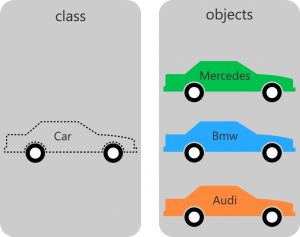
**PHP Object Oriented Programming**

# PHP OOP Basics

**Object Oriented Programming পরিচিতি:**

Object Oriented Programming বুঝার জন্য আমাদেরকে প্রথমে জানতে হবে Object কি? এবং class কি?

[](http://w3programmers.com/bangla/php-oop-basics/php-oop-class-and-object/)

**Object কি?**

আমরা আমাদের চার পাশে যা কিছু দেখি তার সবই Object বা বস্তু। এই ধরুন আপনি আপনার নিজেকে এবং আপনার আসে-পাশের দিকে খেয়াল করুন , আপনার গায়ের জামা, আপনার পরনের প্যান্ট, হাতের মোবাইল, পড়ার টেবিল,বসার চেয়ার ,কম্পিউটার এই সবই একেকটি Object বা বস্তু। এমনকি আপনি / আমি নিজেও একটা Object.

**class কি?**

Object সম্পর্কে তো জানা হলো , চলুন এবার জানি class কি? class হচ্ছে Object এর replica (অবিকল প্রতিরুপ) বা blueprint (নীলনকশা)। ধরুন আমরা একটি চেয়ার বানাতে চাই, শুরুতে আমরা কোন রকম চিন্তা ভাবনা না করে ফু দিয়ে একটা চেয়ার বানিয়ে ফেলতে পারি না। আমরা এর জন্যে পরিকল্পনা করি- চেয়ারটা দেখতে কেমন হবে, এটি দৈর্ঘ্য, প্রস্থ এবং উচ্চতা কতটুকু হবে, চেয়ারটি কি কি কাজ করবে ইত্যাদি। এই পরিকল্পনা গুলো আমরা আমরা কোথাও লিখে রাখি। আমাদের এই লেখা ডকুমেন্টটি আসলে class. আরেকটু ব্যাখ্যা করে বলতে গেলে আমরা বলতে পারি যে একটা অবজেক্ট কেমন হবে তা যেখানে নির্ধারণ করা থাকে তার নাম class। এই ক্লাস দিয়ে তৈরি অবজেক্টকে ঐ ক্লাসের অস্তিত্ব(Instance) বলে।

**Object Oriented Programming কি?**

এখন প্রশ্ন হল, তাহলে Object Oriented Programming কি? Computer Programming এ class এবং Object এর ধারণাকে কাজে লাগিয়ে যে Programming করা হয় তাকেই বলা হয় Object Oriented Programming। অবজেক্ট ওরিয়েন্ট কনসেপ্ট তিনটি ধারণার উপর প্রতিষ্ঠিত:

* **Inheritance**-ইংরেজি শব্দ inheritance অর্থ হচ্ছে উত্তরাধিকার। Object Oriented Programming এ এক class থেকে কিছু ফাংশনালিটি (Methods) ও বৈশিষ্ট্য (Property) অন্য class এ শেয়ার করার একটা পদ্ধতি হচ্ছে Inheritance. বা কোন Class বা Object এ অন্য কোন Class বা Object এর বৈশিষ্ট্য অর্জন করে থাকে যে প্রক্রিয়ায় তাকে উত্তরাধিকার সূত্র বা Inheritance বলে। আরো সহজ ভাবে বলা যায় , একটা ক্লাসকে (Parent Class) Inherit (অনুসরণ) করে তার কিছু বৈশিষ্ট্য আরেকটি উত্তরসূরি (child class) এর মধ্যে ব্যবহার করার প্রক্রিয়াকে Inheritance বলা হয়।
* **encapsulation**-অর্থাৎ সব কিছু encapsulate অবস্থায় থাকা। অনেকটা ক্যাপসুল মেডিসিন এর মতোই , যেভাবে একটা capsule এর বাইরে একটা আবরণ দিয়ে সব কিছু ভেতরে আটকানো থাকে । একই ভাবে Object Oriented Programming এ একটা class এর আবরণ দিয়ে সব কিছু ভেতরে আটকানো থাকে।
* **polymorphism** – বহুরূপিতা। অর্থাৎ একটা অবজেক্ট নানা সময় নানা রকম রূপ ধারণ করতে পারে বা বহুরূপে ব্যবহৃত হতে পারে।

**PHP তে Object Oriented Programming শেখার আগে কি কি বিষয় জানতে হবে?**

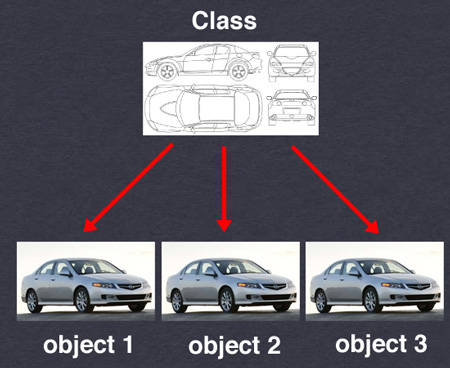
PHP তে Object Oriented Programming শেখার আগে যেসব বিষয় জানা আবশ্যক?

* **Parent Class:** একটি class থেকে যখন অন্য একটা class inherits বা উত্তরাধিকার সূত্রে তৈরী হয় , তখন যেই class থেকে নতুন class টি তৈরী হয়, তাকে Parent class বলা হয়। Parent class কে base class বা super class ও বলে।
* **Child Class:**একটি class যখন অন্য একটা class থেকে inherits বা উত্তরাধিকার সূত্রে তৈরী হয় , তখন তাকে child class বলা হয়। child class কে subclass বা derived class ও বলে।
* **Visibility/Access modifier:**  
  ইংরেজি শব্দ visibility অর্থ দৃশ্যমানতা, অর্থাৎ class এর মধ্যে যেকোনো property,constant অথবা Method এ কে access পাবে তা visibility দিয়ে নির্ধারিত হয়। Visibility গুলো সাধারণত class এর মধ্যে constant,property, method এর আগে ঘোষণা করতে হয়।

**PHP তে Visibility তিন ধরনের:**

* + **private :**Class এর মধ্যে constant, property, method গুলোকে যদি শুধু একই class বা class নিজের মধ্যে ব্যবহার সীমিত রাখতে চাই , তাহলে সেইসব constant, property, method গুলোকে private ঘোষণা করতে হয়।
  + **protected:** Class এর মধ্যে constant, property, method গুলোকে যদি শুধু একই class বা class নিজের মধ্যে এবং তার child class এর মধ্যে ব্যবহার সীমিত রাখতে চাই , তাহলে সেইসব constant, property, method গুলোকে protected ঘোষণা করতে হয়।
  + **public:** Class এর মধ্যে constant, property, method গুলোকে যদি একই class বা class নিজের মধ্যে , child class এবং class এর বাহির থেকে ব্যবহার রাখতে চাই , তাহলে সেইসব constant, property, method গুলোকে public ঘোষণা করতে হয়।
* **Property:**PHP তে class এর মধ্যে অবস্থিত variable গুলিকে Property বলা হয়। Property গুলোকে “attributes” বা “fields” নামেও ডাকা হয়।PHP তে Property গুলোকে যেকোনো একটা visibility অর্থাৎ : public, private, protected এর যেকোনো একটি visibility সহ ডিফাইন করতে হয়।
* **Method:**PHP তে class এর মধ্যে অবস্থিত function গুলিকে Method বলা হয়। PHP তে Method গুলোকে যেকোনো একটা visibility অর্থাৎ : public, private, protected এর যেকোনো একটি visibility সহ declare বা ঘোষণা করতে হয়।
* **static keyword:**class এর যেকোনো property এবং method এ কোন রকম instance বা object ছাড়া access করার সুযোগ দিতে চাইলে বা সরাসরি class দিয়ে access করার সুযোগ দিতে চাইলে static keyword ব্যবহৃত হয়।
* **“::” scope resolution operator** PHP তে class এর মধ্যে অবস্থিত যেকোনো static property, static method এবং constant কে class এর ভিতরে অথবা বাহিরে যেকোনো কাজে ব্যবহার করতে চাইলে “**:: scope resolution operator**” ব্যবহার করতে হয়।
* **“$this”** pseudo-variable: **“$this”** pseudo-variable টি মূলত current class এর object কে ধারণ করে। PHP তে class এর মধ্যে অবস্থিত যেকোনো Non-static property এবং Non-static method কে class এর মধ্যেই যেকোনো কাজে ব্যবহার করতে চাইলে “$this” pseudo-variable ব্যবহার করতে হয়।

২: Making and using of Class, Object and Class Members

[](http://w3programmers.com/bangla/making-and-using-of-class-object-and-class-members/class-object/)

**PHP তে class declare বা ঘোষণা করার নিয়ম কি?**

PHP তে “class” keyword এর পর letter অথবা underscore দিয়ে class name শুরু করতে হয়। class name এর পর curly braces দিয়ে class শুরু করতে হয়, এবং curly braces দিয়ে class শেষ করতে। curly braces এর ভিতরের অংশকে বলা হয় class body . class name হিসাবে PHP এর [reserved](http://php.net/manual/en/reserved.php)word দেওয়া যাবেনা। চলুন একটা উদাহরণ দেখা যাক :

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10 | <?php  class SimpleClass // class Declaration  { //class start      /\*..........      Class Body      .........\*/    } //class end  ?> |

**PHP তে class থেকে object তৈরির নিয়ম কি?**

PHP তে class থেকে অবজেক্ট তৈরী করতে হলে class এর বাহিরে class এর নামের পূর্বে new keyword দিতে হয়। অর্থাৎ new className অথবা new className(). আর object কে যেই variable এ রাখা হয় তাকে object handler বলা হয়। নিচের উদাহরণ দেখুন :

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12 | <?php  class SimpleClass  { //class start      /\*..........      Class Body      .........\*/    } //class end    $objHandler=new SimpleClass(); // Object Call  ?> |

**PHP তে class এর মধ্যে constant, property এবং method ঘোষণা করার নিয়ম কি?**

PHP তে class এর মধ্যে constant, property এবং method ঘোষণা করতে হলে constant, property এবং method গুলোর নামের পূর্বে আপনার প্রয়োজন অনুসারে যেকোনো একটা visibility দিতে হয়। তবে constant এর বেলায় visibility ঐচ্ছিক বা optional.constant এ কোন visibility ঘোষণা না করলে বাই ডিফল্ট public থাকে। নিচের উদাহরণ দেখুন :

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12 | <?php  class SimpleClass  {      public $var = 'a default value'; // public property declaration      private const a="America"; // private constant Declaration        // public method declaration      public function displayVar() {          echo $this->var;      }  }  ?> |

**PHP তে class এর মধ্যে অবস্থিত property, method এবং constant ব্যবহারের নিয়ম কি ?**

PHP তে class এর মধ্যে অবস্থিত property, method এবং constant গুলো দুইভাবে ব্যবহৃত হতে পারে।

১. class এর বাহিরে।  
২. class এর মধ্যে।

**property, method এবং constant কে class এর বাহিরে ব্যবহার**

PHP তে class এর মধ্যে অবস্থিত property, method এবং constant কে class এর বাহিরে দুইভাবে ব্যবহার করা যায়।

* **object handler বা object variable দিয়ে**: PHP তে class এর মধ্যে অবস্থিত যেকোনো public Non-static property এবং public Non-static method কে class এর বাহিরে যেকোনো কাজে ব্যবহার করতে চাইলে সরাসরি object handler বা object variable দিয়ে ব্যবহার করা যায়। নিচের উদাহরণ দেখুন :

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18 | <?php  class SimpleClass  {      // property declaration      public $var = 'Hello Simple Class';        // method declaration      public function displayVar() {          echo $this->var;      }  }  $obj=new SimpleClass;  $obj->displayVar();  /\*  access class method using Object Variable $obj   Result: Hello Simple Class  \*/  ?> |

* **“::” scope resolution operator** PHP তে class এর মধ্যে অবস্থিত যেকোনো public [static](http://w3programmers.com/bangla/php-oop-basics#static) property, public [static](http://w3programmers.com/bangla/php-oop-basics#static) method এবং public constant কে class এর বাহিরে যেকোনো কাজে ব্যবহার করতে চাইলে “**:: scope resolution operator**” ব্যবহার করতে হয়। নিচের উদাহরণ দেখুন :

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19 | <?php  class SimpleClass  {      // property declaration      public $var = 'Hello Simple Class';  public const a="America";        // Static method declaration      public static function sayHello() {          echo "Hello World";      }  }  SimpleClass::sayHello();  echo SimpleClass::a;  /\*  access class method using directly class name and scope resolution Operator ::   Result: Hello World  \*/  ?> |

**property, method এবং constant কে class এর মধ্যেই ব্যবহার**

PHP তে class এর মধ্যে অবস্থিত property, method এবং constant কে class এর মধ্যেই দুইভাবে ব্যবহার করা যায়।

