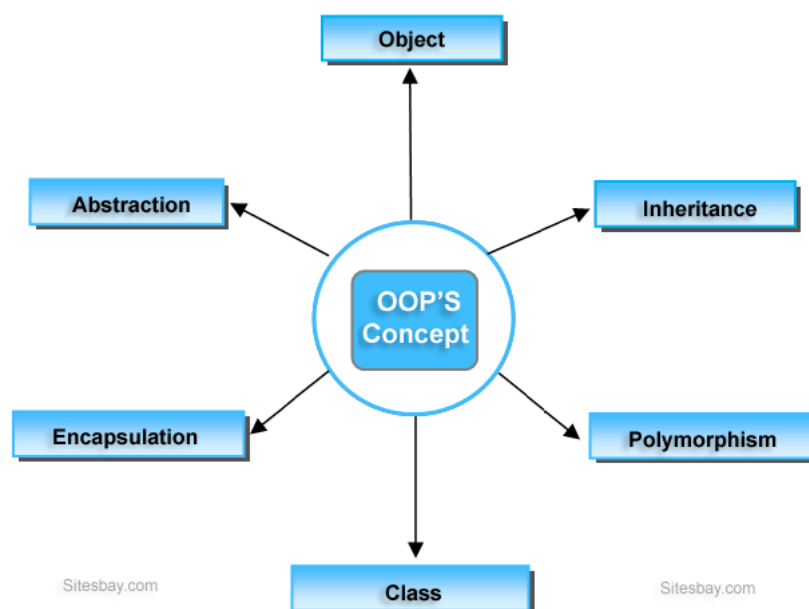


ណែនាំអោយស្គាល់ពី OOP ក្នុង PHP (Object Oriented Programming)

I. ដូចម្តេចទៅដែលហៅថា OO?

បើនិយាយទៅលើ Concept OO គឺច្បាស់ជាយើងផ្ដោតសំខាន់ទៅលើ Object Oriented ដែលសំដៅលើការបង្កើត Class និង Object ដើម្បីប្រើប្រាស់ក្នុង Project របស់យើង ។ Concept OO ត្រូវបានគេប្រើប្រាស់យ៉ាងសំបូរបែបក្នុងការកសាង System មួយ ដែលវាយលើគោលការណ៍សំខាន់ ៣គឺ ៖

- Encapsulation: សំដៅលើការប្រមូលផ្តុំនូវទិន្នន័យ(Data & Method) ជាក្នុងការគ្រប់ដោយការប្រើប្រាស់ Class ។
- Abstraction/Data hiding:
សំដៅលើការបិទបាំងនូវទិន្នន័យមួយចំនួនពីផ្នែក ខាងក្រៅនៃ Class ។



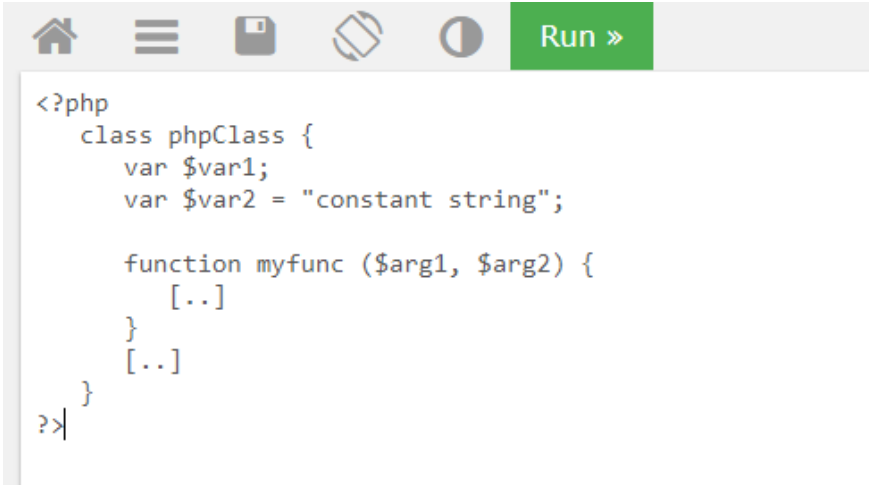


- Inheritance: សំដៅលើការបង្កើតនូវ Class ថ្មីមួយចេញពី Class ដែលមានស្រាប់ ពោលគឺការទទួលយកទិន្នន័យបន្តគ្នា ឬមរតបន្តគ្នា។
- Polymorphism : សំដៅលើទំរង់ច្រើននៃ Object ឬ Method របស់ Class។

