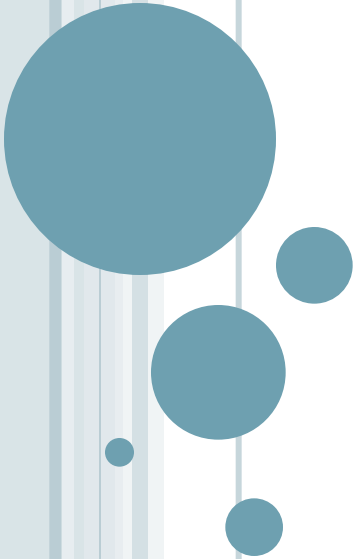
OBJECT-ORIENTED PROGRAMMING WITH PHP



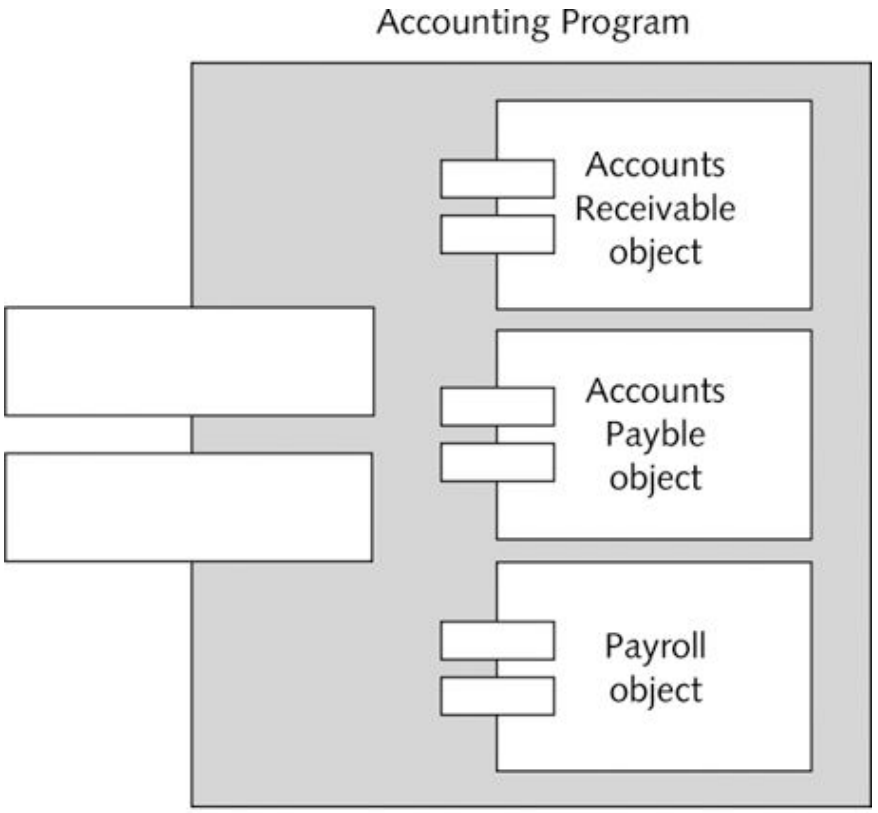
OBJECT-ORIENTED PROGRAMMING

- **Object-oriented programming** (OOP) refers to the creation of reusable software objects that can be easily incorporated into multiple programs
- An **object** refers to programming code and data that can be treated as an individual unit or component
- Objects are often also called **components**

OBJECT-ORIENTED PROGRAMMING

- **Data** refers to information contained within variables or other types of storage structures
- The functions associated with an object are called **methods**
- The variables that are associated with an object are called **properties** or **attributes**

OBJECT-ORIENTED PROGRAMMING



UNDERSTANDING ENCAPSULATION

- Objects are **encapsulated** – all code and required data are contained within the object itself
- Encapsulated objects hide all internal code and data
- An **interface** refers to the methods and properties that are required for a source program to communicate with an object
- **Encapsulated objects allow users to see only the methods and properties of the object that you allow them to see**
- Encapsulation reduces the complexity of the code
- Encapsulation prevents other programmers from accidentally introducing a bug into a program, or stealing code

