Module 7 – Advanced PHP Exercises OOPs Concepts

THEORY EXERCISE:

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

Object-Oriented Programming (OOP) is a programming paradigm based on the concept of **objects**, which are instances of **classes**. It allows developers to structure programs using real-world entities that contain data (properties) and behaviors (methods). OOP improves code reusability, scalability, and maintainability.

Four Main Principles of OOP

1. **Encapsulation**
   Encapsulation is the practice of hiding an object's internal data and allowing access only through public methods. It protects data from direct modification and ensures controlled access.
2. **Inheritance**
   Inheritance allows a class (child/subclass) to inherit properties and methods from another class (parent/superclass). This promotes code reuse.
3. **Polymorphism**
   Polymorphism allows methods to have different behaviors depending on the object that calls them. It enables one interface to be used for different underlying data types.
4. **Abstraction**
   Abstraction hides complex implementation details and shows only essential features of an object. It helps reduce programming complexity.

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

A **class** in PHP is a blueprint used to create objects. It defines the **properties** (variables) and **methods** (functions) that an object will have.

Main components of a PHP class:

1. **Class Declaration**
   A class is declared using the class keyword followed by the class name.
2. **Properties**
   Properties store data of an object. They can have access modifiers:
   - public – accessible from anywhere
   - private – accessible only inside the class
   - protected – accessible in the class and subclasses
3. **Methods**
   Methods define the behavior of the class. They are functions written inside the class.
4. **Constructor**
   A special method __construct() that runs automatically when an object is created. It is used to initialize properties.
5. **Object Creation**
   An object is created using the new keyword.

• What is an object in OOP? Discuss how objects are instantiated from classes in PHP.

An **object** in Object-Oriented Programming (OOP) is an **instance of a class**. While a class is a blueprint or template, an object is the actual entity created from that blueprint. Objects contain **properties** (data) and **methods** (functions) that define their behavior.

Object Instantiation in PHP

In PHP, objects are instantiated from a class using the **new** keyword. When an object is created, the class constructor (__construct()) is automatically called to initialize the object's properties.

Syntax:

$objectName = new ClassName();

If the class has a constructor:

$objectName = new ClassName($value1, $value2);

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

Inheritance in Object-Oriented Programming (OOP)

Inheritance is an OOP concept where a **child (subclass)** acquires the properties and methods of a **parent (superclass)**. It promotes **code reusability**, **logical hierarchy**, and **easy maintenance**.

Inheritance in PHP

In PHP, inheritance is implemented using the **extends** keyword.

- The child class can **use**, **override**, or **add** new properties and methods.
- PHP supports **single inheritance** (a class can extend only one parent class).

Syntax:

```
class ChildClass extends ParentClass {
   // additional or overridden code
}
```

• Discuss method overloading and how it is implemented in PHP.

Method Overloading in OOP

Method overloading means defining **multiple methods with the same name but different parameter lists** (number or type of parameters). The correct method is called based on the arguments passed.

Method Overloading in PHP

Important:
PHP **does NOT support traditional method overloading** (same method name with different parameters).

Instead, PHP implements **method overloading using magic methods**, mainly:

- __call() → handles calls to undefined or inaccessible **object methods**
- __callStatic() → handles calls to undefined **static methods**

Using these magic methods, we can **simulate method overloading** by checking:

- Method name
- Number of arguments

- Argument values

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

Abstraction in OOP

Abstraction is the concept of **hiding implementation details** and showing **only essential features** of an object. It focuses on **what an object does**, not **how it does it**.

In PHP, abstraction is achieved using:

- **Abstract classes**
- **Interfaces**

Interfaces in PHP

An **interface** defines a **contract** that a class must follow.

- Interfaces contain **only method declarations**, not implementations.
- All methods in an interface are **public** by default.
- A class uses the **implements** keyword to implement an interface.
- A class can implement **multiple interfaces**.

Syntax:

```
interface InterfaceName {
   public function methodName();
}
```

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

A **constructor** is a special method in a PHP class that is **automatically called when an object is created**. It is defined using the method name **__construct()**.

Purpose of a Constructor

- Initializes object properties
- Sets default values
- Prepares the object for use
- Reduces repetitive code

Constructor Usage in PHP

- A class can have **only one constructor**
- It runs automatically when new keyword is used
- It can accept parameters

Syntax:

```
public function __construct() {
   // initialization code
}
```

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

A **destructor** is a special method in PHP that is **automatically called when an object is destroyed** or when the script execution ends. It is defined using the method name **__destruct()**.