1.1. Class: គឺជាការប្រមូលផ្តុំនូវ Data(Variable) និង Methods(Function) ដាក់ក្នុងការគ្រប់គ្រង ដោយការប្រើប្រាស់ដោយ Class។ នៅក្នុង Class អាចមាន Data member និង Method Member

- Data member: គឺជាប្រភេទ អញ្ញាតិដែលអាចផ្ទុកទិន្នន័យជា បណ្តុះ អាសន្ន។
- Function Member: គឺជាប្រភេទ function ដែលប្រកាសនៅក្នុង Class ហើយដែលវាសំរាប់ធ្វើអ្វីមួយ ដូចជា void input(), void Output(), void setID(),...។ function អាចជា return function និង Non Return function ជាដើម។

ទំរង់ទូទៅ ៖



```
<?php
class phpClass {
    var $var1;
    var $var2 = "constant string";

    function myfunc ($arg1, $arg2) {
        [..]
    }
    [..]
}
?>
```

ឧទាហរណ៍ ១ ៖

```
<?php
class Books {
    /* Member variables */
    var $price;
    var $title;

    /* Member functions */
    function setPrice($par){
        $this->price = $par;
    }

    function getPrice(){
        echo $this->price . "<br/>";
    }

    function setTitle($par){
        $this->title = $par;
    }

    function getTitle(){
        echo $this->title . " <br/>";
    }
}
?>
```

Object: គឺជារបស់ដែលកើតចេញពីClass ដែលមានលទ្ធភាព

អាច ប្រើប្រាស់ នូវ Data និង Method របស់ Class បាន។

```
$physics = new Books;
$maths = new Books;
$chemistry = new Books;

$physics->setTitle( "Physics for High School" );
$chemistry->setTitle( "Advanced Chemistry" );
$maths->setTitle( "Algebra" );

$physics->setPrice( 10 );
$chemistry->setPrice( 15 );
$maths->setPrice( 7 );

$physics->getTitle();
$chemistry->getTitle();
$maths->getTitle();
$physics->getPrice();
$chemistry->getPrice();
$maths->getPrice();
```



លទ្ធផលទទួលបាន ៖

```
Physics for High School
Advanced Chemistry
Algebra
10
15
7
```

ឧទាហរណ៍ ២ ៖

```
<?php
class Rectangle
{
    // Declare properties
    public $length = 0;
    public $width = 0;

    // Method to get the perimeter
    public function getPerimeter(){
        return (2 * ($this->length + $this->width));
    }

    // Method to get the area
    public function getArea(){
        return ($this->length * $this->width);
    }
}
?>
```

Save: Rectangle.php



```

<?php
// Include class definition
require "Rectangle.php";

// Create multiple objects from the Rectangle class
$obj1 = new Rectangle;
$obj2 = new Rectangle;

// Call the methods of both the objects
echo $obj1->getArea(); // Output: 0
echo $obj2->getArea(); // Output: 0

// Set $obj1 properties values
$obj1->length = 30;
$obj1->width = 20;

// Set $obj2 properties values
$obj2->length = 35;
$obj2->width = 50;

// Call the methods of both the objects again
echo $obj1->getArea(); // Output: 600
echo $obj2->getArea(); // Output: 1750
?>

