MEC ACADEMY

Object oriented programming

MEC Academy

Agenda

- What is OOP ?
- Procedural programming
- What is the class and object ?
- OOP Principles
  - Inheritance
  - Encapsulation
- Static and const keywords
- Self and this keywords

MEC ACADEMY

What is OOP ?

- **object-oriented programming:** is a programming paradigm based on the concept of object
- A **programming paradigm:** is a style of programming , a way of thinking about software construction
- **programming paradigm** : does not refer to a specific language but rather to a way to build a program or a methodology to apply
- some programming languages allow the programmer apply more than one paradigm

Procedural Programming

Procedural programming (PP) :

- also known as inline programming takes a top-down approach. It is about writing a list of instructions to tell the computer what to do step by step. It relies on procedures or routines

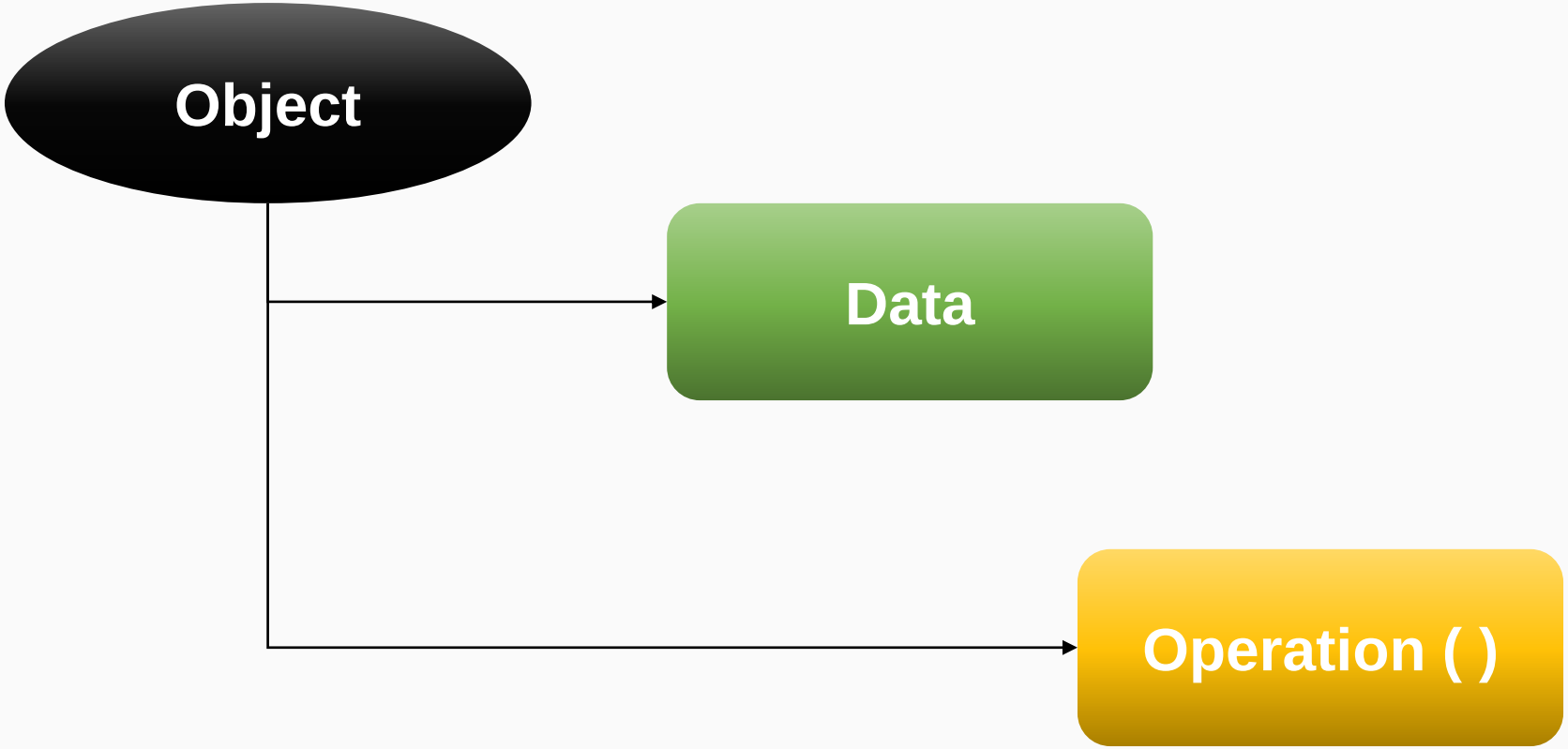
```php
<?php
// Function to calculate the sum of two numbers
function add($num1, $num2) {
    return $num1 + $num2;
}

// Function to display a message
function displayMessage($message) {
    echo $message;
}

// Main program
$num1 = 10;
$num2 = 5;

$result = add($num1, $num2);
displayMessage("The sum is: " . $result);
?>
```

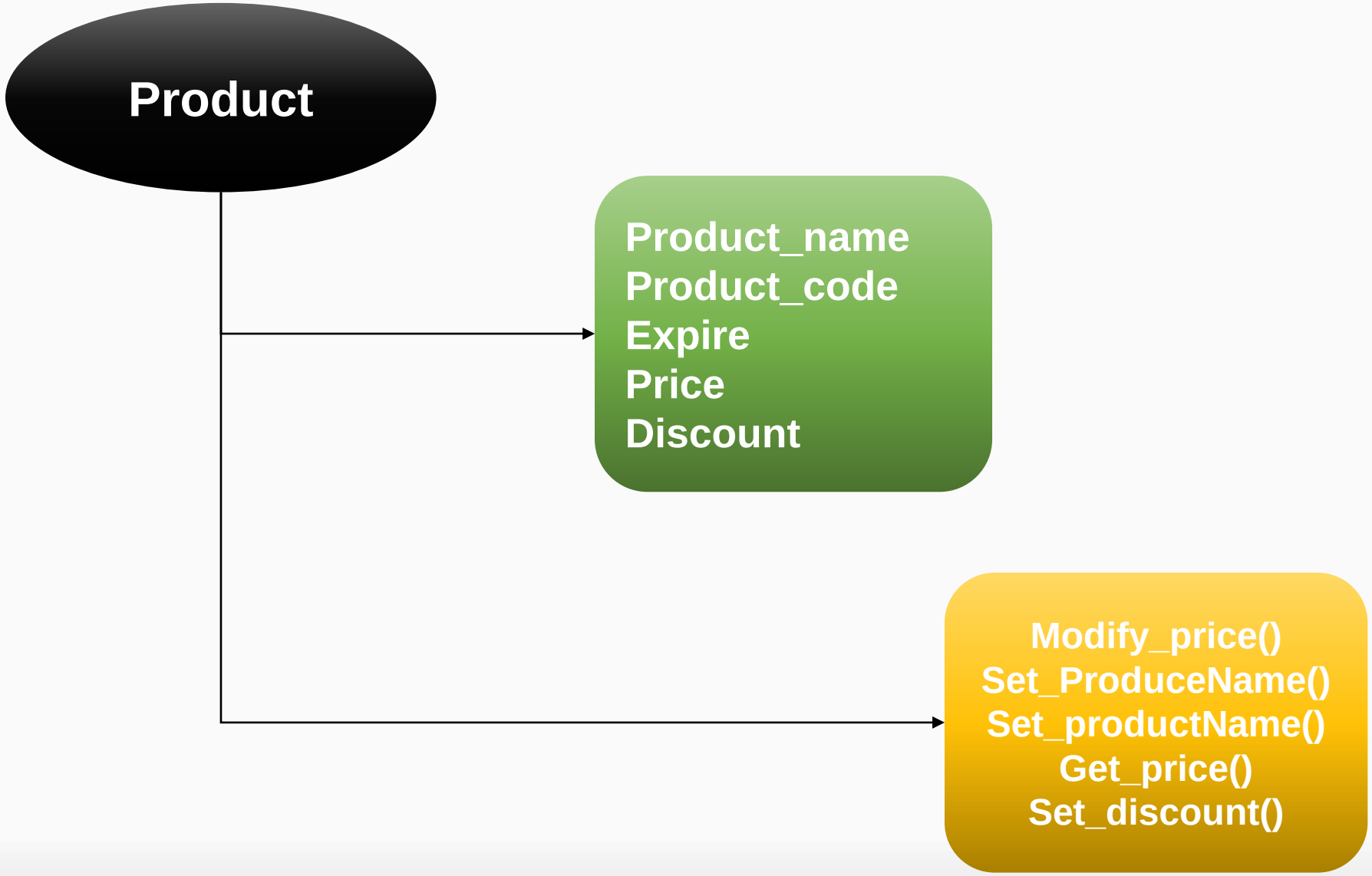
MEC ACADEMY

❑ **object-oriented programming:** is a programming paradigm based on the concept of object