Role of a Destructor

- Frees resources (database connections, file handles, memory)
- Performs cleanup tasks
- Executes code just before an object is removed from memory

When is a Destructor Called?

- When an object goes **out of scope**
- When an object is explicitly destroyed using unset()
- At the **end of the script execution**

Syntax:

```
public function __destruct() {
   // cleanup code
}
```

• Define magic methods in PHP. Discuss commonly used magic methods like get(), set(), and construct().

Magic methods in PHP are special predefined methods that **automatically execute** when certain actions are performed on an object, such as accessing undefined properties, calling inaccessible methods, or creating an object.

Magic methods **start with double underscores (__)** and help developers handle object behavior dynamically.

Commonly Used Magic Methods

1. **__construct()**
   Called automatically when an object is created. Used to initialize properties.
2. **__get($name)**
   Triggered when accessing an **undefined or inaccessible property**.
3. **__set($name, $value)**
   Triggered when assigning a value to an **undefined or inaccessible property**.

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

Scope Resolution Operator (::) in PHP

The **scope resolution operator (::)** is used in PHP to access:

- **Static properties**
- **Static methods**
- **Class constants**
- **Parent class methods or properties**

It allows access to class members **without creating an object**.

Uses of the Scope Resolution Operator

1. **Access static properties**

2. ClassName::$property;
3. **Access static methods**
4. ClassName::method();
5. **Access class constants**
6. ClassName::CONSTANT;
7. **Access parent class members**

   parent::method();

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

Traits in PHP are a mechanism for **code reuse**. They allow you to include methods from multiple sources into a single class, helping to overcome the limitation that PHP **does not support multiple inheritance**.

A trait is **not a class** and **cannot be instantiated** on its own. It simply groups reusable methods that can be used by multiple classes.

Purpose of Traits

- Enable **multiple code reuse**
- Reduce code duplication
- Share common behavior across unrelated classes
- Avoid complex inheritance hierarchies

Traits Syntax

```
trait TraitName {
   public function methodName() {
      // code
   }
}
```

Use traits in a class with the use keyword.

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

Visibility in PHP (Access Modifiers)

Visibility controls **where properties and methods can be accessed** in a PHP class. PHP provides three access modifiers:

1. Public

- Accessible **from anywhere**
- Can be accessed inside the class, outside the class, and by child classes

2. Private

- Accessible **only within the same class**
- **Not accessible** outside the class or in child classes

3. Protected

- Accessible **within the class and its child classes**
- **Not accessible** from outside the class.

Visibility	***Same Class***	***Child Class***	***Outside Class***
Public	✓	✓	✓
Protected	✓	✓	✗
Private	✓	✗	✗

• Explain type hinting in PHP and its benefits.

Type hinting in PHP allows you to **specify the expected data type** of function or method parameters (and return values). It ensures that the correct type of data is passed, helping prevent runtime errors.

Benefits of Type Hinting

- Improves **code reliability**
- Catches errors **early**
- Makes code **easier to understand**
- Helps with **debugging and maintenance**

Common Type Hints in PHP

- Scalar types: int, float, string, bool
- Arrays: array
- Objects: ClassName
- Interfaces
- Return types

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

The **final** keyword in PHP is used to **restrict inheritance and method overriding**.

Purpose of the final Keyword

- Prevent a class from being inherited
- Prevent a method from being overridden in child classes
- Improve **security** and **code stability**
- Ensure critical functionality is not changed

How final Affects Classes and Methods

1. **Final Class**

- A class declared as final **cannot be extended**

```
final class ClassName {
}
```

2. **Final Method**

- A method declared as final **cannot be overridden** in a child class

```
class ParentClass {
   final public function test() {
   }
}
```

• Explain the importance of email security and common practices to ensure secure email transmission.

Importance of Email Security

Email security is important to **protect sensitive information** and prevent misuse such as spam, phishing, hacking, and data leakage. Insecure email handling can allow attackers to inject malicious code or send fake emails.

Common Practices for Secure Email Transmission

- **Input sanitization** to remove malicious characters
- **Email validation** to ensure correct format
- Use of **secure mail servers (SMTP with SSL/TLS)**
- Avoid exposing email addresses publicly
- Prevent **email header injection**
- Use authentication mechanisms (SPF, DKIM, DMARC)

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

File Handling in PHP

File handling in PHP allows a program to **create, open, read, write, and close files** stored on the server. It is commonly used for logging, data storage, and reading configuration or content files.

