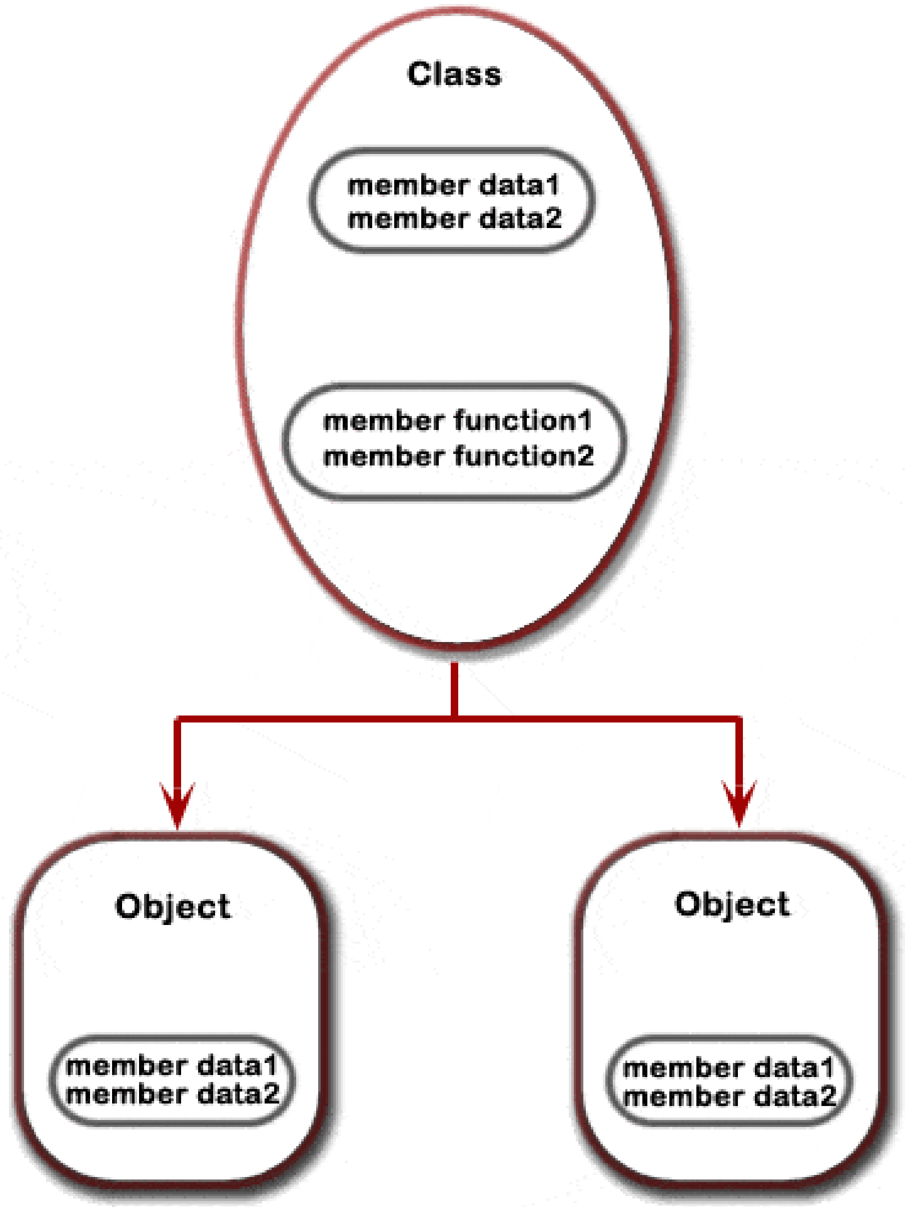
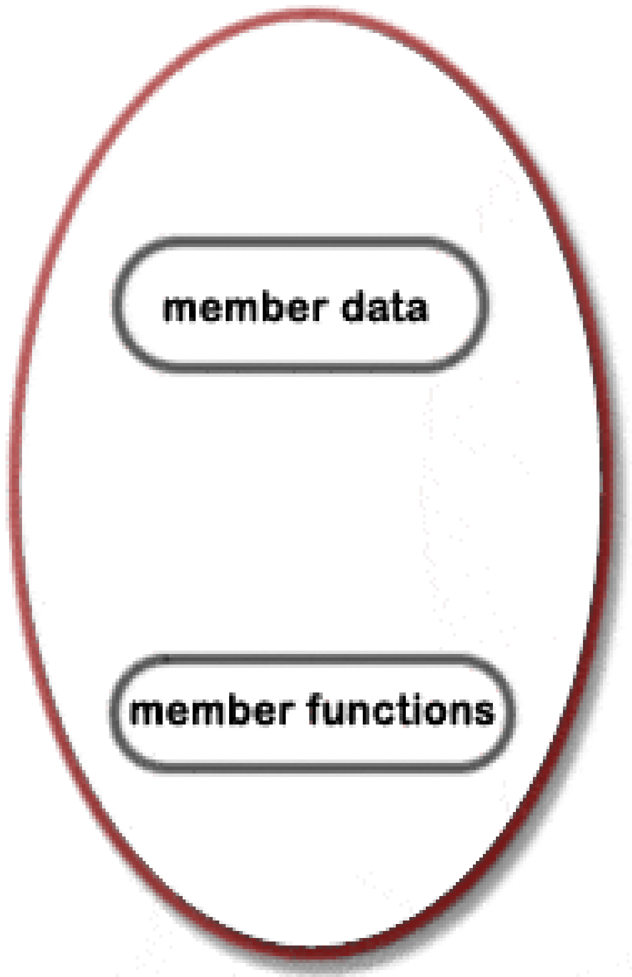
Unit 3

