**Course: Software Construction & Development**

**Course ID: CSCL-2126**

**Faculty: Shahzaib Naveed**

**Lecture # 6**

**Topics Covered:**

1. Local Storage of browser
2. Parsing of data (Array)
3. Oop Concepts in Php
4. Listing of products (Code Discussion)

**Code Reference:**

<?php

//--------------------------------Encapsulation--------------------------------------------

class Animal {

public $name; // Public property

private $age; // Private property

protected $sound; // Protected property

public function \_\_construct($name, $age, $sound) {

$this->name = $name;

$this->age = $age;

$this->sound = $sound;

}

public function makeSound() {

echo "{$this->name} makes a {$this->sound} sound.<br>";

}

public function setAge($age) {

// Public method to set the age

$this->age = $age;

}

public function getAge() {

// Public method to get the age

return $this->age;

}

}

class Dog extends Animal {

public function makeSound() {

// Override the makeSound method

echo "{$this->name} the dog barks: Woof woof!<br>";

}

public function showProperties() {

echo "Name: {$this->name}<br>";

// You can access protected properties in subclasses

echo "Sound: {$this->sound}<br>";

// You cannot access private properties from subclasses

// echo "Age: {$this->age}<br>"; // This would be an error

}

}

$animal = new Animal("Generic Animal", 5, "generic");

$dog = new Dog("Buddy", 3, "bark");

echo "Animal's Name: " . $animal->name . "<br>"; // Public property

// echo "Animal's Age: " . $animal->age . "<br>"; // This would be an error (private)

echo "Animal's Age: " . $animal->getAge() . "<br>"; // Accessing age via a public method

$animal->setAge(6); // Modifying age via a public method

echo "Updated Animal's Age: " . $animal->getAge() . "<br>";

$animal->makeSound(); // Calls the method in the Animal class

echo "<br>---<br>";

echo "Dog's Name: " . $dog->name . "<br>"; // Public property

// echo "Dog's Sound: " . $dog->sound . "<br>"; // This would be an error (protected)

$dog->showProperties(); // Accessing properties in a subclass

$dog->makeSound(); // Calls the overridden method in the Dog class

//--------------------------------Inheritance--------------------------------------------

class Vehicle {

protected $brand;

protected $model;

public function \_\_construct($brand, $model) {

$this->brand = $brand;

$this->model = $model;

}

public function getBrand() {

return $this->brand;

}

public function getModel() {

return $this->model;

}

public function drive() {

echo "The {$this->brand} {$this->model} is moving.<br>";

}

}

class Car extends Vehicle {

private $numWheels;

public function \_\_construct($brand, $model, $numWheels) {

parent::\_\_construct($brand, $model);

$this->numWheels = $numWheels;

}

public function getNumWheels() {

return $this->numWheels;

}

public function drive() {

echo "The {$this->brand} {$this->model} car with {$this->numWheels} wheels is driving.<br>";

}

public function getModel(){

return parent::getModel();

}

}

class Motorcycle extends Vehicle {

private $hasSidecar;

public function \_\_construct($brand, $model, $hasSidecar) {

parent::\_\_construct($brand, $model);

$this->hasSidecar = $hasSidecar;

}

public function hasSidecar() {

return $this->hasSidecar;

}

}

// Create instances of the derived classes

$car = new Car("Toyota", "Camry", 4);

$motorcycle = new Motorcycle("Harley-Davidson", "Sportster", true);

// Call methods on the objects

echo "Car: {$car->getBrand()} {$car->getModel()} with {$car->getNumWheels()} wheels.<br>";

$car->drive(); // Calls the overridden drive method in the Car class

echo "<br>";

echo "Motorcycle: {$motorcycle->getBrand()} {$motorcycle->getModel()}";

if ($motorcycle->hasSidecar()) {

echo " with a sidecar.";

} else {

echo " without a sidecar.";

}

$motorcycle->drive(); // Calls the overridden drive method in the Motorcycle class

echo "yesss: {$car->getModel()} ";

//---------------------------------------------Polymorphism---------------------------------

class MathOperation {

public function calculate($arg1, $arg2 = null, $arg3 = null) {

if ($arg3 !== null) {

return $this->calculateThreeArgs($arg1, $arg2, $arg3);

} elseif ($arg2 !== null) {

return $this->calculateTwoArgs($arg1, $arg2);

} else {

return $this->calculateOneArg($arg1);

}

}

private function calculateOneArg($arg1) {

return "You provided one argument: $arg1";

}

private function calculateTwoArgs($arg1, $arg2) {

return "You provided two arguments: $arg1 and $arg2";

}

private function calculateThreeArgs($arg1, $arg2, $arg3) {

return "You provided three arguments: $arg1, $arg2, and $arg3";

}

}

$math = new MathOperation();

echo $math->calculate(5) . "<br>";

echo $math->calculate(5, 10) . "<br>";

echo $math->calculate(5, 10, 15) . "<br>";

--------------------------------------------------------------------

class Animal {

public function speak() {

return "Animal speaks.";

}

}

class Dog extends Animal {

public function speak() {

return "Woof! Woof!";

}

}

class Cat extends Animal {

public function speak() {

return "Meow!";

}

}

$dog = new Dog();

$cat = new Cat();

echo "Dog says: " . $dog->speak() . "<br>";

echo "Cat says: " . $cat->speak() . "<br>";

?>