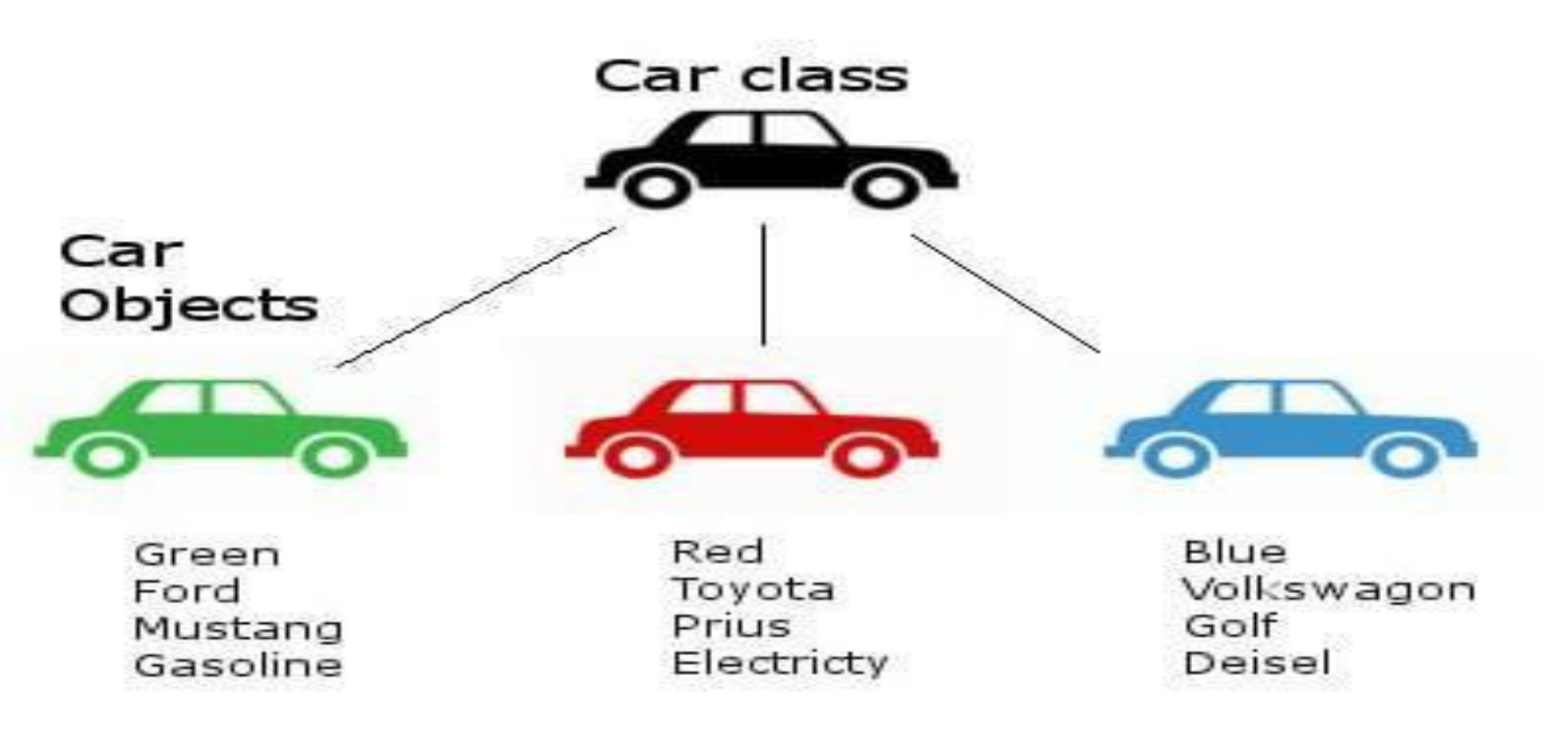
MEC ACADEMY

Object comprised of




Product
Product_name
Product_code
Expire
Price
Discount
Modify_price()
Set_ProduceName()
Set_productName()
Get_price()
Set_discount()

What is Class ?