Object Oriented PHP

Class & Object

- Class:
  - In object-oriented programming, a class is a construct or prototype from which objects are created.
  - A class defines constituent members which enable class instances to have state and behavior.
  - Data field members enable a class object to maintain state and methods enable a class object's behavior.
- Object:
  - The fundamental idea behind an object-oriented language is to enclose a bundle of variables and functions into a single unit and keep both variables and functions safe from outside interference and misuse.
  - Such a unit is called object which acts on data. The mechanism that binds together data and functions are called encapsulation.
  - This feature makes it easy to reuse code in various projects. The functions declared in an object provides the way to access the data.
  - The functions of an object are called methods and all the methods of an object have access to variables called properties.


Class
member data1
member data2
member function1
member function2
Object
member data1
member data2
Object
member data1
member data2


member data
member functions

PHP: Creating classes and Instantiation

- The class definition starts with the keyword class followed by a class name, then followed by a set of curly braces ({}) which enclose constants, variables (called "properties"), and functions (called "methods") belonging to the class.
- A valid class name (excluding the reserved words) starts with a letter or underscore, followed by any number of letters, numbers, or underscores.
- Class names usually begin with an uppercase letter to distinguish them from other identifiers.
- 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.
- To create an object out of a class, the new keyword must be used.
- Classes should be defined prior to instantiation.

- Class member variables are called properties. Sometimes they are referred as attributes or fields.
- The properties hold specific data and related with the class in which it has been defined.
- Declaring a property in a class is an easy task, use one of the keyword public, protected, or private followed by a normal variable declaration.
  - public : The property can be accessed from outside the class, either by the script or from another class
  - private : No access is granted from outside the class, either by the script or from another class.
  - protected : No access is granted from outside the class except a class that’s a child of the class with the protected property or method.
- After an object is instantiated, you can access the property of a class using the object and -> operator. Any member declared with keyword "private" or "protected" cannot be accessed outside the method of the class.

```
<?php
class Myclass
{
 // Add property statements here
 // Add the methods here
}
$myobj = new MyClass;
?>
```

```
<?php
class Myclass
{
 public $font_size =10;
}
$f = new MyClass;
echo $f->font_size;
?>
```