OBJECT-ORIENTED PROGRAMMING AND CLASSES

- The code, methods, attributes, and other information that make up an object are organized into **classes**
- An **instance** is an object that has been created from an existing class
- Creating an object from an existing class is called **instantiating** the object
- An object **inherits** its methods and properties from a class — it takes on the characteristics of the class on which it is based

CREATING A CLASS DEFINITION (CONTINUED)

- The ***ClassName*** portion of the class definition is the name of the new class
- Class names usually begin with an uppercase letter to distinguish them from other identifiers
- Within the class's curly braces, declare the data type and field names for each piece of information stored in the structure

```
class BankAccount {
data member and member function definitions
}
```

```
$Checking = new BankAccount(); // creating object
```

USING OBJECTS IN PHP SCRIPTS

- Declare an object in PHP by using the **`new`** operator with a class constructor

- A **class constructor** is a special function with the same name as its class that is called automatically when an object from the class is instantiated
- The syntax for instantiating an object is:

```
$ObjectName = new ClassName();
```

USING OBJECTS IN PHP SCRIPTS

- The identifiers for an object name:
  - Must begin with a dollar sign
  - Can include numbers or an underscore
  - Cannot include spaces
  - Are case sensitive

```
$Checking = new BankAccount();
```

  - Can pass arguments to many constructor functions

```
$Checking = new BankAccount(01234587, 1021, 97.58);
```

USING OBJECTS IN PHP SCRIPTS (CONTINUED)

- After an object is instantiated, use a hyphen and a greater-than symbol (–>) to access the methods and properties contained in the object
- Together, these two characters are referred to as ***member selection notation***
- With member selection notation append one or more characters to an object, followed by the name of a method or property

USING OBJECTS IN PHP SCRIPTS (CONTINUED)

- With methods, include a set of parentheses at the end of the method name, just as with functions
- Like functions, methods can also accept arguments

```
$Checking->getBalance();
$CheckNumber = 1022;
$Checking->getCheckAmount($CheckNumber);
```

SOME MORE INFORMATION

- You can create object of the class in some different way also. Following is some of the example of creating object of class.

E.g. $checking = 'BankAccount';

$acc1 = new $checking(); (old style)

E.G

$acc1=new BankAccount();

Or $acc1 = new BankAccount;

E.G

$acc1 = new BankAccount;

$acc2=new acc1; (new version style)

AVAILABLE VISIBILITY IN PHP CLASSES

- There are 3 type of visibility available in php for controlling your property or method.
- **Public**: Public method or variable can be accessible from anywhere. Inside the class, outside the class and in child class also.
- **Private**: Method or property with private visibility can only be accessible inside the class. You can not access private method or variable from outside of your class.
- **Protected**: Method or variable with protected visibility can only be access in the derived class. Or in other word in child class. Protected will be used in the process of inheritance.

CONSTRUCTOR OF CLASSES AND OBJECTS

- A constructor allows you to initialize an object's properties upon creation of the object.
- Constructor is nothing but a function defined in your php class.
- Constructor function automatically called when you will create object of the class.
- As soon as you will write $object = new yourClass() your constructor function of the class will be executed.
- From php5 you can also create constructor by defining **magic function __construct.**
- By **creating __construct() function**,: PHP will automatically call this function when you create an object from a class.

