Learn with Rabbil

PHP
OBJECT ORIENTED
PROGRAMING

- Class & Object
- Constructor Method
- Access Modifiers
- Inheritance
- Method Overriding, Overloading
- Abstract Classes
- Static Methods,Properties

Learn with Rabbil

PHP OOP

Object-oriented programming has several advantages over procedural programming.

- OOP is faster and easier to execute
- OOP provides a clear structure for the programs
- OOP helps to keep the PHP code DRY "Don't Repeat Yourself", and makes the code easier to maintain, modify and debug
- OOP makes it possible to create full reusable applications with less code and shorter development time

Learn with Rabbil

Class & Object

- A class is a template for objects
- An object is an instance of class.

index.php

```
3   class MyClass{
4
5   }
6
7   $obj = new MyClass;
8
```

Learn with Rabbil

Constructor

A constructor allows you to initialize an object's properties upon creation of the object.

index.php

```php
class MyClass{

    function __construct() {
        echo "I am constructor";
    }
}

$obj = new MyClass();

```

Learn with Rabbil

Constructor Parameter

Passing constructor parameter as same as you function/method parameter you passed before

index.php

```php
class MyClass{
    function __construct($msg) {
        echo $msg;
    }
}

$obj = new MyClass("I am constructor");
```

Learn with Rabbil

PHP - Access Modifiers

Properties and methods can have access modifiers which control where they can be accessed.

public

The property or method can be accessed from everywhere. This is default

protected

The property or method can be accessed within the class and by classes derived from that class

private

The property or method can ONLY be accessed within the class

Learn with Rabbil

Access Modifier

Example 01

public

The property or method can be accessed from everywhere.

index.php

```php
class MyClass{
    public $name="Rabbil";
    protected $age="33";
    private $city="Dhaka";
}

$Obj = new MyClass();
echo $Obj->name;  // OK
echo $Obj->age;  // ERROR
echo $Obj->city; // ERROR
```

Access Modifiers

Example 02

public

The property or method can be accessed from everywhere.

index.php

Learn with Rabbil

```php
class MyClass{

    public function name($name) {
        echo $name;
    }
    protected function age($age) {
        echo $age;
    }
    private function city($city) {
       echo $city;
    }
}

$obj = new MyClass();
$obj->name("Jack");   // Working Fine
$obj->age("30");      // Error
$obj->city("London");  // Error
```

Learn with Rabbil

Access Modifiers

private

The property or method can ONLY be accessed within the class

index.php

```php
class MyClass{

    private $num1=20;
    private $num2=30;

    public function addTwo() {
        $num3=$this->num1+$this->num2;
        echo $num3;
    }
}

$obj = new MyClass();
$obj->addTwo();

```

Access Modifiers

protected

The property or method can be accessed within the class and by classes derived from that class

index.php

```php
class MyClass{

    protected $num1=20;
    protected $num2=30;

    public function addTwo() {
        $num3=$this->num1+$this->num2;
        echo $num3;
    }
}

$obj = new MyClass();
$obj->addTwo();
```

Learn with Rabbil

Static Methods

Static methods can be called directly - without creating an instance of the class first.

index.php

```
3    class MyClass{
4        static function addTwo() {
5            $num1=20; $num2=30;
6            $num3=$num1+$num2;
7            echo $num3;
8        }
9    }
10
11   MyClass::addTwo();
12
```

Learn with Rabbil

Static Properties

Static properties can be called directly - without creating an instance of a class.

index.php

```
3   class MyClass{
4       public static $cityList = array(
5           'Dhaka',
6           'Rangpur',
7           'Rajshahi'
8       );
9   }
10  echo MyClass::$cityList[1]
```

Learn with Rabbil

Why Static

01

A static method belongs to the class rather than the object of a class.

02

When we no need to use OOP features like Inheritance/ Overriding/ Overloading etc.

03

Static is used for utility action preposes

03

Static is used for utility action preposes

Learn with Rabbil

Static Features

- Static methods can be accessed directly in static and non-static methods.
- We can call other static methods inside another static methods.
- We can access static methods directly inside a non static methods.
- We can only access a static variables inside a static function.
- Non static methods and variables cannot be accessed inside a static method, it throws a compile time error to change the modifier to static.
- We can also access static methods inside a static block.
- Cannot use 'this' keyword in static.
- Static methods cannot be overridden , but static methods can be overloaded.

Learn with Rabbil

Inheritance Concepts

It is a concept of accessing the features of one class from another class. If we inherit the class features into another class, we can access both class properties. We can extends the features of a class by **using 'extends' keyword.**

- It supports the concept of hierarchical classification.
- Inheritance has three types, single, multiple and multilevel Inheritance.
- PHP supports only single inheritance, where only one class can be derived from single parent class.