* **“$this” pseudo-variable** ব্যবহার করে। নিচের উদাহরণ দেখুন :

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14 | <?php  class SimpleClass  {      // property declaration      public $var = 'Hello Simple Class';        //  method declaration      public function sayHello() {          echo $this->var; // use $this to show $var value      }  }  $obj=new SimpleClass;  $obj->sayHello(); // Result: Hello Simple Class  ?> |

* **“::” scope resolution operator** PHP তে class এর মধ্যে অবস্থিত যেকোনো [static](http://w3programmers.com/bangla/php-oop-basics#static) property, [static](http://w3programmers.com/bangla/php-oop-basics#static) method এবং constant কে class এর মধ্যেই যেকোনো কাজে ব্যবহার করতে চাইলে “**:: scope resolution operator**” ব্যবহার করতে হয়। নিচের উদাহরণ দেখুন :

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17 | <?php  class SimpleClass  {      // property declaration     static public $var = 'Hello Simple Class';     const a="America";        //  method declaration      public function sayHello() {          echo self::$var; // use self with scope resolution Operator          echo "<br>", self::a;      }  }  $obj=new SimpleClass;  $obj->sayHello();    ?> |

# ৩: Method Chaining

## PHP তে Method Chaining কি?

[](http://w3programmers.com/bangla/method-chaining/method-chaining-2/)

PHP তে যখন অনেক গুলো method কে একটি single instruction এর মধ্যে call করা হয়, PHP এর পরিভাষায় একে বলা হয় Method Chaining . আরেকটু সহজ ভাবে বুঝিয়ে বলি, ধরুন আপনার Person নামে একটা class আছে, যেখানে আপনার **$name** এবং **$age** নামে দুটি **private property** আছে। আর **$name** এবং **$age** property দুটিতে value set করার জন্য আছে যথাক্রমে **setName()** এবং **setAge()** নামক দুটি method. এবং সর্বশেষে **$name** এবং **$age** property দুটির value দেখানোর জন্য আছে **getInfo()** নামক method. এবং ধরুন আমাদের object টির নাম হচ্ছে **$person**. এখন আমরা যদি এই তিনটি method কে আলাদা আলাদা ভাবে object দিয়ে call না করে একটি single instruction এর মধ্যে ঠিক এইভাবে call করতে চাই : **$person->setName(“Masud Alam”)->setAge(33)->getInfo()**. PHP এর পরিভাষায় এই পদ্ধতিকে বলা হয় Method Chaining . Method Chaining এ class এর মধ্যের প্রত্যেকটি Method অর্থাৎ যেই class এর method সেই class এর object কে return করে। Method Chaining এর জন্য class এর মধ্যে যেকোনো value return এর পরিবর্তে **return $this;** কথাটি লিখতে হয়। চলুন পুরো ব্যাপারটা কয়েকটি উদাহরণের মাধ্যমে বুঝা যাক।

### উদাহরণ ১: সাধারণ একটা class এবং তার রেজাল্ট

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19  20  21  22  23  24  25 | <?php   class person   {      private $name="";      private $age="";        public function setName($name="")      {        $this->name=$name;      }      public function setAge($age="20")      {        $this->age=$age;      }      public function getInfo()      {        echo "Hello, My name is ".$this->name." and my age is ".$this->age." years.";      }  }    $person = new person();    $person->setName("Masud Alam");    $person->setAge("33");    $person->getInfo();  ?> |

#### Result

Hello, My name id Masud Alam and my age is 33 years.

পাঠক লক্ষ্য করুন আমাদের person নামক class এর **$name** এবং **$age** property দুটিতে value set করার জন্য আমরা **setName()** এবং **setAge()** method দুটিকে আলাদা আলাদা ভাবে call করি . এবং সর্বশেষে **$name** এবং **$age** property দুটির value দেখানোর জন্য আলাদা ভাবে **getInfo()** নামক method কে call করি।

## Method Chaining এর ব্যবহার :

উপরের class টিকে Method Chaining এ convert করতে হলে আপনাকে setName() Method এবং setAge() Method এর শেষে return $this; লাইনটি যুক্ত করে দিতে হবে। নিচের উদাহরণটি দেখুন :

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19  20  21  22  23  24  25  26 | <?php  class person  {    private $name="";    private $age="";      public function setName($name="")    {      $this->name=$name;      return $this;    }    public function setAge($age="20")    {      $this->age=$age;      return $this;    }      public function getInfo()    {      echo "Hello, My name is ".$this->name." and my age is ".$this->age." years.";    }  }    $person = new person();    $person->setName("Masud Alam")->setAge("33")->getInfo();   ?> |

#### Result

Hello, My name id Masud Alam and my age is 33 years.

**ব্যাখ্যা :** লক্ষ্য করুন আমরা setName() Method এবং setAge() Method দুটির মধ্যে $this pseudo-variable টি রিটার্ন করি। আর আমরা জানি “$this” pseudo-variable টি মূলত current class এর object কে ধারণ করে। প্রথমে আসা যাক $person->setName(‘Masud Alam’) এ . এটির কাজ হচ্ছে setName() Method কে call করবে এবং Name Property তে Masud Alam নামকে value হিসেবে সেট করবে। তারপর setName() মেথড $this রিটার্ন করবে , অর্থাৎ এটি Object person কে রিটার্ন করবে।

এরপরে আশা যাক ->setAge(’33’) তে , যেহেতু আমরা আগের setName() method থেকে chain object পেয়েছি , সুতরাং সেই person object দিয়েই আমরা setAge(33) কে call করা এবং age Property তে 33 age কে value হিসেবে সেট করবে। আবার এখানেও আমরা $this রিটার্ন করায়, একই ভাবে এখানেও মূলত আমাদের Object person কে chain রিটার্ন করবে।

সর্বশেষে ->getInfo() method দিয়ে আমরা পূর্বের ধারাবাহিকতায় chain রেজাল্ট পাই।

## Static Method Chaining

Static Method Chaining সাধারণ Method Chaining এর মতোই। তবে এখানে প্রত্যেকটি static Method এর মধ্যে **return $this** এর পরিবর্তে **return new static;** কথাটা লিখতে হবে। নিচের উদাহরণ দেখুন :

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19  20  21  22  23  24  25  26  27  28 | <?php  class oop{      public static $val;        public static function init($var){          static::$val=$var;          return new static;      }        public static function add($var){          static::$val+=$var;          return new static;      }        public static function sub($var){          static::$val-=$var;          return new static;      }        public static function out(){          return static::$val;      }    }    echo oop::init(10)->add(2)->sub(5)->out(); //Result: 10+2-5=7  echo "<br>";  echo oop::sub(2)->out(); //Result: 7-2=5; |

# ৪: Constructor Method and Destructor Methods

## PHP তে Constructor Method কি?

[](http://w3programmers.com/bangla/constructor-and-destructor-methods/constructor/)

Class থেকে নতুন কোনো object তৈরী হওয়ার সময় যে মেথড স্বয়ংক্রিয় ভাবে এবং সবার আগে execute হয়, PHP তে তাকে constructor Method বলে। PHP তে Constructor Method একটি PHP Language কর্তৃক Pre-defined Public Method অর্থাৎ PHP language কর্তৃক পূর্ব নির্ধারিত Public Method , যা public function এর পর একসাথে দুইবার underscore (\_\_) দেওয়ার পর construct কথাটি লিখতে হয় অর্থাৎ public function \_\_construct() নাম দিয়ে লিখতে হয়। নিচের উদাহরণ দেখুন :

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19 | <?php  class oop{   public function \_\_construct(){       echo "Welcome to PHP World!";   }   public function sayHello(){       echo "Hello World";   }      }  $obj=new oop; /\* Now Automatically Execute \_\_construct Method                    And Will show "Welcome to PHP World " First.\*/    echo "<br>";    $obj->sayHello(); // Result: Hello World    ?> |

#### Result:

Welcome to PHP World

Hello World

## Constructor Method এর Parameter এ কিভাবে Argument পাঠাবো?

যদি কোনো class এর Constructor Method এ Parameter দেওয়া থাকে, তাহলে সেই Constructor Method এ Argument পাঠাতে হলে class object এর সাথে parenthesis “( )” দিয়ে আর্গুমেনট পাঠাতে হবে। নিচের উদাহরণ দেখুন :

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19 | class Car {    private $model;      //\_\_construct    public function \_\_construct ($model)    {      $this -> model = $model;    }      public function getCarModel()    {      return ' The car model is: ' . $this -> model;    }  }    //We pass the value of the variable once we create the object  $car1 = new Car("Mercedes"); // Passing Argument to Constructor Parameter    echo $car1 -> getCarModel(); |

#### Result

The car model is: Mercedes.

## PHP তে Destructor Method কি?

[](http://w3programmers.com/bangla/class-constructor-and-destructor-methods/destructor/)

Object এর কাজ শেষ হওয়ার সময় class এর মধ্যে যে মেথড স্বয়ংক্রিয় ভাবে execute হয় বা সবার শেষে execute হয়, PHP তে তাকে Destructor Method বলে। PHP তে Destructor Method একটি PHP Language কর্তৃক Pre-defined Public Method অর্থাৎ PHP language কর্তৃক পূর্ব নির্ধারিত Public Method , যা public function এর পর একসাথে দুইবার underscore (\_\_) দেওয়ার পর destruct কথাটি লিখতে হয় অর্থাৎ public function \_\_destruct() নাম দিয়ে লিখতে হয়। নিচের উদাহরণ দেখুন :

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19  20  21 | <?php  class oop{   public function \_\_construct(){       echo "Welcome to PHP World!";   }   public function sayHello(){       echo "Hello World";   }     public function \_\_destruct(){       echo "<br>I'm about to disappear - bye bye!";   }    }  $obj=new oop;    echo "<br>";    $obj->sayHello();    ?> |

#### Result:

Welcome to PHP World!

Hello World

I'm about to disappear - bye bye!

## PHP তে Destructor Method কোনো argument receive করতে পারে ?

না পারেনা এবং সেটার দরকার ও নাই। তবে অন্য যেকোনো method দিয়ে Argument Receive করে যেকোনো property তে রেখে সেই property কে constructor Method দিয়ে রিটার্ন করতে পারবেন। নিচের উদাহরণ টি দেখুন :

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19  20  21  22 | <?php  class oop{  private $name;   public function \_\_construct($val){       return $this->name=$val;   }   public function sayHello(){       echo "Hello World";   }     public function \_\_destruct(){       echo "<br>Bye bye $this->name!";   }    }  $obj=new oop("Imran");    echo "<br>";    $obj->sayHello();    ?> |

#### Result

Hello World

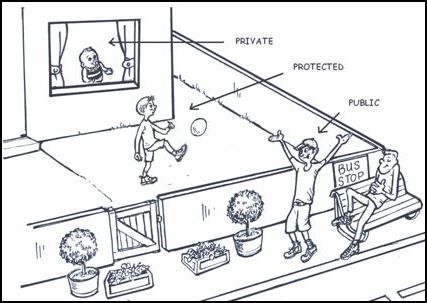
Bye Bye Imran!

## PHP তে Destructor Method কেন ব্যবহার করবো?

কোনো কিছু fininshing বা শেষ করতে চাইলে destructor Method ব্যবহার করা হয়, যেমন : file closing, Database Connection closing এবং Session Destroy এর মতো কাজগুলো Destructor Method দিয়ে করা হয়।

৫: Visibility or Access Modifier in PHP

**PHP তে Visibility বা Access modifier কি ?**

[](http://w3programmers.com/bangla/visibility-or-access-modifier-in-php/sfysv/)

ইংরেজি শব্দ Visibility অর্থ দৃশ্যমানতা, অর্থাৎ class এর মধ্যে যেকোনো property,constant অথবা Method এ কে access পাবে তা visibility দিয়ে নির্ধারিত হয়। Visibility গুলো সাধারণত class এর মধ্যে constant,property, method এর আগে ঘোষণা করতে হয়।

**PHP তে Visibility বা Access modifier কয়টি এবং কি কি ?**

PHP তে Visibility মোট ৩ টি :

* Public
* Protected
* Private

এবার আসুন প্রত্যেকটি Visibility সম্পর্কে উদাহরণসহ বুঝার চেষ্টা করি:

**public:**

Class এর মধ্যে constant, property, method গুলোকে যদি একই class বা class নিজের মধ্যে , child class এবং class এর বাহির থেকে ব্যবহার রাখতে চাই , তাহলে সেইসব constant, property, method গুলোকে public ঘোষণা করতে হয়। চলুন একটা উদাহরণের মাধ্যমে আরো ভালো ভাবে বুঝা যাক :

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19  20  21  22  23  24  25  26  27  28  29  30  31 | <?php  class GrandFather  {      public $name='Masud Alam';  // A public variable      public function getName(){        return $this->name;      }  }    class Daddy extends GrandFather // Inherited class  {      function displayGrandFaName()      {          return $this->name; // The public variable will be available to the inherited class      }    }    $grandFa=new GrandFather;    echo $grandFa->getName(),"\n";    // Inherited class Daddy wants to know Grandpas Name  $daddy = new Daddy;  echo $daddy->displayGrandFaName(),"\n"; // Prints 'Masud Alam'    // Public variables can also be accessed outside of the class!  $outsiderWantstoKnowGrandpasName = new GrandFather;  echo $outsiderWantstoKnowGrandpasName->name; // Prints 'Masud Alam    ?> |