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

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

$apple = new Fruit("Apple");
echo $apple->get_name();
?>
```

- PHP lacks support for declaring multiple constructors of different numbers of parameters for a class unlike languages such as Java. So, if we declare an another constructor in the above example like so. PHP will throw a fatal error

DESTRUCTOR

- Destructor is called when the object is destructed or script is exit or stopped.
- It is created by __destruct() function.
- When __destruct() is created, PHP will automatically call this function at the end of the script.

- E.g constructor.php

PLAYING WITH VISIBILITY AND OTHER FEATURE OF THE CONSTRUCTOR

- Reason behind creating constructor function public is it is accessible from outside of the class.
- This function is executed when we are creating object.
- So php will always through error if you will create your constructor private

1. A **private constructor** is used to prevent the direct creation of objects from the class.
2. The expensive process is performed within the private constructor.
3. The only way to create an instance from the class is by using a static method that creates the object only if it wasn't already created.

- Private_construct.php

STATIC METHODS AND PROPERTIES IN PHP

- Static methods and properties in php can directly accessible without creating object of class.
- Your php class will be static class if your all methods and properties of the class is static.
- Static Methods and Properties in PHP will be treated as public if no visibility is defined.
- It is directly accessible from class with the help of **::**(scope resolution operator)
- You can declare static property using **static** keyword.
- For within the class you can access static property using **self** keyword.
- Static variable or property are the best way to preserve value of the variable within the context of different instance.

STATIC METHODS OR FUNCTIONS

- As in general class various processes are same for methods and properties, **same is with Static Methods and Properties in PHP**.
- You can create your function or method static using **static** keyword.
- You can access all visible static methods for you using **::** like in static variables.

- Static methods

```
Public static function static_method() {
}
```

- Static property

```
static $staticproperty=0;
```

call static method

```
classname::static_method();
```

- Class Test{

```
static
static function static_method() {
    echo "Hello World!";
  }
 public function welcome(){
  self::static_method();
 }
}
 Test::static_method();
```

A static method/properties can be accessed from a method in the same class using the **self** keyword and double colon (::): instead of $this

Staticbasic.php

$THIS AND SELF

self vs. *$this*

$this	self
Represents an instance of the class or object	Represents a class
Always begin with dollar ($) sign	Never begin with dollar($) sign
Is followed by the -> operator	Is followed by the :: operator
The property name after the -> operator does not take dollar ($) sign, e.g., $this->property.	The property name after the :: operator always take the dollar ($) sign.

NAMESPACE

- You know that you cannot have two classes with the same name, since PHP would notknow which one is being referred to when creating a new object. To solve this issue, PHPallows the use of namespaces, which act as paths in a filesystem
- you can have as many classes with the same name as you need, as long as they are all defined in different namespaces.
- Namespaces are qualifiers that solve two different problems:

- They allow for better organization by grouping classes that work together to perform a task
- They allow the same name to be used for more than one class

- Defining namespace
- ?php
- namespace Tutsplus;
-
- // code which is defined here belongs to the Tutsplus namespace
- ?>

```
?php
// mylib.php
namespace Tutsplus\Code;

// define a class
class Tutorial {
   public function __construct() {
      echo "Fully qualified class name: ".__CLASS__.“<br>";

   }
}
 // define a function
function fooBar() {
    echo "Fully qualified function name: ".__FUNCTION__.“<br>";
}
 ./ define a constant
const RECORDS_PER_PAGE = 50;
?>
```

```php
<?php
// app.php
require "mylib.php";

// instantiate the Tutorial class
$objTutorial = new \Tutsplus\Tutorial();

// call the function
echo \Tutsplus\fooBar();

// display the constant
echo \Tutsplus\RECORDS_PER_PAGE;
?>
```

Namespace.php app.php

INHERITANCE IN PHP

- Inheritance in php is introduced from php5 version.
- With the help of inheritance we can get all property and method of one class in another class [child class].
- The class who inherit feature of another class known as **child class**. The class which is being inherited is know as **parent 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
- An inherited class is defined by using the extends keyword.

CALLING A CONSTRUCTOR OF SUPERCLASS IN SUBCLASS

Constructor of the super/parent class can be called in subclass constructor by calling **parent :: __construct() in subclass's constructor method.**

E.g constructor of the BaseClass called into the SubClass using

parent :: _consturct() statements

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

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

```
$obj = new BaseClass();

