Object-Oriented Programming (OOP) — Teaching Material

PART 1: OOP THEORY — The Foundation

Object-Oriented Programming (OOP) organizes code around objects rather than functions. An object represents real-world entities — each with properties (data) and methods (behaviors).

4 MAIN PRINCIPLES OF OOP

1. Encapsulation — 'Data hiding'

Encapsulation bundles data and methods into one unit (a class), protecting internal state from direct modification.

Example (JavaScript):

class BankAccount { #balance = 0; deposit(amount) { this.#balance += amount; } getBalance() { return this.#balance; } }

2. Inheritance — 'Reuse what already works'

Inheritance allows a class to extend another, reusing its properties and methods.
class Vehicle { start() { console.log("Engine started"); } } class Car extends Vehicle { drive() { console.log("Car is moving"); } }

3. Polymorphism — 'Many forms, one interface'

Different classes can have the same method names but different implementations.
class Animal { speak() { console.log("Some sound"); } } class Dog extends Animal { speak() { console.log("Bark"); } } class Cat extends Animal { speak() { console.log("Meow"); } }

4. Abstraction — 'Simplify by hiding complexity'

Abstraction hides unnecessary details, exposing only what is essential.
class CoffeeMachine { start() { console.log("Starting the machine..."); } brew() { console.log("Brewing coffee..."); } }

PART 2: PRACTICAL CLASS EXAMPLES

Example 1: Simple Class

class Student { constructor(name, age) { this.name = name; this.age = age; } introduce() { console.log(`Hi, I'm ${this.name}, ${this.age} years old.`); } }

Example 2: Inheritance

class Person { constructor(name) { this.name = name; } } class Teacher extends Person { teach(subject) { console.log(`${this.name} teaches ${subject}.`); } }

Example 3: Polymorphism

class Shape { area() { return 0; } } class Circle extends Shape { constructor(r){super();this.r=r;} area(){return Math.PI*this.r**2;} } class Square extends Shape { constructor(s){super();this.s=s;} area(){return this.s*this.s;} }

Example 4: Abstraction

class User { login(username, password){ console.log(`User ${username} logged in`); } } class Admin extends User { deleteUser(user){ console.log(`Admin deleted ${user}`); } }

PART 3: TEACHING FLOW PLAN (Recommended for 3 Classes)

Day	Session	Topic	Activity
Day 1	Session 1	What is OOP? Classes, Objects	Explain concepts with real examples
	Session 2	Encapsulation + Code Demo	Students make a Book class
Day 2	Session 1	Inheritance + Polymorphism	Students create Animal subclasses
	Session 2	Mini project	Zoo app where animals make sounds
Day 3	Session 1	Abstraction + Design	Teach User/Admin example
	Session 2	Practice & Quiz	Students create Library or CoffeeShop classes

Mini Project: School Management System

Students build classes: Person (base), Student & Teacher (subclasses), and Course class with enrollments.