MEC ACADEMY

Class and Object

Class

Car

Object

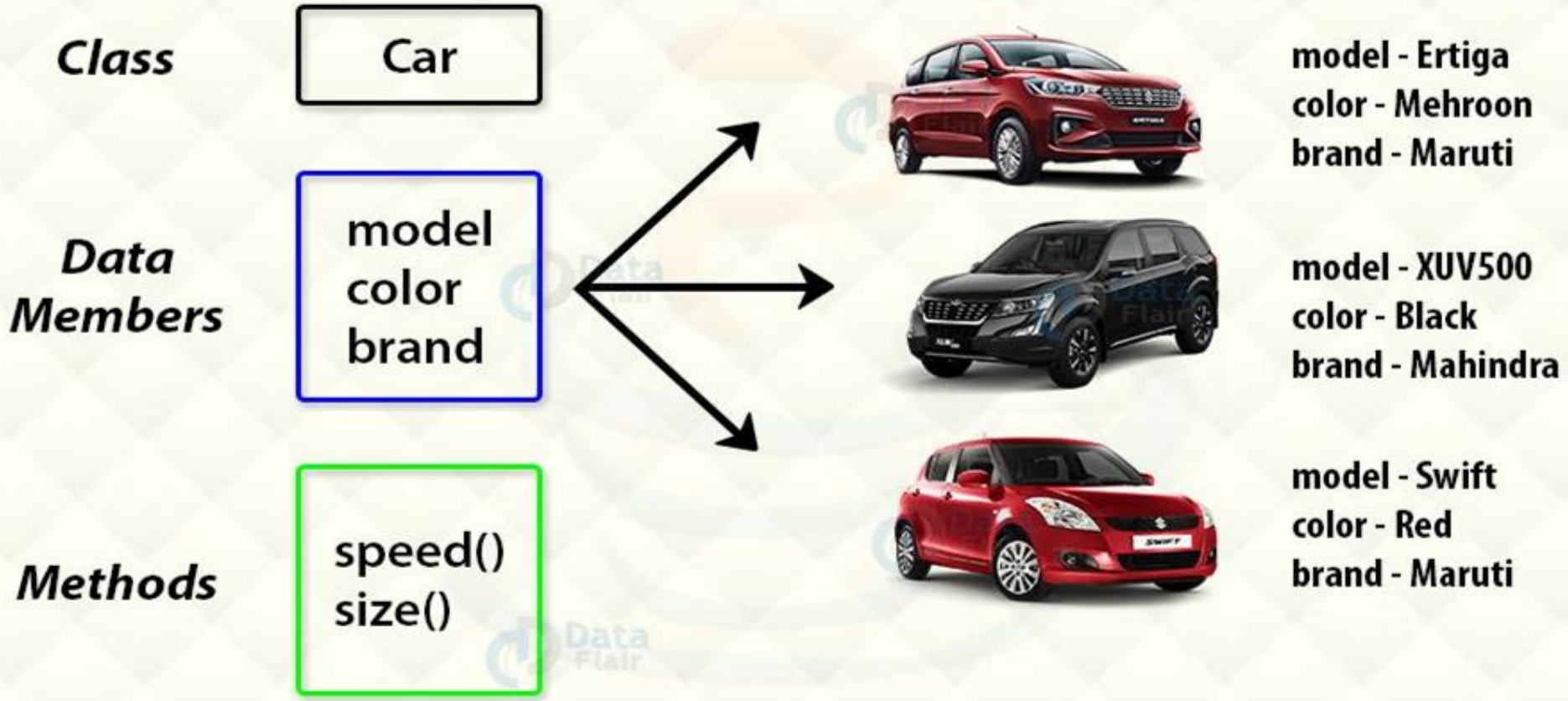
Lamborghini

Ferrari

Electric

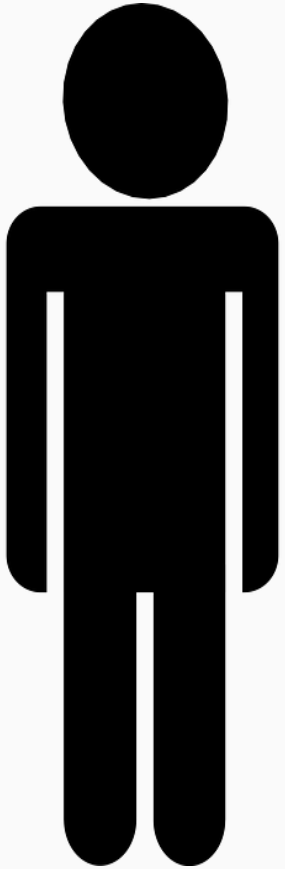
MEC ACADEMY

Class and Object



Class

Man



Object

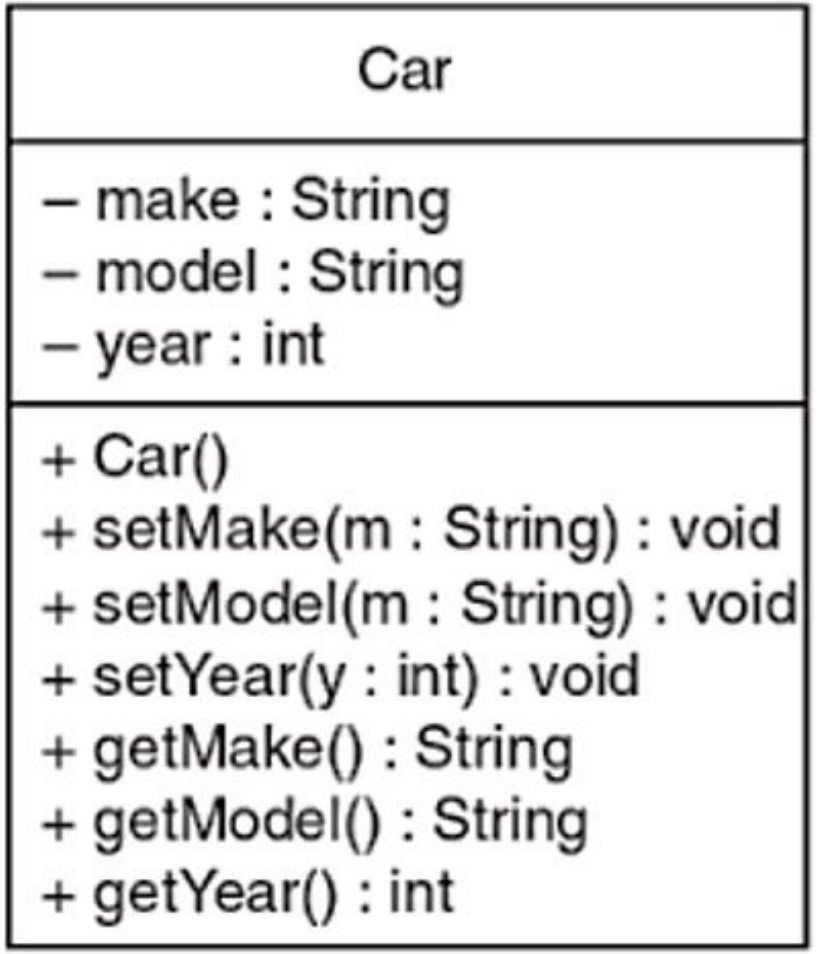
Mo Salah

Object

محمد بن احمد

MEC ACADEMY

UML Digram



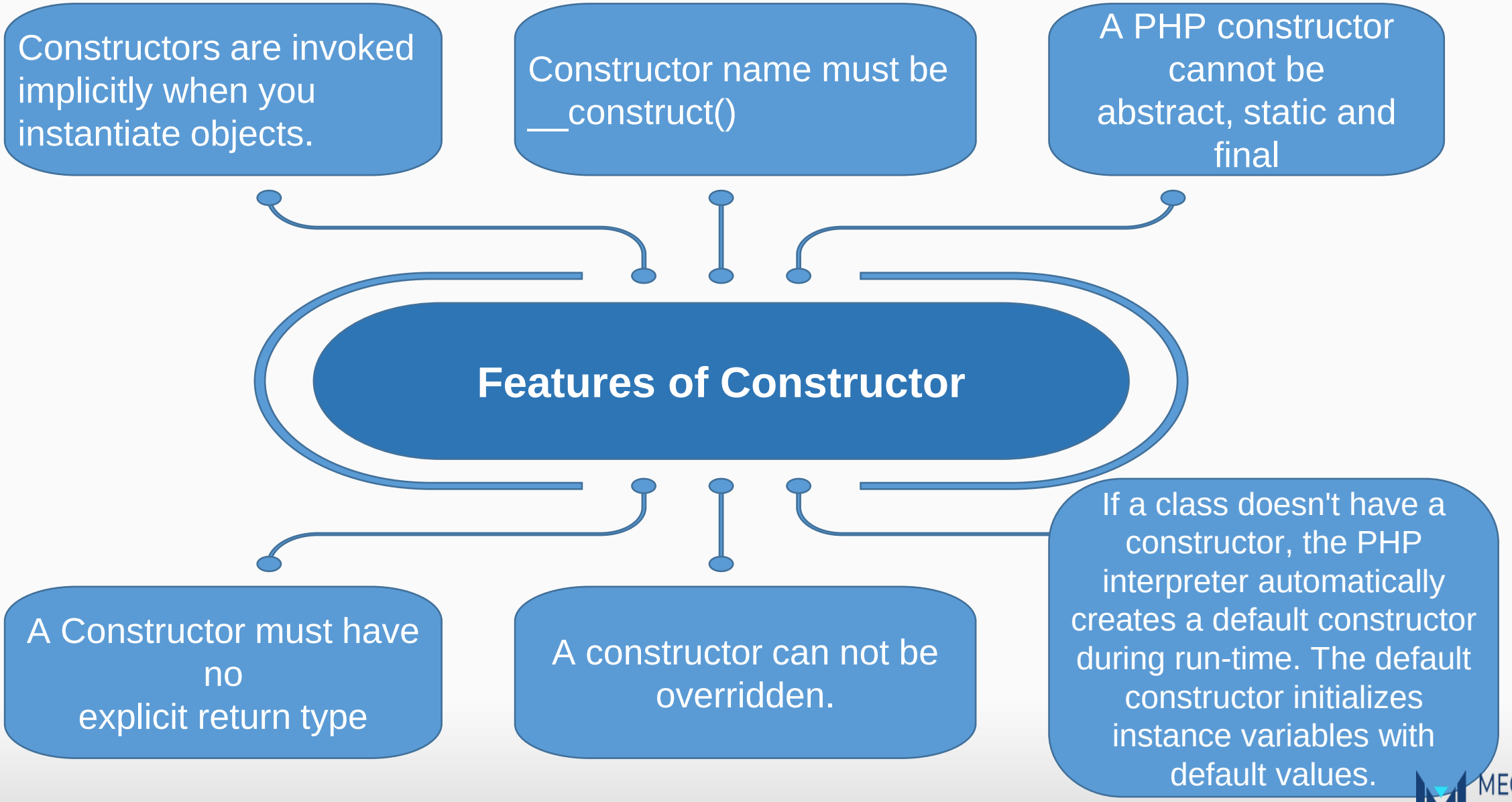
Constructor

A constructor is a method that is automatically called when an object is created

If we talk about a Person class then it will have some class variables (say weight, hair color, and height).

