Class inheritance

Object-oriented programming

VIVIFY
ACADEMY

Problem I

```
class Patient {
        public $firstName;
        public $lastName;
        public $address;
        public $email;
        public $socialNumber;
}

class Doctor {
        public $firstName;
        public $lastName;
        public $address;
        public $email;
        public $speciality;
}
```

VIVIFY
ACADEMY

Problem II

```
class Person {
    public $firstName;
    public $lastName;
    public $address;
    public $email;
    public $medicalSpecialty; // patient doesn't need to have this
    public $socialNumber; // this attribute is not needed for doctor
}
```

VIVIFY ACADEMY

Problems

- How we deal with the situation where we have the same data (attributes) and behavior (methods) in multiple classes?
- Do those classes have something in common?
- How do we get new functionality without changing the class?

VIVIFY ACADEMY

Solution - inheritance

```
class Person {
        public $firstName;
        public $lastName;
        public $address;
        public $email;
}

class Patient extends Person {
        public $socialNumber;
}

class Doctor extends Person {
        public $speciality;
}
```

VIVIFY
ACADEMY

Class inheritance

- Child class inherits properties of parent class
- Child class has possibility to change existing functionality (overriding)
- Classes as types and subtypes
- Keyword used for inheritance is **extends**

VIVIFY
ACADEMY

Extending functionality

```
class Animal {
        public $name;
}

$dog = new Animal();
$dog->name = 'Jacky';
$dog->bark();       // ERROR!
```

VIVIFY
ACADEMY

Extending functionality

```
class Animal {
        public $name;
}

class Dog **extends** Animal {
        public function bark()
        {
                echo 'Woof!';
        }
}

$dog = new Dog();
$dog->name = 'Jacky';
$dog->bark();              // Bark!
```

VIVIFY
ACADEMY

Extending allows us to add functionality and data

- Terminology: if class B extends class A, A is called the “superclass”, “base class” or “parent class”. Class B is called the “subclass”, “child class”, “derived class”, or rarely, “heir class”.
- Extending is also called “inheritance”, because the subclass “inherits” all the properties and methods from the superclass.
- Objects of the subclass have everything that objects of the superclass have, but can also have additional properties and methods.
- Classes are types, and so subclasses are subtypes. This can be checked with the **instanceof** operator.

Extending functionality

```
class Animal {
    public $name;
}

class Dog extends Animal {
    public function bark()
    {
        echo 'Woof!';
    }
}

$someAnimal = new Animal();
$someDog = new Dog();

dump($someDog instanceof Dog);  // true
dump($someDog instanceof Animal);  // true
dump($someAnimal instanceof Animal);  // true
dump($someAnimal instanceof Dog);  // false!
```

An instance of the subclass is also an instance of the superclass!

- Every dog is also an animal, but not every animal is a dog!
- Animal is a type. Dog is a type (more accurately, a subtype) of Animal.

VIVIFY
ACADEMY

Exercise I

Modelovati sledeće klase vodeći računa o pravilnom nasleđivanju:

- Vozilo
- Motorno Vozilo
- Automobil
- Autobus
- Motocikl
- Limuzina
- Bicikl

Postaviti samo atribute i nasledjivanje, nije potrebno implementirati konstuktore i metode. Instancirati objekte klasa *Automobil*, *Bicikl* i *Limuzina*.

VIVIFY
ACADEMY

Protected - access modifier

VIVIFY
ACADEMY

Access modifiers

```
class Person {
    public $firstName;
    public $lastName;
    private $email;
}

class Doctor extends Person {
    public $speciality;
    public $hospital;
}
```

VIVIFY
ACADEMY

Protected

- Private properties and methods are not available when inheriting!
- For those non-public members we want to use in the subclass, we use the **protected** keyword.
- Protected properties function the same as private, except they are accessible in the subclass.
- This doesn’t mean that private members are not contained in the objects of the subclass, only that the subclass can’t access them directly.

Protected - example

```
class Person {
    public $firstName;
    public $lastName;
    private $email;
}

class Doctor extends Person {
    public $speciality;
    public $hospital;

    public function getDoctorEmail()  {  return $this->email;  }            // ERROR!
}
```

VIVIFY
ACADEMY

Protected - example

```
class Person {
    public $firstName;
    public $lastName;
    protected $email;
}

class Doctor extends Person {
    public $speciality;
    public $hospital;

    public function getDoctorEmail()  {  return $this->email;  }            // OK!
}
```

VIVIFY
ACADEMY

Why are private members not accessible?

- Complexity management: if all changes to a property are contained within one class, it’s easier to understand what’s happening with it. This is especially important if there’s multiple levels of inheritance.
- Reduced “coupling”: some members of a class are private because its users (outside of the class) don’t need to know and worry about the internal workings. The same goes for classes inheriting from this class.
- Even if we want to change existing functionality, we shouldn’t care about internals; instead, we can add data we need and use that.

VIVIFY
ACADEMY

Overriding

Changing existing functionality

- If we create a subclass and define a member with the same name as in the superclass, it replaces that member.
- This is called “overriding”. It allows us to change existing functionality of a class.

VIVIFY
ACADEMY

```
class Animal {
    public $name;

    public function speak()
    {
        echo '(We don\'t know what sound this animal makes)';
    }
}

class Dog extends Animal {
    public function speak()
    {
        echo 'Woof!';
    }
}

$dog1 = new Dog();
$dog1->name = 'Snoopy';
$dog1->speak();
```

VIVIFY
ACADEMY

Exercise II

Dodati metodu *kreni* u klase *Vozilo* i *Motorno Vozilo* i postaviti u nju jednostavan ispis (koristiti echo). Pozvati metodu nad kreiranim objektima.

Nakon toga isto ponoviti i za klasu *Automobil* i *Limuzina*. Pozvati metodu nad kreiranim objektima.

VIVIFY
ACADEMY

Parent

VIVIFY
ACADEMY

Referring to the **parent** class

- If we override a method, we don't lose it forever. It's still accessible via the **parent** keyword.
- **parent** is a reference to the parent class. We can call overriden methods from it.
- This is especially useful in constructors, since we can initialize the properties that the object got from its parent class without having to rewrite the code in the parent constructor.
- We can also ignore the parent method, in which case it's not called.

VIVIFY ACADEMY

```
class Animal {
    public $name, $age;

    public function __construct($name, $age)
    {
        $this->name = $name;
        $this->age = $age;
    }

    public function speak()
    {
        echo "(Sorry, we don't know what sound this animal makes.)";
    }
}
```

```
class Dog extends Animal {
    public $breed;

    public function __construct($name, $age, $breed)
    {
        parent::__construct($name, $age);
        $this->breed = $breed;
    }

    public function speak()
    {
        echo "Woof! My name is $this->name and I am a $this->breed.";
    }
}
```

```
$dog1 = new Dog('Buck', 2, 'Bulldog');
$dog1->speak();
```

VIVIFY
ACADEMY

Exercise III

Dodati konstruktore u klasama. Koristiti **parent::** za pozivanje konstruktora roditeljskih klasa.