Marwadi University

Marwadi University

01IT0701 – Advance Web Technology

Unit – 1
Object Oriented PHP

Prof. Jaydeep K. Ratanpara
Department of Computer Engineering

OUTLINE

Marwadi University

- Object Oriented Programming with PHP
- Classes
- Properties
- Methods
- Constructor
- Destructor
- Getter and Setter
- Encapsulation
- Inheritance
- Data Abstraction
- Polymorphism

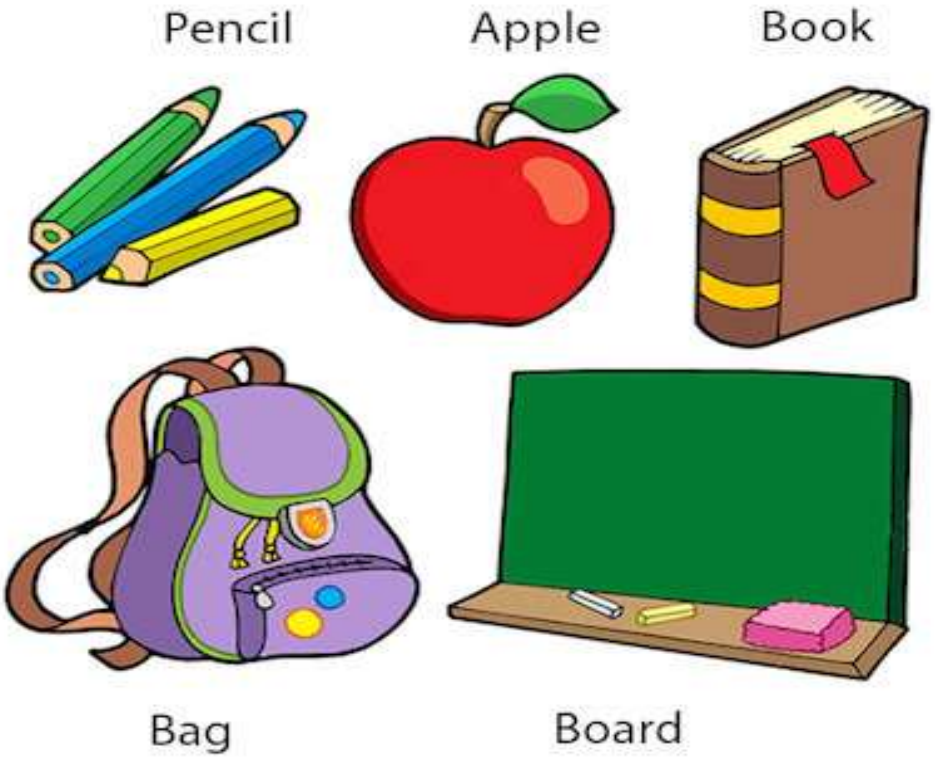
OBJECT ORIENTED PROGRAMMING

Marwadi University

What is OOP?

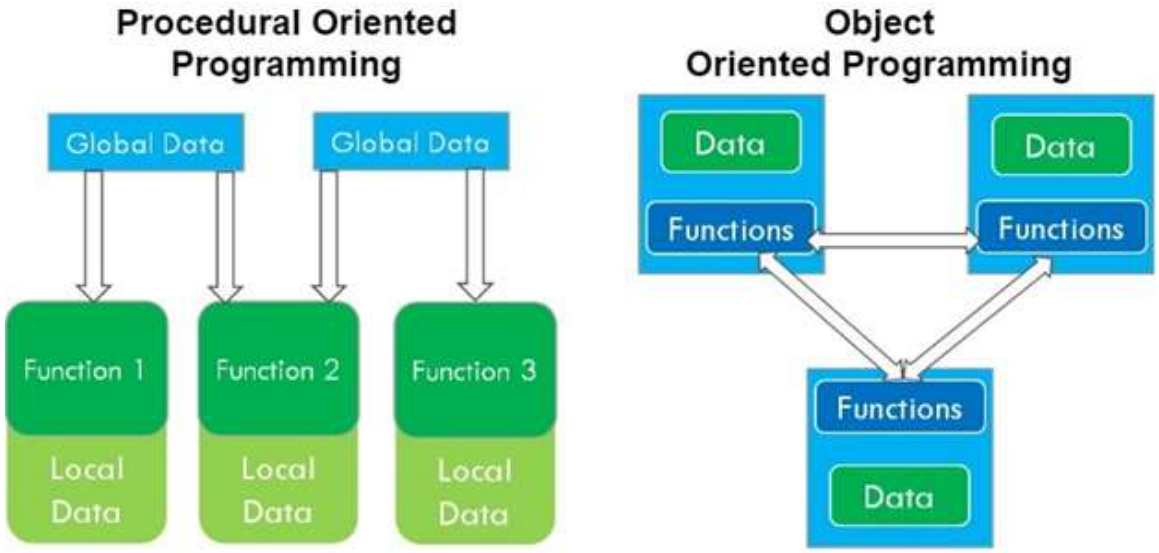
- OOP stands for Object-Oriented Programming.
- The word Object-Oriented is the combination of two words i.e. Object and Oriented.
- The dictionary meaning of the object is an entity that exists in the real world.
- The meaning of oriented is interested in a particular kind of thing or entity.
- In general terms, it is a programming pattern that rounds around an object or entity are called object-oriented programming.

