Dizajni dhe Zhvillimi i Web-it

PHP e Orientuar në Objekte

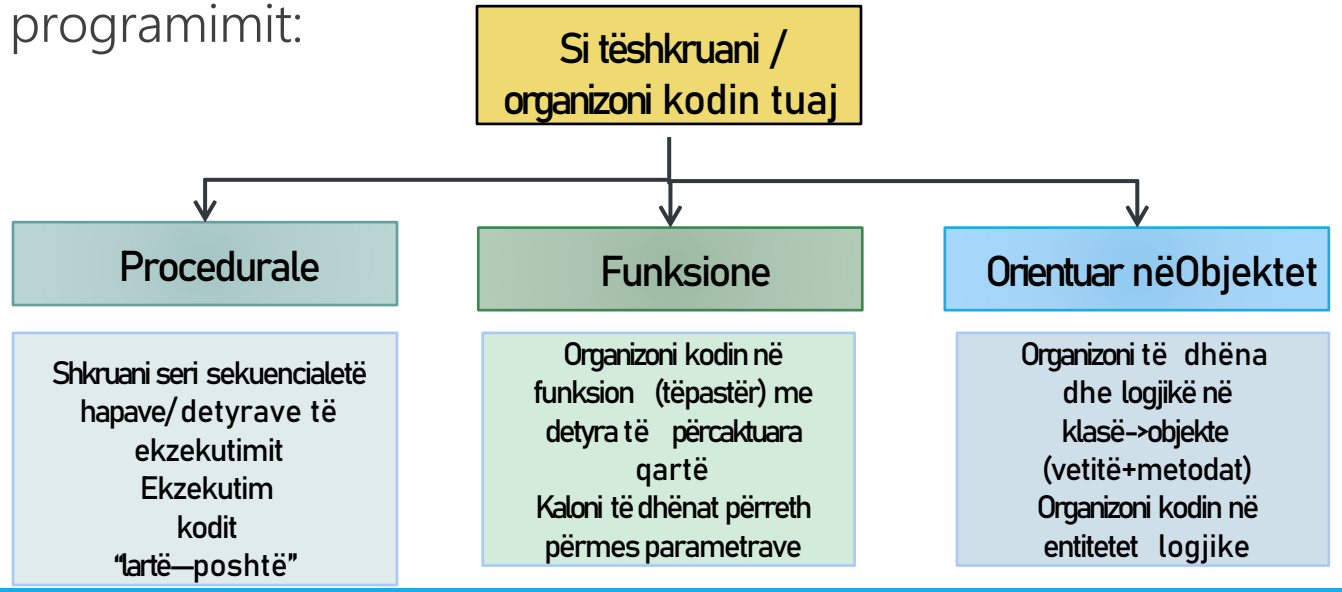
Greta AHMA
Greta.ahma@ubt-uni.net
2022/2023

PHP: Programim i Orientuar në Objektet

PJESA PARË

Paradigmat bazë të Programimit

- **Programimi** është *një proces kreativë* i realizuar nga **programuesit** për të *udhëzuar* **kompjuter** se si të bëjë **një detyrë**.
- Një **program** është një **grup udhëzimesh** që i *tregojnë* një kompjuteri se çfarë të bëjë në mënyrë që të *ralizohet* me një zgjidhje për një problem të veçantë.
- Ekzistojnë një numër i qasjeve alternative ndaj procesit të programimit, të referuara si paradigma të programimit:



