discuss file handling in PHP, including opening, reading, writing, and closing files.

- PHP provides built-in functions to **open**, **read**, **write**, and **close** files. These operations allow you to interact with files on the server for storing or retrieving data.
- File handling is an important part of any web application. You often need to open and process a file for different tasks.

1) PHP Open File - fopen() // fopen("webdictionary.txt", "r") // r / w / r+ / w+ / a / a+

- Syntax: $handle = fopen("data.txt", "r"); // Opens file in read mode

- Modes Description

r Open a file for read only. File pointer starts at the beginning of the file

w Open a file for write only. Erases the contents of the file or creates a new file if it doesn't exist. File pointer starts at the beginning of the file

a Open a file for write only. The existing data in file is preserved. File pointer starts at the end of the file. Creates
a new file if the file doesn't exist
x Creates a new file for write only. Returns FALSE and an error if file already exists
r+ Open a file for read/write. File pointer starts at the beginning of the file
w+ Open a file for read/write. Erases the contents of the file or creates a new file if it doesn't exist. File pointer
starts at the beginning of the file
a+ Open a file for read/write. The existing data in file is preserved. File pointer starts at the end of the file. Creates
a new file if the file doesn't exist
x+ Creates a new file for read/write. Returns FALSE and an error if file already exists

2) PHP readfile() Function:

1. fread():Reads a specific number of bytes
Syntax: $content = fread($handle, filesize("data.txt"));
2. fgets(): Reads one line at a time
Syntax: while (!feof($handle)) { echo fgets($handle); }

3. file_get_contents(): Reads entire file into a string
Syntax: $data = file_get_contents("data.txt");

3) Writing to a File:

1.fwrite(): Writes data to a file
Syntax: $handle = fopen("data.txt", "w"); fwrite($handle, "Hello SVAYAM!");

2.file_put_contents(): Writes entire string to file (simpler)
Syntax: file_put_contents("data.txt", "Feedback saved!");

4) Closing a File: fclose()
Always close the file after you're done to free up system resources.
Syntax: fclose($handle);

MVC Architecture:

16. Discuss the Model-View-Controller (MVC) architecture and its advantages in web development.

- MVC is abbreviated as Model View Controller is a design pattern created for developing applications specifically web applications.
- As the name suggests, it has three major parts. The traditional software design pattern works in an "Input - Process - Output" pattern whereas MVC works as "Controller -Model - View" approach. With the emergence of the MVC model, creation of application takes different aspects individually into consideration.
- Let us discuss the three components of MVC in brief:

1. Model: The Model encloses the clean application related data. But the model does not deal with any logic about how to present the data.MODEL CONNECTED WITH Database + Logic (crud ins/ipd/del/sel)

2. Controller: The Controller is in between the model and the view element. It listens to all the incident and actions triggered in the view and performs an appropriate response back to the events.

3. View: The View element is used for presenting the data of the model to the user. This element deals with how to link up with the model's data but doesn't provide any logic regarding what this data all about or how users can use these data.

Connection with MySQL Database :

17. Explain how to connect PHP to a MySQL data base using mysqli or PDO.

1. Using mysqli (MySQL Improved)

- Procedural Style:

```
$connection = mysqli_connect("localhost", "username", "password", "database");
if (!$connection) {
        die("Connection failed: " . mysqli_connect_error());
         }
        echo "Connected successfully!";
```

- Object-Oriented Style:

```
$connection = new mysqli("localhost", "username", "password", "database");
        if ($connection->connect_error) {
            die("Connection failed: " . $connection->connect_error);
        }
        echo "Connected successfully!";
```

SQL Injection:

18. Define SQL injection and its implications on security.

- SQL injection is a code injection technique that might destroy your database. SQL injection is one of the most common web hacking techniques. SQL injection is the placement of malicious code in SQL statements, via web page input.

- SQL in Web Pages, SQL injection usually occurs when you ask a user for input, like their username/user id, and instead of a name/id, the user gives you an SQL statement that you will unknowingly run on your database.

- Many web applications are connected to a database. The database holds all the information the web application wish to store and use. SQL Injection is a technique which allows attackers to manipulate the SQL ("Structured Query Language").

Session and Cookies:

19. Explain the differences between sessions and cookies in PHP.

- Cookies:
  - Stored on the **client's browser**
  - Size Limit ~4KB
  - Less secure (visible to user)
  - Can persist after browser closes

- Accessible via $_COOKIE
- Use Case :Remember preferences, themes

➢ Sessions:

- Stored on the **server**
- Size Limit- No strict limit (server memory)
- More secure (hidden from user)
- Ends when browser or session expires
- Accessible via $_SESSION
- Use Case :Login, cart, user authentication

File Upload:

20. Discuss file upload functionality in PHP and its security implications.

➢ PHP makes it easy to upload files via HTML forms using the $_FILES superglobal.
➢ **Process Overview**
- User selects a file via an HTML form.PHP receives the file in a temporary location.The file is moved to a permanent directory using move_uploaded_file().

➢ **Security Implications**
- Unsecured file uploads can lead to serious vulnerabilities:
  1. **Malware Uploads**: Attackers may upload executable files like .php, .exe, or .js.
  2. **Remote Code Execution**: Uploaded scripts may be executed on the server.
  3. **Directory Traversal**: Manipulated file paths can access restricted areas.
  4. **Overwriting Files**: Existing files may be replaced if names are reused.
  5. **Denial of Service**: Large files can exhaust server resources.