There is a common mistake to use more than one dollar sign when accessing variables. In the above example there will be no $ sign before font_size (echo $f->font_size)

- Setting Methods:
  - The functions which are declared in a class are called methods.
  - A class method is exactly similar to PHP functions.
  - Declaring a method in a class is an easy task, use one of the keyword public, protected, or private followed by a method name.
  - public : The method can be accessed from outside the class.
  - private : No access is granted from outside the class.
  - protected : No access is granted from outside the class except a class that’s a child of the class with the protected property or method.
  - A valid method name starts with a letter or underscore, followed by any number of letters, numbers, or underscores.
  - The method body enclosed within a pair of braces which contains codes. The opening curly brace ({) indicates the beginning of the method code and the closing curly (}) brace indicates the termination of the method.
  - If the method is not defined by public, protected, or private then default is public.
  - Can access properties and methods of the current instance using $this (Format $this->property) for non static property.

```php
<?php
	class Myclass
	{
		public $font_size ="18px";
		public $font_color = "blue";
		public $string_name = "PHP OOPs Concepts";
		public function customize_print()
		{
			echo "<p style=font-size:".$this->font_size.";color:".$this->font_color.";>".
								$this->string_name."</p>";
		}
	}
	$f = new MyClass;
	echo $f->customize_print();
?>
```

```
class Myclass
{
        public $font_size ="18px";
        public $font_color = "blue";
        public $string_name = "w3resource";
        public function customize_print()
        {
              echo "<p style=font-size:".$this->font_size.";color:".$this->font_color.";>".$this->string_name."</p>";
        }
}
$f = new MyClass;
$f->font_size = "20px";
$f->font_color = "red";
$f->string_name = "Object Oriented Programming";
echo $f->customize_print();
```

PHP Constructor methods

- The constructor is a special built-in method, added with PHP 5, allows developers to declare for classes.
- Constructors allow to initializing object properties (i.e. the values of properties) when an object is created.
- Classes which have a constructor method execute automatically when an object is created.
- The 'construct' method starts with two underscores (__).
- 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.

```
function __construct([argument1, argument2, ..., argumentN])
{
/* Class initialization code */
}
```

```
class Myclass
{
        private $font_size;
        private $font_color;
        private $string_value;
        function __construct($font_size,$font_color,$string_value)
        {
                $this->font_size = $font_size;
                $this->font_color = $font_color;
                $this->string_value = $string_value;
        }
        function customize_print()
        {
                echo "<p style=font-size:".$this->font_size.";color:".$this->font_color.";>".$this->string_value."</p>";
        }
}
$f = new MyClass('20px','red','Object Oriented Programming');
echo $f->customize_print();
```

PHP Destructors methods

- The destructor is the counterpart of constructor.
- A destructor function is called when the object is destroyed
- 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
- The 'destructor' method starts with two underscores (__).

```
function __destruct
{
		//code here
}
```

PHP: Scope Resolution Operator (::)

- The double colon (::), is a token which allows access to static, constant, and overridden properties or methods of a class.

PHP: Class Constants

- A special entity that remains fixed on an individual class basis.
- Constant names are not preceded by a dollar sign ($) like a normal variable declaration.
- Interfaces may also include constants.
- When calling a class constant using the $classname :: constant syntax, the classname can actually be a variable.
- As of PHP 5.3, you can access a static class constant using a variable reference (Example: className :: $varConstant).

```
<?php
 class MyClass
 {
 const constant1 = 'PHP Class Constant';
 function PrintConstant()
 {
 echo  self::constant1 . "<br>";
 }
 }
 echo MyClass::constant1 . "<br>";
 $classname = "MyClass";
 echo $classname::constant1 . "<br>"; // As of PHP 
5.3.0
 $class = new MyClass();
 $class->PrintConstant();
 echo $class::constant1."<br>"; // As of PHP 5.3.0
?>
```