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

Output

```
// Creating Baseclass object
$obj = new BaseClass();
// In BaseClass constructor

// creating SubClass object and call Subclass constructor
$obj = new SubClass();


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

CALLING A METHOD OF SUPERCLASS IN SUBCLASS

- In inheritance public or protected method of super/parent class is called into the subclass/child class.
- If the same method-name is not available[overridding is not done] in the subclass then superclass method can be called using
- **$this->baseclassmethodname()**
-

```php
<?php
class BaseClass {
    function __construct() {
        Echo "<br> In BaseClass constructor";
    }
    protected function display(){
            echo "<br> function of the superclass is called";
    }
}
```

```php
Class SubClass extends BaseClass {
    function __construct() {
        parent::__construct();
        echo "<br> In SubClass constructor";
    }
   public function subdisplay(){

        $this->display();
        echo "<br> function of the subclass is called";
    }

  $obj = new BaseClass; // Baseclass constructor called
 echo "<br> Creting Subclass object";
 $obj1 = new SubClass;  // SubClass constructor called
 $obj1->subdisplay();

}
```

OVERRIDING OF METHOD

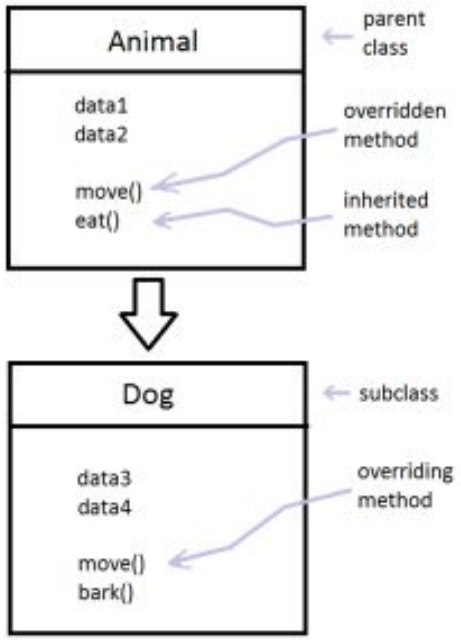
- Overriding is an object-oriented programming feature that enables a child class to provide different implementation for a method that is already defined and/or implemented in its parent class or one of its parent classes.
- In function overriding, both parent and child classes should have same function 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.

METHOD OVER-RIDDING

- **It is also called as Runtime Polymorphism or late binding**

- Method overriding, in object-oriented programming, is a language feature that allows a subclass or child class to provide a specific implementation of a method that is already provided by one of its superclasses or parent classes. It allows for a specific type of polymorphism (subtyping)

- The implementation in the subclass overrides (replaces) the implementation in the superclass by providing a **method that has same name, same parameters or signature, and same return type as the method in the parent class.**



- The version of a method that is executed will be determined by the object that is used to invoke it.
- if an object of a parent class is used to invoke the method, then the version in the parent class will be executed, but if an object of the subclass is used to invoke the method, then the version in the child class will be executed.
- It is used to replace parent method in child class. The purpose of overriding is to change the behavior of parent class method.

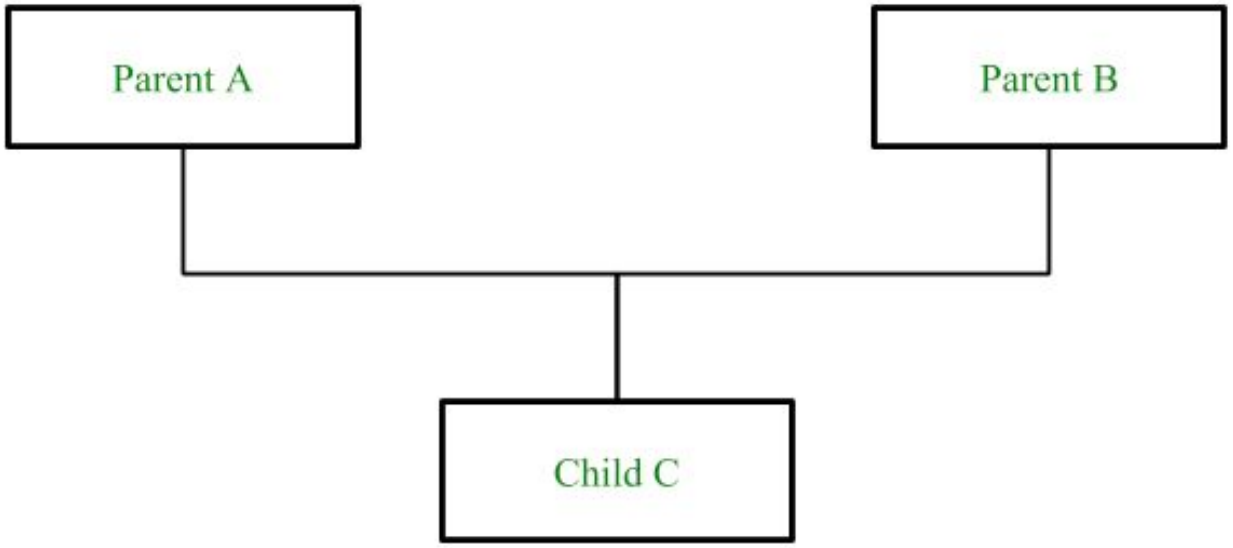
- Overriddingdemo.php
- For html [override.html , overridedemo.php]

MULTILEVEL AND MULTIPLE INHERITANCE IN PHP

- In php multilevel inheritance is possible but multiple inheritance is not possible.
- But hierarchical inheritance is possible.
- Hierarchical means Parent inherit property of grand parent class. Grand child inherit property of parent class.
- So in multilevel inheritance child can get some property of from grand parent class also.

MULTIPLE INHERITANCE

- Multiple Inheritance is the property of the Object Oriented Programming languages in which child class or sub class can inherit the properties of the multiple parent classes or super classes.



- PHP OOP does not allow multiple inheritance, **it allow only multilevel inheritance.**

MULTILEVEL INHERITANCE

- In the multi-level inheritance, we will inherit the one base class into a derived class, and then the derived class will become the base class for another derived class.