Objects: Real World Examples



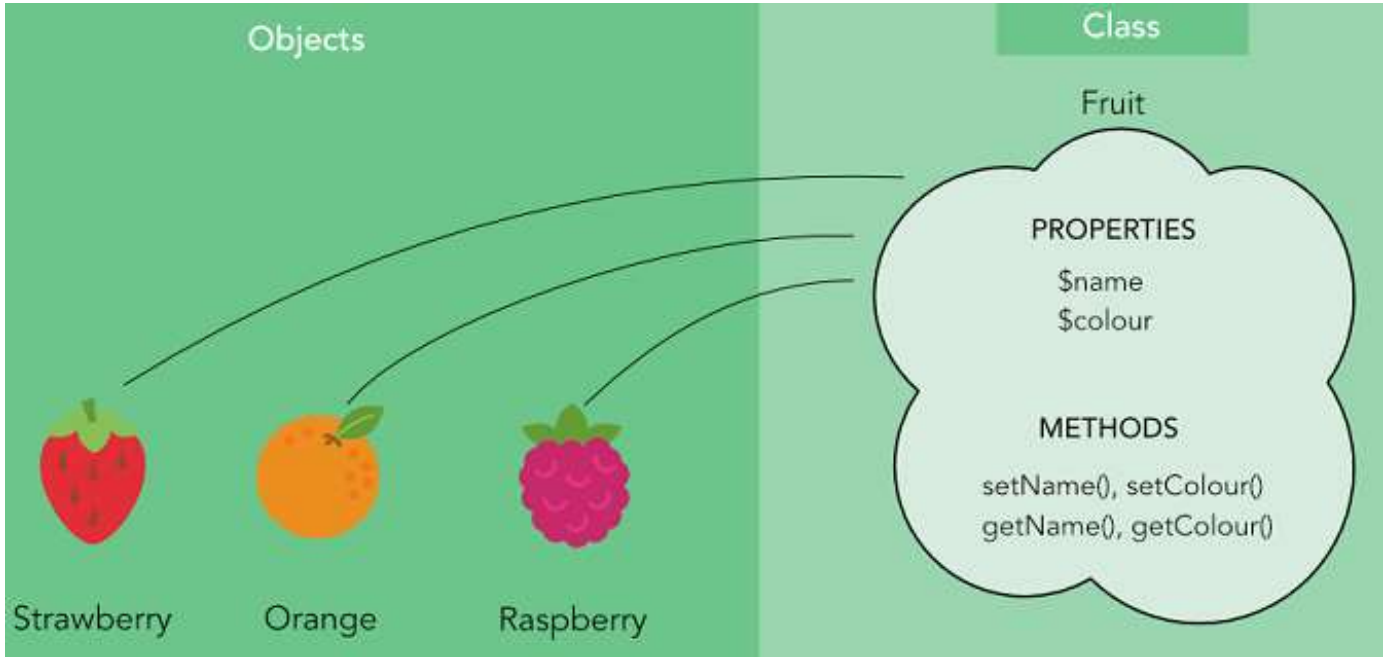
OBJECT ORIENTED PROGRAMMING

- Object-oriented programming: Creating objects that contain both data and functions.
- Procedural programming: Writing procedures or functions that perform operations on the data.