Programimi: Procedural *vs* Orientuar nga Objektet

- Qasja **procedural** e programimit:
  - Në stilin procedural të programimit, të gjitha *funksionet* dhe *variablat qëndrojn së bashku* në fushë-veprimi global.
  - Kodi *organizohet* në skripta sekuenciale *ekzekutuese* në pëjes të vogla ose me vetëm *disa* file.
  - Ky model i programimit rrjedh nga programimi i *strukturuar*, bazuar në konceptin të *thirrjës* së procedures.
- **Programimi i Orientuar në Objektet (OOP)** për çfarë është mirë?
  - OOP është një teknikë që zgjidh problemet duke i zbërthyer ato në objekte.
  - **Grupon** të gjitha **vetit (variablat)** dhe **funksionet (sjelljet)** e një **teme** të veçantë në **një klasë të vetme**. *Eliminojnë* nevojën për **përsëritje**.
  - Është një **stil i programimit** që e bazon të *menduarit* në **objektet e botës reale**.
  - Kodi brenda klasave është i in-kapsuluar brenda *fushëveprimit* të klasës, jashtë mundësive të fushëshveprimit global.

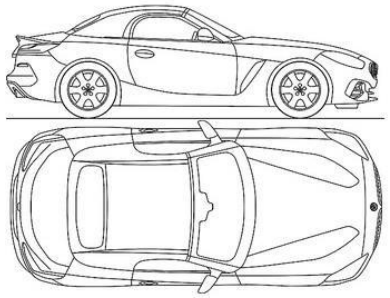
Konceptet baze në OOP

- OOP- **Konceptet** paraqiten si **objekte**
  - Objektet kanë **veti** të cilat përmbajnë **informacion** rreth objektit
  - **Funksionet** e lidhura me objektet quhen **metoda**
- Parimet e programimit të orientuar në objekteve
  - PHP është një gjuhë **skriptimi** e **orientuar në objekte**; mbështet të gjitha parimet e mëposhtem.
- Parimet e mëposhtem arrihen nëpërmjet;
  - **Enkapsulimi** *(fshehja e informacionit-* duke deklarua *vetit* **private***):* nëpërmjet përdorimit të metodave të *qasjes* - të definohen **public** "**get()**" dhe "**set()**"etj.
  - **Trashigimia:** nëpërmjet përdorimit të fjalës kyçe " **extends**". Një **fëmijë-klasë** ose **nënklasë** në OOP mund të *trashëgojë* të gjitha **tiparet** e prindit ose superklasës së saj.
  - **Polimorfizmi:** nëpërmjet përdorimit të fjalës kyçe "**implements**". Klasa kanë *funksione të ndryshme* ndajnë një **interface** të *përbashkët* ose klasa **abstrakte**. (bazuar në metodat *overloading* dhe *overriding*)

Klasat dhe Objektet

- Ç'ka është klasa?
  - Në OOP, një klasë është një *plan* për *krijimin* e objekteve (të një *strukture* të veçantë të të dhënave) *ofron* vetit ose atributet dhe *implementimin* e sjelljeve (*funksioneve* ose *metodave*). Ose thënë ndryshe një plan që përshkruan detajet, formën dhe mënyrën se si funksionon një object.
- Ç'ka është Objekt?
  - Objekti është instancë e *klases*, ju mund të *ndërtoni* më shumë se një objekt nga një klasë.
  - p.sh si *krijimi* i **veturave** të shumta nga **një plan**.

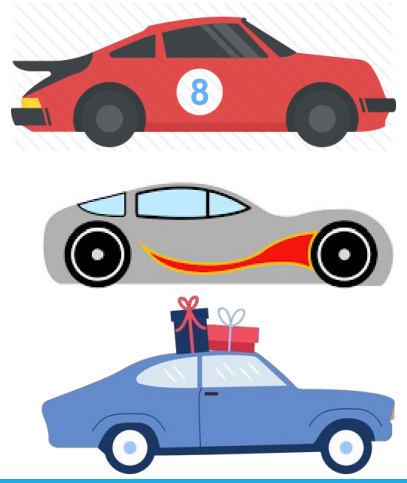
Klasa: `Vetura`



prodhuesi:
modeli: ngjyra:

```
straton()
ndalon()
ndrronSpejtsin()
```

Objektet



Një **klasë** *shndërrohet në një* **objekt** *kur instantohet me fjalën kyçe "***new***".*