**ব্যাখ্যা :** পাঠক লক্ষ্য করুন, আমাদের GrandFather class এর $name নামক Property টি public হওয়ায়, $name property টি আমরা একই সাথে current class, child class এবং class এর বাহির থেকেও access করতে পেরেছি।

**protected:**

Class এর মধ্যে constant, property, method গুলোকে যদি শুধু একই class বা class নিজের মধ্যে এবং তার child class এর মধ্যে ব্যবহার সীমিত রাখতে চাই , তাহলে সেইসব constant, property, method গুলোকে protected ঘোষণা করতে হয়। চলুন একটা উদাহরণের মাধ্যমে আরো ভালো ভাবে বুঝা যাক :

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19  20  21  22  23  24  25  26  27  28  29  30  31 | <?php  class GrandFather  {      protected $name='Masud Alam';  // A Protected variable      public function getName(){        return $this->name;      }  }    class Daddy extends GrandFather // Inherited class  {      function displayGrandFaName()      {          return $this->name; // The Protected Property will be available to the inherited class      }    }    $grandFa=new GrandFather;    echo $grandFa->getName(),"\n"; //Prints 'Masud Alam'    // Inherited class Daddy wants to know Grandpas Name  $daddy = new Daddy;  echo $daddy->displayGrandFaName(),"\n"; // Prints 'Masud Alam'    // Protected Property can not be accessed outside of the class!  $outsiderWantstoKnowGrandpasName = new GrandFather;  echo $outsiderWantstoKnowGrandpasName->name; // // Results in a Fatal Error    ?> |

**ব্যাখ্যা :** পাঠক লক্ষ্য করুন, আমাদের GrandFather class এর $name নামক Property টি protected হওয়ায়, $name property টি আমরা একই সাথে current class এবং child class থেকে access করতে পেরেছি কিন্তু current class অথবা child class এর বাহির থেকে access করতে পারবোনা, করলে আপনাকে fatal error দেখাবে।

**private :**

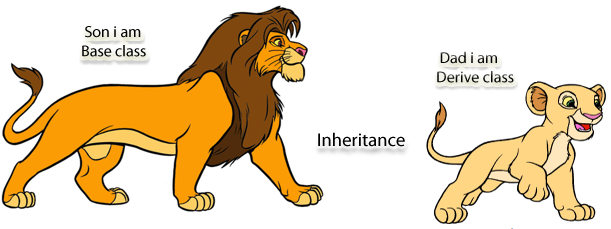
Class এর মধ্যে constant, property, method গুলোকে যদি শুধু একই class বা class নিজের মধ্যে ব্যবহার সীমিত রাখতে চাই , তাহলে সেইসব constant, property, method গুলোকে private ঘোষণা করতে হয়। চলুন একটা উদাহরণের মাধ্যমে আরো ভালো ভাবে বুঝা যাক :

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19  20  21  22  23  24  25  26  27  28  29  30  31  32 | <?php  class GrandFather  {      private $name='Masud Alam';  // A Protected variable      public function getName(){        return $this->name;      }  }    class Daddy extends GrandFather // Inherited class  {      function displayGrandFaName()      {          return $this->name; // The Private Property will not available to the inherited class and will show notice      }    }    $grandFa=new GrandFather;    echo $grandFa->getName(),"\n"; //Prints 'Masud Alam'  echo $gradFa->name, "\n"; //Results in a Notice    // Inherited class Daddy wants to know Grandpas Name  $daddy = new Daddy;  echo $daddy->displayGrandFaName(),"\n"; // Results in a Notice    // Protected Property can not be accessed outside of the class!  $outsiderWantstoKnowGrandpasName = new GrandFather;  echo $outsiderWantstoKnowGrandpasName->name; // // Results in a Fatal Error    ?> |

**ব্যাখ্যা :** পাঠক লক্ষ্য করুন, আমাদের GrandFather class এর $name নামক Property টি private হওয়ায়, $name property টি আমরা শুধু মাত্র current class এর ভিতরে ব্যবহার করতে পেরেছি, কিন্তু current class এর বাহির থেকে access বা ব্যবহার করতে পারবোনা, করলে আপনাকে error দেখাবে।

# ৬: PHP OOP Inheritance

## Inheritance কি?

[](http://w3programmers.com/bangla/inheritance/inheritance-in-php-oop/)

ইংরেজি শব্দ inheritance অর্থ হচ্ছে উত্তরাধিকার। Object Oriented Programming এ এক class থেকে কিছু ফাংশনালিটি (Methods) ও বৈশিষ্ট্য (Property) অন্য class এ শেয়ার করার একটা পদ্ধতি হচ্ছে Inheritance. বা কোন Class বা Object এ অন্য কোন Class বা Object এর বৈশিষ্ট্য অর্জন করে থাকে যে প্রক্রিয়ায় তাকে উত্তরাধিকার সূত্র বা Inheritance বলে। আরো সহজ ভাবে বলা যায় , একটা ক্লাসকে (Parent Class) Inherit (অনুসরণ) করে তার কিছু বৈশিষ্ট্য আরেকটি উত্তরসূরি (child class) এর মধ্যে ব্যবহার করার প্রক্রিয়াকে Inheritance বলা হয়।

## PHP তে কিভাবে child class তৈরী হয় ?

child class তৈরী করতে হলে প্রথমে class কীওয়ার্ড এর পর child class এর নাম দিতে হয়, তারপর extends কীওয়ার্ড লিখে Parent Class এর নাম লিখতে হয়। যেমন:

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9 | <?php  class ParentClass {    // properties and methods here  }    class ChildClass extends ParentClass {    // additional properties and methods here  }  ?> |

## PHP তে Inheritance কিভাবে কাজ করে?

মূলত Parent Class এবং Child Class ধারণার উপর ভিত্তি করে Inheritance ব্যাপারটি আসে। আপনি যখন একটা child class তৈরী করবেন , সেটি তার parent class এর সব protected এবং public Property, Method, Constant গুলো inherit করবে। আরো সহজ ভাবে বলা যায় , একটি class যখন আরেকটি class কে extend করে তখন Parent Class এর সব Protected এবং public Property, Method এবং constant, child class এ define না করেও ব্যবহার বা access করা যায় । পাশাপাশি child class আরো নতুন নতুন property এবং method অন্তর্ভুক্ত করতে পারবে। চলুন একটা Web Forum Application এর উদাহরণ দিয়ে আরো ভালো ভাবে বুঝা যাক:

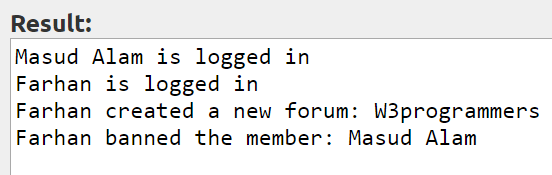
|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15 | <?php  class Member {    public $username = "";    private $loggedIn = false;    public function login() {      $this->loggedIn = true;    }    public function logout() {      $this->loggedIn = false;    }    public function isLoggedIn() {      return $this->loggedIn;    }  }  ?> |

**ব্যাখ্যা:**উপরের Member Class টি একটি Web Forum Application এর জন্য করা একটা Class। যেখানে ফোরাম মেম্বারদের জন্য login(), logout(), isLoggedIn() method তৈরী করা হয়েছে । যেহেতু ফোরাম অ্যাডমিনিস্ট্রেটররাও Member, এখন আপনি Member Class থেকে একটি নতুন Child class তৈরী করতে পারেন, যেটার নাম হবে Administrator Class . আর তাতে সুবিধা হচ্ছে Administrator class , Member class এর সব protected এবং public property, Method এবং constant গুলোকে Inherit বা উত্তরাধিকার সূত্রে পাবে। চলুন এবার Member Class থেকে child class হিসেবে Administrator Class টি তৈরী করা যাক:

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10 | <?php  class Administrator extends Member {    public function createForum( $forumName ) {      echo "$this->username created a new forum: $forumName<br>";    }    public function banMember( $member ) {      echo "$this->username banned the member: $member->username<br>";    }  }  ?> |

**ব্যাখ্যা:**লক্ষ্য করুন Administrator class এ আমাদের অনেক কাজ নতুন করে করা লাগবেনা , Administrator Class তার Member Parent Class দিয়ে login, logout এবং user logged In কিনা সেগুলো inherit করে নিজে ব্যবহার করতে পারবে। এখন আপনি administrator স্পেসিফিক Method গুলো যেমন : createForum(), deleteForum() এবং banMember() আপনার administrator class এ যোগ করতে পারবেন। আর এতে সুবিধা হচ্ছে আপনি administrator class এ একই সাথে সাধারণ Member Class এর সুবিধা পাচ্ছেন, পাশাপাশি Administrative সুবিধা গুলোও পাচ্ছেন। আর এটাই inheritance. চলুন এবার Parent এবং Child Class দুটির উদাহরণ একসাথে দেখা যাক

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19  20  21  22  23  24  25  26  27  28  29  30  31  32  33  34  35  36  37  38  39  40 | <?php  class Member {      public $username = "";      private $loggedIn = false;      public function login() {          $this->loggedIn = true;      }      public function logout() {          $this->loggedIn = false;      }      public function isLoggedIn() {          return $this->loggedIn;      }  }    class Administrator extends Member {      public function createForum( $forumName ) {          echo "$this->username created a new forum: $forumName<br>";      }      public function banMember( $member ) {          echo "$this->username banned the member: $member->username<br>";      }  }    // Create a new member and log them in  $member = new Member();  $member->username = "Masud Alam";  $member->login();  echo $member->username . " is " . ( $member->isLoggedIn() ? "logged in" : "logged out" ) . "<br>";    // Create a new administrator and log them in  $admin = new Administrator();  $admin->username = "Farhan";  $admin->login();  echo $admin->username . " is " . ( $member->isLoggedIn() ? "logged in" : "logged out" ) . "<br>";  // Displays "Mary created a new forum: W3programmers"  $admin->createForum( "W3programmers" );  // Displays "Farhan banned the member: Masud Alam"  $admin->banMember( $member );  ?> |

[](http://w3programmers.com/bangla/inheritance/php-inheritance-example/)

**ব্যাখ্যা:**লক্ষ্য করুন, লাইন নম্বর ৩৩,৩৪,৩৫ এ Administrator Object handler দিয়ে যেই সব property এবং Method গুলো ব্যবহার করেছি সেগুলোর কোনোটিই Administrator Class এর Property এবং Method না। সবগুলো Member Class থেকে child হিসেবে Inherit বা উত্তরাধিকার হিসেবে পাওয়া। আর এই জন্যই এগুলো Administrator class এর object দিয়ে ব্যবহার করতে পেরেছি।

**Inheritance ব্যবহার করে আরেকটি উদাহরণ :**

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19  20  21  22  23  24  25  26  27  28  29  30  31  32  33  34  35  36  37  38  39  40  41  42  43  44  45  46  47  48  49  50  51  52  53  54  55  56 | <?php  /\*\*  Check User Exist or Not  \*/  class Member  {      protected $isLoggedIn=false;      protected $role;      protected $users=[          ["username"=>"abc","password"=>"xyz","role"=>"subscriber"],          ["username"=>"acb","password"=>"yxz","role"=>"admin"],          ["username"=>"bca","password"=>"zxy","role"=>"super\_admin"]      ];      public function Login($user,$pass){              foreach($this->users as $user\_info){              extract($user\_info);                if($username==$user && $password==$pass){                  $this->isLoggedIn=true;                  $this->role=$role;              }          }      }  }    /\*\*  \* Child Class Check User Role  \*\*/  class Role extends Member  {        function \_\_construct($user,$pass)      {          $this->Login($user,$pass);              if($this->isLoggedIn){                  if($this->role=='subscriber'){                      echo "You're Logged in as a Subscriber";                  }                  if($this->role=='admin'){                      echo "You're Logged in as a Administrator";                  }                    if($this->role=='super\_admin'){                      echo "You're Logged in as a Super Administrator";                  }              }              else{                      echo "Invalid Username/Password";                  }        }  }    $login=new Role("acb","yxz"); |

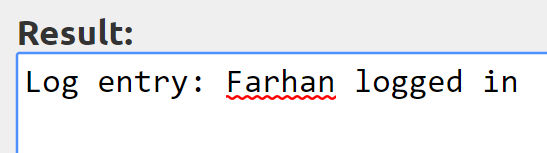
## Method Overriding কি?