But when it comes to creating its object(i.e Box will now exist in computer’s memory), then can a box be there with no value defined for its dimensions. The answer is no.

So constructors are used to assign values to the class variables at the time of object creation, either explicitly done by the programmer or by PHP itself (default constructor).


Constructors are invoked implicitly when you instantiate objects.
Constructor name must be __construct()
A PHP constructor cannot be abstract, static and final
Features of Constructor
A Constructor must have no explicit return type
A constructor can not be overridden.
If a class doesn't have a constructor, the PHP interpreter automatically creates a default constructor during run-time. The default constructor initializes instance variables with default values.

Features of Constructor

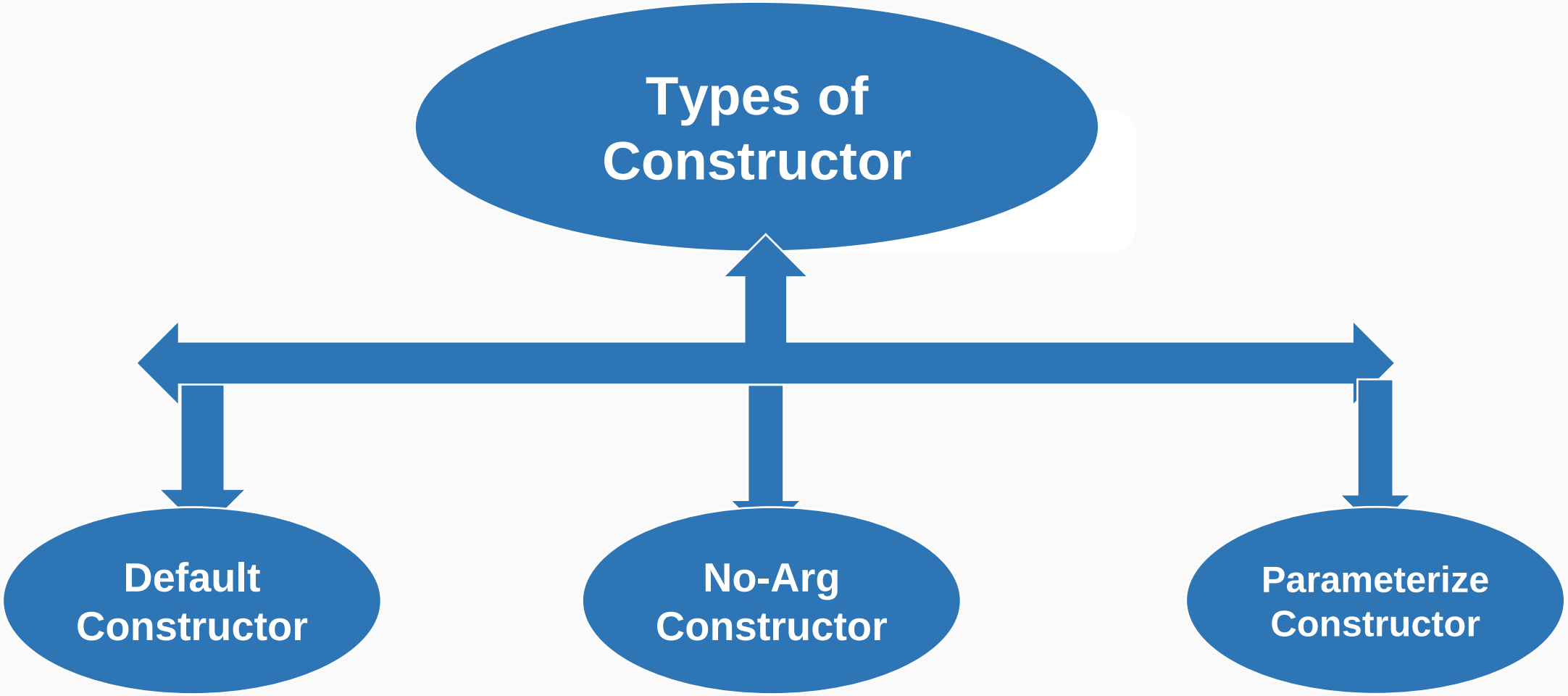
- When is a Constructor called ?

In PHP, a constructor is called when an object is created from a class. It is a special method within a class that has the same name as the class itself and is used to initialize the object's properties or perform any necessary setup tasks.

The constructor is automatically invoked whenever an object is instantiated using the new keyword. When the constructor is called, it allocates memory for the object and initializes its properties according to the code defined within the constructor.

```php
class MyClass {
    public $name;

    // Constructor
    public function __construct($name) {
        $this->name = $name;
        echo "Constructor called!";
    }
}

$obj = new MyClass("John");
```

MEC ACADEMY


Types of Constructor
Default Constructor
No-Arg Constructor
Parameterize Constructor

Default Constructor

When an object is created, its constructor is always called. But what if we do not write a constructor in the object's class?

If you do not write a constructor in a class, PHP automatically provides one when the class is compiled. The constructor that PHP provides is known as the default constructor.

The default constructor doesn't accept arguments

The only time that PHP provides a default constructor is when you do not write your own constructor for a class.

Integer Num is zero (0).

Floating Num is zero (0.0f, 0.0d).

Object or Reference variables is null.

Rectangle r = new Rectangle(); // Calls the default constructor

No-Arg Constructor

Constructor with no arguments ,The signature is same as default constructor, however body can have any code unlike default constructor

where the body of the constructor is empty,

constructor doesn’t accept arguments, so it is considered a no-Arg constructor. In addition,

you can write your own no-Arg constructor. For example, suppose we wrote the following

constructor for the Rectangle class:

```
Public function  __constract ()
{
$this->length = 1.0;
$this->width = 1.0;
}
```

MEC ACADEMY

Parameterize Constructor

Constructors can also take parameters, which is used to initialize attributes.

The following example adds an int y parameter to the constructor. Inside the constructor we set x to y (x=y). When we call the constructor, we pass a parameter to the constructor (5), which will set the value of x to 5

```
Public function __constract ($Len, $w) {
 $this->length= $Len;
  $this->Width= $w;}
//Call code
Rectangle r = new Rectangle (  5  ,   2.4  );
```

Special Properties

Constructors have a few special properties that set them apart from normal method

- Constructors have the __construct name
- Constructors have no return type(not even void)
- Constructors may not return any values
- Constructors are typically public but can be private ex: in **single tone design pattern**

OOP Principles

OOP Principles

1) Encapsulation
2) Inheritance
3) Abstraction
4) Polymorphism

MEC ACADEMY

Encapsulation

Encapsulation

Encapsulation : involves bundling data variables and their related functions within a single unit, or class, shielding them from external interference. Rather than direct data access, PHP encapsulates attributes, ensuring they remain private. However, public getter (GET) and setter (SET) methods facilitate controlled attribute manipulation.

