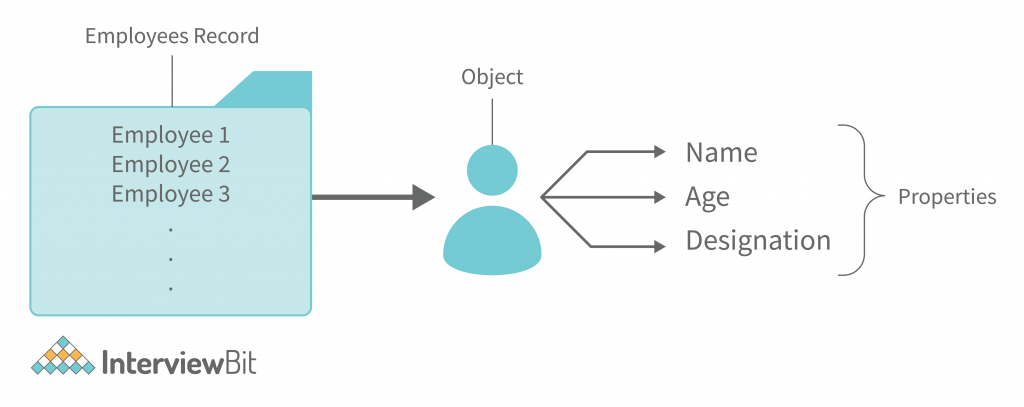
* **What is Object Oriented Programming?**

Object-oriented programming (OOP) is a software design approach that focuses on breaking large programs into smaller, more manageable components called objects. This allows for easier maintenance and improved program organization. OOP enables applications to be more flexible and extensible. It also promotes code reuse, which helps to reduce development costs. One of the key benefits of object-oriented programming is that it makes it easier to write modular code. Because the individual parts of an application are independent objects, it’s possible to write code that can be reorganized and reused without affecting other parts of the program. This makes it possible to build complex applications with fewer lines of code than would otherwise be required. Another advantage of OOP is that it allows an application to be more flexible and extensible. Because each component of an application is an object, it’s possible to make changes to one part without having to update or redesign the entire program. This means that applications can be built with a greater degree of customization than they could otherwise be.

Finally, OOP promotes code reuse. Because each component is an object, it’s possible to reuse existing code in new applications without having to write it from scratch. This reduces development costs and makes it easier for developers to build complex applications with fewer lines of code than would otherwise be required.

**Example:**



An example of an employee record is provided above where each employee can be viewed as an “object”. Since every employee has a name, age, and job role, these can be considered the “properties” of that employee.

* **What Are Properties Of Object Oriented Systems?**

**Classes**

In the oops concept, a class is a construct that is used to describe an individual type. The class is instantiated into instances of itself – referred to as class instances or simply objects. A class defines ingredient members that allow its instances to have position and behavior. Member variables or instance variables facilitate a class instance to maintain its position. On the other hand, other kinds of members, especially methods, allow the behavior of class instances. Simply classes consequently define the type of their instances. A class usually represents a person, place or thing, or something.

For example, a “Bird” class would symbolize the properties and functionality of birds. A single, particular bird would be an instance of the “Bird” class, an object of the type “Bird”. There is a set of access specifiers in classes. private (or class-private) specifiers restrict the entrance to the class itself. Only the methods that are elements of a similar class only can access private members. protected (or class-protected) specifies enables the class itself and all classes under it (sub-classes) to access the member and public means that member can be accessed by its name using any code.

**Objects**

An object is a self-contained segment with the attributes and processes needed to make data usable in programming terms. From an object-oriented perspective, objects are the main building pieces of programs. In each application you create, you may employ a variety of objects of various sorts. Each kind of object is derived from a specific class of that type. Consider an object to be a sculpt of the real-world perceptions, processes, or objects that are important to the application you’re designing.

A variable, function, or data structure may all be considered an object. The term “object” in object-oriented programming refers to a specific instance of a class. Objects are used in software development to combine data components with methods that alter them, allowing for the usage of abstract data structures. Objects in object-oriented programming are answers to the idea of inheritance, resulting in improved program dependability, simpler software maintenance, library administration, and task division in programmer teams. Of basic terms, “Objects” are the fundamental data types in object-oriented programming languages and are used to build object-oriented programming.