সাধারণত যখন একটা child class তৈরী হয়, তখন সেটি তার parent class এর সব Protected Property এবং Method গুলোকে Inherit করে নিয়ে আসে। কিন্তু কখনো কখনো আমাদেরকে Parent Class এর যেকোনো Method কে Child Class এ নতুন করে একই নামে লেখার দরকার হয়, এটা সাধারণত তখনই দরকার হয় , যখন দেখা যায়, Parent Class এর Method টি child class এর সব পারপাস পূরণ করতে পারছেনা। অর্থাৎ, আরো কিছু feature add করার দরকার হয়। Object Oriented Programming এর পরিভাষায় এটাকে বলা হয় Method Overriding. এক কোথায় , Parent Class এর কোনো Method কে Child Class এ একই নামে এবং সমান সংখ্যক parameter দিয়ে পুনরায় declare করলে Object Oriented Programming এর পরিভাষায় বলা হয় Method Overriding. নিচের উদাহরণটি দেখুন :

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11 | class ParentClass {    public function myMethod($x) {      // (method code here)    }  }  class ChildClass extends ParentClass {    public function myMethod($x) {      // For ChildClass objects, this method is called      // instead of the parent class's MyMethod()    }  } |

এবার একটা practical উদাহরণ দেওয়া যাক : দরুন আমাদের Member class টির login Method টি শুধু Login কাজটি করে। এখন যদি Administrator Class চায় , কখন কে লগইন করলো তার একটা log Entry রাখা দরকার। আর এর জন্য আমরা Method Overriding Feature টি ব্যবহার করবো। চলুন দেখা যাক :

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19  20  21  22  23  24  25  26  27 | <?php  class Member {    public $username = "";    private $loggedIn = false;    public function login() {      $this->loggedIn = true;    }    public function logout() {      $this->loggedIn = false;    }  }  class Administrator extends Member {    public function login() {      $this->loggedIn = true;      echo "Log entry: $this->username logged in<br>";    }  }  // Create a new member and log them in  $member = new Member();  $member->username = "Masud Alam";  $member->login();  $member->logout();  // Create a new administrator and log them in  $admin = new Administrator();  $admin->username = "Farhan";  $admin->login();           // Displays "Log entry: Farhan logged in"  $admin->logout(); |

[](http://w3programmers.com/bangla/inheritance/php-method-overriding-example/)

## Parent Class এর কোনো Method কে Child Class এর Method এর মধ্যে ব্যবহারের নিয়ম কি?

Parent Class এর কোনো Method কে child Class ব্যবহার করতে চাইলে Parent Keyword এবং scope resolution operator **::** ব্যবহার করতে হয় । ঠিক এই রকম :

|  |  |
| --- | --- |
| 1 | parent::parentMethod(); |

এবার একটা practical উদাহরণ দেওয়া যাক : লক্ষ্য করবেন , উপরের Method Overriding উদাহরণে আমরা child class এ login Method এ Parent Class এর login এর সব কোডই ব্যবহার করেছি। কিন্তু আমরা যদি parent কীওয়ার্ড টি ব্যবহার করতাম, তাহলে আমাদের কষ্ট করে Parent Class এর login এর সব কোড আবার লেখা প্রয়োজন হতো না। নিচের উদাহরণ দেখুন।

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19  20  21  22  23  24  25  26  27 | <?php  class Member {    public $username = "";    private $loggedIn = false;    public function login() {      $this->loggedIn = true;    }    public function logout() {      $this->loggedIn = false;    }  }  class Administrator extends Member {    public function login() {      parent::login(); // Using parent keyword      echo "Log entry: $this->username logged in<br>";    }  }  // Create a new member and log them in  $member = new Member();  $member->username = "Masud Alam";  $member->login();  $member->logout();  // Create a new administrator and log them in  $admin = new Administrator();  $admin->username = "Farhan";  $admin->login();           // Displays "Log entry: Farhan logged in"  $admin->logout(); |

**ব্যাখ্যা :** খেয়াল করুন ১৪ নম্বর লাইনে আমরা parent class এর হুবহু code copy না করে parent keyword ব্যবহার করেছি।

## Inheritance বন্ধ (prevent) করার উপায় কি?

[](http://w3programmers.com/bangla/inheritance/preventing-inheritance-in-php/)

Class Inheritance বন্ধ বা Prevent করা মানে হচ্ছে ঐ class থেকে child class তৈরী এবং Method Overriding বন্ধ করা। অনেক সময় Inheritance এবং Method Overriding এর কারণে class এর security এর সমস্যা হতে পারে, আবার program complex আকার ধারণ করতে পারে। আর সে জন্য Class এবং Method এর security এর জন্য child class তৈরী এবং Method Overriding বন্ধ রাখতে হয়, আর child class তৈরী এবং Method Overriding বন্ধ করার জন্য PHP তে class এর নামের এবং Method এর নামের আগে final keyword ব্যবহার করতে হয়। চলুন প্রথমে দেখা যাক Final keyword দিয়ে কিভাবে Method Overriding বন্ধ করা যায়:

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19  20  21  22 | <?php  class Member {    public $username = "";    private $loggedIn = false;    public final function login() {      $this->loggedIn = true;    }    public function logout() {      $this->loggedIn = false;    }    public function isLoggedIn() {      return $this->loggedIn;    }  }    class NaughtyMember extends Member {    public function login() {      $this->loggedIn = true;      // Do something bad    }  }  ?> |

#### Result:

**Fatal error: Cannot override final method Member::login()**

এবার চলুন দেখা যাক Final keyword দিয়ে কিভাবে class Inheritance বন্ধ করা যায়:

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8 | <?php  final class Member {    // This class can't be extended at all  }  class Administrator extends Member{  // Will show Fatal Error  }  ?> |

#### Result:

**Fatal error: Class Administrator may not inherit from final class (Member) in on line 7**

৭: Static Method, Properties and Late Static Binding

**Static Method এবং Property কি ?**

[](http://w3programmers.com/bangla/static-keyword-and-late-static-binding/static-method-and-properties-in-php-3/)

Object ছাড়া সরাসরি class এর Method এবং Property গুলোকে access করার জন্য সেই method এবং property গুলোকে কে static ঘোষণা করতে হয়। আর static ঘোষণা করার জন্য Method এবং Property নামের পূর্বে static কীওয়ার্ড ব্যবহার করতে হবে। static keyword টি কে access modifier ও বলা যায়। চলুন একটা উদাহরণ দিয়ে দেখা যাক:

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15 | <?php  class testStatic{    // static methods and properties are defined with the static keyword.    //You can add static keyword before or after visibility (Public, Private, Protected)    public static $count = 0;    static public $counter=1;      public static function counter(){          }    static public function anotherCounter(){          }  }  ?> |

**Static Method এবং Property গুলো কিভাবে access বা ব্যবহার করব?**

Static Method এবং Property গুলোর ব্যবহার দুইভাবে ভাগ করা যায়।  
১. class এর বাহিরে ব্যবহার  
২. class এর ভিতরে ব্যবহার

**Static Method এবং Property গুলোর class এর বাহিরে ব্যবহার :**

Static Method এবং Property গুলোকে class এর বাহির থেকে Object ছাড়া ব্যবহার করতে চাইলে প্রথমে class এর নামের সাথে (**::**) scope resolution operator দিতে হয় , তারপর static Method অথবা Property এর নাম লিখতে হয় । আরো ভালো ভাবে বুঝার জন্য নিম্নের উদাহরণটি দেখুন :

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18 | <?php  class testStatic{      public static $count = 0;        public static function counter(){        echo "Its Static Counter";    }    }    //Accessing Static Property and Method Outside of the Class  testStatic::$count=5; // Assigning Value to static Property  echo testStatic::$count; // Calling Static Property with Updated Value  echo "<br>";  testStatic::counter(); // Calling Static Method with Updated Value  ?> |

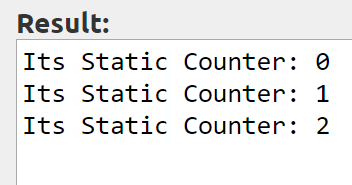
**Note:** scope resolution operator এর পর property নাম এর সাথে অবশ্যই **$** (dollar sign) দিতে হবে।

**Static Method এবং Property গুলোর class এর ভিতরে ব্যবহার :**

Static Method এবং Property গুলোকে class এর ভিতরে $this pseudo-variable ছাড়া ব্যবহার করতে চাইলে প্রথমে class এর নাম অথবা self keyword এর সাথে (**::**) scope resolution operator দিতে হয় , তারপর static Method অথবা Property এর নাম দিতে হয় । আরো ভালো ভাবে বুঝার জন্য নিম্নের উদাহরণটি দেখুন :

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17 | <?php  class testStatic{      public static $count = 0;        public static function counter(){        echo "Its Static Counter: ";        echo self::$count++,"<br>"; // accessing static Property inside class    }    }  // Calling Static Method Outside Class  testStatic::counter();  testStatic::counter();  testStatic::counter();  ?> |

**Result:**

[](http://w3programmers.com/bangla/static-keyword-and-late-static-binding/static-method-example-in-php/)

**Late Static Binding কি ?**

PHP তে Static Method এবং Property গুলোকে class এর ভিতরে কল করার জন্য $this pseudo-variable এর পরিবর্তে class এর নাম অথবা “self” keyword এর সাথে (**::**) scope resolution operator দিয়ে ব্যবহার করা হয় , তখন যেই class এ “self” keyword ব্যবহার করা হয়েছে, “self” keyword শুধু ঐ ক্লাসকেই নির্দিষ্ট করে নেয়। আর তাতে Parent Class এর static Method এবং Property গুলোকে child class এ inherit হয়না। আর সেই জন্য PHP 5.3 থেকে Parent Class এর Static Method এবং Property গুলোকে কে child class এ inherit করার জন্য static Method এবং Property কল করার সময় “self” keyword এর পরিবর্তে “static” keyword ব্যবহার প্রয়োগ করা হয়েছে, আর এটাকেই PHP 5.3 থেকে Late Static Binding বলা হয়। আরো ভালো ভাবে বুঝার জন্য নিম্নের উদাহরণ দুটি দেখুন :

**উদাহরণ: “self” keyword এর ব্যবহার**

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13 | <?php  class Course{      protected static $courseName = 'Professional PHP';      public static function getCourseName(){          return self::$courseName;      }  }    class Student extends Course{      protected static $courseName = 'Laravel';  }    echo Student::getCourseName(); // Professional PHP, not Laravel |

**ব্যাখ্যা:**

* প্রথমত, আমরা Course নামক একটি Class তৈরী করি যার মধ্যে আছে $courseName নামে একটি static property যার Value হচ্ছে “PHP Professional” এবং getCourseName নামে একটি static Method তৈরী করি যা $courseName Property এর value return করবে। লক্ষ্য করবেন, যে আমরা Course Class এর ভিতরে static property তে access করার জন্য self এবং “::” scope resolution Operator ব্যবহার করেছি।
* দ্বিতীয়ত , আমরা Student নামে Course class থেকে একটি child class তৈরী করি, এবং Student class এর মধ্যেও $courseName নামে একটি static property আছে যার value হচ্ছে “Laravel” .
* তৃতীয়ত , আমরা আমাদের Student child class দিয়ে Course class থেকে getCourseName method কল করি। যাতে আমরা আমাদের Student Child Class এর course name হিসেবে “laravel” পাই। যাই হোক Course class এর getCourseName Method টি আমাদের কে ফলাফল হিসেবে “Laravel” এর পরিবর্তে “PHP Professional” রিটার্ন করে, এর কারণ হচ্ছে self keyword যেই class এ কল করা হয় সেই class ছাড়া child বা অন্য কোনো class কে নির্দিষ্ট করেনা।

উপরের সমস্যা কে সমাধান করার জন্য PHP 5.3 থেকে Late Static Binding সুবিধা যোগ করে। মূলত, self keyword এর পরিবর্তে static keyword টি ব্যবহার করা হয়েছে, যেন যেই class থেকে call করা হবে যেন সেই class কে নির্দিষ্ট করে। চলুন উপরে আমাদের উদাহরণ পরিবর্তন করা যাক:

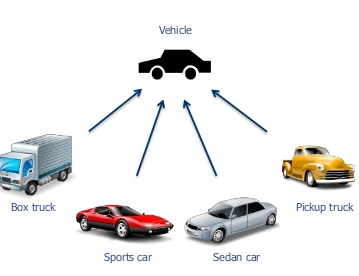
**উদাহরণ: “static” keyword বা Late Static Binding এর ব্যবহার**

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13 | <?php  class Course{      protected static $courseName = 'Professional PHP';      public static function getCourseName(){          return static::$courseName;      }  }    class Student extends Course{      protected static $courseName = 'Laravel';  }    echo Student::getCourseName(); // Laravel |

এখন আমরা আমাদের প্রত্যাশিত ফলাফল পাব।

৮: Abstract Class and Methods

**PHP তে Abstract Class কি?**

[](http://w3programmers.com/bangla/abstract-class-and-methods/abstract-class/)

