



# Въведение в PHP

Обектно-ориентирано програмиране

*Георги Гроздев*

*[g.grozdev@viscomp.bg](mailto:g.grozdev@viscomp.bg)*

*Атанас Василев*

*[a.vasilev@viscomp.bg](mailto:a.vasilev@viscomp.bg)*

*[www.viscomp.bg](http://www.viscomp.bg)*

# About us

## ■ Лектори

- Георги Гроздев – [g.grozdev@viscomp.bg](mailto:g.grozdev@viscomp.bg)
- Атанас Василев – [a.vasilev@viscomp.bg](mailto:a.vasilev@viscomp.bg)

## ■ Инфо и слайдовете ще намерите на:

- <http://phplab.viscomp.bg>

vis|comp  
we develop the web

# Design Patterns

- Установен подходи за решаване на различни проблеми в даден контекст
- Всеки шаблон идентифицира проблем, който се повтаря в обкръжението ни и описва такова решение за него, че то да може да се използва многократно.
- Шаблоните правят по-лесно използването на доказани дизайни и архитектури
- Представянето на проверени техники под формата на шаблони ги прави по-достъпни за разработчиците на нови системи
- Шаблоните ни помагат да избираме такива алтернативи за дизайн, които да направят системата ни използвана многократно

# Design Patterns

## ■ *Factory* и *Factory Method*

- Когато заради по-голяма гъвкавост искаме обектите да не се инстанциират директно, а да се произвеждат от централен метод. Така ако искаме да променим типа на произведените методи, ще трябва да променим само централния метод.
- Когато даден клас не може да знае, обекти от какъв клас ще произвежда.
- Когато е по-добре производните класове да определят какъв обект ще произведе базовият клас.

# Design Patterns

## ■ Singleton

- Когато ни е необходима точно една инстанция на даден клас и тя трябва да е достъпна за клиентите отвсякъде.
- Когато единствената инстанция трябва да може да се разширява и клиентите да могат да ползват разширения вариант без да е необходимо да променят кода си.
- Когато е по-добре производните класове да определят какъв обект ще произведе базовият клас.

# Design Patterns

## ■ Decorator

- Когато искаме да добавяме функционалност към отделни обекти динамично и прозрачно, без да променяме други обекти.
- Когато искаме да добавим отговорности, които да могат по-късно да бъдат отнети лесно.
- Когато искаме да добавим дадена функционалност не по принцип, а за един конкретен компонент и то в определени от нас моменти – в този случай наследяването не е опция, понеже така разширението би се приложило за всички инстанции.
- Когато не е възможно разширяване чрез наследяване. Например всички възможни комбинации от многобройни разширения биха довели до твърде много класове.

# ООП

## ■ Клас

- Най-общо - съвкупност от данни и код. Обединява структурата и поведението на обектите.

```
class Dog
{
    public $name;
    public $breed = 'Golden retriever';

    function __construct($name) {
        $this->name = $name;
    }
    public function bark() {...}
}
```

- Стойностите по подразбиране трябва да са константни изрази, а не променлива, извикване на функция или метод на клас

# Въпроси?

**Благодаря ви  
за вниманието!**