```
<?php
//PHP program to demonstrate the multi-level inheritance.
class Base
{
    function BaseFun()
    {
        echo "BaseFun() called<br>";
    }
}
class Derived1 extends Base
{
    function Derived1Fun()
    {
        echo "Derived1Fun() called<br>";
    }
}
```

```
class Derived2 extends Derived1
{
    function Derived2Fun()
    {
        echo "Derived2Fun() called<br>";
    }
}

$dObj = new Derived2();

$dObj->BaseFun();
$dObj->Derived1Fun();
$dObj->Derived2Fun();

?>
```

Output

BaseFun() called
Derived1Fun() called
Derived2Fun() called

e.g multilevel_inheritance.php , multilevel2.php

AUTOLOADING

- Autoloading allows you to automatically include the class files.
- When you are working in a quite large PHP project, you will have a large number of class files. In each PHP file, you will need to have a bunch of include or require statements at the beginning to use those classes.
- <?php
- include 'Class1.php';
- include 'Class2.php';
- ....
- And more
- ?>

- PHP's autoloading feature doesn't need such explicit inclusion. Instead, when a class is used (for declaring its object etc.) PHP parser loads it automatically,
- if it is registered with spl_autoload_register() function. Any number of classes can thus be registered. This way PHP parser gets a last chance to load class/interface before emitting error.