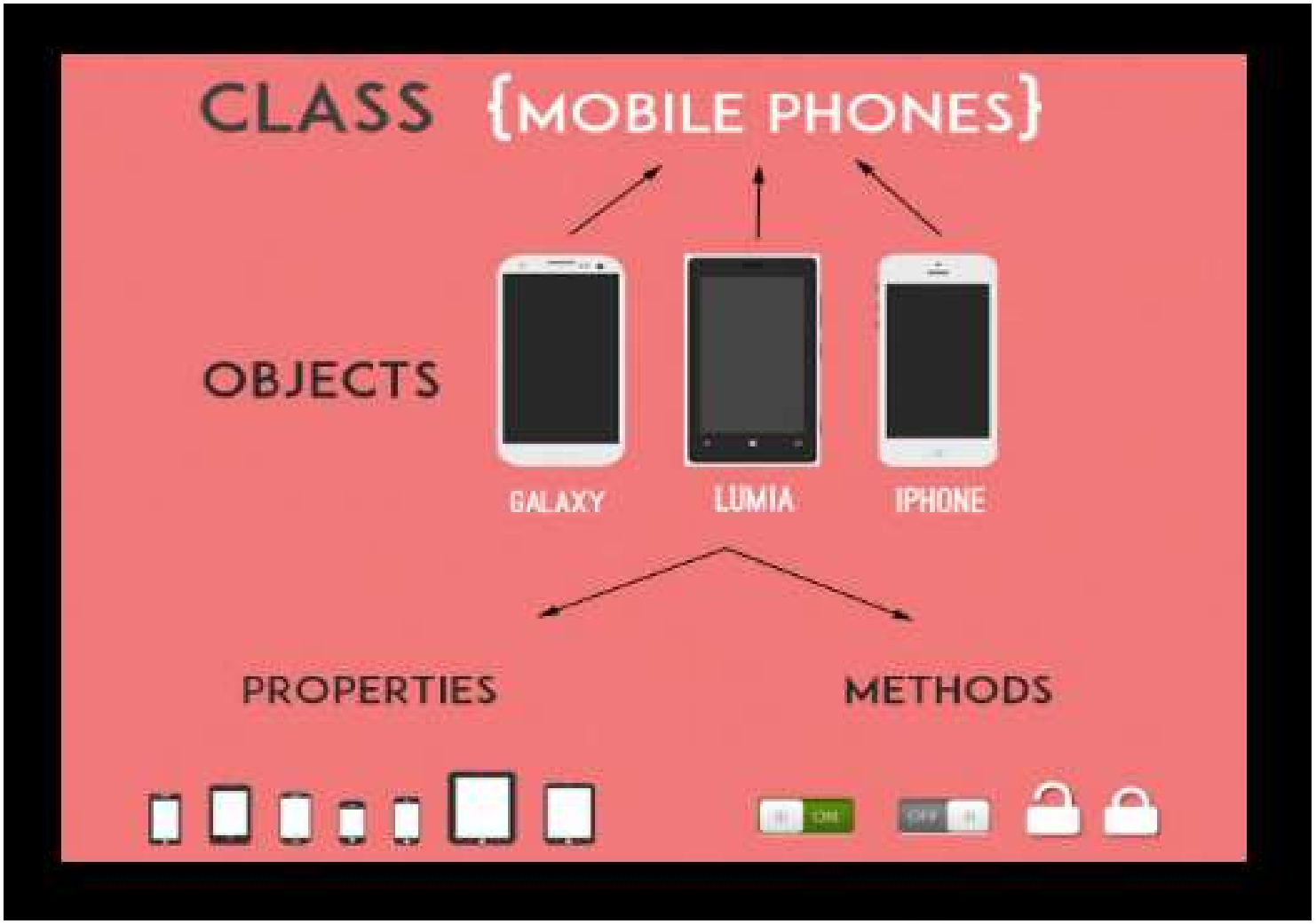


CLASSES

- Classes and objects are the two main aspects of object-oriented programming.
- a class is a template for objects, and an object is an instance of a class.
- When the individual objects are created, they inherit all the properties and behaviors from the class, but each object will have different values for the properties.



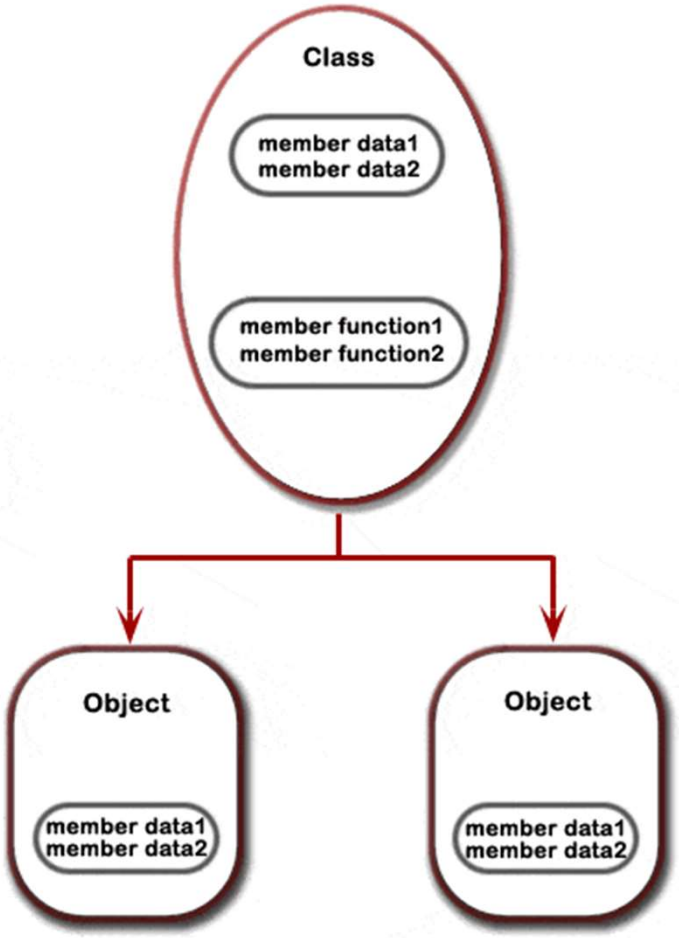
CLASSES


CLASS {MOBILE PHONES}
OBJECTS
GALAXY
LUMIA
IPHONE
PROPERTIES
METHODS

CLASSES

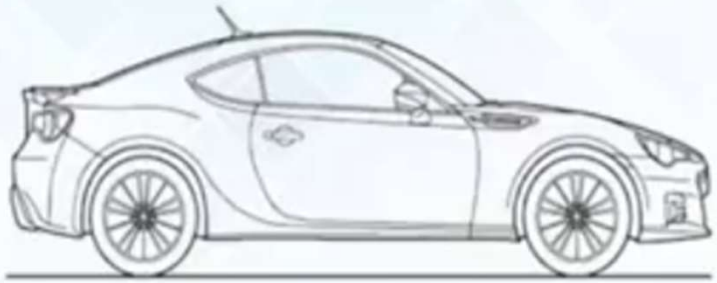
- Programmer-defined data type, which includes local functions as well as local data.

- Blueprint or Prototype or a set of instruction to build a specific type of object



CLASSES

Marwadi University



Class - Car
(Blueprint)

- **States of a Car** : Color - Orange
  Model - 1
  Price - 50000
- **Behavior of Car** : Speed up
  Change gear

CLASSES

Marwadi University

Syntax of Classes

```
<?php
        class Myclass
        {
                //Add property statements here
                //Add the methods here
        }
?>
```

Class = Template for objects

Object = Instance of class

CLASSES

Marwadi University

- The class definition starts with the keyword class followed by a class name, then followed by a set of curly braces({ }).
- Classes enclose constants, variables (called "properties"), and functions (called "methods") belonging to the class.
- Class names usually begin with an Uppercase letter to distinguish them from other identifiers.

CLASSES

Marwadi University

Class Example

```php
<?php
class Fruit
{
  // Properties
  public $name;
  public $color;

  // Methods
  function set_name($name)
  {
    $this->name = $name;
  }
  function get_name()
  {
    return $this->name;
  }
}
?>
```

CLASSES

Marwadi University

Class Example

```php
<?php
class Fruit {
  // Properties
  public $name;
  public $color;

  // Methods
  function set_name($name)
  {
    $this->name = $name;
  }
  function get_name()
  {
    return $this->name;
  }
function set_color($color)
  {
    $this->color = $color;
  }
function get_color()
  {
    return $this->color;
  }
}

$apple = new Fruit();
$apple->set_name('Apple');
$apple->set_color('Red');

echo "Name: " . $apple->get_name();
echo "<br>";
echo "Color: " . $apple->get_color();
?>
```

PROPERTIES

Marwadi University

- Class member variables are called properties or attributes or fields.
- The properties hold specific data and related with the class in which it has been defined.
- Use one of the keyword public, protected, or private followed by a normal variable declaration.
- If declared using var (compatibility with PHP 4), the property will be defined as public.