Inheritance Concepts

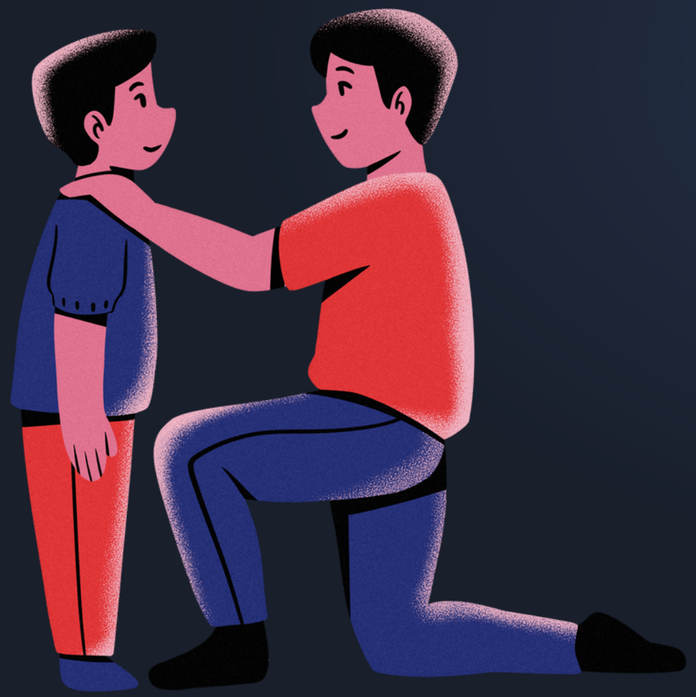
index.php

```php
class Father{

}


class Son extends Father{

}
```

Inheritance

Example 01

How father's method can use his son directly



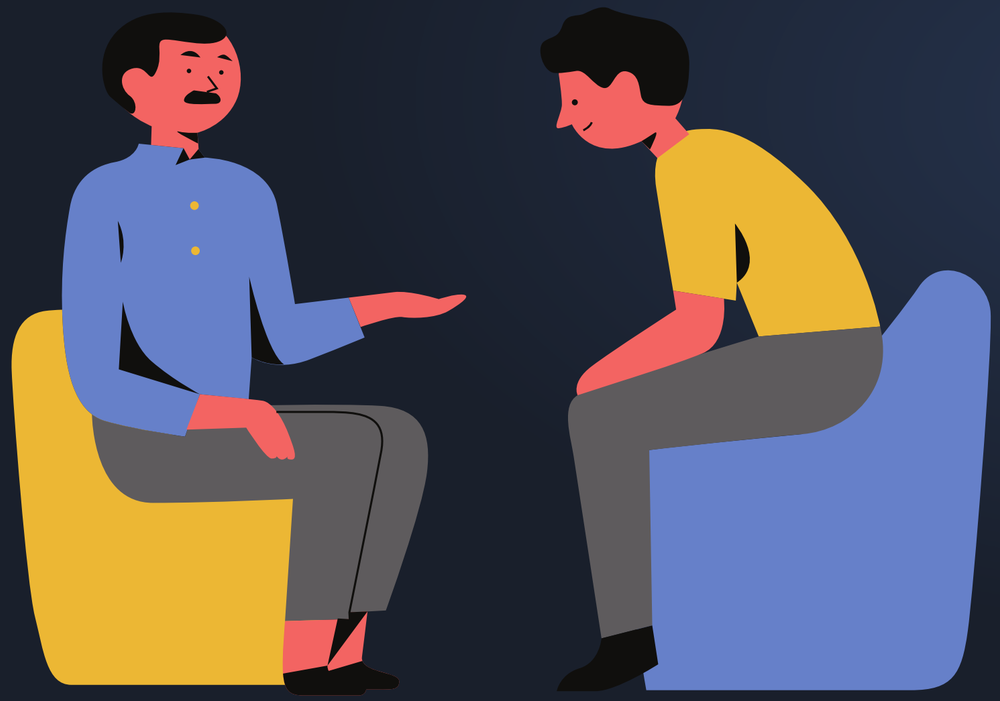
index.php

Learn with Rabbil

```
3   class Father{
4       function addTwo(){
5        $num1=10;
6        $num2=10;
7        $num3= $num1+$num2;
8        echo $num3;
9       }
10  }
11
12  class Son extends Father{
13
14  }
15
16  $obj=new Son();
17  $obj->addTwo();
```

Inheritance Override

Son can modify or change father's properties



index.php

Learn with Rabbil

```php
class Father{
    function addTwo(){
     $num1=10; $num2=10;
     $num3= $num1+$num2;
     echo $num3;
    }
}
class Son extends Father{
     // Method overriding
     function addTwo(){
          $num1=30; $num2=40;
          $num3= $num1+$num2;
          echo $num3;
     }
}
$obj=new Son();
$obj->addTwo();
```

Abstract Class

We can create direct object when any class declared as abstract

index.php

Learn with Rabbil

```php
// Class Abstract
abstract class Father{
    function addTwo(){
     $num1=10; $num2=10;
     $num3= $num1+$num2;
     echo $num3;
    }
}
class Son extends Father{

}
// This is correct
$obj2=new Son();
$obj2->addTwo();

// This will error
$obj1=new Father();
$obj1->addTwo();
```

Learn with Rabbil

PHP Namespaces

Namespaces are qualifiers that solve two different problems:

01

They allow for better organization by grouping classes that work together to perform a task

02

They allow the same name to be used for more than one class

PHP Namespaces

PHP Namespaces are the way of encapsulating items so that same names can be reused without name conflicts

index.php

Learn with Rabbil

```php
namespace Namespace1 {

    class MyClass{

        function addNumber(int $num1,int $num2){
         echo $num1+$num2;
        }

     }
}


namespace Namespace2 {

    class MyClass{

        function addNumber(int $num1,int $num2){
         echo $num1+$num2;
        }

     }
}
```