**Inheritance**

In layman’s terms, the attributes that you inherit from your parents are a simple illustration of inheritance. Classes may inherit characteristics from other classes thanks to inheritance. Parent classes, in other words, extend properties and behaviors to child classes. Reusability is aided via inheritance. Prototyping is another name for inheritance in JavaScript. A prototype object serves as a base from which another object may derive its features and actions. Thus, you may use multiple prototype object templates to form a prototype chain. Inheritance is passed down from one generation to the next parent.

Consider the application Polygon, which represents several Shapes. We’re expected to make two distinct sorts of polygons: a Rectangle and a Triangle.

**Encapsulation**

Encapsulation is the process of enclosing all critical information inside an object and only revealing a subset of it to the outside world. For example, code inside the class template defines attributes and behaviours.

The data and methods are then enclosed in the object when it is created from the class. Inside a class, encapsulation conceals the underlying software code implementation and the internal data of the objects. Encapsulation necessitates designating certain fields as private while others are made public.

* Methods and attributes only available from other methods in the same class make up the private/internal interface.
* Methods and attributes that are available from outside the class are known as the public / external interface.

**Encapsulation Demonstration in Real-Time**

One of the most practical examples of encapsulation is a school bag. Our books, pencils, and other items may be kept in our school bag.

The following are some of the advantages of encapsulation:

* **Data Hiding:** In this case, the user will be unaware of the class’s internal implementation. Even the user will have no idea how the class stores data in variables. He or she will only be aware that the values are sent to a setter method and that variables are initialised with that value.
* **Increased Flexibility:** Depending on our needs, we may make the variables of the class read-only or write-only. If you want to make the variables read-only, remove the setter methods like setName(), setAge(), and so on from the above programme. If you want to make the variables write-only, remove the get methods like getName(), getAge(), and so on from the above programme.
* It also promotes**reusability** and makes it simple to alter to meet new needs.

**Abstraction**

Abstraction refers to the user’s interaction with just a subset of an object’s characteristics and operations. To access a complicated item, abstraction uses simpler, high-level techniques.

* Simple items are used to show complexity.
* Keep complicated information hidden from the user.

Simple classes are used to indicate complexity in abstraction. Encapsulation is an extension of abstraction.

**A Real-Life Example of Abstraction**

Abstraction reveals just the most significant facts to the user while hiding the underlying intricacies. For example, when we ride a bike, we only know how to ride it but not how it works. We also have no idea how a bike works on the inside.

**Advantages of Abstraction**

* It simplifies the process of seeing things in their entirety.
* Code duplication is avoided, and reusability is increased.
* Because just the most necessary information is shown to the user, it helps to enhance the security of an application or software.

**Polymorphism**

Polymorphism refers to the creation of items that have similar behavior. For example, objects may override common parent behaviors with particular child behaviors through inheritance. Method overriding and method overloading are two ways that polymorphism enables the same method to perform various actions.

Examine how Polymorphism and the actual world are interconnected with examples.

Take, for example, your mobile phone. It has the capability of storing your Contacts. Consider the following scenario: you wish to store two numbers for one individual. You may do this by storing the second number under the same name as the first.

Consider the following scenario: you wish to store two numbers for the same individual in an object-oriented language such as Java. Create a function that will accept as arguments two integers and the name of the individual to some function void createContact that will be defined later (String name, int number1, int number2).

**Method Overriding**

Method overriding is used in runtime polymorphism. When a child class overrides a parent class’s method, the child class might offer an alternative implementation.

Consider a family of three, consisting of the father, mother, and son. The father makes the decision to teach his kid to shoot. As a result, he brings him to the range with his favorite rifle and teaches him how to aim and fire at targets. The father, on the other hand, is right-handed, while the kid is left-handed. So they each have their own way of handling the pistol! Because of their differing orientations, the father was concerned that he may not be able to teach his son how to shoot.

The son, on the other hand, was astute and chose to flip his father’s hands, putting his dominant hand on the trigger rather than the father’s. Specifically, the right hand. By significantly changing the learning process, the son was able to grasp the skill of shooting!

Method overriding is the term used in programming to describe this idea.

**Method Overloading**

Method overloading is used in Compile Time Polymorphism. Although two methods or functions may have the same name, the number of arguments given into the method call may vary. Therefore, depending on the number of parameters entered, you may obtain different results.

With the help of a simple example, it may be comprehended in simple words. A class addition contains two add() methods, one with arguments int a and int b and the other with three integer parameters, int a, int b, and int c. As a result, the add() function is considered overloaded.