```

ឧទាហរណ៍ ៣ ៖

```

<?php
class Students {
    /* Member variables */
    var $id;
    var $name;
    var $sex;
    var $score;
    /* Member functions */
    function setID($i){
        $this->id = $i;
    }
    function setName($n){
        $this->name = $n;
    }
    function setSex($s){
        $this->sex = $s;
    }
    function setScore($s){
        $this->score = $s;
    }
    function output(){
        echo "ID=" . $this->id . "<br/>";
        echo "Name=" . $this->name . "<br/>";
        echo "Gender=" . $this->sex . "<br/>";
        echo "Score=" . $this->score . "<br/>";
    }
}
?>

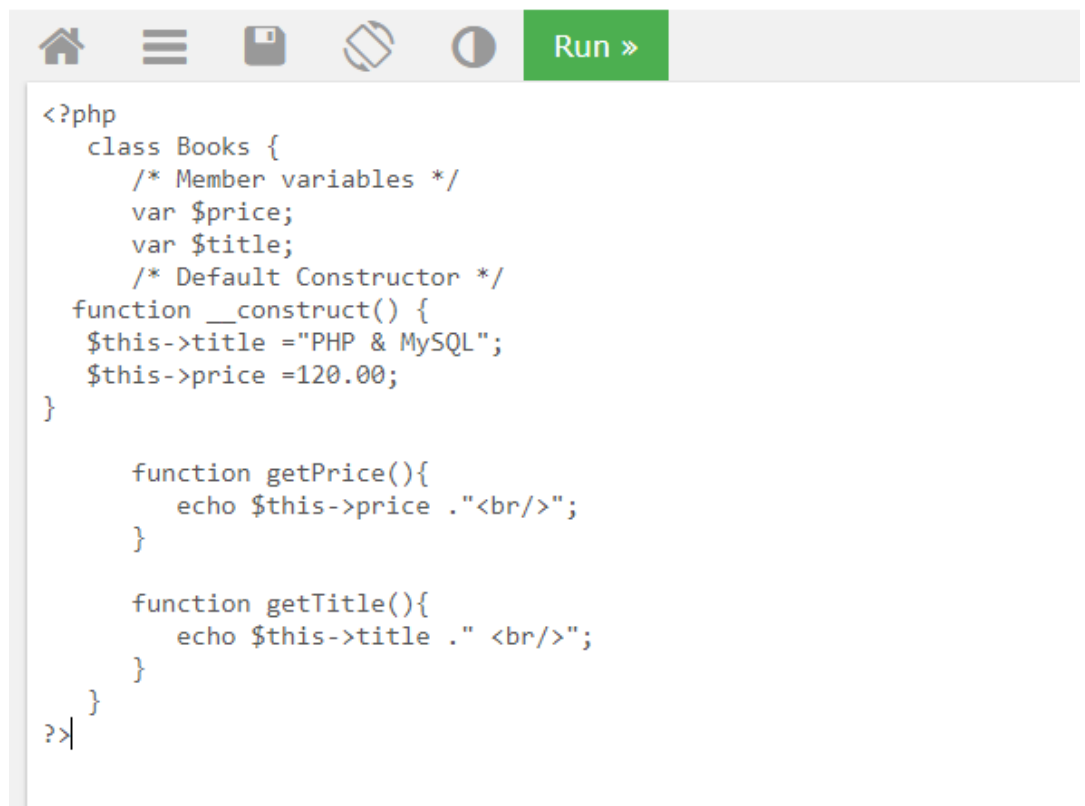
```

```
<?php
    $stu = new Students;
    $stu->setID("1001");
    $stu->setName("Sok Dara");
    $stu->setSex("Male");
    $stu->setScore(100);
    $stu->output();
?>
```

1.2. ការបង្កើតនូវ Constructor

Constructor គឺជាប្រភេទ Function ពិសេសមួយដែលដំណើរការដោយស្វ័យប្រវត្តិ ពេល គឺវាបានភ្ជាប់ជាមួយនិង object រួចតែម្តង លក្ខណៈ សំគាល់របស់ Constructor មាន ឈ្មោះដូច Class គ្មាន return type សូម្បីតែ void ។ Constructor ត្រូវបានគេបែងចែកជា ២ប្រភេទ គឺ៖

១) **Default Constructor:** គឺប្រភេទ Function ពិសេសមួយដែលគេប្រើប្រាស់នៅ ពេលដែល Object មួយកកើតឡើងវាទទួលបានតំលៃជាស្វ័យប្រវត្តិ។



```
<?php
class Books {
    /* Member variables */
    var $price;
    var $title;
    /* Default Constructor */
    function __construct() {
        $this->title = "PHP & MySQL";
        $this->price = 120.00;
    }

    function getPrice(){
        echo $this->price . "<br/>";
    }

    function getTitle(){
        echo $this->title . "<br/>";
    }
}
?>
```

```

<?php
$physics = new Books();
$maths = new Books ();
$chemistry = new Books ();

/* Get those set values */
$physics->getTitle();
$chemistry->getTitle();
$maths->getTitle();

$physics->getPrice();
$chemistry->getPrice();
$maths->getPrice();

?>