prodhuesi: Porsche
modeli: 911
ngjyra: Kuqe

prodhuesi:
Audi modeli:
A9 ngjyra: Hiri

prodhuesi: Ford
modeli: Mustang
ngjyra: Kalter

Deklarimi i Klasës (vetit & metodat) në PHP

- Sintaksa e deklarimit të **klasës** në PHP:
  - Deklarimi i klasës në PHP fillon me fjalën kyçe '**class**' duke me emrin e klasës tuaj si '**KlasaIme**'.

```php
<?php
class KlasaIme
{
  //Ketu shtoni deklarimi e vetive.
  //Ketu shtoni deklarimi e metodave
}
?>
```

```php
class Vetura{
    public $prodhuesi;
    public $modeli;
    public $ngjyra;

    public function straton(){...;}
    public function ndalon(){...}
    public function ndrronSpejtsin(){...}
}
```

Deklarimi i vetive

Deklarimi i metodave

- Krijimi i një Objeti
  - *Krijimi* i një objekti nga një klasë ekzistuese quhet **instancimi i objektit**
  - Për të krijuar një objekt *jashtë* një klase, duhet të përdoret fjalen '**new**'.

```php
<?php
  $objIm = new KlasaIme;
?>
```

```php
Porsche = new Vetura();
Audi = new Vetura();
Ford = new Vetura();
```

Definimi i vetive në të një Klase në PHP

- **Variablat** e anëtarëve të klasës *quhen* veti. Ndonjëherë ato referohen si atribute ose fusha.
  - Vetit *mbajnë* të dhënat përkatese dhe *lidhur* me klasën në të cilën ajo ka qenë e definuar.

- **Vetia/atributi** deklarura me **`var`** (PHP 4) do të përcaktohet si public.

- **Menaxhimi e qasjes** (vizibiliteti) së anëtarëve/variableve.
  - **public:** Vetia mund të jetë e qasshme nga jashtë klasës, ose nga file ose nga një klasë tjetër
  - **private:** Asnjë qasje nuk lejohet përveq mbrenda klasës ku është definuar, as nga file, as nga një klasë tjetër.
  - **protected**: Nuk lejohet qasje nga jashtë klasës, përveç një klase që është një fëmijë i klasës prind.

Deklarimi i Objekteve të OOP në PHP

- Ne mund të krijojmë objekte të shumta nga një klasë.
  - Këto objekte quhen instanca të klasës, ky proces quhet instancimi.

```php
class Vetura
{
  public $brendi="Audi";
  public $lloji="A7 3.0 TDI";
 }

$brendV1 = new Vetura();//mundeni edhe vetem new Vetura
echo "Une vozis Veture: ",$brendV1->brendi."-",$brendV1->lloji;
```

- Pas deklaratës së klasës, mund të krijom instanca nga ajo.

```php
$brendV1 = new Vetura();//mundeni edhe vetem: $brendV1 = new Vetura
```

Qasjan të antareve të klasës duke përdorur objektin

- **Operatori i objektit** (->)
  - Për të thirrur një metodë jasht klases përdorni operatorin e objektit (->)

```
$brendV1 = new Vetura();
echo "Une voziti Veture: ", $brendV1->brendi."-",$brendV1->lloji;
```

- Çfarë është **$this** në PHP?
  - Në PHP fjalë kyçe **$this** i referohet objektit aktual të klasës.
  - **$this** ju lejon të **qasemi vetive** dhe **metodave** të **objektit** aktual brenda klasës duke përdorur **operatorin** e objektit **(->)**:

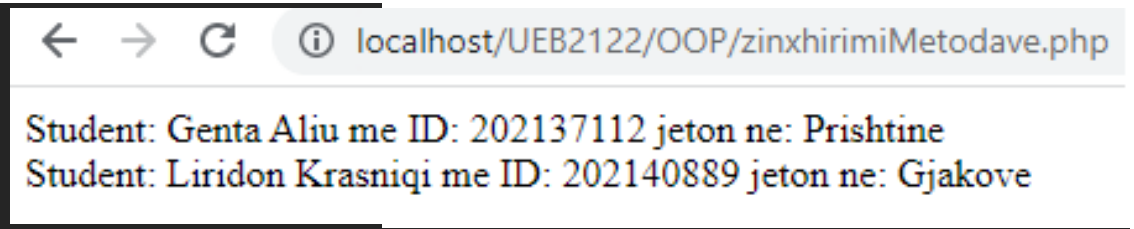
```
$this -> emriVetis;
$this -> emriMetodes();
```

shembull: përdorimi i fjales kyqe $this

```php
<?php
    class Vetura{
        //Deklarimi i Vetive
        public $brendi="Audi";
        public $modeli;
        public $lloji;

        //Deklarimi i metodes
        public function shfaqVeturen(){
            return $this->brendi;
        }
    }
    $vetura01=new Vetura();
    echo "<h3> Vetura qe une vozis eshte:<i> ".$vetura01->shfaqVeturen()."</i></h3>";
?>
```

lidhja zingjirore e Metodave & Vetive në PHP

```php
<?php
class dhenatStudentit {

 public $id = '';
 public $emri = '';
 public $adresa ='';

 public function setStudentId($id) {
  $this->id = $id;
  return $this;
 }

 public function setStudentEmri($emri) {
  $this->emri = $emri;
  return $this;
 }

 public function setStudentAdresa($adresa) {
  $this->adresa = $adresa;
  return $this;
 }

 public function getStudentInfo() {
  return 'Student: '. $this->emri. ' me ID: '.$this->id. ' jeton ne: '. $this->adresa;
 }

}
```



Lidhja e metodave me zinxhirimi

```php
$obj = new dhenatStudentit();
//Zinxhirimi i metodave
$infoStudentit=$obj->setStudentId(202137112)
                   ->setStudentEmri('Genta Aliu')
                   ->setStudentAdresa('Prishtine')
                   ->getStudentInfo()."<br>";
 echo ($infoStudentit);

$obj = new dhenatStudentit();
//Printimi i rezultatit pa zhinxhirim
$obj->setStudentId(202140889);
$obj->setStudentEmri('Liridon Krasniqi');
$obj->setStudentAdresa('Gjakove');
echo ($obj->getStudentInfo());
```

Deklarimi i Metodave të OOP në PHP

- Metodat përdoren për të kryer veprime.
  - Në OOP në PHP, metodat janë funksione *brenda* klasave.
- Deklarimi i metodave në OOP-PHP:

```php
<?php
    class Vetura{
        //Deklarimi i Vetive
        public $brendi;
        public $modeli;
        public $lloji;

        //Deklarimi i metodes
        public function shfaqVeturen($brendi){
            $this->brendi=$brendi;
        }
    }
    $bmw=new Vetura();
    $bmw->shfaqVeturen('BMW');
    echo "<h3> Vetura qe une vozis eshte:<i> ".$bmw->brendi."</i></h3>";
?>
```

Deklarimi i metodave në OOP-PHP

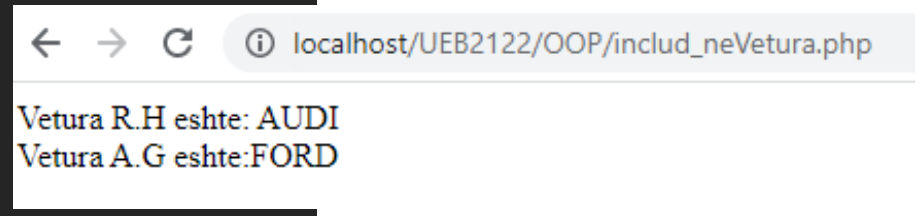
Qasja në metoda dhe shfaqja e rezultatit

```php
<?php
 include 'deklarimiMetodes.php';
?>
<!DOCTYPE html>
<htm>
<head>
    <title>Vetit dhe Metodat POO-PHP</title>
</head>
<body>
<?php
    //Objekti 1-audi
     $audi = new Vetura();
     $audi-> shfaqVeturen("AUDI");
    echo 'Vetura R.H eshte: ',$audi->brendi,"<br>";
    //Objekti 2-ford
    $ford = new Vetura();
    $ford-> shfaqVeturen("FORD");
    echo 'Vetura A.G eshte:',$ford->brendi;
 ?>
</body>
</htm>
```