```
spl_autoload_register(function ($class_name) {
   include $class_name . '.php';
});
```

e.g Shape.php , Circle.php , objdemo.php

ABSTRACT CLASSES

- Abstract classes are those classes which can not be directly initialized and that you can not create object of it.
- If your class has at least one method abstract than your class is abstract class.
- **Abstract method is only declared but not defined**.
- You can create abstract classes in php using **abstract** keyword.
- If you have an abstract method in your abstract class and once you inherit your abstract class then it is necessary to define your abstract method.
- You can define your abstract method in child class with the same visibility or less restricted visibility..
- Abstract class can contain the constructor

- E.g abstract_basic.php
-

WHAT IS INTERFACE ?

- Object interfaces allow you to create code which specifies which methods a class must implement, without having to define how these methods are handled.
- Interfaces are defined using the *interface* keyword, in the same way as a standard class, but without any of the methods having their contents defined.
- All methods declared in an interface must be public; this is the nature of an interface.

IMPLEMENTS

- To implement an interface, the *implements* operator is used.
- All methods in the interface must be implemented within a class;
- Classes may implement more than one interface if desired by separating each interface with a comma.
- It's possible for interfaces to have constants.
- Interface constants works exactly like class constants except they cannot be overridden by a class/interface that inherits them.
- You can not implement two interfaces with same function names.
- If you are using default argument then you can change your value of the argument.

- E.G interface_basic.php

INTERFACES

- Note:-
  - A class cannot implement two interfaces that share function names, since it would cause ambiguity.
  - Interfaces can be extended like classes using the extends operator.
  - The class implementing the interface must use the exact same method signatures as are defined in the interface. Not doing so will result in a fatal error.

DIFFERENCES BETWEEN ABSTRACT CLASS AND INTERFACE

- In abstract classes this is not necessary that every method should be abstract. But in interface every method is abstract.
- Multiple and multilevel both type of inheritance is possible in interface. But single and multilevel inheritance is possible in abstract classes.
- Method of php interface must be public only. Method in abstract class in php could be public or protected both.
- In abstract class you can define as well as declare methods. But in interface you can only define your methods.

FINAL KEYWORD

- 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.

EXAMPLE :-

```
<?php
    class BaseClass {
       public function test() {
          echo "BaseClass::test() called<br>";
       }

       final public function moreTesting() {
          echo "BaseClass::moreTesting() called<br>";
       }
    }
       class ChildClass extends BaseClass {
       public function moreTesting() {
          echo "ChildClass::moreTesting() called<br>";
       }
    }
 ?>
```

FINAL CLASS EXAMPLE

```
<?php
 final class BaseClass {
       public function test() {
          echo "BaseClass::test() called<br>";
       }
       // Here it doesn't matter if you specify the function as final or not
       final public function moreTesting() {
          echo "BaseClass::moreTesting() called<br>";
       }
    }
 class ChildClass extends BaseClass{}
 // Results in Fatal error: Class ChildClass may not inherit from final cl
     ass (BaseClass)
 ?>
```

POLYMORPHISM

- **Polymorphism in PHP can be implemented by either the use of interfaces or abstract classes.**

OVERLOADING AND OVERRIDING IN PHP

- Overriding is a process by which you can re-declare your parent class method in child class.
- Overriding is required when your parent class has some method, but in your child class you want it with different behavior.
- Overloading means assigning extra work to the existing method.
- We can not implement overloading by creating two functions with same names in class.
- So to implement overloading in php we will take help of magic method __call.
- Magic method __call invoked when method called by class object is not available in class.

MAGIC METHOD

- Magic methods are special methods which override PHP's default's action when certain actions are performed on an object.
- Read document
- https://www.php.net/manual/en/language.oop5.magic.php

__call :

is triggered when invoking inaccessible methods in an object context.

__callStatic() is triggered when invoking inaccessible methods in a static context.

E.G calltest.php

EXCEPTION HANDLING

- Exception Handling in PHP is almost similar to exception handling in all programming languages.
- PHP provides following specialized keywords for this purpose.

- **try:** It represent block of code in which exception can arise.
- **catch:** It represent block of code that will be executed when a particular exception has been thrown.
- **throw:** It is used to throw an exception. It is also used to list the exceptions that a function throws,
- finally: It is used in place of catch block or after catch block basically it is put for cleanup activity in PHP code. Which will be execute even exception is not occur.

- E.g