```

PHP & MySQL
 PHP & MySQL
 PHP & MySQL

120.00
 120.00
 120.00

លទ្ធផលទទួលបាន ៖

២) Constructor With Parameter

```

<?php
class Books {
    /* Member variables */
    var $price;
    var $title;

    /* Constructor */
    function __construct( $par1, $par2 ) {
        $this->title = $par1;
        $this->price = $par2;
    }

    /* Member functions */

    function getPrice(){
        echo $this->price . "<br/>";
    }

    function getTitle(){
        echo $this->title . " <br/>";
    }
}

$physics = new Books( "Physics for High School", 10 );
$maths = new Books ( "Advanced Chemistry", 15 );
$chemistry = new Books ("Algebra", 7 );

/* Get those set values */
$physics->getTitle();
$chemistry->getTitle();
$maths->getTitle();

$physics->getPrice();
$chemistry->getPrice();
$maths->getPrice();

?>

```



លទ្ធផលទទួលបាន ៖

Physics for High School
Advanced Chemistry
Algebra
10
15
7|

លំហាត់អនុវត្តន៍

ចូរធ្វើការបង្កើត Class មួយឈ្មោះ Employee ដែលមាន data member ដូចជា Id, name, sex, add និង salary ហើយនិងមាន Constructor ពីរគឺ Employee(), Employee(_____,_____,_____) និង Method set_() ,get_() និង Output() បន្ទាប់មកបង្ហាញទិន្នន័យចេញមកក្រៅវិញ។

Object Array

នៅក្នុងការបង្កើតនូវ Object Array នៃ Class វាសំដៅលើការផ្ទុកទិន្នន័យនៅ Object បានច្រើន។

```

<html>
<head>
<title>Online PHP Script Execution</title>
</head>
<body>
<?php
    class Car
    {
        public $color;
        public $type;
    }

    $myCar1 = new Car();
    $myCar1->color = 'red';
    $myCar1->type = 'Highlander';

    $myCar2 = new Car();
    $myCar2->color = 'Green';
    $myCar2->type = 'Lexus';

    $myCar3 = new Car();
    $myCar3->color = 'White';
    $myCar3->type = 'Toyota';

    $cars = array($myCar1, $myCar2,$myCar3);

    foreach ($cars as $car) {
        echo 'This car is a ' . $car->color . ' ' . $car->type . "\n";
    }
?>

```


ណែនាំអោយស្គាល់ Inheritance ក្នុង PHP

I. ដូចម្តេចទៅដែលហៅថា Inheritance?

Inheritance គឺជាដំណើរនៃការកើត Class ថ្មីមួយចេញពី Class ដែលមានស្រាប់ ដែល Class ថ្មី គេហៅថា Sub Class ឬ Derived Class និង Class មានស្រាប់គេហៅថា Base Class ឬ Super Class។

ទំរង់ទូទៅ៖

```
<?php
class Sub_Class extends Super_Class{
    ....
}
```

ឧទាហរណ៍ ១៖ ការ Accessing function member របស់ super class ទៅកាន់ Sub Class

```
1 // The parent class has its properties and methods
2 class Car {
3     //A private property or method can be used only by the parent.
4     private $model;
5
6     // Public methods and properties can be used by both the parent and the child classes.
7     public function setModel($model)
8     {
9         $this -> model = $model;
10    }
11
12    public function getModel()
13    {
14        return $this -> model;
15    }
16 }
17
```

```
18 //The child class can use the code it inherited from the parent class,
19 // and it can also have its own code
20 class SportsCar extends Car{
21
22     private $style = 'fast and furious';
23
24     public function driveItWithStyle()
25     {
26         return 'Drive a ' . $this -> getModel() . ' <i>' . $this -> style . '</i>';
27     }
28 }
29
30 //create an instance from the child class
31 $sportsCar1 = new SportsCar();
32
33 // Use a method that the child class inherited from the parent class
34 $sportsCar1 -> setModel('Ferrari');
35
36 // Use a method that was added to the child class
37 echo $sportsCar1 -> driveItWithStyle();
```

Result:

លទ្ធផលទទួលបាន ៖

Drive a Ferrari *fast and furious*.