PHP তে Abstract Class হচ্ছে এক ধরনের বিশেষ class যার থেকে কোনো Object instantiate বা তৈরী করা যাবেনা। কিন্তু Abstract Class গুলো থেকে child class তৈরী বা inherit করা যাবে। কোনো class কে abstract ঘোষণা করতে হলে class keyword এর সামনে abstract keyword টি দিতে হয়। “যেমন- abstract class className{....}” . কোন class এর মধ্যে যদি এক বা একাধিক abstract Method থাকে, সেটিকে অবশ্যই Abstract class হিসেবে Declare করতে হবে । তবে Abstract Class এ abstract এবং Non-Abstract দুই ধরনের মেথডই থাকতে পারে।

**PHP তে Abstract Method কি?**

PHP তে Abstract Method একধরণের বিশেষ Abstract Class Method যার বডিতে কোনো code define করা থাকেনা শুধু Method Signature থাকে , অর্থাৎ শুধুমাত্র Method এর নাম এবং Parameter সমূহ Declare করা থাকে। কোনো Method কে abstract ঘোষণা করতে হলে function keyword এর সামনে abstract keyword টি দিতে হয়। তারপর perentheses “( )” তারপর semicolon ” ; ” দিতে হয়। “যেমন- abstract function functionName();“. Abstract Method গুলোকে সম সংখ্যক Parameter সহ child class এ অবশ্যই implement করতে হবে। অর্থাৎ, আপনি চাইলেই child class এ কোন Method এর একটি Parameter যোগ বা বাদ দিতে পারবেন না । তবে default value সহ Parameter যোগ করতে পারবেন। Abstract Method গুলোর Visibility একই অথবা বেশি Open রাখতে হবে অর্থাৎ protected থাকলে protected অথবা public রাখতে হবে। abstract method কে private ঘোষণা করা যায়না।

এবার চলুন abastract class নিয়ে কয়েকটা উদাহরণ দেখা যাক।

**উদাহরণ ১:**

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15 | <?php  abstract class AbstractClass{      // Our abstract method only needs to define the required arguments      abstract protected function getName($name);      }    class childClass extends AbstractClass{      public function getName($name) {          return 'Hi '.$name.' !';      }  }    $class = new childClass;  echo $class->getName('Shahriar'), '\n';  ?> |

**Result:**

Hi! Shahriar

**উদাহরণ ২:**

এবার আমরা দেখবো সম সংখ্যক Parameter ছাড়া child class এ implement করলে কি সমস্যা হতে পারে।

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15 | <?php  abstract class AbstractClass{      // Our abstract method only needs to define the required arguments      abstract protected function getName($name);      }    class childClass extends AbstractClass{      public function getName($name,$prefix) {          return 'Hi'.$name.' !';      }  }    $class = new childClass;  echo $class->getName('Shahriar', 'Mr. '), '\n';  ?> |

**Result:**

Fatal error: Declaration of childClass::getName($name, $prefix) must be compatible with AbstractClass::getName($name) in [...][...]on line 11

**ব্যাখ্যা:** লক্ষ্য করুন parent class এ getName Method এ Parameter ছিল একটি , এখন child class এ কোনো default value ছাড়া অতিরিক্ত Parameter দেয়াই PHP একটি fatal error দেখাচ্ছে।

**উদাহরণ ৩:**

এবার আমরা দেখবো সম সংখ্যক Parameter ছাড়া child class এ implement করার পদ্ধতি ।

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15 | <?php  abstract class AbstractClass{      // Our abstract method only needs to define the required arguments      abstract protected function getName($name);      }    class childClass extends AbstractClass{      public function getName($name,$prefix='Mr. ') {          return 'Hi '.$prefix.$name.' !';      }  }    $class = new childClass;  echo $class->getName('Shahriar'), '\n';  ?> |

**Result:**

Hi Mr. Shahriar !

**ব্যাখ্যা:** লক্ষ্য করুন parent class এ getName Method এ Parameter ছিল একটি , এখন child class এ default value সহ অতিরিক্ত Parameter দেওয়ার পরও PHP কোনো error ছাড়াই ফলাফল দেখাচ্ছে।

৯: PHP Object Interfaces

**Object Oriented Programming এ Interface কি?**

[](http://w3programmers.com/bangla/php-object-interfaces/interface/)

Object Oriented Programming এ interface এর ধারণা টা অনেকটা একজন Web Developer এর কাজের ধরণের মত। একজন Web Developer যখন কোনো একটি HTML Template বানায় প্রথমে তাকে একটি PSD ডিজাইন দেয়া হয়। এর মাধ্যমে Web Developer বুঝতে পারে তাকে HTML দিয়ে কি বানাতে হবে। অর্থাৎ এর রং , মেনু এবং বাটন এবং এর input গুলো দেখতে কেমন হবে এসবই বর্ননা করা থাকে PSD ডিজাইনে। একজন Web Developer চাইলেই কিন্তু PSD ডিজাইনকে সরাসরি ব্যবহার করতে পারেনা অথবা ব্যবহার করা সম্ভবও না। এর মানে হলো আপনাকে অবশ্যই PSD থেকে HTML এ কনভার্ট করে একে ব্যবহার করতে হবে। আর একে বলা হয় Implementation। তাহলে এখানে PSD টিই হলো পুরো Template এর একটি Interface অর্থাৎ Template টি কেমন হবে তা এখানেই বর্নানা করা হয়েছে, আর Object Oriented Programming এর interface ও একই। আপনার একটি interface থাকবে যেখানে আপনি class গুলো কি কাজ করবে তা থাকবে। কিন্তু কিভাবে কাজ করবে সেটি বলা থাকবে না। আর interface গুলোকে আপনি সরাসরি ব্যবহার করতে পারবেন না, সেজন্য অবশ্যই একে class এর উপর implement করে নিতে হবে।। abstract class আর interface প্রায় একই, শুধু পার্থক্য হচ্ছে abstract class এ property, constant, abstract এবং non-abstract সব ধরনের Method ই থাকতে পারে এবং সবগুলো Method এর Visibility public অথবা protected থাকতে পারে । কিন্তু interface এর মধ্যে শুধু method থাকতে পারবে এবং সবগুলো Method বাই ডিফল্ট abstract Method হিসেবে থাকে এবং সবগুলো Method এর Visibility অবশ্যই public হতে হবে। সাধারণত: একটা child class শুধু মাত্র একটা Parent Class থেকেই inherit করতে পারে, অর্থাৎ, একই সময়ে একাধিক class থেকে inherit করতে পারেনা। কিন্তু একটা class এর উপর একই সময়ে একাধিক interface implement করা যায়। interface এর সবগুলো Method কে সম সংখ্যক Parameter সহ যেই class এ implement করা হয়েছে সেই class এ অবশ্যই implement করতে হবে। অর্থাৎ, আপনি চাইলেই implement কৃত class এ কোন Method এর একটি Parameter যোগ বা বাদ দিতে পারবেন না । তবে default value সহ Parameter যোগ করতে পারবেন।

**interface ঘোষণা করার নিয়ম কি?**

interface ঘোষণা করার জন্য interface keyword এর পর interface এর নাম দিতে হয়। নিচের সিনটেক্স টি দেখুন :

|  |  |
| --- | --- |
| 1  2  3  4  5  6 | <?php  interface MyInterface {    public function aMethod();    public function anotherMethod();  }  ?> |

**interface কে class এর মধ্যে implement করার নিয়ম কি?**

interface কে কোনো class এ implement করতে হলে class এর নামের পর implements keyword দিতে হয়, তারপর interface এর নাম গুলোকে কমা (,) দিয়ে লিখতে হয়। নিচের সিনটেক্স টি দেখুন :

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10 | <?php  class MyClass implements MyInterface1,...,MyInterfaceN {    public function aMethod() {      // (code to implement the method)    }    public function anotherMethod() {      // (code to implement the method)    }  }  ?> |

**interface ব্যবহার করে এইবার একটা বাস্তব উদাহরণে আশা যাক:**

উদাহরণস্বরূপ , একটি Web Forum এর কথা ধরা যাক , Forum মেম্বারদের information কে Database থেকে তুলে আনা এবং সংরক্ষণ করার জন্য একটা Member class থাকবে। এবং ফোরাম মেম্বারদের তৈরী করা টপিক গুলোকে তুলে আনা এবং সংরক্ষণ করার জন্য একটা Topic Class থাকবে। এবং এগুলো একটা আরেকটা থেকে আলাদা থাকবে , কারণ তারা প্রত্যেকে আলাদা আলাদা ভাবে কাজ করে। কিন্তু দুটি class ই Data তুলে আনা এবং সংরক্ষণ এর মত কিছু common কাজ করে।

যাইহোক, আমরা Member এবং Topic দুটি Object দিয়ে MySQL Database থেকে Data তুলে আনা এবং সংরক্ষণ এর মত কিছু common কাজগুলো করার জন্য Persistable নামে একটা interface তৈরী করব। যা Database এর common কাজগুলো করার জন্য Method গুলোকে নির্দিষ্ট করে দেবে। নিচের উদাহরণটি দেখুন :

|  |  |
| --- | --- |
| 1  2  3  4  5 | interface Persistable {    public function save();    public function load();    public function delete();  } |

এখন আসুন আমরা আমাদের Member Class তৈরী করব , এবং এর মধ্যে Persistable interface কে implement করব। এর মানে হল যে Member Class এর মধ্যে অবশ্যই save(), load() এবং delete() Method গুলো implement করতে হবে। পাশাপাশি class এর নিজস্ব Method গুলো ও থাকবে। নিচের উদাহরণটি দেখুন :

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19  20  21  22  23  24  25  26  27  28  29  30 | <?php  class Member implements Persistable {    private $username;    private $location;    private $homepage;    public function \_\_construct( $username, $location, $homepage ) {      $this->username = $username;      $this->location = $location;      $this->homepage = $homepage;    }    public function getUsername() {      return $this->username;    }    public function getLocation() {      return $this->location;    }    public function getHomepage() {      return $this->homepage;    }    public function save() {      echo "Saving member to database<br>";    }    public function load() {      echo "Loading member from database<br>";    }    public function delete () {      echo "Deleting member from database<br>";    }  }  ?> |

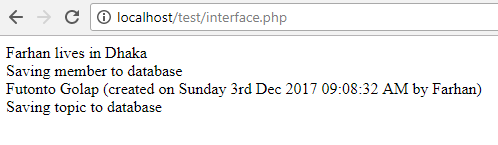
একই ভাবে Topic Class তৈরী করব , এবং এর মধ্যে Persistable interface কে implement করব। এর মানে হল এখানেও Topic Class এর মধ্যে অবশ্যই save(), load() এবং delete() Method গুলো implement করতে হবে। পাশাপাশি class এর নিজস্ব Method গুলো ও থাকবে। নিচের উদাহরণটি দেখুন :

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19  20  21  22  23  24  25  26 | <?php  class Topic implements Persistable {    private $subject;    private $author;    private $createdTime;    public function \_\_construct( $subject, $author ) {      $this->subject = $subject;      $this->author = $author;      $this->createdTime = time();    }    public function showHeader() {      $createdTimeString = date( 'l jS M Y h:i:s A', $this->createdTime );      $authorName = $this->author->getUsername();      echo "$this->subject (created on $createdTimeString by $authorName)<br>";    }    public function save() {      echo "Saving topic to database<br>";    }    public function load() {      echo "Loading topic from database<br>";    }    public function delete () {      echo "Deleting topic from database<br>";    }  }  ?> |

এখন আমরা Member Object তৈরী করব getUsername() মেথড call করার জন্য। এবং Topic Object তৈরী করব showHeader() মেথড call করার জন্য। একই সাথে আমরা interface method গুলো যেমন save(), load() এবং delete() গুলো call করব, যেগুলো দুটি class এ implement করা হয়েছে।

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8 | <?php  $aMember = new Member( "Farhan", "Dhaka", '<http://w3programmers.com/>' );  echo $aMember->getUsername() . " lives in " . $aMember->getLocation() ."<br>";  $aMember->save();  $aTopic = new Topic( "Futonto Golap", $aMember );  $aTopic->showHeader();  $aTopic->save();  ?> |

**Result**

[](http://w3programmers.com/bangla/php-object-interfaces/interface-example/)

১০: PHP OOP Overloading

**PHP OOP তে Overloading কি?**

[](http://w3programmers.com/bangla/php-oop-overloading/php-oop-overloading-2/)

class এর বাহির থেকে class ব্যবহারকারীর চাহিদা অনুযায়ী স্বয়ংক্রিয়ভাবে class এর মধ্যে (যা পূর্ব থেকে define করা নাই এমন) যেকোনো property অথবা Method তৈরী করার পদ্ধতিকে PHP Object Oriented Programming এর পরিভাষায় Overloading বলা হয়। Overloading কে আমরা class ব্যবহারকারীর চাহিদা অনুযায়ী স্বয়ংক্রিয়ভাবে property এবং Method তৈরী করার পদ্ধতিও বলতে পারি।