Why we use the Encapsulation ?

1- to hide class properties from external and prevent direct access them outside the class.

MEC ACADEMY

```php
class Car {
    private $brand;
    private $model;
    private $year;

    public function __construct($brand, $model, $year) {
        $this->brand = $brand;
        $this->model = $model;
        $this->year = $year;
    }

    public function getBrand() {
        return $this->brand;
    }

    public function getModel() {
        return $this->model;
    }

    public function getYear() {
        return $this->year;
    }

    public function setYear($year) {
        $this->year = $year;
    }
}

// Create a Car object
$car = new Car("Toyota", "Camry", 2020);

// Access and display the brand, model, and year
echo "Brand: " . $car->getBrand() . "\n";
echo "Model: " . $car->getModel() . "\n";
echo "Year: " . $car->getYear() . "\n";

// Update the year using the setYear() method
$car->setYear(2022);
```

Access Modifiers

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

There are 3 access modifiers:-

- **Public:** the property or method can be accessed from everywhere; This is default.
- **Private:** the property or method can ONLY be accessed within the class.
- **Protected:** the property or method can be accessed within the class and by classes derived from that class.

MEC ACADEMY

Inheritance


INHERITANCE
2
Google Developer Student Clubs
Suez Canal University
NEXT

MEC ACADEMY

Inheritance

Inheritance : is a relationship that the mechanism in PHP by which one class is allow to inherit the features (fields and methods) of another class.

Why we use the Inheritance ?

1- For Method Overriding (so runtime polymorphism can be achieved), we will learn later.

2- For Code Reusability that when we want to create a new class and there is already a class that includes some of the code that we want, we can derive our new class from the existing class.

MEC ACADEMY

Superclass and subclass

Super Class : is the class where a subclass inherits the features. It is also called a (**Parent Class**).

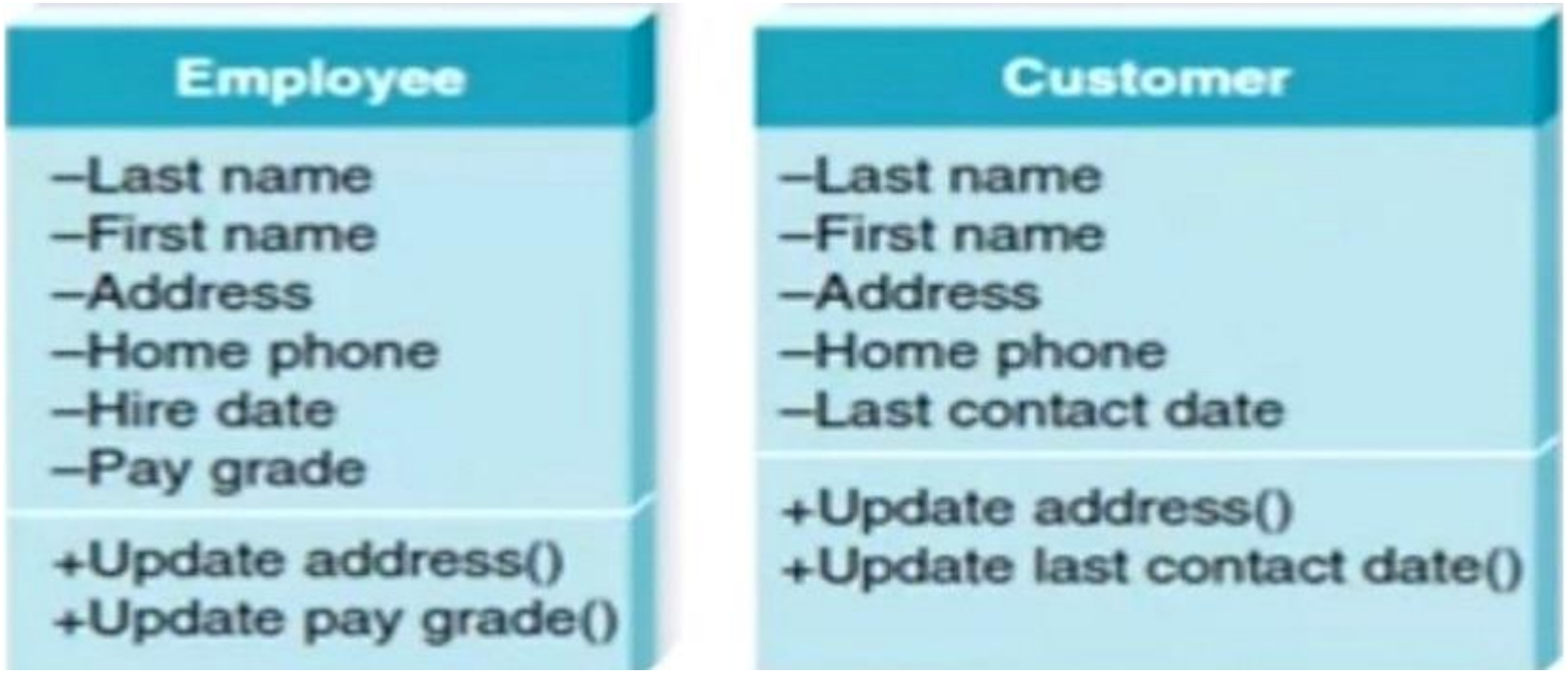
Sub Class : Subclass is a class which inherits the other class.

It is also called a (**Child Class**) , The subclass have own fields and methods in addition to the superclass fields and methods.