ឧទាហរណ៍ ២ ៖ ការ Accessing data member របស់ super class ទៅកាន់ Sub Class ដោយប្រើប្រាស់ private Accessing.

```
1 // The parent class
2 class Car {
3     //The $model property is private, thus it can be accessed
4     // only from inside the class
5     private $model;
6
7     //Public setter method
8     public function setModel($model)
9     {
10         $this -> model = $model;
11     }
12 }
13
14
15 // The child class
16 class SportsCar extends Car{
17     //Tries to get a private property that belongs to the parent
18     public function hello()
19     {
20         return "beep! I am a <i>" . $this -> model . "</i><br />";
21     }
22 }
```

```
24 //Create an instance from the child class
25 $sportsCar1 = new SportsCar();
26
27 //Set the class model name
28 $sportsCar1 -> setModel('Mercedes Benz');
29
30 //Get the class model name
31 echo $sportsCar1 -> hello();
```

Result:

លទ្ធផលទទួលបាន ៖

Notice: Undefined property: SportsCar::\$model

ឧទាហរណ៍ ៣ ៖ ការ Accessing data member របស់ super class ទៅកាន់ Sub Class ដោយប្រើប្រាស់ protected Accessing.

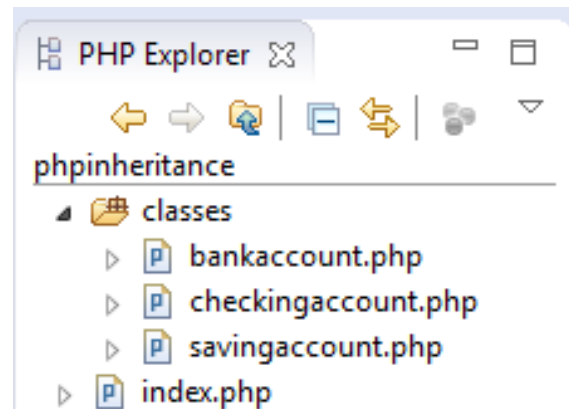
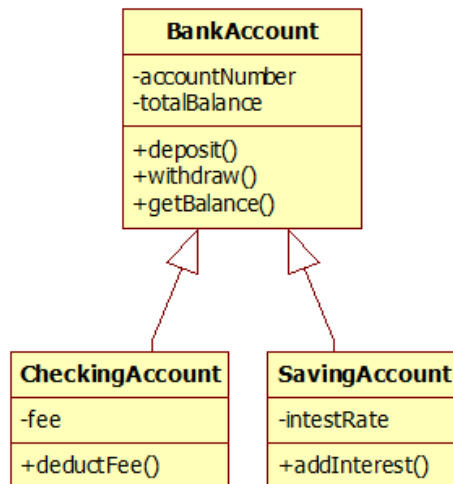
```
1 // The parent class
2 class Car {
3     //The $model property is now protected, so it can be accessed
4     // from within the class and its child classes
5     protected $model;
6
7     //Public setter method
8     public function setModel($model)
9     {
10         $this -> model = $model;
11     }
12 }
13
14 // The child class
15 class SportsCar extends Car {
16     //Has no problem to get a protected property that belongs to the parent
17     public function hello()
18     {
19         return "beep! I am a <i>" . $this -> model . "</i><br />";
20     }
21 }
22
23 //Create an instance from the child class
24 $sportsCar1 = new SportsCar();
25
26 //Set the class model name
27 $sportsCar1 -> setModel('Mercedes Benz');
28
29 //Get the class model name
30 echo $sportsCar1 -> hello();
```