```php
<?php
//user-defined function with an exception
function checkNumber($num) {
  if($num>=1) {
    //throw an exception
    throw new Exception("Value must be less than 1");
  }
  return true;
}
```

```php
//trigger an exception in a "try" block
try {
  checkNumber(5);
  //If the exception throws, below text will not be display
  echo 'If you see this text, the passed value is less than 1';
}

//catch exception
catch (Exception $e) {
  echo 'Exception Message: ' .$e->getMessage();
}
?>
```

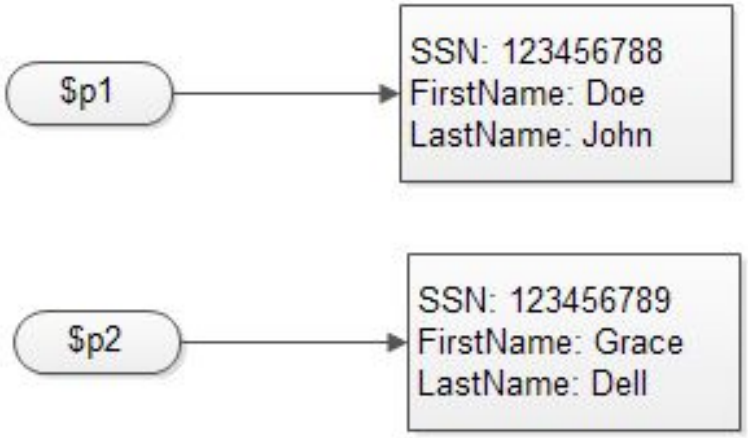
OBJECT CLONING IN PHP

- If you will directly copy objects in php, then it will copy by reference, not by value.
- Means if you will change main object data then copied object will be affected.
- Also if you will change value of the copied object then main object value will be changed.
- So if you want to create copy of the **object which should never referenced to original object then you can take help of object cloning in php**.

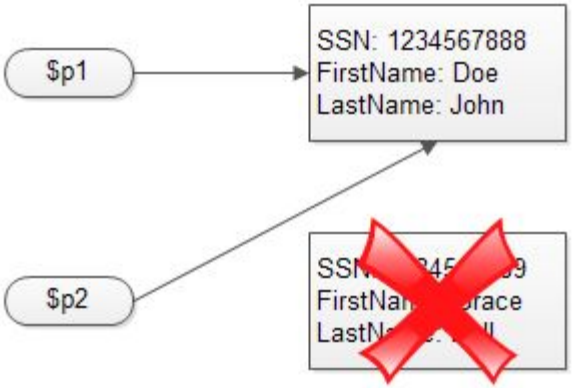
OBJECT CLONING WITH MAGIC METHOD __CLONE

- Suppose you want to change value of your property a of the test class in case of cloning of object in php.
- We can change behavior of the clone object in php using magic method __clone.
- Magic method clone is executed when object cloning is performed. As soon as php execute statement $c = clone $a, _ clone method invoked.

CLONE


$p1
SSN: 123456788
FirstName: Doe
LastName: John
$p2
SSN: 123456789
FirstName: Grace
LastName: Dell

```
$p2 = $p1;
```


$p1
SSN: 1234567888
FirstName: Doe
LastName: John
$p2

MAGIC METHODS

- PHP reserves all function names starting with __ as magical. It is recommended that you do not use function names with __ in PHP unless you want some documented magic functionality.

WHAT ARE MAGIC METHODS?

- Magic methods in php are **some predefined function by php compiler which executes on some event**.
- PHP functions that start with a double underscore – a "__" – are called magic functions (and/or methods) in PHP. E.g. __call, __get, __set.
- They are functions always **defined *inside* classes**, and are **not** stand-alone (outside of classes) functions.
- __construct is a magic method which will be automatically called on creating object of the classes.
- There are various magic methods in php.

class_exists()

is_a()

_call

Chaining