The amount of arguments given in the method calling statement determines which method is performed. For example, add(20,30) calls the two-parameter add() function, whereas add(10,20,30) calls the three-parameter add method.

**Constructors and Destructors**

Constructors in most object-oriented languages have the same name as the class and are public. Constructors may be overloaded, which means that multiple argument lists can be used with the same name. The function Object() { [native code] } in PHP 5.0 is the function \_construct (). Normally, attribute values would be initialised in a function Object() { [native code] }. The \_destruct() method is optional, although it might be used to implement code that cleans up once an object is destroyed, such as shutting files or database connections.

**OOP Advantages**

* Complex things are modeled as repeatable, basic structures in OOP.
* Thus, OOP objects are reusable and may be utilized in several applications.
* Modularity for easier troubleshooting.
* Classes are easier to debug since they generally include all relevant information.
* Reuse of code through inheritance.
* **What Is Difference Between Class And Interface?**

To understand the difference between class and interface lets understand first class and interface.

## What is Class?

In OOP, everything is considered as an object. It is not possible to create an object without a class. A class is a blueprint to create an object. When building a house, the architect draws the plan. The plan is similar to a class. The house is similar to the object. The class is the plan to build an object. An object is what is created using the class.

The class contains the properties and methods. A student can have properties such as name, grade, index number. A student can have methods such as reading, walking, studying. A class is created with the necessary properties and methods.

The syntax for creating a class in many programming languages is as follows. It is created using the keyword class.

class class\_name {

// properties

//methods

}

Programming languages such as [C#](https://www.differencebetween.com/difference-between-c-and-vs-c-sharp/) and [Java](https://www.differencebetween.com/difference-between-java-and-c-language/) follow a similar syntax to create an object using a class. Assume that the class name is Student.

Student s1= new Student ();

This s1 is the object. The “new” keyword is used to allocate memory for the properties. A class also has a constructor to initialize properties or variables.

Class members such as properties and methods have access modifiers. Access specifiers describe the accessibility and visibility of those members to other classes. Members of the class can have access specifiers such as public, private and protected. Public members are accessible by other classes. Private members are only accessible to the class. Protected members are accessible within the class and relevant subclasses.

## What is Interface?

[Abstraction](https://www.differencebetween.com/difference-between-encapsulation-and-vs-abstraction/) is a pillar of Object Oriented programming. It is to hide the implementation details and to display the functionality to the user. Abstraction is achieved using [abstract classes](https://www.differencebetween.com/difference-between-abstract-class-and-vs-concrete-class/) and interfaces. An abstract method does not have an implementation. A class that contains at least one abstract method is called an abstract class.

When there are two abstract classes, the methods declared in those classes should be implemented. A new class is used to implement those methods. If both classes had the same method, it might cause an ambiguity problem. Therefore, programming languages such as Java and C# have an interface.

Interfaces contain only the declaration of methods. There is no method of implementation. Also, interfaces cannot be used to create objects. They are used to support multiple [inheritances](https://www.differencebetween.com/difference-between-inheritance-and-vs-containership/)and to secure the code.

The syntax of Interface is as follows. Interface use the keyword “interface”.

interface interface\_name{

type method1(parameter\_list);

type method2(parameter\_list);

}

According to above, interfaces only have the declaration. There is no definition. So, interfaces cannot instantiate objects. It only gives an abstract view of what the interface is. Methods declared in the interface can be implemented by one or many classes. A class use the keyword “implement” to implement an interface. Refer below example written using Java.



**Figure 01: Program using Interfaces**

According to the above program, A and B are interfaces. Interface A has a method declaration which is the sum(). Interface B has a method declaration sub(). Class C is implementing both interfaces which are A and B. Therefore, class C define both sum() and sub() methods. After creating the object of type C, it is possible to call both methods sum() and sub().

Methods declared inside the interface must always be public because the implementing classes define them. An interface can also inherit from another interface.

## What are the Similarities Between Class and Interface?

* Both are reference types.
* Both relate to Object-Oriented Programming.