PROPERTIES

Marwadi University

Properties can be

- **Public:** the property or method can be accessed from everywhere. This is default. either by the script or from another class
- **Protected :** the property or method can be accessed within the class and by classes derived from that class
- **Private :** the property or method can ONLY be accessed within the class. No access is granted from outside the class, either by the script or from another class.

PROPERTIES

Marwadi University

Properties Example

```php
<?php
class Fruit
{
  public $name;
  protected $color;
  private $weight;
}

$mango = new Fruit();
$mango->name = 'Mango'; // OK
$mango->color = 'Yellow'; // ERROR
$mango->weight = '300'; // ERROR
?>
```

PROPERTIES

Marwadi University

Properties Example

```php
<?php
class Fruit
{
  public $name;
  public $color;
  public $weight;

  function set_name($n)
  { // a public function (default)
    $this->name = $n;
  }
  protected function set_color($n)
  { // a protected function
    $this->color = $n;
  }
  private function set_weight($n)
  { // a private function
    $this->weight = $n;
  }
}

$mango = new Fruit();
$mango->set_name('Mango'); // OK
$mango->set_color('Yellow'); // ERROR
$mango->set_weight('300'); // ERROR
?>
```

TWO WAYS TO INITIALIZE PROPERTIES

Marwadi University

PHP - The $this Keyword

- The $this keyword refers to the current object, and is only available inside methods.

- So, where can we change the value of the $name property? There are two ways:

- Inside the class (by creating method)
- Outside the class (by directly changing the property value)

TWO WAYS TO INITIALIZE PROPERTIES

Marwadi University

1. Inside the class (by creating method)

```php
<?php
class Fruit
{
  public $name;
  function set_name($name)
  {
    $this->name = $name;
  }
}
$apple = new Fruit();
$apple->set_name("Apple");

echo $apple->name;
?>
```

TWO WAYS TO INITIALIZE PROPERTIES

Marwadi University

2. Outside the class (by directly changing the property value)

```php
<?php
class Fruit

{
  public $name;
}

$apple = new Fruit();
$apple->name = "Apple";

echo $apple->name;
?>
```

METHODS

Marwadi University

- The functions which are declared in a class are called methods.
- A class method is exactly similar to PHP functions.
- Use one of the keyword public, protected, or private followed by a method name.
- A valid method name starts with a letter or underscore, followed by any number of letters, numbers, or underscores.

METHODS

Marwadi University

Example

```php
<?php
class Car
{
 // Properties
 public  $name;
 public $color;

 // Methods
 function set_name($name)
 {
   $this->name = $name;
 }
 function get_name()
{
   return $this->name;
 }
 function set_color($color)
{
   $this->color = $color;
 }
 function get_color()
 {
   return $this->color;
 }
}

$audi = new Car();
$audi->set_name('Audi');
$audi->set_color('Black');

echo $audi->get_name();
echo "<br>";
echo $audi->get_color();
?>
```

var_dump()

It is used to display the structured information (type and value) about one or more objects.

```
<?php
        class Fruit
        {
                public $name;
                public $color;
        }
        $obj = new Fruit();
        $obj ->name=“Orange”;
        var_dump($obj);
?>
```

Output:

object(Fruit)#1 (2) { ["name"]=> string(12) "“Orange”" ["color"]=> NULL }

instanceof( )

Marwadi University

It is used to check if object belongs to a specific class or not.

```
<?php
        class Fruit
        {
                public $name;
        }
        $obj = new Fruit();
        var_dump($obj instanceof Fruit);
?>
```

Output:

bool(true)

isset() Function

The isset() function checks whether a variable is set, which means that it has to be declared and is not NULL.

This function returns true if the variable exists and is not NULL, otherwise it returns false.

Note: If multiple variables are supplied, then this function will return true only if all of the variables are set.

Syntax

isset(variable, ...);

isset() Function

Marwadi University

```php
<?php
$a = 0;
// True because $a is set
if (isset($a)) {
  echo "Variable 'a' is set.<br>";
}

$b = null;
// False because $b is NULL
if (isset($b)) {
  echo "Variable 'b' is set.";
}
?>
```

Output : Variable 'a' is set.

unset() Function

The unset() function unsets a variable.

Syntax : unset(variable, ...);

```php
<?php
$a = "Hello world!";
echo "The value of variable 'a' before unset: " . $a . "<br>";
unset($a);
echo "The value of variable 'a' after unset: " . $a;
?>
```

Output:
The value of variable 'a' before unset: Hello world!
The value of variable 'a' after unset:

MAGIC METHOD: CONSTRUCTOR

Marwadi University

- Constructors are special member functions.
- A constructor allows you to initialize an object's properties upon creation of the object.
- It is a special built-in method, added with PHP 5.
- Classes which have a constructor method, execute automatically when an object is created.
- The constructor is not required if you don't want to pass any property values or perform any actions when the object is created.
- PHP only ever calls one constructor.

MAGIC METHOD: CONSTRUCTOR

Marwadi University

Advantages of Constructor

- Encourage re-usability avoiding re-initializing whenever instance of the class is created .
- The ability to pass parameters which are helpful in automatic initialization of the member variables during creation time
- It can call class member methods and functions.
- It can call other Constructors even from Parent class.

MAGIC METHOD: CONSTRUCTOR

Marwadi University

Syntax of Constructor

- The 'construct' method starts with two underscores (__)

```
function __construct([argument1,,.., argumentN])
{
        /* Class initialization code */
}
```

The type of argument1,.....,argumentN are mixed.

MAGIC METHOD: CONSTRUCTOR

Marwadi University

Example

```php
<?php
class Fruit
{
  public $name;
  public $color;

  function __construct($name, $color)
  {
    $this->name = $name;
    $this->color = $color;
  }
  function get_name()
  {
    return $this->name;
  }
  function get_color()
  {
    return $this->color;
  }
}

$apple = new Fruit("Apple", "red");
echo $apple->get_name();
echo "<br>";
echo $apple->get_color();
?>
```

MAGIC METHOD: CONSTRUCTOR

Marwadi University

Calling of parent constructor

- Two ways to define constructor using "__ construct( )" or same name as class name.
- Parent constructors are not called implicitly if the child class defines a constructor.
- In order to run a parent constructor, a call to parent::__construct( ) within the child constructor is required.