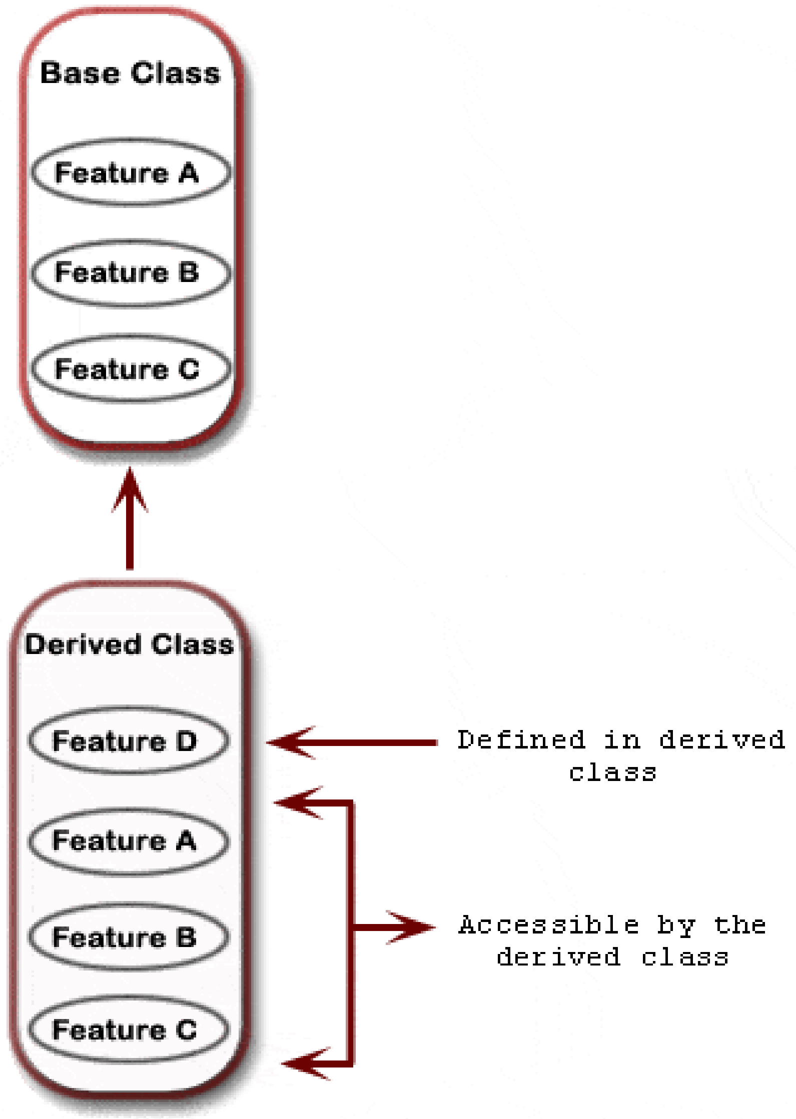
Understanding Access Specifiers

- Properties can be public, private or protected.
  - Public means that properties can be accessed everywhere,
  - private means properties can be accessed by the class that defines the member and
  - protected means properties can be accessed only within the class itself and by inherited and parent classes.
- Methods can be public, private or protected.
  - Public means that methods can be accessed everywhere,
  - private means methods can be accessed by the class that defines the member and
  - protected means methods can be accessed only within the class itself and by inherited and parent classes.

```
class Myclass
{
 public $font_size ="18px";
 private $font_color = "blue";
 protected $string_name = "w3resource";
 function property_print()
 {
 echo $this->font_size;
 echo $this->font_color;
 echo $this->string_name;
 }
}
$obj = new MyClass;
echo $obj->font_size; //Display 18px
echo $obj->font_color; //Fatal error: Cannot access private property Myclass::$font_color in F:\wamp\..
echo $obj->string_name; //Fatal error: Cannot access protected property Myclass::$string_name in F:\wamp\..
$obj->property_print(); //Display 18pxbluew3resource
```

PHP: Inheritance

- Inheritance is a well-established programming principle.
- Inheritance enables classes to form a hierarchy like a family tree.
- Allows subclasses to share the methods and properties (which are public or protected) of its superclass.
- Superclass is the parent class.
- A subclass can add properties and methods.
- Inheritance allows reusing code.



