# OOP interfaces, static methods and classes

PHP WebDevelopment 2020

Милена Томова
Vratsa Software

https://vratsasoftware.com/

## Table of Contents

## Interfaces

## Interfaces

It is like an abstract class having only abstract methods.

All classes that implement an interface must implement

the methods declared in the interface.

## Interfaces

Interfaces allow you to create code which specifies which methods a class must implement, without having to define how these methods are implemented.

Usually this method implementation is different in different classes.

It is possible to declare a constructor in an interface, which can be useful in some contexts.

## Interfaces

One class can implement more than one interface.

An interface can implemented by different, not isA connected classes.

Can interfaces be implemented by isA connected classes?

## interfaces

```
interface iInfo {
      public function get_data( $param );
      public function display_data();
}
```

## interfaces

Враца Софтуер Общество

```
class User implements iInfo {
        public function get_data( $param ){
                //gets user from db
                //returns user data from db
    }
        public function display_data(){
        $data = $this->get_data( $this->username );
        //display retrieved data from the db
    }
}
```

# interfaces

Враца Софтуер Общество

```
class Page implements iInfo {
        public $content;
        ...
        public function __construct( $page_id ){
                $this->content = $this->get_data( $page_id );
                …
        }
        public function get_data( $param ){
                //gets page data from db
                //returns data
    }
        public function display_data(){
        //display $this->content
        //display ....
    }
}
```

# interfaces

One class can implement more than one interface.

```
class User implements iInfo, iAnotherInterface {
        public function get_data( $param ){
                //gets user from db
                //returns user data from db
    }
        public function display_data(){
        $data = $this->get_data( $this->username );
        //display retrieved data from the db
    }
    ...
}
```

## Summary

Враца Софтуер Общество

- ➔ The class that implements an interface must implement all the methods from the interface definition.
- ➔ If the method is declared in the interface that it will use param(s), the methods implemented in the class must be implemented with the same number of parameters.
- ➔ In an interface method access specifiers are only public.
- ➔ An interface can be extended by another interface using the **extends keyword**.

## Static properties

## Static properties

Properties that belong to the class rather than to the instance.

Used without any class instance.

```
class Car {

    static public $type = 'car';

}
```

## Static properties

A static property is accessed

inside a class -

```
public function get_car_type(){
        return self::$type;
}
```

A static property is accessed

outside a class -

```
echo Car::$type;
```

# Static propertis

*Static properties values in the classes that extend a class*

```
Class Toyota extends Car {

    static public $type = ‘toyota’;

    public function get_toyota_type(){
            echo self::$type;
   }
}

$toyota = new Toyota();
$toyota->get_car_type();//car
$toyota->get_toyota_type();//toyota
```

```
Class Car {

    static public $type = ‘car’;

    public function get_car_type(){
            echo static::$type;
    }
}

$toyota = new Toyota();
$toyota->get_car_type();//toyota
$toyota->get_toyota_type();//toyota
```

## Static methods

## Static properties

Methods that belong to the class, not to the class instances.

Static methods are called on the class, not on the class instances.

Static methods use only static properties.

```
class Car {
    static public $type = 'car';
    static public function get_car_type(){
        return self::$type;
    }
}
```

```
Car::get_type();//car
Toyota::get_type();//car
```

## Late Static Binding

## Static propertis

Replacing **self** with **static** keyword allows to work with the current class`s value of a static property.

```
Class Car {

     static public $type = ‘car’;

     public function get_car_type(){
               echo static::$type;
     }
}

class Toyota {
       static public $type = ‘toyota’;
}

Car::get_car_type();//car
Toyota::get_car_type();//toyota
```

## Static properties and methods

*Task*: Using static properties and/or methods implement displaying the current instance number.

*Describe the process - the methods to be defined and properties to be used.*

## Constants

## constants

Constants are defined by

**define(‘OPERATOR’, ‘driver’);**

in procedural programming.

Constant names are in **uppercase** by covention.

In a class constants are defined by

```
class Car {
    const OPERATOR =’driver’;
}
```

using the keyword **const**.

# constants

Constants can be used inside of the class -

```
public get_const() {
    echo self::OPERATOR
}
```

or outside of the class

Carr::OPERATOR

## constants

A class constant is vosible and can be used in the classes that inherit the class where the constant is defined.

```
Class ToyotaCar extends Car{

    public function toyota_description(){

        echo 'Usually a Toyota is driven by an ' .

         self::OPERATOR;

    }

}

    echo ToyotaCar::OPERATOR
```

## constants

A common example of using constants in OOP is a class that implements a database connection.

The data values needed to connect a Data base are stored in constants.

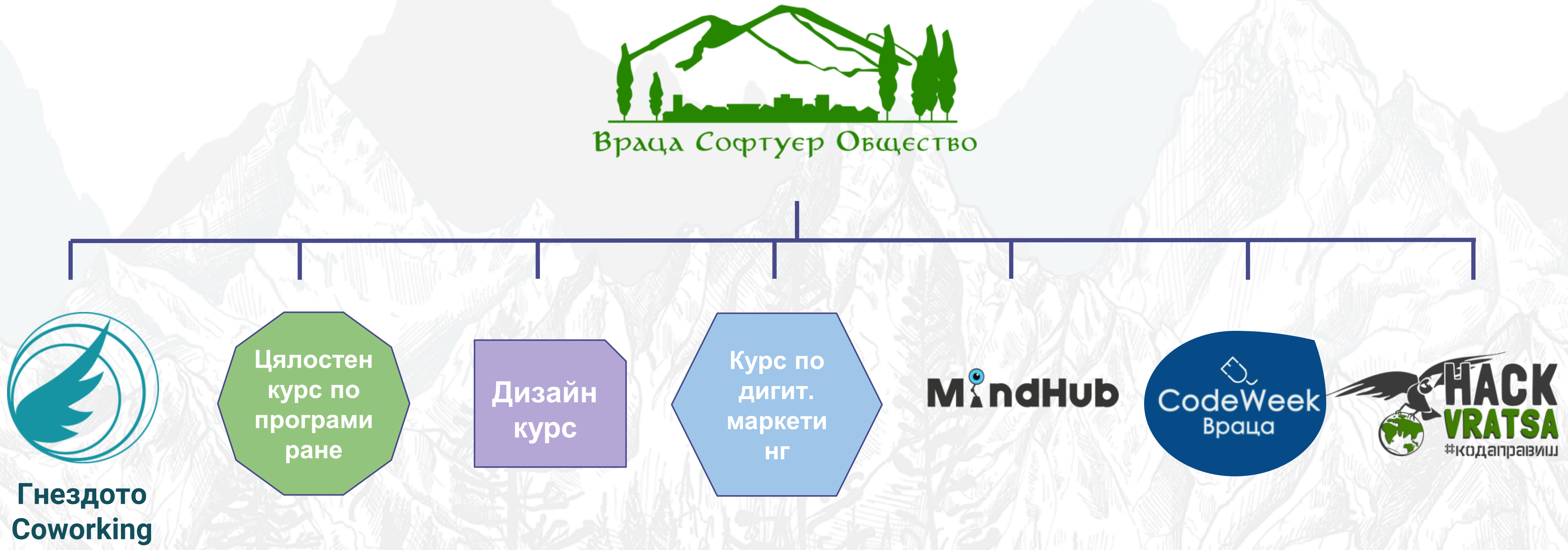
```
const dbHost =

const dbName =

const dbUsername =

const dbPassword =
```

# 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

NOVA ПРОМЯНАТА

# 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