Common File Handling Functions in PHP

1. **Opening a File**

```
fopen("file.txt", "r");
```

Modes:

- r → Read only
- w → Write (creates/overwrites)
- a → Append
- r+, w+, a+ → Read & write

2. **Reading a File**

- fread() – reads file content
- fgets() – reads one line
- file_get_contents() – reads entire file

3. **Writing to a File**

- fwrite() – writes data to file

4. **Closing a File**

fclose($file);

Closing a file is important to **free system resources**.

• Explain how to send emails in PHP using the mail() function and the importance of validating email addresses.

Sending Emails in PHP using mail()

PHP provides the built-in **mail()** function to send emails directly from a script. It uses the server's mail transfer agent (MTA) to deliver messages.

Syntax:

mail($to, $subject, $message, $headers);

Parameters

- $to → Receiver's email address
- $subject → Email subject
- $message → Email body
- $headers → Sender information (From, Reply-To, etc.)

Importance of Email Validation

Validating email addresses is important to:

- Prevent **invalid or fake emails**
- Avoid **email header injection attacks**
- Reduce spam and errors
- Ensure successful delivery

PHP provides filter_var() for **safe email validation**.

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

MVC (Model–View–Controller) is a software design pattern used in web development to **separate application logic into three interconnected components**:

1. **Model**
2. **View**
3. **Controller**

This separation of concerns makes applications **organized, maintainable, and scalable**.

Components of MVC

1. Model

- Handles **data and business logic**
- Interacts with database
- Does NOT deal with UI

2. View

- Responsible for **presentation / UI**
- Displays data to the user
- Contains HTML, minimal PHP

3. Controller

- Acts as a **bridge between Model and View**
- Receives user requests
- Fetches data from Model
- Passes data to View

Advantages of MVC Architecture

- Separation of concerns
- Easier maintenance
- Code reusability
- Better scalability
- Supports team development
- Clean and structured code

• Explain how to connect PHP to a MySQL database using mysqli or PDO.

Connecting PHP with MySQL

To connect PHP with a MySQL database, mainly **two methods** are used:

1 MySQLi (MySQL Improved)

- Specially designed for MySQL
- Supports both **Procedural** and **Object-Oriented** styles
- Fast and easy for beginners

Syntax (OOP style):

```
$conn = new mysqli($host, $user, $password, $database);
```

2 PDO (PHP Data Objects)

- Supports **multiple databases** (MySQL, Oracle, PostgreSQL, etc.)
- More **secure** (supports prepared statements)
- Flexible and professional approach

Syntax:

```
$conn = new PDO("mysql:host=$host;dbname=$db", $user, $password);
```

Importance of Error Handling

- Database credentials may be incorrect
- Server may be down
- Handling errors is important for **security** and **debugging**

Define SQL Injection and Its Implications on Security.

SQL Injection is a security attack in which an attacker injects malicious SQL code into a database query through user input.

Through this attack, an attacker can:

- View, modify, or delete database data
- Bypass login authentication
- Completely compromise the database

Implications on Security

- Unauthorized access
- Data theft
- Data loss
- Application hacking
- Legal and privacy issues

Example of SQL Injection Input

```
' OR '1'='1
```

Explain the Differences Between Sessions and Cookies in PHP

What are Sessions in PHP?

- Sessions are stored on the **server**

- They temporarily store user data
- Sessions are destroyed when the browser is closed
- More secure (data is not visible on the client side)

What are Cookies in PHP?

- Cookies are stored on the **client (browser)**
- They store limited data
- Have an expiration time
- Users can modify them (less secure)

Difference between Session and Cookies

Feature	**Session**	**Cookie**
Storage	Server	Browser
Security	More secure	Less secure
Size	Large data	Small data
Lifetime	Browser close	Expiry based
Access	Server only	Client & server

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

File Upload Functionality in PHP

File upload functionality in PHP allows users to upload files from their local computer to the web server. PHP uses the $_FILES superglobal array to handle uploaded files. When a file is uploaded, PHP temporarily stores it on the server and then allows the developer to move it to a permanent directory using the move_uploaded_file() function.

Security Implications of File Upload

Improper handling of file uploads can lead to serious security issues such as:

- Uploading malicious scripts
- Server compromise
- Unauthorized access to sensitive files

Security Best Practices

- Validate file types (extensions)
- Limit file size
- Rename files before saving
- Store files in a secure directory
- Prevent execution of uploaded files