Referencimi/lidhja e klasës me Index në OOP-PHP

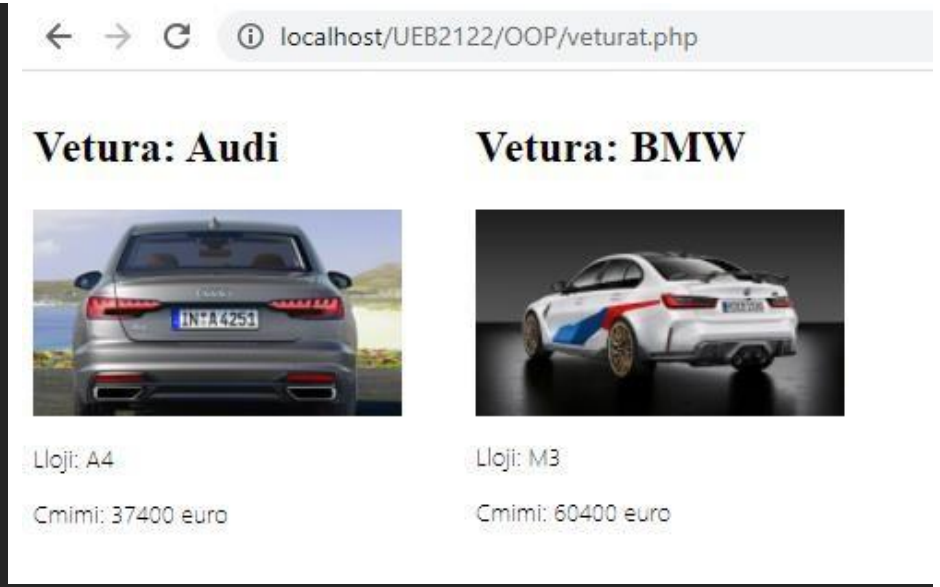
Deklarimi i instances së Objektit

Thirrja e *metodes* nga Klasa në Objekt



Deklarimi i Metodave të OOP në PHP...

```php
<?php
class Vetura{
    const br='<br/>';
    public $brendi;
    public $lloji;
    public $cmimi;
    public $foto;
    public $folderi='fotoVetura/';

    public function setVetura($newBrendi,$newLloji,$newCmimi,$newFoto){
        $this->brendi=$newBrendi;
        $this->lloji=$newLloji;
        $this->cmimi=$newCmimi;
        $this->foto=$newFoto;
    }

    public function printo_InfotVetures(){
        echo "<div style='float:left;margin-right:40px;'>";
        echo "<h2>Vetura: {$this->brendi}</h2>";
        echo "<img style='width:200px;' src='{$this->folderi}{$this->foto}'>".self::br;
        echo "<p style='font-family:Segoe UI Light;font-size:10pt;'>Lloji:
                {$this->lloji}</p>";
        echo "<p style='font-family:Segoe UI Light;font-size:10pt;'>Cmimi:
                {$this->cmimi} euro</p>";
        echo "</div>";
    }
}
```