លទ្ធផលទទួលបាន ៖

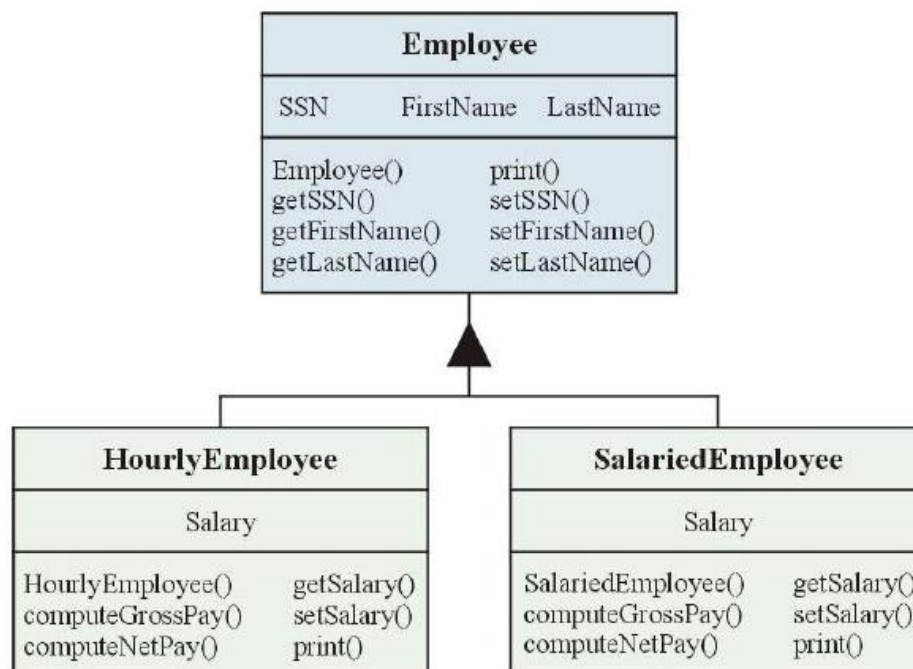
Result:

beep! I am a *Mercedes Benz*

លំហាត់អនុវត្តទី១ ចូរបង្កើតនូវ Class ទៅតាម Model ដូចខាងក្រោម ៖



លំហាត់អនុវត្តទី ២ ចូរបង្កើតនូវ Class ទៅតាម Model ដូចខាងក្រោម ៖



Overriding Methods: គឺសំដៅលើការផ្តល់លទ្ធភាពអោយ Sub Class អាច Overriding លើ Method ដែលមានស្រាប់នៅក្នុង Super Class ។ ការ Overriding លើ Method វាបានផ្តល់លទ្ធភាពអោយអោយយើងកែទិន្នន័យ ឬបន្ថែមទិន្នន័យលើ Method មានដែល មានក្នុង Super Class ។

ឧទាហរណ៍ ១ ៖

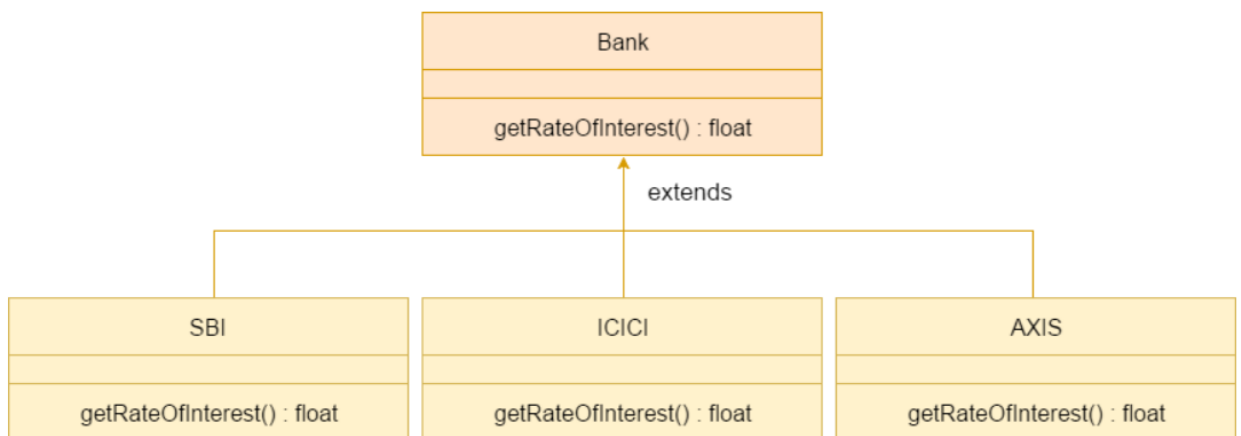
```
1 <?php
2 class Accounts {
3     /* Member variables */
4     protected $ename;
5     protected $esex;
6     protected $dob;
7     protected $address;
8     //Constructor
9     function __construct() {
10        $this->ename = "Sok San";
11        $this->esex = "Male";
12        $this->dob="01/02/1990";
13        $this->address="Phnom Penh";
14
15    }
16    // Methods for overloading
17    function view_Account()
18    {
19        echo "Employee Name=".$this->ename."<br/";
20        echo "Employee Sex=".$this->esex."<br/";
21        echo "Employee DOB=".$this->dob."<br/";
22        echo "Employee Address=".$this->address."<br/";
23    }
24 }
25 class User extends Accounts{
26     private $uname;
27     private $pass;
28     private $cpass;
29     private $utype;
30
31     function __construct() {
32        $this->uname = "sa";
33        $this->pass = "123";
34        $this->cpass = "123";
35        $this->utype = "Admin";
36
37    }
38 }
```

```

39 //Methods Overloading
40 function view_Account()
41 {
42     Accounts::view_Account();
43     echo "User Name=".$this->uname."<br/>";
44     echo "Password=".$this->pass."<br/>";
45     echo "Confirm Password=".$this->cpass."<br/>";
46     echo "User Type=".$this->utype."<br/>";
47 }
48 }
49
50
51
52 }
53
54 $u = new User();
55 $u->view_Account();
56
57 ?>