## What is the Difference Between Class and Interface?

|  |  |
| --- | --- |
| Class vs Interface | |
| A class is a reference type that is a blueprint to create an object. | An interface is a reference type that cannot be instantiated. |
| **Object Instantiation** | |
| A class is used to instantiate an object. | An interface cannot be instantiated because the methods are unable to perform any action. |
| **Constructor** | |
| A class contain a constructor, to initialize the variables. | An interface does not contain a constructor because they are hardly any variables to initialize. |
| **Keyword** | |
| A class uses the keyword “class”. | An interface uses the keyword “interface”. |
| **Access Specifier** | |
| Members of the class can be private, public and protected. | Members of the interface should be always public because the implementing classes define them. |

* **What Is Overloading?**

Method overloading is used in Compile Time Polymorphism. Although two methods or functions may have the same name, the number of arguments given into the method call may vary. Therefore, depending on the number of parameters entered, you may obtain different results.

With the help of a simple example, it may be comprehended in simple words. A class addition contains two add() methods, one with arguments int a and int b and the other with three integer parameters, int a, int b, and int c. As a result, the add() function is considered overloaded.

The amount of arguments given in the method calling statement determines which method is performed. For example, add(20,30) calls the two-parameter add() function, whereas add(10,20,30) calls the three-parameter add method.

* **What Is T\_PAAMAYIM\_NEKUDOTAYIM (Scope Resolution Operator (::) with Example?**

The scope resolution operator also known as *Paamayim Nekudotayim* or more commonly known as the double colon is a token that allows access to static, constant, and overridden properties or methods of a class.

It is used to refer to blocks or codes in context to classes, objects, etc. An identifier is used with the scope resolution operator. The most common example of the application of the scope resolution operator in PHP is to access the properties and methods of the class.

The following examples show the usage of the scope resolution operator in various scenarios.

<?php

// Declaring parent class

class demo{

    public static $bar=10;

    public static function func(){

        echo static::$bar."\n";

    }

}

// Declaring child class

class Child extends demo{

    public static $bar=20;

}

// Calling for demo's version of func()

demo::func();

// Calling for child's version of func()

Child::func();

?>

* **What are the differences between abstract classes and interfaces?**

|  |  |
| --- | --- |
| Abstract class | Interface |
| * Abstract class comes under partial abstraction. * Abstract classes can maintain abstract methods and non-abstract methods. * In abstract classes, we can create the variables. * In abstract classes, we can use any access specifier. * By using 'extends' keyword we can access the abstract class features from derived class.   Multiple inheritance is not possible. | * Interface comes under fully abstraction. * Interfaces can maintain only abstract methods. * In interfaces, we can't create the variables. * In interface, we can use only public access specifier. * By using 'implement' keyword we can get interface from derived class. * By using interfaces multiple inheritance is possible. |

* **Define Constructor and Destructor?**

**Constructors:** Constructors are special member functions for initial settings of newly created object instances from a class, which is the key part of the object-oriented concept in **PHP5**.  
Constructors are the very basic building blocks that define the future object and its nature. You can say that the Constructors are the blueprints for object creation providing values for member functions and member variables.  
Once the object is initialized, the constructor is automatically called. Destructors are for destroying objects and automatically called at the end of execution.  
In this article, we are going to learn about object-oriented concepts of constructors and destructors.   
Both are special member functions of any class with different concepts but the same name except destructors are preceded by a **~ Tilda operator**.  
**Syntax:**

**\_\_construct():** 

function \_\_construct()

{

// initialize the object and its properties by assigning

//values

}

**Destructor:** Destructor is also a special member function which is exactly the reverse of constructor method and is called when an instance of the class is deleted from the memory. Destructors (\_\_destruct ( void): void) are methods which are called when there is no reference to any object of the class or goes out of scope or about to release explicitly.   
They don’t have any types or return value. It is just called before de-allocating memory for an object or during the finish of execution of PHP scripts or as soon as the execution control leaves the block.   
Global objects are destroyed when the full script or code terminates. Cleaning up of resources before memory release or closing of files takes place in the destructor method, whenever they are no longer needed in the code. The automatic destruction of class objects is handled by PHP Garbage Collector.

**Syntax:**

function \_\_destruct()

{

// destroying the object or clean up resources here

}

* **How to Load Classes in PHP?**

PHP load classes are used for declaring its object etc. in object oriented applications. PHP parser loads it automatically, if it is registered with spl\_autoload\_register() function. PHP parser gets the least chance to load class/interface before emitting an error.

**Syntax:**

spl\_autoload\_register(function ($class\_name) {

include $class\_name . '.php';

});

* **How to Call Parent Constructor?**

We will face two cases while calling the parent constructor method in child class.

## Case1

We can't run directly the parent class constructor in child class if the child class defines a constructor. In order to run a parent constructor, a call to parent::\_\_construct() within the child constructor is required.