**Overloading কত প্রকার ?**

**PHP তে Overloading মূলত দুইপ্রকার:**  
১. Property Overloading  
২. Method Overloading

**Property Overloading কি?**

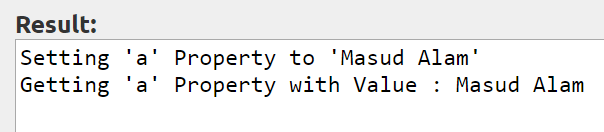
class এর বাহির থেকে class ব্যবহারকারীর চাহিদা অনুযায়ী স্বয়ংক্রিয়ভাবে class এর মধ্যে (যা পূর্ব থেকে define করা নাই এমন) যেকোনো property তৈরী করার পদ্ধতিকে PHP Object Oriented Programming এর পরিভাষায় Property Over Loading বলা হয়। Property Overloading কে আমরা class ব্যবহারকারীর চাহিদা অনুযায়ী স্বয়ংক্রিয়ভাবে property তৈরী করার পদ্ধতিও বলতে পারি।

PHP তে Property Overloading এ আমরা \_\_set() function দিয়ে Overloaded property তে value সেট করি, \_\_get() function দিয়ে Overloaded property এর value get করি, \_\_unset() function দিয়ে Overloaded property এর value, class এর বাহির থেকে খালি করি এবং \_\_isset() function দিয়ে Overloaded property এর value আছে কি না তা class এর বাহির থেকে চেক করি

**PHP তে Property Overload করার জন্য \_\_set() এবং \_\_get ফাঙ্কশনের কাজ কি?**

PHP তে Property Overload করার জন্য \_\_set() function দিয়ে class এর বাহির থেকে নতুন property তৈরী করা যায়। এবং তৈরিকৃত property তে value set করা যায়। \_\_get() function দিয়ে Overloaded property এর value কে class এর বাহির থেকে get করা যায়।

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19  20  21  22  23  24  25  26  27  28  29 | <?php  class PropertyTest  {      /\*\*  Location for overloaded data.  \*/      private $data = array();        public function \_\_set($name, $value)      {          echo "Setting '$name' Property to '$value'\n";          $this->data[$name] = $value;      }        public function \_\_get($name)      {          echo "Getting '$name' Property with Value : ";          if (array\_key\_exists($name, $this->data)) {              return $this->data[$name];          }      }    }      $obj = new PropertyTest;    $obj->a = "Masud Alam";  echo $obj->a . "\n";    ?> |

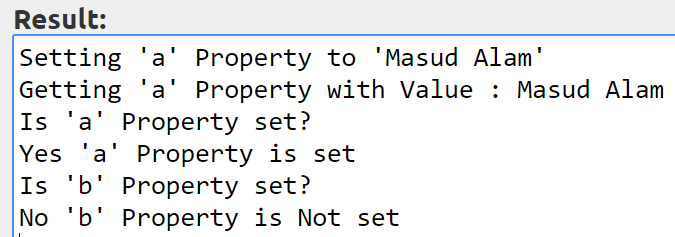
[](http://w3programmers.com/bangla/php-oop-overloading/property-overloading-value-set-and-get/)

**ব্যাখ্যা:** প্রথমে আমরা \_\_set() function দিয়ে class টির মধ্যে $a নামে একটি property তৈরী করি (যা আমাদের class এ define করা ছিলোনা ) একই সাথে $a তে value সেট করি। এবং একই সাথে $data নামে একটা private array property তে property name এবং তার value টি সংরক্ষণ করে রাখি। তারপর \_\_get() function দিয়ে class টির মধ্যে $data array property থেকে $a নামে property এর value গেট করি।

**PHP তে Property Overload করার জন্য \_\_isset() ফাঙ্কশনের কাজ কি?**

class এর বাহির থেকে যখন কোনো overloaded property কে isset() function দিয়ে চেক করা হয়, তখন class এর মধ্যে \_\_isset() function টি স্বয়ংক্রিয় ভাবে execute হয়। এবং overloaded property এর value আছে কি না তা চেক করার সুযোগ করে দেয়।

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19  20  21  22  23  24  25  26  27  28  29  30  31  32  33  34  35  36  37  38 | <?php  class PropertyTest  {      /\*\*  Location for overloaded data.  \*/      private $data = array();        public function \_\_set($name, $value)      {          echo "Setting '$name' Property to '$value'\n";          $this->data[$name] = $value;      }        public function \_\_get($name)      {          echo "Getting '$name' Property with Value : ";          if (array\_key\_exists($name, $this->data)) {              return $this->data[$name];          }      }      // Checking Property Value is set?      public function \_\_isset($name)      {          echo "Is '$name' Property set?\n";          echo isset($this->data[$name])?"Yes '$name' Property is set":"No '$name' Property is Not set";          echo "\n";      }    }      $obj = new PropertyTest;    $obj->a = "Masud Alam";  echo $obj->a . "\n";  isset($obj->a);  isset($obj->b);    ?> |

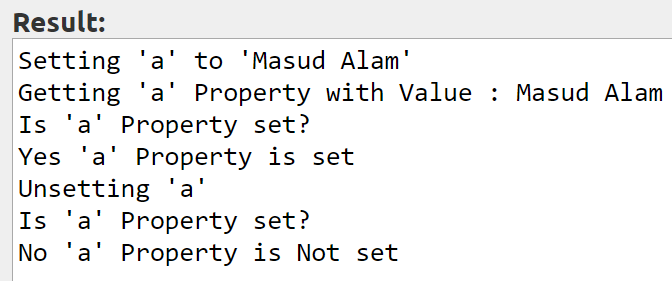
[](http://w3programmers.com/bangla/php-oop-overloading/php-property-overloading-check-property-value-is-set/)

**ব্যাখ্যা:** লক্ষ্য করুন , লাইন নম্বর ৩৫ এবং ৩৬ এ ফাঙ্কশন isset() দিয়ে দুটি unknown property কে class এর মধ্যে value সেট আছে কিনা চেক করি, আর isset () function call করার সাথে সাথে আমাদের class এর মধ্যে ২১ নম্বর লাইনে \_\_isset() ফাঙ্কশনটি স্বয়ংক্রিয় ভাবে execute হয়। overloaded property গুলোর value আছে কিনা তা চেক করে।

**PHP তে Property Overload করার জন্য \_\_unset() ফাঙ্কশনের কাজ কি?**

class এর বাহির থেকে যখন কোনো overloaded property কে unset() function দিয়ে চেক করা হয়, তখন class এর মধ্যে \_\_unset() function টি স্বয়ংক্রিয় ভাবে execute হয়। এবং overloaded property এর value কে unset করার সুযোগ করে দেয়।

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19  20  21  22  23  24  25  26  27  28  29  30  31  32  33  34  35  36  37  38  39  40  41  42  43  44  45  46 | <?php  class PropertyTest  {      /\*\*  Location for overloaded data.  \*/      private $data = array();        public function \_\_set($name, $value)      {          echo "Setting '$name' to '$value'\n";          $this->data[$name] = $value;      }        public function \_\_get($name)      {           echo "Getting '$name' Property with Value : ";          if (array\_key\_exists($name, $this->data)) {              return $this->data[$name];          }      }        public function \_\_isset($name)      {          echo "Is '$name' Property set?\n";          echo isset($this->data[$name])?"Yes '$name' Property is set":"No '$name' Property is Not set";          echo "\n";      }        public function \_\_unset($name)      {          echo "Unsetting '$name'\n";          unset($this->data[$name]);      }    }      $obj = new PropertyTest;    $obj->a = "Masud Alam";  echo $obj->a . "\n";  echo isset($obj->a);  unset($obj->a);  echo isset($obj->a);  echo "\n";    ?> |

[](http://w3programmers.com/bangla/php-oop-overloading/php-property-overloading-unset-property-value/)

**ব্যাখ্যা:** লক্ষ্য করুন , লাইন নম্বর ৪২ এ ফাঙ্কশন unset() দিয়ে unknown property এর value unset করার জন্য call করি, আর unset () function call করার সাথে সাথে আমাদের class এর মধ্যে ২8 নম্বর লাইনে \_\_unset() ফাঙ্কশনটি স্বয়ংক্রিয় ভাবে execute হয়। overloaded property গুলোর value unset করে।

**Method Overloading কি?**

class এর বাহির থেকে class ব্যবহারকারীর চাহিদা অনুযায়ী স্বয়ংক্রিয়ভাবে class এর মধ্যে (যা পূর্ব থেকে define করা নাই এমন) যেকোনো Method তৈরী করার পদ্ধতিকে PHP Object Oriented Programming এর পরিভাষায় Method Over Loading বলা হয়। Method Overloading কে আমরা class ব্যবহারকারীর চাহিদা অনুযায়ী স্বয়ংক্রিয়ভাবে Method তৈরী করার পদ্ধতিও বলতে পারি।

PHP তে Method Overloading এ আমরা \_\_call() function দিয়ে class এর বাহিরের থেকে Overloaded Method তৈরী করি এবং Overloaded Method এ Parameter set করি। এবং \_\_callStatic() function দিয়ে class এর বাহিরের থেকে Overloaded static Method তৈরী করি এবং Overloaded static Method এ Parameter set করি

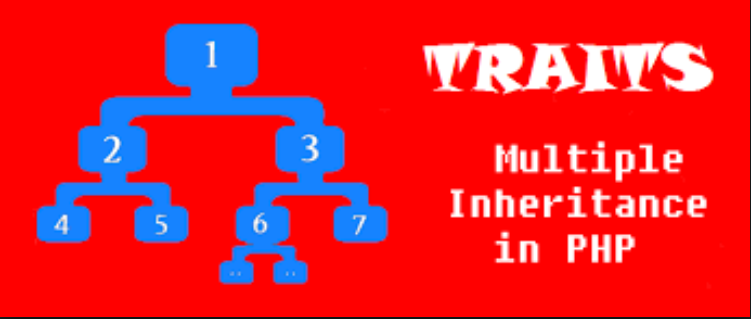
|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19  20  21 | <?php  class MethodTest  {      public function \_\_call($name, $arguments)      {          echo "Calling object method '$name' "               . implode(', ', $arguments). "\n";      }        public static function \_\_callStatic($name, $arguments)      {          echo "Calling static method '$name' "               . implode(', ', $arguments). "\n";      }  }    $obj = new MethodTest;  $obj->runTest('in object context');    MethodTest::runTest('in static context');  ?> |