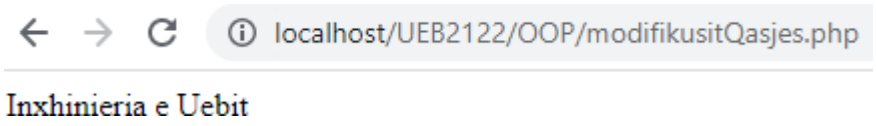

```php
//Krijimi i Objekteve
$vetura = new Vetura;
$vetura->brendi='Audi';
$vetura->lloji='A4';
$vetura->cmimi=37400;
$vetura->foto='AudiA4.jpg';

$vetura->printo_InfotVetures();

$vetura2 = new Vetura;
$vetura2->setVetura('BMW','M3',60400,'BMWM3.jpg');
$vetura2->printo_InfotVetures();
```

modifikuesit e qasjes në PHP

- PHP ka tre modifikues të qasjes: **public**, **private** dhe **protected**. Ne do të përqendroheni në public dhe private.
  - **Modifikuesi i qasjes public** ju ***lejon të përdorni vetitë*** dhe ***metodat*** si nga brenda ashtu edhe nga jashtë **klasës**.
  - **Modifikuesi i qasjes private** ju ***pengon të qasni vetive*** dhe ***metodave*** nga jashtë **klasës**.
- **Modifikuesi i qasjes public**
  - nëse vetia dhe metoda përcaktohen si **public**, kodi **jashtë klasës** mund të ***ndërveprojë drejtpërdrejt*** **me to**.
- **Modifikuesi i qasjes private**
  - Ne mund të ***parandalojmë qasjen*** në ***vetitë*** dhe ***metodat*** **brenda klasave** tona.
  - nëse i përcaktojmë me modifikuesin e qasjes **private** në vend të modifikuesit të qasjes **public**.

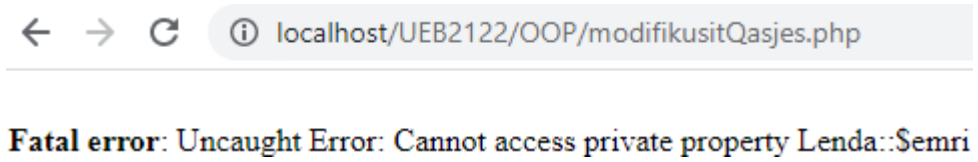
modifikuesi i qasjes public

```php
class Lenda
{
    public $emri;

    public function shfaqEmri()
    {
        return $this->emri;
    }
}
// Krijon objektin 'inxhUebit'
$inxhUebit = new Lenda();

// set emrin e objektit "Inxhinieria e Uebit'
$inxhUebit->emri = 'Inxhinieria e Uebit';

// get emrin e objektit "Inxhinieria e Uebit' dhe shfaq
echo $inxhUebi->shfaqEmri(); // Inxhinieria e Uebit
```



modifikuesi i qasjes private

```php
class Lenda
{
    //public $emri;
    private $emri;

    public function shfaqEmri()
    {
        return $this->emri;
    }
}
$bDH = new Lenda();
$bDH->emri ='Sistemet Bazes se te Dhenave';
echo $bDH->shfaqEmri(); // error
```



Qasja në vetie private

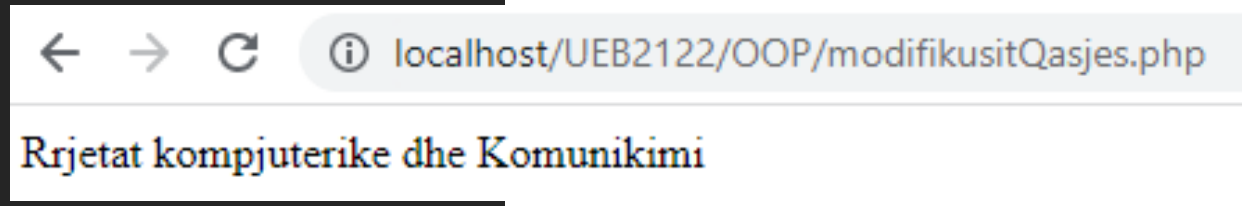
- Si të qasemi një vetie **private**?