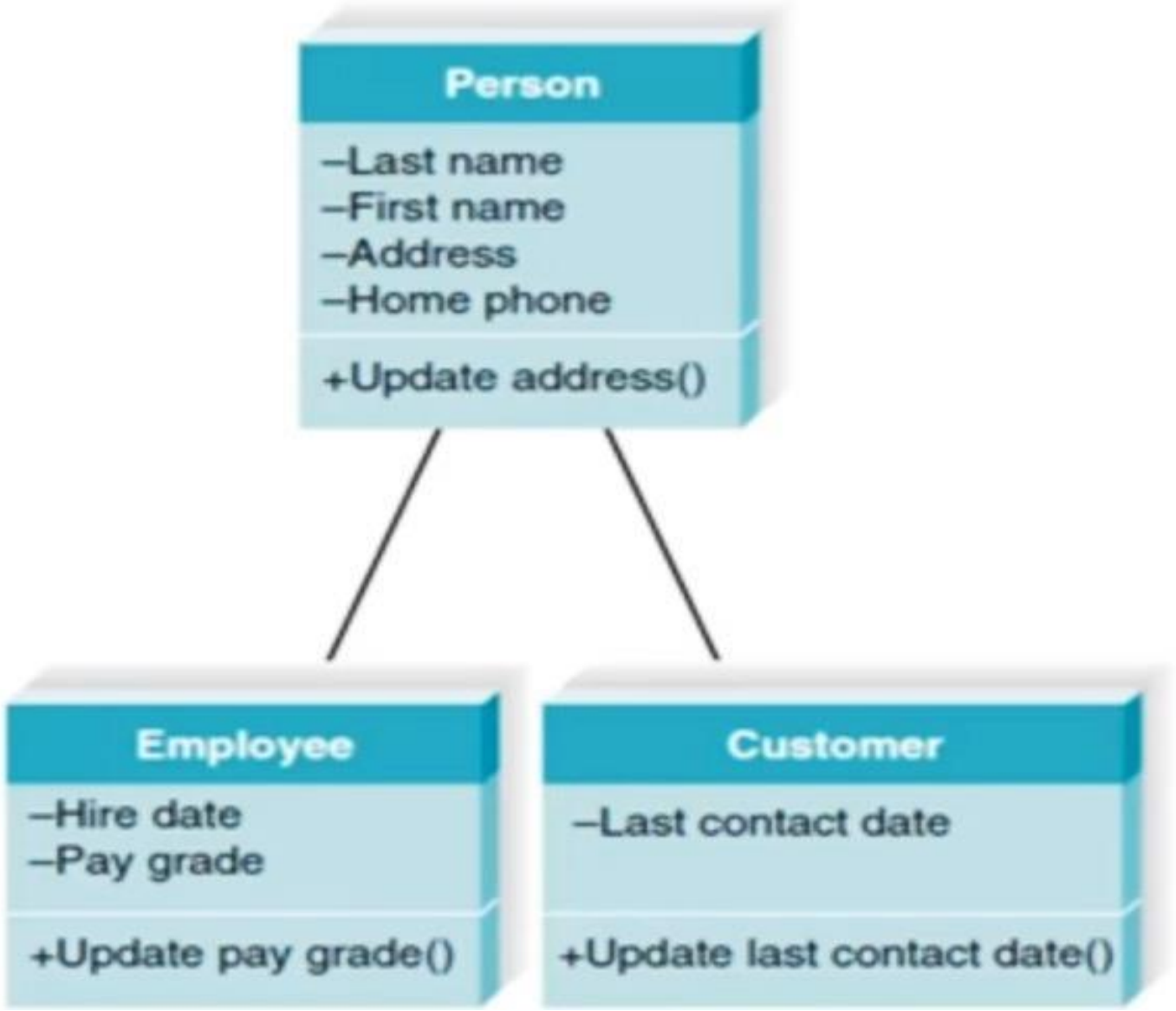
MEC ACADEMY

An Example to show reusability of the code :

Without Inheritance



With Inheritance



Hint :

The features of the superclass that are marked protected are inherited by the subclass, may only be accessed from the subclass by public methods of the superclass.

MEC ACADEMY

Inheritance syntax

- The keyword used for inheritance is **'extends'**

```
class Super {
      .....
            .....
 }
class Sub extends Super {
     .....
          .....
}
```

MEC ACADEMY

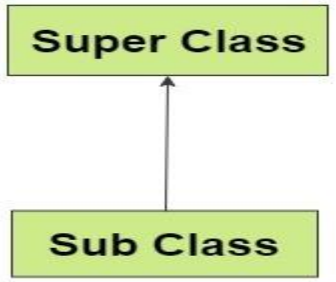
Calling Super class Constructor from Sub class

```
class Super {
 protected $name;
      function __construct($name){
            $this->name = $name;
      }
 }
class Sub extends Super {
public $address;
 function __construct($name, $address){
             parent:: __construct($name);
            $this->address = $address;
      }
}
```

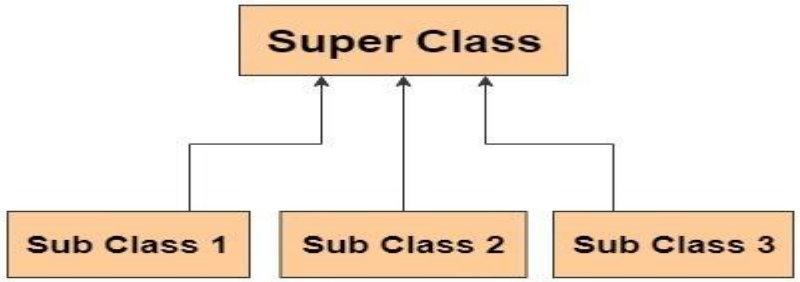
MEC ACADEMY

Types of Inheritance

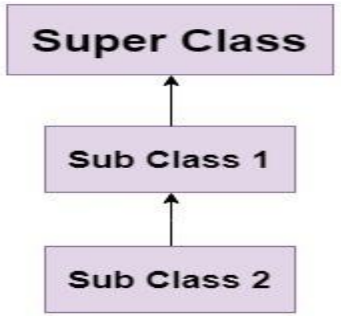
Single Inheritance



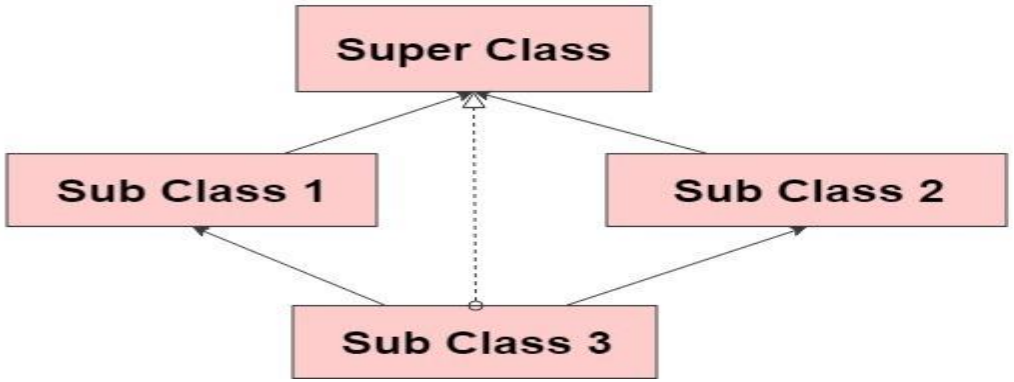
Hierarchial Inheritance



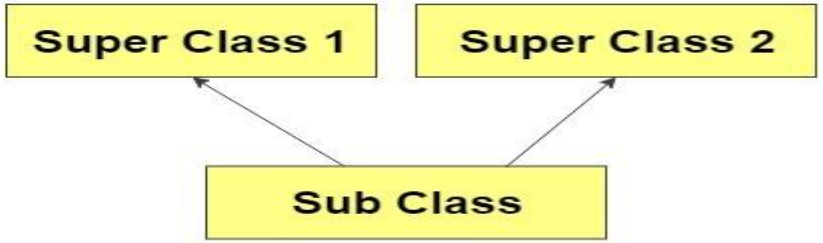
MultiLevel Inheritance



Hybrid Inheritance



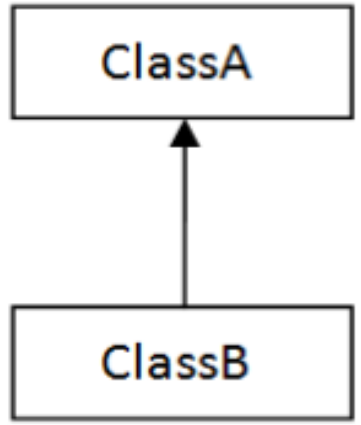
Multiple Inhertance



MEC ACADEMY

1) Single Inheritance

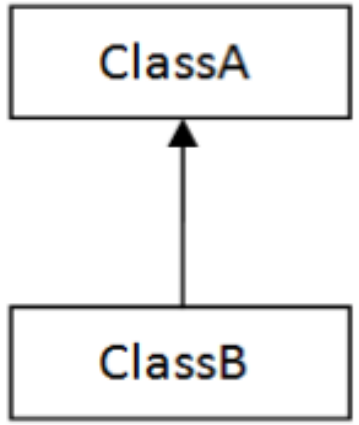
When a class extends another one class only then we call it a single inheritance. The below flow diagram shows that class B extends only one class which is A. Here A is a parent class of B and B would be a child class of A.