MAGIC METHOD: CONSTRUCTOR

Marwadi University

Calling of parent constructor Example

```php
<?php
    class TC2
    {
        public function TC2()
        {
                echo "constructor TC2";
        }
    }
class Abatch extends TC2
 {
        public function __construct()
        {
                parent::TC2();
                echo "<br>";
                echo "constructor A batch";
        }
}
```

Constructor using class name

Constructor using __construct() method

```php
$obj= new Abatch();
?>
```

Output:
constructor TC2
constructor A batch

MAGIC METHOD: CONSTRUCTOR

Marwadi University

Calling of parent constructor Example

```
<?php
class BaseClass
{
   function __construct()
  {
      print "In BaseClass constructor";
      echo "<br>";
   }
}

class SubClass extends BaseClass
{
   function __construct()
  {
      parent::__construct();
      print "In SubClass constructor";
      echo "<br>";
   }
}

class OtherSubClass extends BaseClass
{
   // inherits BaseClass's constructor
}

// In BaseClass constructor
$obj = new BaseClass();

// In BaseClass constructor
// In SubClass constructor
$obj = new SubClass();

// In BaseClass constructor
$obj = new OtherSubClass();
?>
```

Output:

```
In BaseClass constructor
In BaseClass constructor
In SubClass constructor
In BaseClass constructor
```

MAGIC METHOD: DESTRUCTOR

Marwadi University

- The destructor is the counterpart of constructor.
- A destructor is called when the object is destructed or the script is stopped or exited.
- If you create a __destruct( ) function, PHP will automatically call this function at the end of the script.
- A destructor function cleans up any resources allocated to an object after the object is destroyed.
- A destructor function is commonly called in two ways: When a script ends or manually delete an object with the unset( ) function.

MAGIC METHOD: DESTRUCTOR

Marwadi University

Syntax

- The destructor method starts with two underscores (__).
- Cleaning up of resources before memory release or closing of files takes place in the destructor method, whenever they are no longer needed in the code.

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

MAGIC METHOD: DESTRUCTOR

Marwadi University

```php
<?php
class Car
{
        public $name;
        public $color;

        function __construct($name, $color)
        {
                $this->name = $name;
                $this->color = $color;
        }
        function __destruct()
        {
                echo "The Car is {$this->name} and the color is {$this->color}.";
        }
}
        $car1 = new Car("Audi", "red");
?>
```

Output : The Car is Audi and the color is red.

The example has a __construct( ) function that is automatically called when you create an object from a class, and a __destruct( ) function that is automatically called at the end of the script:

MAGIC METHOD: DESTRUCTOR

Marwadi University

```php
<?php
class Fruit
{
  public $name;

  function __construct($name)
  {
                $this->name = $name;
  }
  function __destruct( )
  {
                echo "The fruit is {$this->name}.". "<br>";
                echo "The " . $this->name . " is being deleted" . "<br>";
  }
}

$apple = new Fruit("Apple");
$orange= new Fruit("Orange");
?>
```

Output:

```
The fruit is Orange.
The Orange is being deleted
The fruit is Apple.
The Apple is being deleted
```

MAGIC METHOD: DESTRUCTOR

Marwadi University

```
<?php
class cars
{
public static $count=0;
public function __construct($type, $year)
{
                $this->type= $type;
                $this->year= $year;
                cars::$count++;
}

public function __destruct()
{
echo "The " . $this->type . " is being deleted" . "<br>";
}
}
```

```
$car1= new cars("Toyota", 2014);
$car2= new cars("BMW", 2015);
$car3= new cars("Tesla", 2016);
$car4 = new cars("Maruti",2017);

unset($car2);

echo "The number of objects is : " . cars::$count;
echo "</br>";
?>
```

Output:

```
The BMW is being deleted
The number of objects is : 4
The Maruti is being deleted
The Tesla is being deleted
The Toyota is being deleted
```

GETTER AND SETTER METHODS

Marwadi University

- Getters and Setters are object methods that allow you to control access to a certain class variables / properties
- A getter allows to you to retrieve or get a given property.
- A setter allows you to set the value of a given property.
- If any property has private access modifier then you can not directly called that property so create getter and setter function to access it.
- **Usage of Getter & Setter**
- The getter method retrieves an attribute in other classes.
- The setter method modifies/changes the value of the attribute/data.

GETTER AND SETTER METHODS

Marwadi University

Example

```
<?php
  class Person
  {
        private $name;
        public function setName($name)
        {
              $this->name = $name;
        }
        public function getName( )
        {
              return ‘Welocme'. $this->name;
        }
  }
    $person = new Person( );
    $person->setName('Alex');
    $name = $person->getName( );
    echo $name;
?>
```

Output:
WelocmeAlex

ENCAPSULATION

Marwadi University

- It states the concept of binding data (attributes) and methods (functions) into a single unit (Class). This term is also referred to as Information Hiding.
- we are hiding the real implementation of data from the user and also does not allow anyone to manipulate data members except by calling the desired operation.
- In encapsulation, a class's variables are hidden from other classes and can only be accessed by the methods of the class in which they are found.
- Encapsulation in PHP can be achieved using the implementation of access specifiers (Private).