নিচে Method Overloading feature ব্যবহার করে একটা CRUD উদহারণ দেওয়া হলো

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19  20  21  22  23  24  25  26  27  28  29  30  31  32  33  34  35  36  37  38  39  40  41  42  43  44  45  46  47  48  49  50  51  52  53  54  55  56  57  11: PHP OOP Class Autoloading  **PHP OOP তে class autoloading বিষয়টা কি?**  [PHP Class Autoloading](http://w3programmers.com/bangla/php-oop-class-autoloading/autoloading/)  মূলত: আগে থেকেই কোনো class include না করে রেখেই class object তৈরী করা কালীন স্বয়ংক্রিয় ভাবে class load হওয়ার পদ্ধতিকে PHP তে class autoloading বলা হয়। অর্থাৎ, একজন PHP প্রোগ্রামার যেকোনো বড় Project Development কালীন একাধিক class library তৈরী করতে হয়। এবং Project Development এর সুবিধার্তে বিভিন্ন সময়ে ভিন্ন ভিন্ন class এর ভিন্ন ভিন্ন Object তৈরী করতে হয়। আর এর জন্য প্রত্যেকটি PHP class file কে include অথবা require function দিয়ে program বা code এর শুরুতে সংযুক্ত বা include করে রাখতে হয়, আর এতে একজন প্রোগ্রামার কে আগে থেকেই অনেক অপ্রয়োজনীয় class গুলো include করে রাখতে হয়, আর তাতে project এর performance সহ বিভিন্ন সমস্যা তৈরী হয়। আর এই সমস্যা সমাধানের জন্য PHP-5 থেকে class autoloading সুবিধা সংযুক্ত করা হয়েছে। class autoloading করার জন্য class এর নাম এবং class file এর নাম একই হতে হয়।  **PHP OOP তে কিভাবে class autoload করা হয়?**  PHP তে **spl\_autoload\_register()** function এর মাধ্যমে class autoload করা হয়। চলুন আরেকটু ভালো ভাবে ব্যাখ্যা করা যাক :  উদাহরণস্বরূপ, একটি Web Forum Application তৈরী করবেন, এবং Web Forum Application এর মধ্যে আপনি Member class এর জন্য Member.php নামে একটি ফাইল এবং topic class এর জন্য Topic.php নামে একটি ফাইল তৈরী করলেন। সাধারণত আপনি আপনার class file গুলো আপনার ওয়েব সার্ভারের কোনো folder এ রাখবেন ঠিক এইরকম।  classes/  Member.php  Topic.php  এখন আপনি যদি আপনার application এর জন্য Member class থেকে একটা object তৈরী করার দরকার হয় বা মনে করেন, তাহলে আপনাকে প্রথমে Member.php ক্লাস ফাইল অন্তর্ভুক্ত করতে হবে, তারপর object তৈরী করতে হবে ঠিক নিচের মত করে:   |  |  | | --- | --- | | 1  2  3  4 | <?php  require\_once( "classes/Member.php" );  $member = new Member();  ?> |   এখন একই page এ আপনি যদি আপনার application এর জন্য Topic class থেকে একটা object তৈরী করার দরকার হয় বা মনে করেন, তাহলে আপনাকে আবার Member.php class file এর মত Topic.php class file অন্তর্ভুক্ত করতে হবে, তারপর object তৈরী করতে হবে ঠিক নিচের মত করে:   |  |  | | --- | --- | | 1  2  3  4  5  6 | <?php  require\_once( "classes/Member.php" );  require\_once( "classes/Topic.php" );  $member = new Member();  $member = new Topic();  ?> |   এখন যদি আমরা PHP Class Autoloading Feature টি ব্যবহার করি, তাহলে বার বার class file include করার প্রয়োজন পরবেনা, আর PHP class Autoloading Feature টি ব্যবহার করতে চাইলে আপনাকে প্রথমে একটি function তৈরী করতে হবে, ঠিক নিচের উদাহরণটির মত করে।   |  |  | | --- | --- | | 1  2  3  4  5 | <?php  function loadClass($class\_name) {      include "Classes/$class\_name.php";  }  ?> |   **বিঃ দ্রঃ** loadClass function এ include function এর মধ্যে class file পাথটি ঠিক আছে কিনা তা খেয়াল রাখবেন। পাথ ঠিক না থাকলে class autoload হবেনা।  **বিঃ দ্রঃ** function এর নাম loadClass হতে হবে এমন কোনো কথা নেই। আপনি যেকোনো নামই ব্যবহার করতে পারবেন।  এখন আমাদেরকে spl\_autoload\_register() function দিয়ে আমাদের loadClass() function কে autoload এর জন্য রেজিস্টার করে নিতে হবে। নিচের উদাহরণটি দেখুন :   |  |  | | --- | --- | | 1  2  3  4  5  6 | <?php  function loadClass($class\_name) {      include "Classes/$class\_name.php";  }  spl\_autoload\_register("loadClass");  ?> |   এবার আপনার কাজ শেষ , এখন যখনি আপনি যেকোনো class এর object তৈরী করবেন সাথে সাথে সেই class টি autoload হয়ে যাবে। নিচের উদাহরণটি দেখুন :   |  |  | | --- | --- | | 1  2  3  4  5  6  7  8  9  10  11  12 | //File: class\_autoload.php  <?php  function loadClass($class\_name) {      include "Classes/$class\_name.php";  }    spl\_autoload\_register("loadClass");    new Member;  echo "<br>";  new Topic;  ?> |   [Class Autoloading](http://w3programmers.com/bangla/php-oop-class-autoloading/class-autoloading/)  বুঝার সুবিদার্থে আপনার Member.php এবং Topic.php File দুটিতে নিচের কোড গুলো লিখুন। তারপর class\_autoload.php file টি execute করুন।  File: Member.php   |  |  | | --- | --- | | 1  2  3  4  5  6  7 | <?php  class Member{      public function \_\_construct(){          echo "Hello Member Class";      }  }  ?> |   File:Topic.php   |  |  | | --- | --- | | 1  2  3  4  5  6  7 | <?php  class Topic{      public function \_\_construct(){          echo "Hello Topic Class";      }  }  ?> |   **Anonymous function দিয়ে class autoloading**  spl\_autoload\_register() function এ আমরা কোনো function এর নাম ব্যবহারের পরিবর্তে Anonymous function ব্যবহার করতে পারি। নিচের উদাহরণ দেখুন :   |  |  | | --- | --- | | 1  2  3  4  5  6  7  8  9  10 | <?php  spl\_autoload\_register(          function ($className){              include "Classes/$className.php";          });    new Member;  echo "<br>";  new Topic;  ?> |   **Class Autoloading এ exception handling এর ব্যবহার**  অনেক সময় আমরা Object তৈরী করার সময় class এর নাম ভুল করে ফেলি, আর তাতে PHP engine আমাদের কিছু unexpected error দেখায় , এতে project security, Project Performance এ মারাত্মক সমস্যা সৃষ্টি করে, আর class Autoloading এর ক্ষেত্রে এই বিষয়টা আমরা খুব সহজে exception handling দিয়ে solve করতে পারি। নিচের উদাহরণটি দেখুন:   |  |  | | --- | --- | | 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19 | <?php  spl\_autoload\_register(function ($className){       if (file\_exists("classes/$className.php")) {            require\_once "classes/$className.php";        }        else{      throw new Exception("Unable to load $className Class.");      }  });    try {      new Member;      echo "<br>";      new NotExist;  } catch (Exception $e) {      echo $e->getMessage(), "\n";  }    ?> |   **Result**  Hello Member Class  Unable to load NotExist Class.  **ব্যাখ্যা:** লাইন নম্বর ১২ এ আমাদের classes folder এর মধ্যে Member class থাকায় কোনো error দেখায় নাই। কিন্তু ১৪ নম্বর লাইনের NotExist class টি না থাকায় আমাদের custom error দেখাচ্ছে।  **Mulitple Autoloads**  আমাদের প্রোগ্রামিং লাইফ এ class file এর নাম গুলো সব সময় একই রকম হবে তার কোনো বিশ্বাস নেই। এমন কি file গুলো সব সময় একই folder এ থাকবে তার ও কোনো নিশ্চয়তা নাই , এবং file extension গুলো এক রকম নাও হতে পারে , আর এর জন্য PHP SPL autoload এ আছে Multiple Autoload। চলুন একটা উদাহরণ দিয়ে দেখা যাক :   |  |  | | --- | --- | | 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19  20  21  22  23  24  25  26  27  28  29  30  31  32  33  34  35  36  37  38  39  40 | <?php        /\*\*\* nullify any existing autoloads \*\*\*/      spl\_autoload\_register(null, false);        /\*\*\* specify extensions that may be loaded \*\*\*/      spl\_autoload\_extensions('.php, .class.php, .lib.php');        /\*\*\* class Loader \*\*\*/      function classLoader($class)      {          $filename = strtolower($class) . '.class.php';          $file ='classes/' . $filename;          if (!file\_exists($file))          {              return false;          }          include $file;      }        function libLoader($class)      {          $filename = strtolower($class) . '.lib.php';          $file ='libs/' . $filename;          if (!file\_exists($file))          {              return false;          }          include $file;      }        /\*\*\* register the loader functions \*\*\*/      spl\_autoload\_register('classLoader');      spl\_autoload\_register('libLoader');        /\*\*\* a new instance of norman \*\*\*/      $norman = new norman;        /\*\*\* make norman do some thing \*\*\*/      $norman->do\_something(); |   **Interfaces Autoload**  উপরের উদাহরণ গুলোতে আমরা দেখেছি কিভাবে class কে autoload করা যায়। তবে আপনি চাইলে interface কেও autoload করতে পারেন। নিচের উদাহরণটি দেখুন :   |  |  | | --- | --- | | 1  2  3  4  5  6  7  8  9  10  11 | <?php      /\*       \* icontroller.class.php       \* interface to ensure all classes have an index method       \*       \*/      interface iController      {          public function index();      }  ?> | | 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19  20  21  22  23  24  25  26  27  28  29  30  31  32  33  34  35  36 | <?php      /\*\*\* nullify any existing autoloads \*\*\*/      spl\_autoload\_register(null, false);        /\*\*\* specify extensions that may be loaded \*\*\*/      spl\_autoload\_extensions('.php, .class.php');        /\*\*\* class Loader \*\*\*/      function classLoader($class)      {          $filename = strtolower($class) . '.class.php';          $file ='classes/' . $filename;          if (!file\_exists($file))          {              return false;          }          include $file;      }        /\*\*\* register the loader functions \*\*\*/      spl\_autoload\_register('classLoader');        class blog implements iController      {          public function index()          {              echo 'hello from the index';          }      }        /\*\*\* a new blog instance \*\*\*/      $blog = new blog;        /\*\*\* run the index method \*\*\*/      $blog->index();  ?> | | <?php  class Db{      private $conn;      public function \_\_construct($host,$user,$pass,$db){          $this->conn=new mysqli($host,$user,$pass,$db);          if($this->conn->connect\_errno){              die("Connection Fail for: ".$this->conn->connect\_error);          }      }      public function \_\_call($method,$params){          //echo $params[1];          extract($params[0]);          switch($action){              case "read":              if(isset($where)){                  $sql="SELECT $cols FROM $table WHERE $where";                  $result=$this->conn->query($sql);                  if($result->num\_rows>0){                      return $result->fetch\_assoc();                  }              }              else{                  $sql="SELECT $cols FROM $table";                  $result=$this->conn->query($sql);                  if($result->num\_rows>0){                      return $result->fetch\_all(MYSQLI\_ASSOC);                  }              }              break;              case "insert":              $sql="INSERT INTO $table SET $cols";              $result=$this->conn->query($sql);              if($this->conn->affected\_rows>0){                  return true;              }              break;              case "update":              $sql="UPDATE $table SET $cols WHERE $where";              $result=$this->conn->query($sql);              if($this->conn->affected\_rows>0){                  return true;              }              break;              case "delete":              $sql="DELETE FROM $table WHERE $where";              $result=$this->conn->query($sql);              if($this->conn->affected\_rows>0){                  return true;              }              break;              default:              return false;              break;          }      }    }    $connect=new Db("localhost","root","","php85");  echo "<pre>";  print\_r($connect->read(array("table"=>"students","cols"=>"\*","action"=>"read")));  //print\_r($connect->readAll(array("table"=>"students","cols"=>"\*")));  /\*print\_r($connect->getById(array("table"=>"students","cols"=>"name,mobile","where"=>"id=19","action"=>"read")));\*/  //echo $connect->Add(array("table"=>"students","cols"=>"name='Iftekhar',mobile='0122113323',address='Laxmipur,Bangladesh'","action"=>"insert"))?"Insert Success":"Insert Fail";    //echo $connect->removeData(array("table"=>"students","action"=>"delete","where"=>"id=19"))?"Delete Success":"Delete Fail";  echo "</pre>"; |

১৪: PHP OOP Traits

**PHP OOP তে trait কি?**

[](http://w3programmers.com/bangla/php-oop-traits/php-oop-trait/)

সাধারণত PHP কে বলা হয় Single Inheritance Language অর্থাৎ, PHP Language টি Multiple Inheritance সাপোর্ট করেনা। আর Trait হচ্ছে PHP OOP তে Single Inheritance এর সীমাবদ্ধতা দূর করার এবং Multiple Inheritance ব্যবহার করার একটি নতুন concept . যা PHP 5.4 এ প্রথম ব্যবহার করা হয়। Trait অনেকটা class এর মতোই, Trait কে Define করা হয় ক্লাসের মত করেই trait কিওয়ার্ডটি ব্যবহার করে। তবে এর থেকে class এর মত object তৈরী করা যায়না। কিন্তু একাধিক trait এর property এবং Method গুলোকে একটি single class এর মধ্যে ব্যবহার করা যায়। এবার চলুন একটা উদাহরণের মাধ্যমে আরো ভালো ভাবে বুঝা যাক :

File Name: **foo.php**

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11 | <?php  trait Foo   {      public function sayHello(){           return "Hello";      }      public function sayWorld(){       return "World";      }   }  ?> |

use কীওয়ার্ড ব্যবহার করে Trait কে ক্লাসে ব্যবহার করা হয়। চলুন trait ব্যবহার করে একটা উদাহরণ দেখা যাক :

File Name: **bar.php**

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13 | <?php  include("foo.php");  class Bar{      // Using the Trait Here       use Foo;  }     $obj = new Bar;     // Executing the method from trait   $obj->sayHello(); //Hello   $obj->sayWorld(); // World  ?> |

**ব্যাখ্যা:** লক্ষ্য করুন foo নামক trait কে use keyword দিয়ে bar class এর মধ্যে ব্যবহার করি, আর এতে আমরা bar এর object দিয়ে খুব সহজে foo trait এর method এবং property গুলো ব্যবহারের সুযোগ পাই।

**traits এবং class এর মধ্যে কার Precedence বা অগ্রাধিকার আগে?**

কাজ করার ক্ষেত্রে অনেক সময় দেখা যায়, একই নামের function একই সাথে trait, parent class এবং child class এ পাওয়া যায়, তখন প্রথমে child class এর Method পাবে, তারপর trait এর মধ্যে method টি পাবে, তারপর parent class এর method টি পাবে।  
চলুন একটা উদাহরণের মাধ্যমে দেখা যাক :

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19  20  21  22  23  24  25  26  27 | <?php   class Base{       public function sayHello(){           echo "say hello from base";      }  }    trait trt{      public function sayHello(){          echo "say hellow from trait";      }  }    class Child extends Base{        use trt;      public function sayHello(){          echo "hello from child class";      }  }    $objCls = new Child;    $objCls->sayHello();    ?> |

**Result**

hello from child class

এখন যদি আপনি child class এর sayHello() Method টি off অর্থাৎ comment দিয়ে hide করে রাখেন, তাহলে trt trait এর sayHello() Method execute হবে। চলুন একটা উদাহরণের মাধ্যমে দেখা যাক :

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19  20  21  22  23  24  25  26  27 | <?php   class Base{       public function sayHello(){           echo "say hello from base";      }  }    trait trt{      public function sayHello(){          echo "say hellow from trait";      }  }    class Child extends Base{        use trt;     /\* public function sayHello(){          echo "hello from child class";      }\*/  }    $objCls = new Child;    $objCls->sayHello();    ?> |

**Result:**

say hellow from trait

**কিভাবে একটা class এ একাধিক trait ব্যবহার করা যায়?**

একটা class এ একাধিক trait ব্যবহার করতে হলে class এর মধ্যে use keyword এর পর প্রতিটি trait কে একটির পর আরেকটি comma দিয়ে দিয়ে লিখতে হয়। চলুন একটা উদাহরণের মাধ্যমে দেখা যাক :

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19  20  21  22  23  24  25  26  27  28  29  30  31  32  33  34  35  36  37  38  39  40  41  42 | <?php  trait Subscriber{      public function subscriberLogin() {          echo "You\'re Logged in as Subscriber". '<br/>';      }  }    trait Contributor{      public function contributorLogin() {          echo "You're Logged in as Contributor". '<br/>';      }    }    trait Author{      public function AuthorLogin() {          echo "You're Logged in as Author." . '<br/>';      }  }    trait Administrator{      public function AdministratorLogin(){          echo "You're Logged in as Administrator" . '<br/>';      }  }    class Member{      use Subscriber, Contributor, Author, Administrator;      public function run() {          $this->subscriberLogin();          $this->contributorLogin();          $this->AuthorLogin();          $this->AdministratorLogin();          echo 'Members Login...done' . '<br/>';      }  }    $authentication = new Member();    $authentication->run();    ?> |

**Result:**

You’re Logged in as Subscriber

You’re Logged in as Contributor

You’re Logged in as Author.