1) Single

1) Single Inheritance

```php
class Animal {
    protected $name;

    public function __construct($name) {
        $this->name = $name;
    }

    public function eat() {
        echo $this->name . " is eating. ";
    }
}

class Dog extends Animal {
    public function bark() {
        echo $this->name . " is barking. ";
    }
}
```

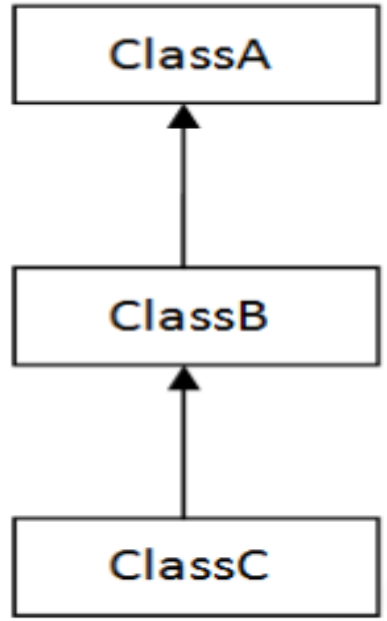


1) Single

MEC ACADEMY

2) Multilevel Inheritance

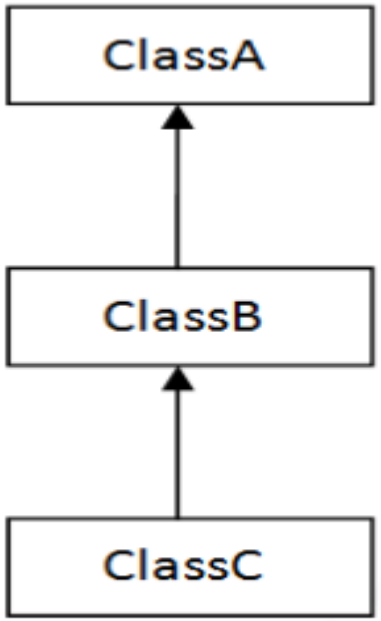
Multilevel inheritance refers to a mechanism in OO technology where one can inherit from a derived class, there by making this derived class the base class for the new class.



2) Multilevel

MEC ACADEMY

```php
class Animal {
    protected $species;

    public function __construct($species) {
        $this->species = $species;
    }

    public function eat() {
        echo "The animal is eating." . PHP_EOL;
    }
}

class Mammal extends Animal {
    protected $name;

    public function __construct($species, $name) {
        parent::__construct($species);
        $this->name = $name;
    }

    public function sleep() {
        echo $this->name . " is sleeping." . PHP_EOL;
    }
}

class Dog extends Mammal {
    public function bark() {
        echo $this->name . " is barking." . PHP_EOL;
    }
}
```

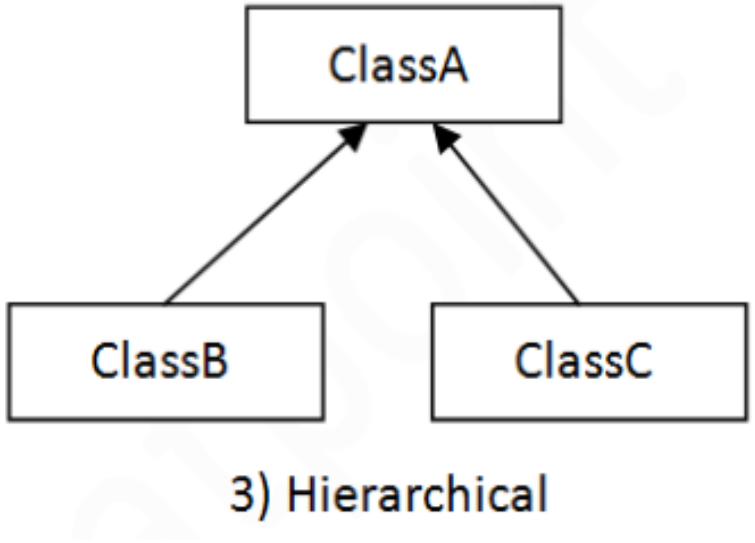


2) Multilevel

MEC ACADEMY

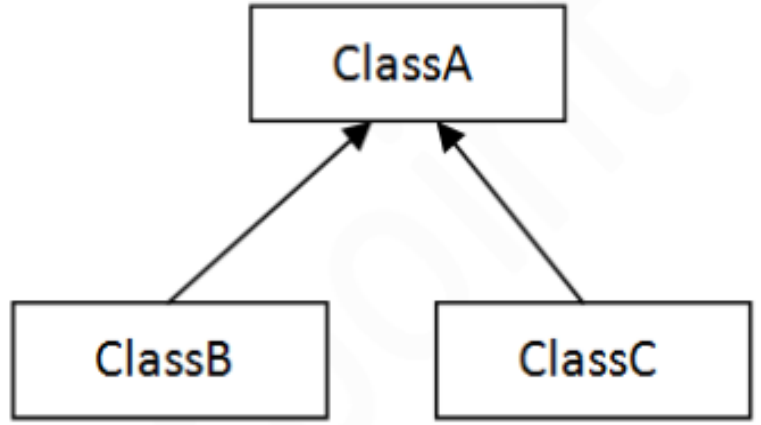
3) Hierarchical Inheritance

In such kind of inheritance one class is inherited by many sub classes. In below example class B and C inherits the same class A. A is parent class (or base class) of B and C.



3) Hierarchical

```php
class Animal {
    protected $name;

    public function __construct($name){
        $this->name = $name;
    }

    public function eat() {
        echo $this->name . " is eating.\n";
    }
}

class Dog extends Animal {
    public function bark() {
        echo $this->name . " is barking.\n";
    }
}

class Cat extends Animal {
    public function meow() {
        echo $this->name . " is meowing.\n";
    }
}
```

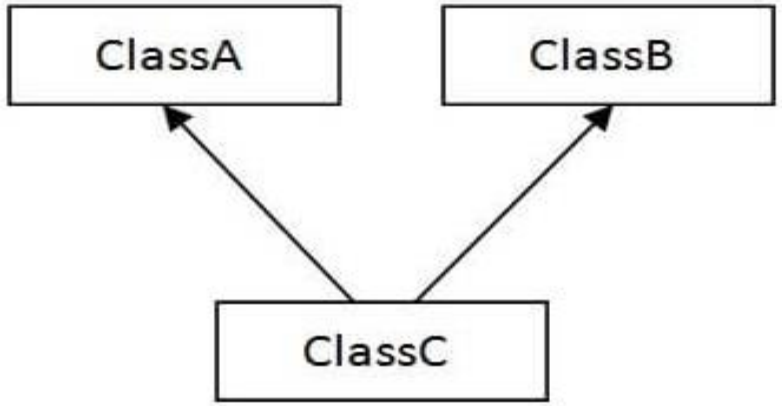


3) Hierarchical

MEC ACADEMY

4) Multiple Inheritance

Multiple inheritance is **not supported in PHP through class.** In PHP, we can achieve multiple inheritance only through Interfaces. We will learn about interfaces later.

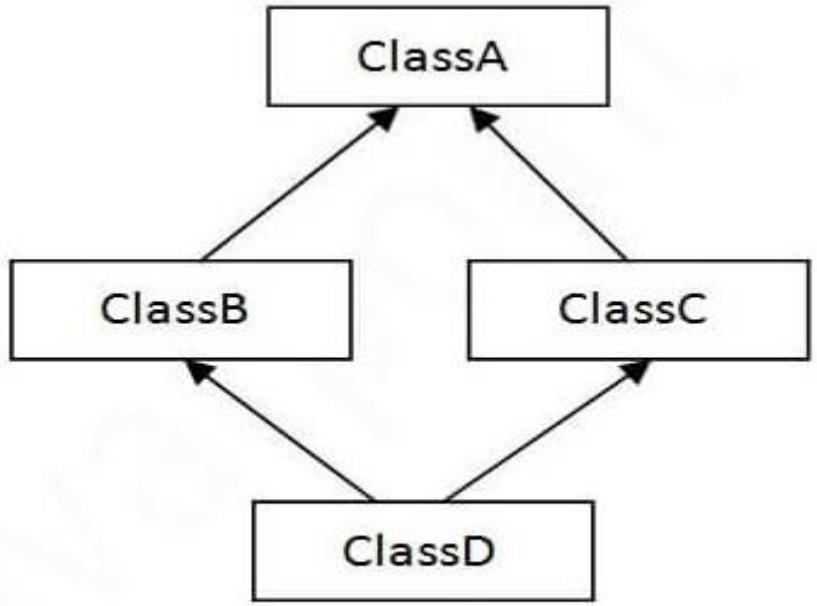


4) Multiple

MEC ACADEMY

5) Hybrid Inheritance

It is a mix of two or more of the above types of inheritance. **PHP doesn't support Hybrid inheritance with classes**, the hybrid inheritance is also not possible with classes. In PHP, we can achieve hybrid inheritance only through Interfaces.



