00P inheritance

уеб разработка с РНР/2021

Милена Томова Vratsa Software

https://vratsasoftware.com/

Table of Contents



- 1. Class inheritance
- 2. Method "overloading" and "overriding"
- 3. isA and hasA relations between classes
- 4. Access specifiers in class extension
- 5. Final keyword in class definition





```
class Page {
       public $title;
       public $content;
       public $footer;
       public function __construct($t, $c, $f) {
               $this->title = $t;
               $this->content = $c;
               $this->footer = $f;
       public function render header(){
           $str = $this->title();
           return $str;
       public function render_title() {
               $str = '<h1>'.$this->title.'</h1>';
               return $str;
       public function render content() {
               $str .= ''.$this->content.'';
               return $str;
       public function render_footer() {
               $str .= ''.$this->footer.'';
               return $str;
```

Task



Задача Създайте приложение, което

Страниците на приложението са Home, Contacts, About Us, Content Само Home страницата има слайдер и реклама. Всички страници имат

- header
- content
- footer



Properties

class HomePage {

public \$title; public \$content; public \$footer; public \$slider; public \$banner;

=> every type of page

=> HomePage specific

••••

We avoid code repetion by using class inheritance

```
class HomePage extends Page {

public $slider;
public $banner;
}

Every object, instance of HomePage class
will have all Page class properties + HomePage class properties.
```







Methods

All instances of HomePage class will possess the Page class methods.

also

The instances of HomePage class can have their specific methods, defined in HomePage class.

also

The instances of HomePage class can override or extend parent methods /the methods defined in Page class/.



```
class HomePage extends Page {
     public $slider;
     public $banner;
     public function __construct($h, $c, $f, $s, $b){
            parent::__construct($h, $c, $f); //call parent constructor if it is
                                            defined in the parent class
                        $this->slider = $s;
            $this->banner = $b;
```



```
class HomePage extends Page {
     public $slider;
     public $banner;
     public function __construct($h, $c, $f, $s, $b){
        parent::__construct($h, $c, $f);
        $this->slider = $s;
       $this->banner = $b;
     public function render_slider() {
          $str .= ''.$this->slider.'';
          return $str;
     public function render_banner() {
          $str .= ''.$this->banner.'';
          return $str;
```



Method overriding

В класът HomePage предефинирахме __construct(). Освен на header, content, footer, той задава стойн свойствата.

Можем да предефинираме и други методи в класовете наследници

- В наследника създаваме едноименен метод с този в родителския
- Метода ще функционира във формата, дефиниран
 - В родителския клас, ако се извиква от негови обекти
 - В дъщерния клас, ако се извиква от негови обекти.

"overloading" and "overriding" class methods



Methods

In homePage class we "overrided" the parent __construct() method:

Page class __construct()

```
public function __construct($t, $c, $f) {
        $this->title = $t;
        $this->content = $c;
        $this->footer = $f;
}
```

HomePage class __construct()

```
public function __construct($h, $c, $f, $s, $b){
    parent::__construct($h, $c, $f);
    $this->slider = $s;
    $this->banner = $b;
}
```

"overloading" and "overriding" class methods



```
// Page class method definition
       public function render_header(){
          $str = $this->title();
          return $str;
//HomePage class overriding the method definition
      public function render_header(){
          $str = parent::render_header(); //call parent method definition only to extend parent method logic ...
          $str .= $this->slider();
          return $str;
```