- Për të manipuluar vlerën e një vetie **private**, ju duhet të përcaktoni një metodë **public** dhe të përdorni metodën për *menaxhimin e vetis private*.

- Zakonisht duhet të përcaktoni *dy lloje metodash publike* për të *menaxhuar* një veti **private**:
  - Një **get()** kthen vlerën e vetis private.
  - Një **set()** vendos një vlerë të re për vetin private.

Qasja në vetie private: përdorimi i get() dhe set()

```php
class Lenda
{
    /*Modifikuesi i qasjes privat mohon qasjen
    në metodë jashtë fushës së veprimit të klasës*/
    private $emri;

    public function getEmri()
    {
        return $this->emri;
    }
    /*modifikuesi i qasjes publik lejon
      qasjen në metodë nga jashtë klasës*/
    public function setEmri($newEmri)
    {
        $this->emri=$newEmri;
    }
}
$rrjeta = new Lenda();
//Përcakton ermin e lendes
$rrjeta ->setEmri('Rrjetat kompjuterike dhe Komunikimi');
//Merr emrin e lendes
echo $rrjeta->getEmri(); //
```



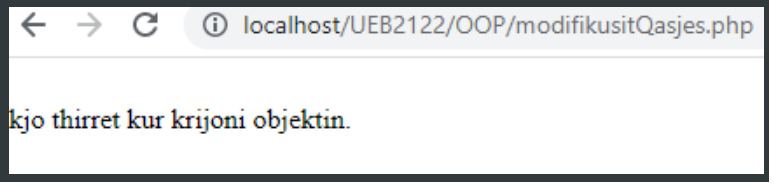
Metodat magjike

- Këto janë metoda që thirren automatikisht kur ndërmerren veprime të caktuara, megjithëse mund të thirren edhe manualisht.
- Zakonisht parashtesa me dy nënviza (**__metoda(parametrat,.....)**)
- Metodat magjike
  - **__construct** metodën e konstruktorit; thirret në një objekt sapo të instantcohet. Mund të pranojë parametra.
  - **__destruct** metodën e destruktorit; thirret kur një objekt ndalon së përdoruri/referencuar ose gjatë sekuencës së mbylljes
  - **__toString** thirret kur doni të përdorni një objekt si String (p.sh. echo $my_object) ku kthen stringun të objekti
  - **__clone** quhet kur provoni te klononi një objekt (p.sh. $new_object = clone $old_object) i dobishëm për mos-referencimin e ndonjë ID unike, etj.

Deklarimi i __construct()

- Ne përdorim **__construct()** për të bërë diçka **sapo të krijojmë një objekt** nga një klasë.
  - Një metodë e këtij lloji quhet **konstruktor**.
  - Zakonisht ne përdorim **konstruktorin** për të **vendosur** (**set**'at) një **vlerë** në një veti/variable.

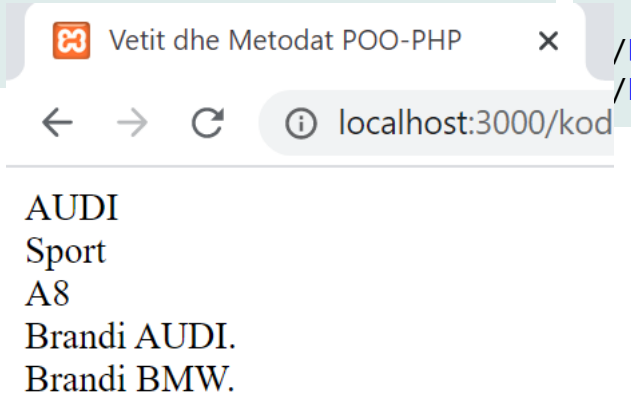
```php
class MyClass {
    private $someProperty = 'a default value';
    private $anotherPropert;
    public function __construct()
    {
    echo "<br>kjo thirret kur krijoni objektin.";
    }
    public function myMethod()
    {
    echo "Kjo është metoda ime që po ekzekutohet";
    }
}
//Krijon instance
$a = new MyClass;
```