5) Hybrid

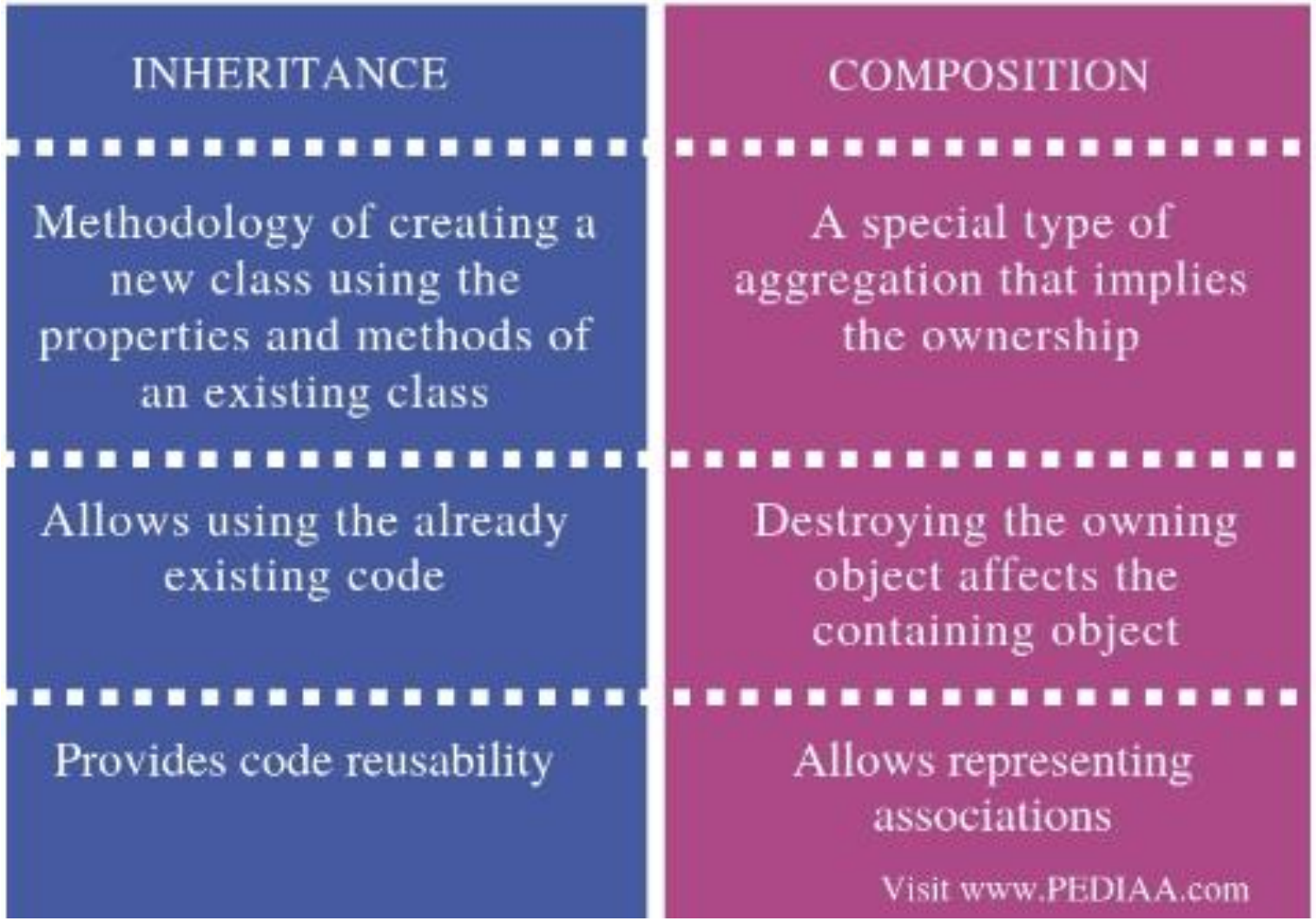
composition

When there are two classes in a model need to communicate with each other, there must be link between them , that can be represented by an Composition.

Composition is **has a /** a **part-of** relationship (strong relationship) there are two classes called class A and B. If the object of class B cannot exist if the object of class A is destroyed, then that is a composition. A book consists of many pages. If the book is destroyed, the pages will also destroy. The page objects cannot exist without the book object.

```php
<?php

class Engine
{
    private $fuelType;

    public function __construct($fuelType)
    {
        $this->fuelType = $fuelType;
    }

    public function start()
    {
        echo "Engine started. Fuel type: {$this->fuelType}\n";
    }
}

class Car
{
    private $model;
    private $engine;

    public function __construct($model, Engine $engine)
    {
        $this->model = $model;
        $this->engine = $engine;
    }

    public function startCar()
    {
        echo "Starting car: {$this->model}\n";
        $this->engine->start();
    }
}

// Create an engine
$engine = new Engine('Gasoline');
```

Inheritance VS composition



this

In PHP, this is a **reference variable** that refers to the current object :

1) this can be used to refer current class instance variable.

```php
class Car {
  public $brand;
  public $model;

  public function __construct($brand, $model) {
    $this->brand = $brand;
    $this->model = $model;
  }

  public function getCarInfo() {
    echo "Brand: " . $this->brand . ", Model: " . $this->model;
  }
}

$car1 = new Car("Toyota", "Corolla");
$car1->getCarInfo(); // Output: Brand: Toyota, Model: Corolla
```

MEC ACADEMY

Self

- In PHP, this is a **reference variable** that refers to the Class.

- refer to class and used to access **const** properties and **static** properties and methods

MEC ACADEMY

static

1) Variable (also known as a class variable) : A static field is a part of a class and that is not part of the object to the value of the static field is shared among all objects.

```php
class Counter {
    public static $count = 0;

    public function __construct() {
        self::$count++;
    }
}

$counter1 = new Counter();
echo Counter::$count; // Output: 1

$counter2 = new Counter();
echo Counter::$count; // Output: 2
```

MEC ACADEMY

Const

1) Variable (also known as a class variable) : A static field is a part of a class and that is not part of the object to the value of the Const field is shared among all objects.

```
Class rectangle {
      const pi=3.14;

     public function getConst() {
          return self::pi;
     }
}
```

MEC ACADEMY

Difference Between the static method and instance method

Instance Methods	Static Methods
It requires an object of the class.	It doesn't require an object of the class.
It can access all attributes of a class.	It can access only the static attribute of a class.
The methods can be accessed only using object reference.	The method is only accessed by class name.

Tasks

- What is single tone design pattern?
- How can apply Multiple Inheritance?
- What is destructor ?

Tasks

❑ You have person class

- have id, name properties private;
- give the values of properties while creating object.
- create function display person data.
- create functions to set and get id only and name only.

❑ Create anther class called Employee class inherit from Person Class

- Have company name and salary properties private.
- give the values of properties while creating object.
- create function display Employee data.
- create functions to set and get salary only and company name only.

Next Session

- Abstract class.
- Interface.
- Final keyword.
- Traits.
- Namespace.

THANKS
Any questions?

MEC ACADEMY