## Example

<?php

   class grandpa{ //----------------Parent constructor----------------

      public function \_\_construct(){

         echo "I am in Tutorials Point"."";

      }

   }

   class papa extends grandpa{ //--------Child constructor that uses properties of parent function--

      public function \_\_construct(){

         parent::\_\_construct();

         echo "I am not in Tutorials Point";

      }

   }

$obj = new papa();

?>

**Output:**

I am in Tutorials Point

I am not in Tutorials Point

## Explanation

In the above example, we have used parent::\_\_construct() to call the parent class constructor.

## Case2

If the child does not define a constructor then it may be inherited from the parent class just like a normal class method(if it was not declared as private).

## Example

<?php

   class grandpa{

      public function \_\_construct(){

         echo "I am in Tutorials point";

      }

   }

   class papa extends grandpa{

   }

   $obj = new papa();

?>

* **Are Parent Constructor Called Implicitly When Create An ObjectOf Class?**

Parent constructors are not called implicitly if the child class defines a constructor. In order to run a parent constructor, a call to parent::\_\_construct() within the child constructor is required. If the child does not define a constructor then it may be inherited from the parent class just like a normal class method (if it was not declared as private).

* **What Happen, If Constructor Is Defined As Private Or Protected?**

If a constructor is declared as private, then its objects are only accessible from within the declared class. You cannot access its objects from outside the constructor class.

* **What are PHP Magic Methods/Functions? List them**

Following are PHP Magic Methods/Functions

\_\_construct(), \_\_destruct(), \_\_call(), \_\_callStatic(), \_\_get(), \_\_set(), \_\_isset(), \_\_unset(), \_\_sleep(), \_\_wakeup(), \_\_serialize(), \_\_unserialize(), \_\_toString(), \_\_invoke(), \_\_set\_state(), \_\_clone(), and \_\_debugInfo().

* **Write program for Static Keyword in PHP?**

The static keyword is used to declare properties and methods of a class as static. Static properties and methods can be used without creating an instance of the class.

The static keyword is also used to declare variables in a function which keep their value after the function has ended.

**Example:-**

<?php

function add1() {  
  static $number = 0; //------------Static Variable------------  
  $number++;  
  return $number;  
}  
  
echo add1();  
echo "<br>";  
echo add1();  
echo "<br>";  
echo add1();  
?>

* **Create multiple Traits and use it in to a single class?**

<?php

trait abc //----Traits are used to reduce reusable code…

{

public function test()

{

echo "This is test method";

}

}

trait xyz

{

public function sample()

{

echo "this is sample method";

}

}

class c

{

use abc,xyz; //---------------------------- Multiple inheritance-------------------------

}

$obj=new c();

$obj->test();

$obj->sample();

?>

Write PHP Script of Object Iteration?

<?php

class myclass{

   private $var;

   protected $var1;

   public $x, $y, $z;

   public function \_\_construct(){

      $this->var="private variable";

      $this->var1=TRUE;

      $this->x=100;

      $this->y=200;

      $this->z=300;

   }

   public function iterate(){

      foreach ($this as $key => $value) {

         print "$key => $value  
";

      }

   }

}

$obj = new myclass();

foreach($obj as $key => $value) {

   print "$key => $value  
";

}

echo "  
";

$obj->iterate();

?>

Output:-

The output is as below −

x => 100

y => 200

z => 300

var => private variable

var1 => 1

x => 100

y => 200

z => 300

* **Use of The $this keyword**

**$this** is a reserved keyword in PHP that refers to the calling object. It is usually the object to which the method belongs, but possibly another object if the method is called statically from the context of a secondary object. This keyword is only applicable to internal methods.

**Example 1:** A simple program to show the use of *$this* in PHP.

<?php

class simple{

    public $k = 9;

    public function display(){

        return $this->k; //------------------------- $this keyword---------------

    }

}

$obj = new simple();

echo $obj->display();

?>

**Output:**

9

**$this – a pseudo-variable:** Unlike other reserved keywords used in the context of class like the static, parent, etc does not need to be declared with the dollar sign (‘$’). This is because in PHP *$this* is treated as a pseudo-variable.  
In PHP, this is declared like a variable declaration (with the ‘$’ sign) even though it is a reserved keyword. More specifically, *$this* is a special read-only variable that is not declared anywhere in the code and which represents a value that changes depending on the context of program execution.