Inheritance

This lesson introduces the concept of Inheritance focusing on base and derived classes.

We'll cover the following

What is Inheritance

Inheritance is a very useful and popular concept that enhances the object-oriented programming experience. It allows a programmer to create and define classes that can inherit and build upon functionalities already present in existing classes without having to duplicate a lot of the code.

- Provides a way to create a **new** class from an **existing** class.
- The **new** class is a *specialized* version of the **existing** class.
- Allows the **new** class to override or redefine inherited *methods* from the **existing** class to perform differently than how they are defined in the **existing** class.

Terminology

- **Base Class** (or Parent): *inherited* by **child** class.
- **Derived Class** (or child): *inherits* from base class.

Characteristics

A **derived** class has:

- All *members* defined in the **derived** class.

- All *members* declared in the **base** class.

A **derived** class can:

- Use all `public` and `protected` members defined in the **derived** class.
- Use all `public` and `protected` members defined in the **base** class.
- Override an *inherited* member

A **derived** class cannot:

- *Inherit constructors* and **destructors**

Inheritance is one of the major reasons for using object-oriented programming in PHP.

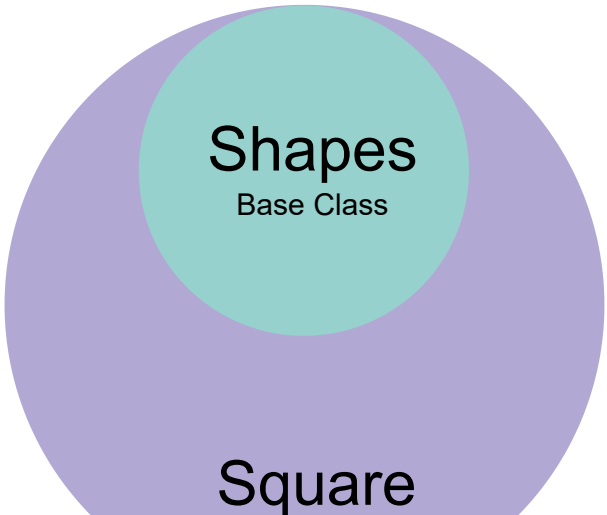
Notation

Classes can *inherit* the properties and methods of another class using the keyword `extends`.

Let's take a look at the notation below.

```
class Shape { //base class
 //body
}

//derived class inherits from another class by using the keyword extends
class Square extends Shape {
      //body
}
```



Extended Class

Example

Let's consider an example with the *base* class `Shape` and the *derived* class `Square`.

```
<?php
class Shape
{ //base class
    public $sides = 0;

    public $name = " ";

    public function __construct($name, $sides)
    { //base class constructor
        $this->sides = $sides;
        $this->name = $name;
    }

    public function description()
    {
        return "A $this->name with $this->sides sides.";
    }
}

class Square extends Shape
{ //derived class inheriting from base class
    public $sideLength = 0;

    public function __construct($sideLength)
    {
        parent::__construct("square", 4); //calling parent class constructor
        $this->sideLength = $sideLength;
    }

    public function perimeter()
    {
        return $this->sides * $this->sideLength;
    }

    public function area()
    {
        return $this->sideLength * $this->sideLength;
    }
}

$mySquare = new Square(10);
$mySquare->description();
echo "Perimeter of the square is " . $mySquare->perimeter() . "\n";
echo "Area of the square is ";
echo $mySquare->area();
?>
```

In the above example, the **extended class** `Square` *extends* from the **base class**.

`Shape`. While doing so, it is inheriting all of its properties, `$sides` and `$names`, and its methods, `description()`.

It then goes on to define new properties: `$sideLength` and new methods: `perimeter()` and `area()`.

As seen in the example above, a derived class cannot inherit the base class constructor however it can call the base class constructor as seen in **line 25**. Hence, once the derived class constructor is called it will further call upon the parent class constructor, where it'll set the `name` and `sides`, and after that, it'll set the `sideLength` as well.

This marks the end of our discussion on inheritance. In the next chapter, we will discuss *exception handling*.