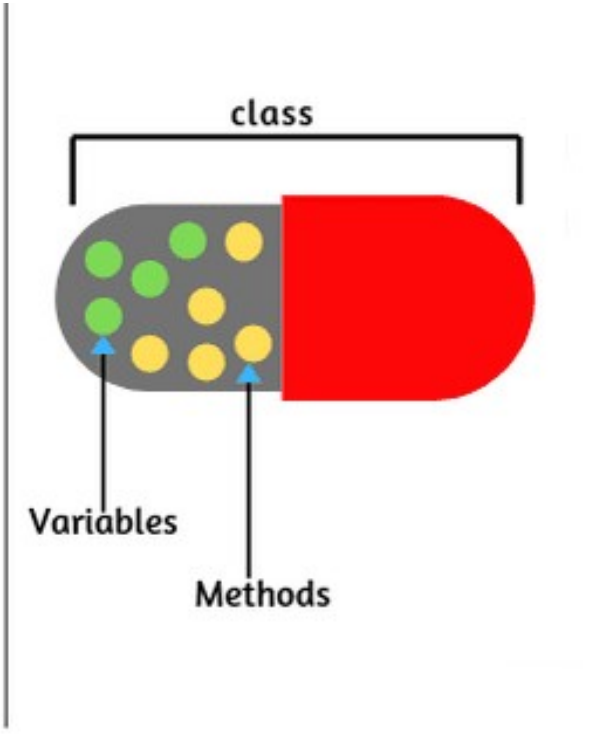
ENCAPSULATION

Marwadi University

```
class
{

  data members
        +
  methods (behavior)

}
```



ENCAPSULATION

Marwadi University

Advantages of Encapsulation

- Data Hiding
- Data security
- Reduces complexity
- Reliability
- Easier testing of code

ENCAPSULATION

Marwadi University

Example

```
<?php
class Student
{
        private $name;
        private $gender;
        public function setname($name)
        {
                $this->name=$name;
                echo ‘Name is “.$name;
                echo “<br>”;
        }
        public function setgender($gender)
        {
                if($gender == “Male” || $gender == “Female”)
                {
                        $this->gender=$gender;
                        echo "Gender is ".$gender;
                }
        }
```

Programs continues…..

ENCAPSULATION

Marwadi University

```
        public function getname( )
        {
                return $this->name;
        }
        public function getgender( )
        {
                return $this->gender;
        }
}
        $std1 = new Student( );
        $std1 ->setname(“Student 1”);
        $std1 ->setgender(“Male”);
?>
```

ENCAPSULATION

Marwadi University

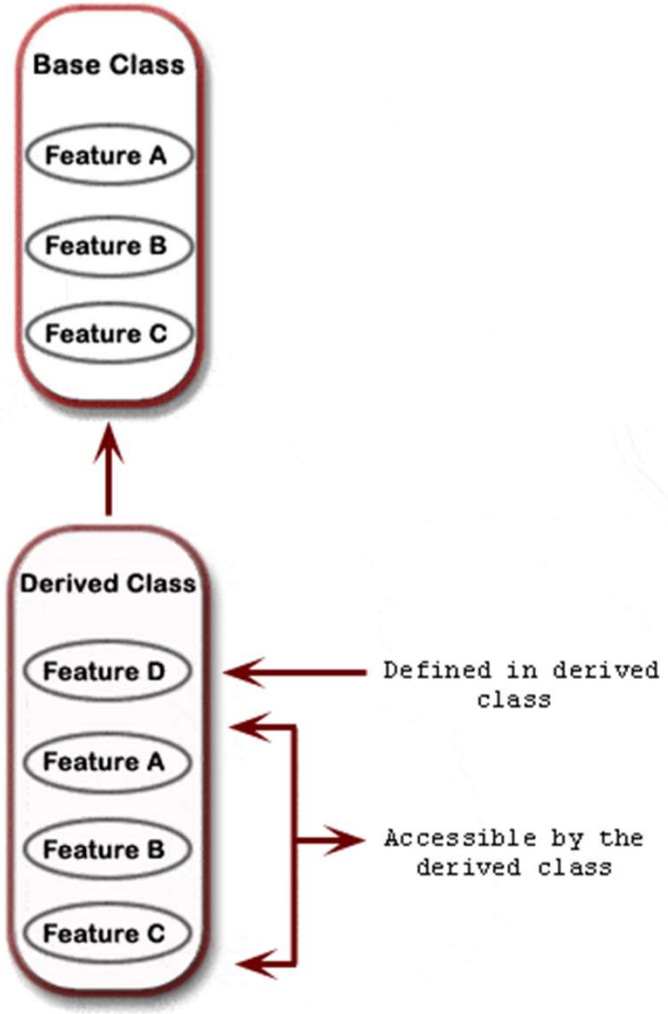
```php
<?php
class ATM
{
    private $custid;
    private $atmpin;
    public function PinChange($custid,$atmpin)
    {
        ---------perform tasks-----
    }
    public function CheckBalance($custid,$atmpin)
    {
        ---------perform tasks-----
    }
    public function miniStatement($custid)
    {
        ---------perform tasks-----
    }
}
$obj = new ATM( );
$obj -> CheckBalance(10005285637,1234);
?>
```

INHERITANCE

Marwadi University

- Inheritance = When a class derives from another class.
- The child class will inherit all the public and protected properties and methods from the parent class. In addition, it can have its own properties and methods.
- Inheritance allows reusing code.
- Super class is the base class or parent class.
- Sub class or derived class or child class can add properties and methods.
- An inherited class is defined by using the extends keyword.