"overloading" and "overriding" class methods

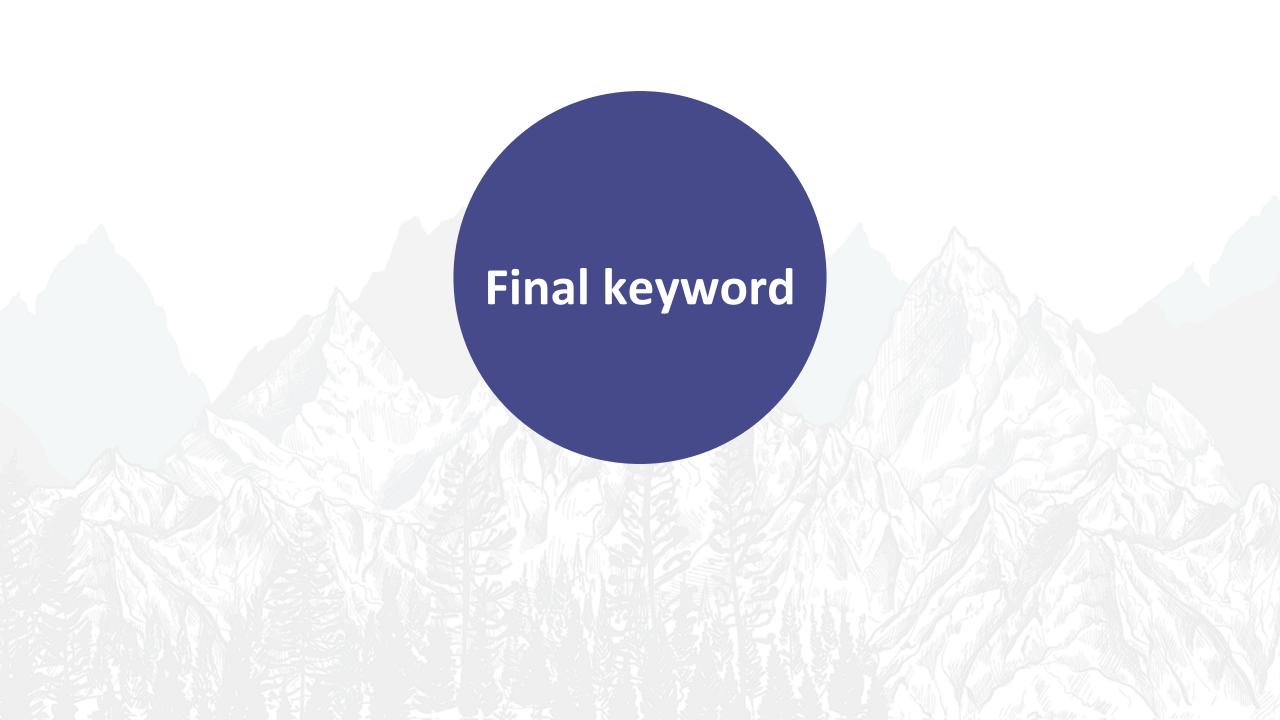


Methods cannot be overloaded in classes written in php.

Two methods with the same name will invoke an error.







Final keyword



When a method definition starts with the keyword final, this will prevent the method to be overridden in the classes that will extend the current class.

```
class Test {
 final public function moreTesting() {
      echo "I am a final method
                 and cannot be overriden!";
```

Final keyword



When a <u>class</u> definition starts with the keyword final,

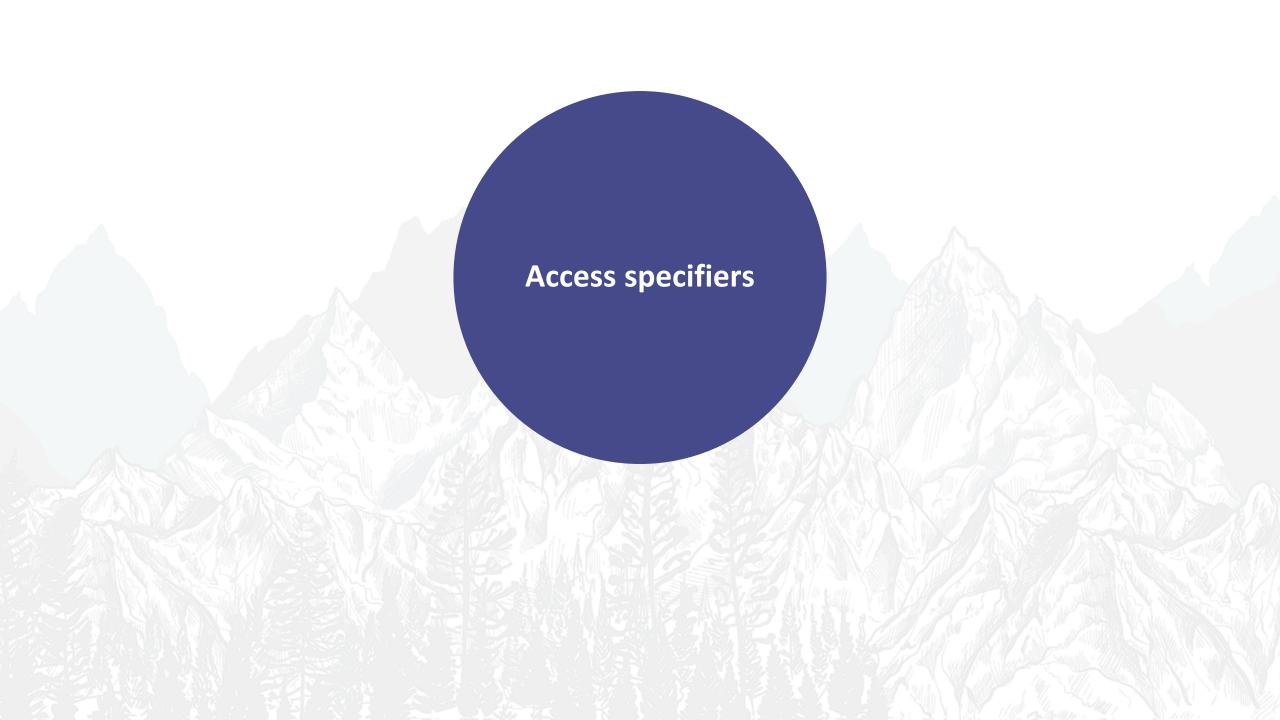
this will prevent the class to be extended.

```
final class BaseClass {
   public function test() {
      //
   }

   // Here it doesn't matter if you specify the function as final or not
   final public function moreTesting() {
      //
   }
}
```

```
class ChildClass extends BaseClass {
}

// will result in Fatal error: Class
ChildClass may not inherit from final
class (BaseClass)
```



Access specifiers (modifiers)





more info

- 1. public class or its members defined with this access modifier will be publicly accessible from anywhere, even from outside the scope of the class.
- private class members with this keyword will be accessed within the class itself. It
 protects members from outside class access with the reference of the class
 instance.
- 3. protected same as private, except by allowing subclasses to access protected superclass members.
- 4. abstract This keyword can be used only for PHP classes and its functions. For containing abstract functions, a PHP class should be an abstract class./to be explained later/
- final It prevents subclasses to override super class members defined with final keyword.

Questions?



Partners















Trainings @ Vratsa Software



- Vratsa Software High-Quality Education, Profession and Jobs
 - www.vratsasoftware.com
- The Nest Coworking
 - www.nest.bg
- Vratsa Software @ Facebook
 - www.fb.com/VratsaSoftware
- Slack Channel
 - www.vso.slack.com