```

ឧទាហរណ៍ ២ ៖



```
1  <?php
2  //Create Super Class
3  class Bank{
4      function getRateOfInterest(){return 0;}
5  }
6  //Creating child classes.
7  class SBI extends Bank{
8      function getRateOfInterest(){return 8;}
9  }
10
11 class ICICI extends Bank{
12     function getRateOfInterest(){return 7;}
13 }
14 class AXIS extends Bank{
15     function getRateOfInterest(){return 9;}
16 }
17 //Test class to create objects and call the methods
18 $s=new SBI();
19 $i=new ICICI();
20 $a=new AXIS();
21 echo "SBI Rate of Interest: ".$s->getRateOfInterest()."\n";
22 echo "ICICI Rate of Interest: ".$i->getRateOfInterest()."\n";
23 echo "AXIS Rate of Interest: ".$a->getRateOfInterest()."\n";
24
25 ?>
```

ណែនាំអោយស្គាល់ Polymorphism ក្នុង PHP

ពាក្យថា Poly + morphism គឺសំដៅលើការប្រើប្រាស់ទំរង់ច្រើនទៅលើ Object Super Class និង លក្ខណៈរបស់ Methods បានច្រើនទំរង់ទាំងក្នុង Super Class និង Sub Class នៅក្នុងចំណុចនេះអ្នកនឹងសិក្សាលើ៖

1. Abstract Class

2. Interface

1. Abstract Class: គឺជាប្រភេទ Class ដែលបង្កើតចេញពី Keyword Abstract ដែល Class ប្រភេទនេះ មាននូវ Methods Abstract មួយយ៉ាងតិច។ Class ប្រភេទនេះអាចអោយ Sub Class extends ទៅ Override លើ Method Abstract បាន។

Abstract Method: គឺជាប្រភេទ Method ដែល ប្រកាសចេញពី Keyword Abstract ហើយគ្មានខ្លួន ។ ប្រភេទ Method បែបនេះ វាមានភាពងាយស្រួល ដល់ការ Overriding លើ Method ពី Sub Class។

ឧទាហរណ៍ ១ ៖

```
1 <?php
2 abstract class AbstractClass
3 {
4     // Force Extending class to define this method
5     abstract protected function getValue();
6     abstract protected function prefixValue($prefix);
7
8     // Common method
9     public function printOut() {
10         print $this->getValue() . "n";
11     }
12 }
```



```
14 class ConcreteClass1 extends AbstractClass
15 {
16     protected function getValue() {
17         return "ConcreteClass1";
18     }
19
20     public function prefixValue($prefix) {
21         return "{$prefix}ConcreteClass1";
22     }
23 }
24
25 class ConcreteClass2 extends AbstractClass
26 {
27     public function getValue() {
28         return "ConcreteClass2";
29     }
30
31     public function prefixValue($prefix) {
32         return "{$prefix}ConcreteClass2";
33     }
34 }
35
36 $class1 = new ConcreteClass1;
37 $class1->printOut();
38 echo $class1->prefixValue('Hello_') . "n";
39
40 $class2 = new ConcreteClass2;
41 $class2->printOut();
42 echo $class2->prefixValue('FOO_') . "n";
43 ?>
```

