OOP inheritance

уеб разработка с PHP/2019

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 inheritance

Class inheritance

```
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() {
                $str .= ‘<p>’.$this->content.‘</p>’;
                return $str;
        }

        public function render_footer() {

                $str .= ‘<p>’.$this->footer.‘</p>’;

                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

Class inheritance

Properties

```
class HomePage {

        public $title;
        public $content;                                  => every type of page
        public $footer;
        public $slider;
        public $banner;                                   => HomePage specific

….
```

Class inheritance

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.

Class inheritance

Врађа Софтуер Общество

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 inheritance

```
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 inheritance

```
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() {
          $str .= ‘<p>’.$this->slider.‘</p>’;
          return $str;
     }
     public function render_banner() {
          $str .= ‘<p>’.$this->banner.‘</p>’;
          return $str;
     }
}
```

‘overloading’ and ‘overriding’ class methods

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;
}
```

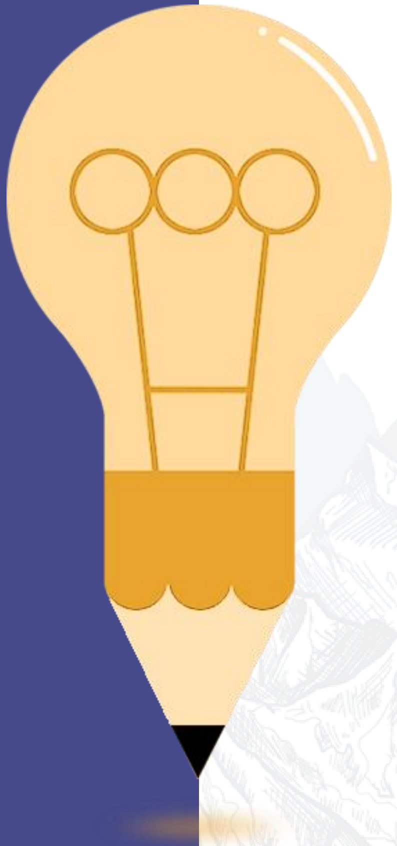
HomePage class __construct()

```
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.

IsA / HasA demo

Final keyword

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 class definition starts with the keyword final, this will prevent the class to be extended.

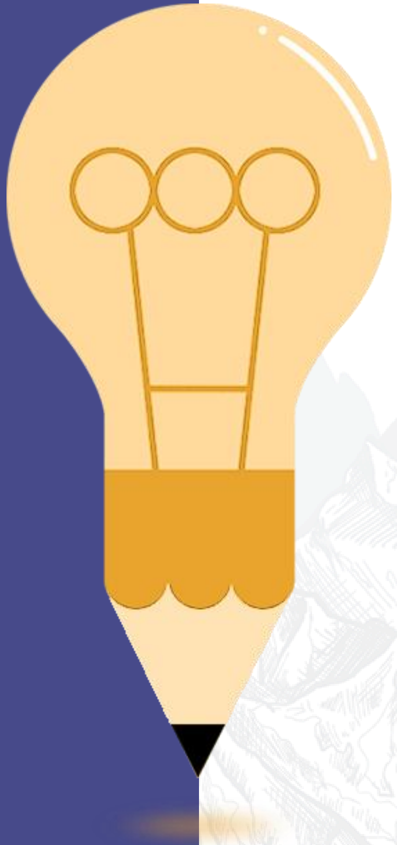
```php
final class BaseClass {
   public function test() {
      //
   }

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

```php
class ChildClass extends BaseClass {

}

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

Access specifiers

Access specifiers (modifiers)

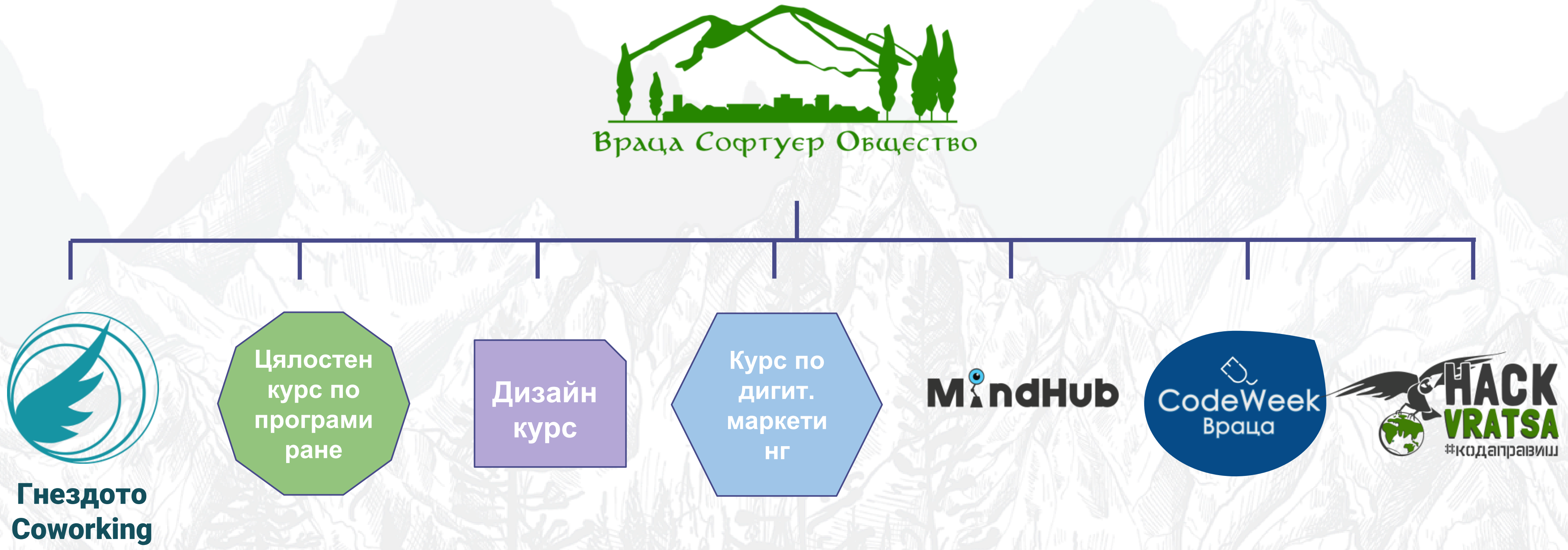
Враца Софтуер Общество



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.
2. *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/
5. *final* – It prevents subclasses to override super class members defined with *final* keyword.

Questions?


Враца Софтуер Общество
Гнездото Coworking
Цялостен курс по програми ране
Дизайн курс
Курс по дигит. маркети нг
MindHub
CodeWeek Враца
HACK VRATSA
#кодаправиш

Partners

Враца Софтуер Общество

AMERICA FOR BULGARIA FOUNDATION

ВРАЦА

Telerik Academy

НАЦИОНАЛНА МРЕЖА ЗА ДЕЦАТА

MindHub

ПРОМЯНАТА

NOVA

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