You’re Logged in as Administrator

Members Login…done

**PHP OOP তে trait এর Conflict Resolution কি?**

অনেক সময়, যদি একটি class এ একাধিক trait ব্যবহার করে এবং এবং একাধিক class এ trait এ যদি একই নামে Method থাকে, আপনি যদি একাধিক trait ই একটি class এ ব্যবহার করেন তবে এটি আপনাকে fatal error দেখাবে। আর এই ব্যাপারটিকে বলা হয় Conflict Resolution. আর আমরা যদি insteadof operator দিয়ে compiler কে বলে দেন, যে একই নামের একাধিক Method এর মধ্যে কোন Method টি ব্যবহার হবে,অথবা method গুলোকে alias করে নেই , তাহলে আর কোনো error দেখাবেনা। চলুন একটা উদাহরণের মাধ্যমে দেখা যাক :

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19  20  21  22  23  24  25  26  27 | <?php    trait Foo{      public function first\_function(){          echo "From Foo Trait";      }  }    trait Bar{      public function first\_function(){          echo "From Bar Trait";      }  }    class FooBar{      use Foo, Bar{          // This class will now call the method          // first function from Foo only          Foo::first\_function insteadof Bar;      }  }    $obj = new FooBar;    $obj->first\_function();    ?> |

**Result**

From Foo Trait

**ব্যাখ্যা:** মূলতঃ আমরা ১৯ নম্বর লাইনে বলে দিয়াছি , এখানে Foo trait এর first\_function টি ব্যবহৃত হোক, trait Bar এর টা নয়।

এবার চলুন aliasing ব্যবহার করে একটা উদাহরণ দেখা যাক :

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19  20  21  22  23  24  25  26  27  28  29  30  31  32  33  34  35  36 | <?php  trait Foo{      public function first\_function(){          echo "From Foo Trait";      }  }    trait Bar{      public function first\_function(){          echo "From Bar Trait";      }    }    class FooBar{      use Foo, Bar{          // This class will now call the method          // first function from Foo Trait only\          Foo::first\_function insteadof Bar;          // first\_function of Bar can be          // accessed with second\_function          Bar::first\_function as second\_function;      }  }    $obj = new FooBar;    // Output: From Foo Trait    $obj->first\_function();    // Output: From Bar Trait    $obj->second\_function();    ?> |

**ব্যাখ্যা:** মূলতঃ আমরা ১৯ নম্বর লাইনে বলে দিয়াছি , এখানে Foo trait এর first\_function টি ব্যবহৃত হোক, trait Bar এর টা নয়। আবার ২২ নম্বর লাইনে আমরা Bar trait এর first\_function টিকে aliasing করে second\_function নাম দিয়েছি।

**Class এর মধ্যে ব্যবহৃত Trait এর method গুলোর default visibility পরিবর্তনের উপায় কি?**

সাধারণত class এর মতো trait এর method গুলোর visibility public,private এবং protected ঘোষণা করা যায়, তখন আপনি চাইল trait এর মধ্যের Method গুলোকে class এর মধ্যে as কীওয়ার্ড দিয়ে পরিবর্তন করতে পারবেন। নিচের উদাহরণটি দেখুন :

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19  20  21  22  23  24  25  26  27  28  29  30  31 | <?php  trait visible{      public function pub(){          echo "this is public method";      }      private function priv(){          echo "this is private";      }      protected function proc(){          echo "echo this is protected function";      }  }    class cls{      use visible{          priv as public;      }      function callPriv(){          $this->priv();      }  }    $objCls = new cls();    $objCls->pub();//echo this is public method    //$objCls->priv();//This is private    $objCls->callPriv(); //this is private    ?> |

**trait এর মধ্যে একাধিক trait এর ব্যবহার**

**use** কীওয়ার্ড ব্যবহার করে আমরা একটি trait এর মধ্যে একাধিক trait এর ব্যবহার করতে পারি। নিচের উদাহরণটি দেখুন :

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19  20  21  22  23  24  25  26 | <?php  trait Hello{      function sayHello() {          echo "Hello";      }  }    trait World{      function sayWorld(){          echo "World";      }  }    trait HelloWorld{      use Hello, World;  }    class MyWorld{      use HelloWorld;  }    $world = new MyWorld();    echo $world->sayHello() . " " . $world->sayWorld(); //Hello World    ?> |

**ব্যাখ্যা:** লাইন নম্বর ১৫ তে লক্ষ করুন , ট্রেইট HelloWorld এ আমরা Hello এবং World নামে দুটি trait ব্যবহার করেছি।

**traits এর মধ্যের properties গুলোকে class এ ব্যবহার :**

কোনো trait কে class এর মধ্যে ব্যবহার করার পর , trait এ অবস্থিত একই নামের property কে class এ define করা যায়না, তারপর ও যদি কেও trait এ অবস্থিত একই নামের property কে class এ define করে , সেক্ষেত্রে যদি property value যদি same হয় তাহলে PHP কোনো error বা warning দেখাবেনা , আর যদি value ও same না হয় , তাহলে PHP Fatal Error দেখাবে। নিচের উদাহরণটি দেখুন :

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14 | <?php  trait Calories {      public $banana = 105;      public $cake = 300;      public $donation = 205;  }    class Cookbook {      use Calories;  }    $c = new Cookbook;  echo $c->banana;  ?> |

**Result**

105

এবার আমরা ভিন্ন value দিয়ে check করব।

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15 | <?php  trait Calories {      public $banana = 105;      public $cake = 300;      public $donation = 205;  }    class Cookbook {      use Calories;      public $cake=400;  }    $c = new Cookbook;  echo $c->banana;  ?> |

**Output**

**Fatal error**: Cookbook and Calories define the same property ($cake) in the composition of Cookbook. However, the definition differs and is considered incompatible. Class was composed in **[...][...]** on line **8**

**ব্যাখ্যা:** লক্ষ্য করুন , লাইন নম্বর ১০ এ আমরা $cake এর 300 এর পরিবর্তে 400 দিয়েছি, যার দরুন PHP আমাদের কে fatal error দেখাচ্ছে।

**Trait এর মধ্যে static Property এবং Method এর ব্যবহার :**

PHP তে Trait এর মধ্যে static Property এবং Method দুটিই support করে। নিচের উদাহরণটি দেখুন :

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19  20  21 | <?php  trait Calories {      public $banana = 105;      public $cake = 300;      static public $donation = 205;        public static function test(){          return self::$donation;      }  }    class Cookbook {      use Calories;    }    $c = new Cookbook;  echo Cookbook::$donation;  echo "\n";  echo Cookbook::test();  ?> |

**Trait এর মধ্যে abstract Method এর ব্যবহার :**

আপনি চাইলে Trait এ abstract Method ঘোষণা করতে পারেন। আর যদি কোনো trait এ abstract Method থাকে, তখন abstract Method যুক্ত trait টি যেই class এ ব্যবহৃত হবে, সেই ক্লাস এ অবশ্যই abstract Method টি body সহ পুনরায় ঘোষণা করতে হবে। নিচের উদাহরণটি দেখুন :

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17 | <?php    trait Helper {      abstract public function greet();  }    class Foo {      use Helper;      public function greet($name){          printf('Hi there %s !', $name);      }  }    $foo = new Foo;    $foo->greet('Awesome Man');  ?> |

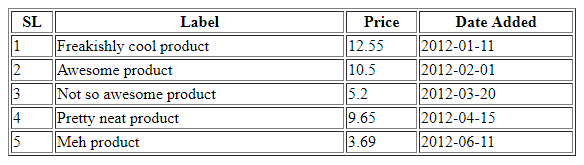
**Result**

Hi there Awesome Man !

**Trait ব্যবহার করে একটা sorting উদাহরণ :**

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19  20  21  22  23  24  25  26  27  28  29  30  31  32  33  34  35  36  37  38  39  40  41  42  43  44  45  46  47  48  49  50  51  52  53  54  55  56  57  58  59  60  61  62  63  64  65  66  67  68  69  70  71  72  73  74  75  76  77  78  79  80 | <?php  trait SortStrategy {      private $sort\_field = null;      private function string\_asc($item1, $item2) {          return strnatcmp($item1[$this->sort\_field], $item2[$this->sort\_field]);      }      private function string\_desc($item1, $item2) {          return strnatcmp($item2[$this->sort\_field], $item1[$this->sort\_field]);      }      private function num\_asc($item1, $item2) {          if ($item1[$this->sort\_field] == $item2[$this->sort\_field]){              return 0;          }              return ($item1[$this->sort\_field] < $item2[$this->sort\_field] ? -1 : 1 );      }      private function num\_desc($item1, $item2) {          if ($item1[$this->sort\_field] == $item2[$this->sort\_field]){              return 0;          }              return ($item1[$this->sort\_field] > $item2[$this->sort\_field] ? -1 : 1 );      }      private function date\_asc($item1, $item2) {          $date1 = intval(str\_replace('-', '', $item1[$this->sort\_field]));          $date2 = intval(str\_replace('-', '', $item2[$this->sort\_field]));          if ($date1 == $date2){              return 0;          }          return ($date1 < $date2 ? -1 : 1 );      }      private function date\_desc($item1, $item2) {          $date1 = intval(str\_replace('-', '', $item1[$this->sort\_field]));          $date2 = intval(str\_replace('-', '', $item2[$this->sort\_field]));          if ($date1 == $date2){              return 0;          }          return ($date1 > $date2 ? -1 : 1 );      }  }    class Product {      public $data = array();      use SortStrategy;      public function get() {          // do something to get the data, for this ex. I just included an array          $this->data = array(              101222 => array('label' => 'Awesome product', 'price' => 10.50, 'date\_added' => '2012-02-01'),              101232 => array('label' => 'Not so awesome product', 'price' => 5.20, 'date\_added' => '2012-03-20'),              101241 => array('label' => 'Pretty neat product', 'price' => 9.65, 'date\_added' => '2012-04-15'),              101256 => array('label' => 'Freakishly cool product', 'price' => 12.55, 'date\_added' => '2012-01-11'),              101219 => array('label' => 'Meh product', 'price' => 3.69, 'date\_added' => '2012-06-11'),          );      }      public function sort\_by($by = 'price', $type = 'asc') {          if (!preg\_match('/^(asc|desc)$/', $type)) $type = 'asc';          switch ($by) {              case 'name':                  $this->sort\_field = 'label';                  uasort($this->data, array('Product', 'string\_'.$type));                  break;              case 'date':                  $this->sort\_field = 'date\_added';                  uasort($this->data, array('Product', 'date\_'.$type));                  break;              default:                  $this->sort\_field = 'price';                  uasort($this->data, array('Product', 'num\_'.$type));              }          }      }    $product = new Product();  $product->get();  $product->sort\_by('date');  ?>  <table border="1" width="100%">      <tr>          <th>SL</th>          <th>Label</th>          <th>Price</th>          <th>Date Added</th>      </tr>      <?php          $i=1;          foreach($product->data as $value){              extract($value);      ?>      <tr>          <td><?php echo $i++; ?></td>          <td><?php echo $label; ?></td>          <td><?php echo $price; ?></td>          <td><?php echo $date\_added; ?></td>      </tr>      <?php      }      ?>  </table> |

**Output**

[](http://w3programmers.com/bangla/php-oop-traits/php-sorting-with-trait/)