INHERITANCE



INHERITANCE

```php
<?php
class books
{
        public $title;
        public $author;
        public $price;
        public function __construct($t,$a,$p)
        {
                $this->title=$t;
                $this->author=$a;
                $this->price=$p;
        }
        public function get_values(){
                echo "Title is:".$this->title."<br>";
                echo "Author is:".$this->author."<br>";
                echo "Price is:".$this->price;
        }
}
```

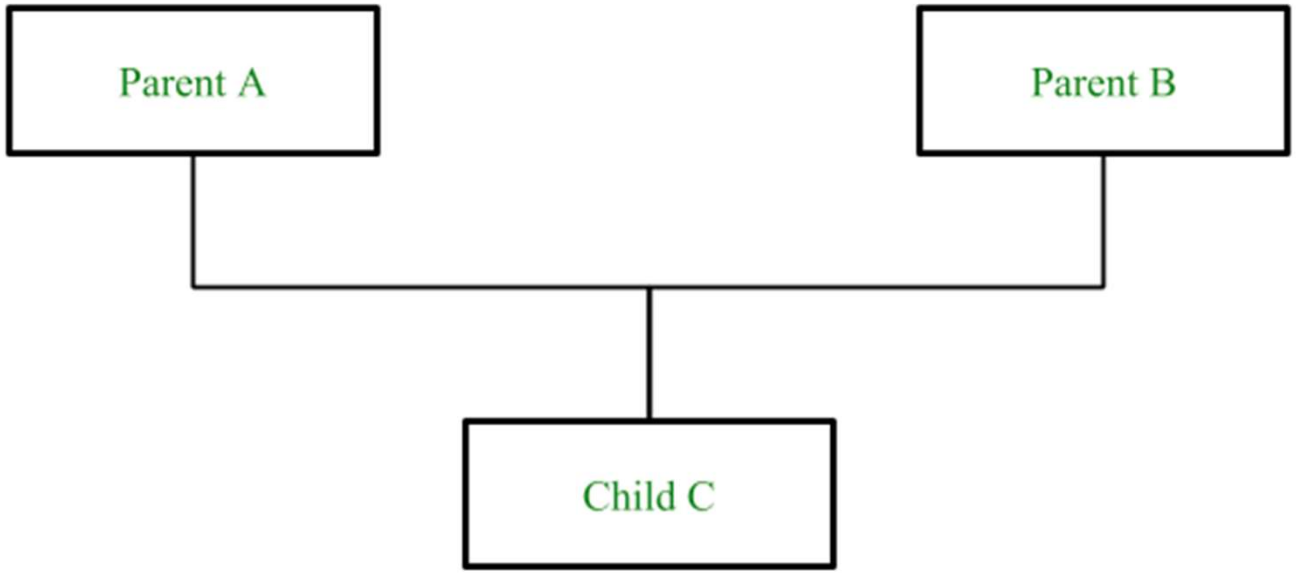
INHERITANCE

Marwadi University

```
class technicalbooks extends books
{
        public $publication;
        public function set($p)
        {
                $this->publication=$p;
        }
        public function get( )
        {
            echo "<br>Publication is:".$this->publication;
        }
}
$AWT=new technicalbooks("Advance web Technology", "XYZ","350");
$AWT->set("Tata McGraw Hill");
$AWT->get_values();
$AWT->get();
?>
```

INTERFACE

- PHP doesn’t support multiple inheritance but by using Interfaces in PHP or using Traits in PHP instead of classes, we can implement it.



INTERFACE

Marwadi University

- The class that is fully abstract is called an interface.
- An interface is similar to a class except that it cannot contain code.
- The interface does not contain any concrete methods (methods that have code). All the methods of an interface are abstract methods.
- An interface can define method names and arguments, but not the contents of the methods.
- Any classes implementing an interface must implement all methods defined by the interface.
- Interfaces can't maintain Non-abstract methods.

INTERFACE

Marwadi University

- Derived classes may implement more than one interface.
- Interfaces may inherit from other interfaces using the extends keyword.
- All methods are assumed to be public in the interface definition can be defined explicitly as public or implicitly.
- **USAGE OF INTERFACE**
- To implement multiple inheritance
- It referred to as the next level of abstraction(Define method but how is it implemented that is not given)

INTERFACE

Marwadi University

SYNTAX

```
interface MyInterface
{
   function method1( );
   function method2( );
}
class MyClass implements MyInterface
{
   function method1( )
    {
                  // definition of method1
    }
   function method2( )
    {
                 // definition of method2
    }
}
```

INTERFACE

Marwadi University

EXAMPLE -1

```php
<?php
interface Getdata
{
        public function get( );
}
interface Printdata
{
         public function put( );
}
class class1 implements Getdata, Printdata
{
        public function get( )
        {
                echo "Get data method is called";
        }
        public function put( )
        {
                echo "<br>Put data method is called";
        }
}
```

```php
$obj=new class1;
$obj->get();
$obj->put();
?>
```

INTERFACE

Marwadi University

EXAMPLE -2

```php
<?php
interface Animal
{
  public function makeSound( );
}

class Cat implements Animal
{
        public function makeSound( )
        {
        echo "Meow";
        }
}

$animal = new Cat( );
$animal->makeSound( );
?>
```

INTERFACE

Marwadi University

EXAMPLE -3

```php
<?php
interface Animal
{
  public function makeSound( );
}
class Cat implements Animal
{
        public function makeSound( )
        {
                echo “ Meow ";
        }
}
class Dog implements Animal
{
        public function makeSound( )
        {
                echo " Bark ";
        }
}
class Mouse implements Animal
{
        public function makeSound( )
        {
        echo " Squeak ";
        }
}

// Create a list of animals
$cat = new Cat( );
$dog = new Dog( );
$mouse = new Mouse( );
$animals = array($cat, $dog, $mouse);

// Tell the animals to make a sound
foreach($animals as $animal)
{
  $animal->makeSound();
}
?>
```

TRAITS

Marwadi University

- What if a class needs to inherit multiple behaviors?
- The trait is a type of class which enables multiple inheritance.
- Traits are used to declare methods that can be used in multiple classes.
- Traits can have methods and abstract methods that can be used in multiple classes, and the methods can have any access modifier
- Reduces code duplication because no need to redeclare same methods over and over again.