PHP: Interfaces

- Provides methods to implement.
- 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.
- When a class implements multiple interfaces there cannot be any naming collision between methods defined in the different interfaces.

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

PHP: Object Cloning

- Cloning is used to create a copy of an object.
- An object copy is created by using the clone keyword.
- When an object is cloned, PHP 5 will perform a shallow 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.
- Once the cloning is complete, if a __clone() method is defined, then the newly created object's __clone() method will be called, to allow any necessary properties that need to be changed.

Final keyword

- PHP 5 introduces 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.
-
```
<?php
class BaseClass {
   public function test() {
       echo "BaseClass::test() called\n";
   }
   final public function moreTesting() {
       echo "BaseClass::moreTesting() called\n";
   }
}
class ChildClass extends BaseClass {
   public function moreTesting() {
       echo "ChildClass::moreTesting() called\n";
   }
}
// Results in Fatal error: Cannot override final method BaseClass::moreTesting()
?>
```

-
  ```
  <?php
  final class BaseClass {
     public function test() {
         echo "BaseClass::test() called\n";
     }
     // Here it doesn't matter if you specify the function as final or not
     final public function moreTesting() {
         echo "BaseClass::moreTesting() called\n";
     }
  }
  class ChildClass extends BaseClass {
  }
  // Results in Fatal error: Class ChildClass may not inherit from final class (BaseClass)
  ?>
  ```
- **Note:** Unlike Java final keyword in PHP can only be used for methods and classes not for variables.We use the keyword 'const'.

Why do I have to use final?

1. Preventing massive inheritance chain of doom
2. Encouraging composition
3. Force the developer to think about user public API
4. Force the developer to shrink an object's public API
5. A final class can always be made extensible
6. extends breaks encapsulation
7. You don't need that flexibility
8. You are free to change the code

When to avoid final: Final classes only work effectively under following assumptions:

1. There is an abstraction (interface) that the final class implements
2. All of the public API of the final class is part of that interface

Serialization

- Most often we need to store a **complex array** in the database or in a file from PHP. Some of us might have surely searched for some built-in function to accomplish this task. Complex arrays are arrays with elements of more than one data-types or array.There are two popular methods of serializing variables.
  - **serialize()**
  - **unserialize()**
- We can serialize any data in PHP using the serialize() function. The serialize() function accepts a single parameter which is the data we want to serialize and returns a serialized string. Below program illustrate this:

```
<?php
$myvar = array( 'hello', 42, array(1, 'two'), 'apple'); // a complex array
$string = serialize($myvar);   // convert to a string
echo $string;
$newvar = unserialize($string); // unserializing the data in $string
 ?>
```

- Serialization is used to convert an object to a byte-stream representation and vice-versa.
- This is useful when we need to pass object data in the form of string of text between scripts and applications.
- There are various situations in which we need to pass objects in the form of string such as:
  - Converting object to string and storing it in a text file or a database table.
  - Converting and passing objects in URL query string.
  - Carrying objects between webpages in the session, etc.
- When the object is serialized, the content is placed with some type of a specifier followed by a colon, then followed with the actual data followed by a semi-colon.

- When we serialize an object it stores the class name and all its properties. It doesn’t store methods of the class. Hence to unserialize the serialized object the class should be present in the script before unserializing it.
- Note that static properties of class is not serialized.