លទ្ធផលទទួលបាន៖

```
ConcreteClass1
FOO_ConcreteClass1
ConcreteClass2
FOO_ConcreteClass2
```

ឧទាហរណ៍ ២ ៖

```
1  <?php
2  abstract class AbstractClass
3  {
4      // Our abstract method only needs to define the required arguments
5      abstract protected function prefixName($name);
6
7  }
8
9  class ConcreteClass extends AbstractClass
10 {
11
12     // Our child class may define optional arguments not in the parent's signature
13     public function prefixName($name, $separator = ".") {
14         if ($name == "Pacman") {
15             $prefix = "Mr";
16         } elseif ($name == "Pacwoman") {
17             $prefix = "Mrs";
18         } else {
19             $prefix = "";
20         }
21         return "{$prefix}{$separator} {$name}";
22     }
23 }
24
25 $class = new ConcreteClass;
26 echo $class->prefixName("Pacman"), "\n";
27 echo $class->prefixName("Pacwoman"), "\n";
28 ?>
29
```

លទ្ធផលទទួលបាន៖

```
Mr. Pacman
Mrs. Pacwoman
```

2. **Interface:** មានលក្ខណៈស្រដៀងទៅនឹងការបង្កើតនូវ Abstract Class ដែរតែគ្រាន់តែវាមិនបង្កើតឡើងដោយការប្រើប្រាស់នូវ Keyword Class តែវាបង្កើតឡើងដោយ Keyword interface ដែលជាពិសេសវាអនុញ្ញាតិអោយបង្កើតនូវ Methods ដែលគ្មាននូវ body ។ Interface អាចអោយ Class ណាមួយផ្សេងទៀត ទៅ implement ទាញយក នូវ Methods ដែលគ្មាននូវ body មកប្រើប្រាស់បានតាមធម្មតា។ Interface មិនអនុញ្ញាតិអោយអ្នកអាចបង្កើតនូវ Object នោះទេ។

ឧទាហរណ៍ ១ ៖

```
1 interface Man
2 {
3     public function __construct($name, $age, $height);
4
5     public function giveFirmHandshakes();
6
7     public function beStubborn();
8
9     public function notPutToiletPaper();
10
11    public function isActive();
12 }
13 class AthleticMan implements Man
14 {
15     public $name;
16     public $age;
17     public $height;
18
19     public function __construct($name, $age, $height)
20     {
21         $this->name = $name;
22         $this->age = $age;
23         $this->height = $height;
24     }
25
26     public function giveFirmHandshakes()
27     {
28         return "I give firm handshakes.";
29     }
30
31     public function beStubborn()
32     {
33         return "I am stubborn.";
34     }
35 }
```



```

36 public function notPutToiletPaper()
37 {
38     return "It's not humanly possible to remember to put toilet
39 }
40
41 public function isActive()
42 {
43     return "I am a very active athlete.";
44 }
45 }
46 $jack = new AthleticMan('ETEC', '26', '5 feet 6 inches');
47 echo $jack->isActive();
48 echo $jack->notPutToiletPaper();
49 echo $jack->giveFirmHandshakes();
50 echo $jack->beStubborn();

```

លំហាត់អនុវត្តន៍

ចូរបង្កើតថ្នាក់ Class មួយឈ្មោះ Person(id,name, sex, dob) និង
 function ដូចជា View_Info() និង Construct with Parameter ចំនួន ១ ហើយបង្កើតថ្នាក់
 Sub Class ចំនួនពីរទៀតគឺ Employee(Salary) និង Students(Score) ហើយ
 Overriding Method ទាំងពីររបស់ Person បន្ទាប់មកបង្កើតថ្នាក់ Web form ចំនួន ៣
 ដើម្បី Test ទៅលើ Object នៃ Class ទាំង ៣ នេះ ?

Good Luck!