TRAITS

Marwadi University

SYNTAX

```
<?php
trait TraitName
{
        // some code...
}
?>
```

To use a trait in a class, use the **use** keyword:

```
<?php
class MyClass
{
        use TraitName;
}
?>
```

TRAITS

EXAMPLE

```php
<?php
trait message1
{
    public function msg1( )
    {
        echo "Trait 1: msg1 called";
    }
}
trait message2
{
    public function msg2( )
    {
        echo "Trait 2: msg2 called";
    }
}
class class1
{
    use message1;
 }
class class2
{
    use message1, message2;
 }
$obj = new class1( );
$obj->msg1( );
echo "<br>";
$obj2 = new class2( );
$obj2->msg1( );
echo "<br>";
$obj2->msg2( );
?>
```

DATA ABSTRACTION

Marwadi University

- Abstraction is the any representation of data in which the implementation details are hidden.
- An abstract class is a class that contains at least one abstract method.
- An abstract method is a method that is declared, but not implemented in the code.
- Abstract class can not be instantiated.
- Abstract class can contain abstract methods or non-abstract methods or constructors also.

DATA ABSTRACTION

Marwadi University

When a child class is inherited from an abstract class,

- The child class method must be defined with the same name and it redeclares the parent abstract method.
- The child class method must be defined with the same or a less restricted access modifier.
- The number of required arguments must be the same. However, the child class may have optional arguments in addition.

DATA ABSTRACTION

Marwadi University

SYNTAX OF ABSTRACT CLASS

An abstract class or method is defined with the **abstract** keyword:

```
<?php
        abstract class ParentClass
        {
                abstract public function someMethod1();
                abstract public function someMethod2($name, $color);
                public function someMethod3( )
                {
                        //Some code here
                }
        }
?>
```

DATA ABSTRACTION

Marwadi University

EXAMPLE

```php
<?php
abstract class ParentClass
{
  // Abstract method with an argument
  abstract protected function prefixName($name);
}

class ChildClass extends ParentClass
{
   public function prefixName($name)
   {
      if ($name == "John Doe")
      {
            $prefix = "Mr.";
      }
      else if ($name == "Jane Doe")
      {
           $prefix = "Mrs.";
      }
   else
   {
       $prefix = "";
   }
    return "{$prefix} {$name}";
  }
}

$class = new ChildClass;
echo $class->prefixName("John Doe");
echo "<br>";
echo $class->prefixName("Jane Doe");
?>
```

OutPut:

Mr. John Doe
Mrs. Jane Doe

OBJECT CLONING

Marwadi University

- Cloning is used to create a copy of an object by using the clone keyword.
- When an object is cloned, PHP 5 will perform a shadow copy of all of the object's properties.
- Any properties that are references to other variables, will remain references.
- PHP provides a special method __clone to copy an object.
- if a __clone() method(magic method) is defined, then the newly created object's __clone() method will be called, to allow any necessary properties that need to be changed.

OBJECT CLONING

Marwadi University

```
<?php
   class class1{
      public $x;
      private $y;
      public function __construct($x,$y){
         $this->x=$x;
         $this->y=$y;
      }
      function __clone(){
         $this->x="Z";
      }
   }
   $a = new class1("AWT","Sem 7");
   $b = clone $a;
   var_dump($a);
   echo "<br>";
   var_dump($b);
 ?>
```

OBJECT CLONING

Marwadi University

```
object(class1)#1 (2) {
   ["x"]=>
   string(3) "AWT"
   ["y":"class1":private]=>
   string(5) "Sem 7"
}
<br>object(class1)#2 (2) {
   ["x"]=>
   string(1) "Z"
   ["y":"class1":private]=>
   string(5) "Sem 7"
}
```

POLYMORPHISM

Marwadi University

- Poly (meaning many) and morph (meaning forms).
- Two types of Polymorphism; they are:
    1. Compile time (function overloading)
    2. Run time (function overriding)
- **Method Overloading:** Same method name with different signature, since PHP doesn't support method overloading concept (magic method __call() is used to achieve this)
- **Method Overriding:** When same methods defined in parents and child class with same signature i.e know as method overriding

POLYMORPHISM

Marwadi University

Method overloading

- Method overloading contains same method name and that method performs different task according to number of arguments.
- For example, find the area of certain shapes where radius are given then it should return area of circle if height and width are given then it should give area of rectangle and others.
- Like other OOP languages method overloading can not be done by native approach.
- In PHP method overloading is done with the help of magic method _call(). This method takes method name and arguments.
- The __call(). method is invoked when a non existing method or an inaccessible method is called.

POLYMORPHISM

Marwadi University

EXAMPLE – METHOD OVERLOADING

```php
<?php
class shape
{
// __call is magic function which accepts function name & arguments
    function __call($name_of_function, $arguments)
    {
       // It will match the function name
        if($name_of_function == 'area')
        {
            switch (count($arguments))
            {

              // If there is only one argument area of circle
                case 1: return 3.14 * $arguments[0];

              // IF two arguments then area is rectangle;
                case 2: return $arguments[0]*$arguments[1];
            }
        }
    }
}
```

```php
$s1 = new Shape;

// Function call
echo($s1->area(2));
echo "\n";

// calling area method for rectangle
echo ($s1->area(4, 2));
?>
```

POLYMORPHISM

Marwadi University

Method overriding

- In method overriding, both parent and child classes should have same method name with and number of arguments.
- It is used to replace parent method in child class. The purpose of overriding is to change the behavior of parent class method.
- The two methods with the same name and same parameter is called overriding.

POLYMORPHISM

EXAMPLE – METHOD OVERRIDING (Run time Polymorphism)

```
<?php
class Base
{
    function display( )
    {
      echo "\nBase class function declared final!";
    }
    function demo( )
    {
      echo "\nBase class function!";
    }
}
class Derived extends Base
{
    function demo()
    {
      echo "\nDerived class function!";
    }
}
  $ob = new Base;
  $ob->demo();
  $ob->display();
  $ob2 = new Derived;
  $ob2->demo();
  $ob2->display();
?>
```

Output

```
Base class function!
Base class function declared final!
Derived class function!
Base class function declared final!
```

Marwadi University

THANK YOU