Polymorphism in PHP

According to the Polymorphism (Greek for “many forms”) principle, methods in different classes that do similar things should have the same name.

The Polymorphism principle, methods in different classes that do similar things should have the same name.

How to implement the polymorphism principle?

In order to implement the polymorphism principle, we can choose between abstract classes and interfaces.

**Example:**

interface Shape {

public function calcArea();

}

class Circle implements Shape {

private $radius;

public function \_\_construct($radius)

{

$this -> radius = $radius;

}

// The method calcArea calculates the area of circles

public function calcArea()

{

return $this -> radius \* $this -> radius \* pi();

}

}

//////////////////////////////////////////////////////////////////////////

class Rectangle implements Shape {

private $width;

private $height;

public function \_\_construct($width, $height)

{

$this -> width = $width;

$this -> height = $height;

}

// calcArea calculates the area of rectangles

public function calcArea()

{

return $this -> width \* $this -> height;

}

}

// Now, we can create objects from the concrete classes:

$circ = new Circle(3);

$rect = new Rectangle(3,4);

echo $circ -> calcArea();

echo $rect -> calcArea();

Result:

28.274333882308

12