Shembull i përdorimit të Konstruktorëve & Destruktorëve

```php
<?php
    class VeturaIME02{
    //Deklarimi i Vetive
    public $brendi;
    public $modeli;
    public $lloji;

    //Konstruktorët
    public function __construct($newBrendi, $newModel, $newLloji){
        $this->brendi=$newBrendi;
        $this->modeli=$newModel;
        $this->lloji=$newLloji;
    }
    //Destruktorët
     public function __destruct() {
        echo "Brendi {$this-> brendi}.<br>";
     }
    }
?>
```

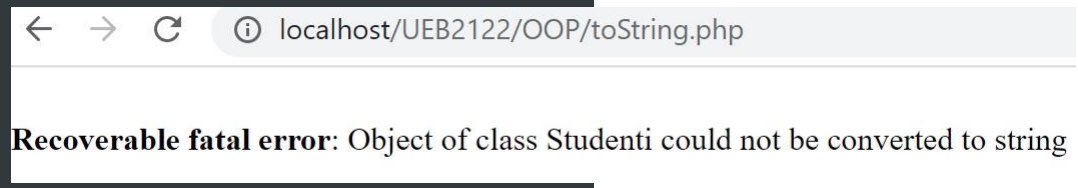
```php
<?php
 include 'klasaVetura02.php';
?>
<!DOCTYPE html>
<htm>
<head>
    <title>Vetit dhe Metodat POO-PHP</title>
</head>
<body>
    <?php
     //Objekti01-Vetura01
      $vetura01 = new VeturaIME02('AUDI','Sport','A8');
      echo $vetura01->brendi,"<br>";
      echo $vetura01->modeli,"<br>";
      echo $vetura01->lloji,"<br>";

      $vetura01 = new VeturaIME02("BMW","SUV","X5");
    ?>
/body>
/htm>
```



Metoda _toString() në PHP

- Metoda __toString() nuk pranon asnjë parametër dhe kthen një varg Stringu.
  - P.sh rasti i shtypjes se rezultati direkt nga instancimi i konstruktorit, shfaqja e errorit.

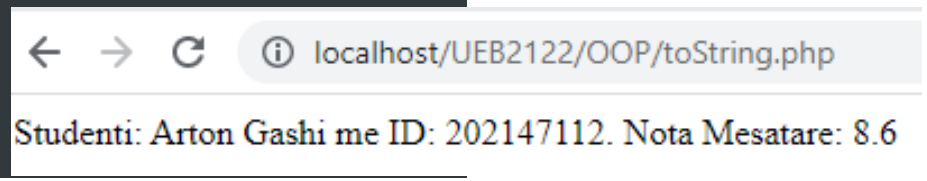
```php
<?php
class Studenti
{
    private $studentID;
    private $emriStudentit;
    private $notaMesatare;

    public function __construct($studentID,$emriStudentit,$notaMesatare) {
        $this->studentID = $studentID;
        $this->emriStudentit = $emriStudentit;
        $this->notaMesatare = $notaMesatare;
    }
}

$studenti1 = new Studenti('202147112', 'Arton Gashi',8.6);
echo $studenti1;
```



Metoda __toString() në PHP...

```php
class Studenti
{
    private $studentID;
    private $emriStudentit;
    private $notaMesatare;

    public function __construct($studentID,$emriStudentit,$notaMesatare) {
        $this->studentID = $studentID;
        $this->emriStudentit = $emriStudentit;
        $this->notaMesatare = $notaMesatare;
    }

    public function __toString()
    {
        return "Studenti: $this->emriStudentit me ID:
            $this->studentID. Nota Mesatare: $this->notaMesatare";
    }
}
$studenti1 = new Studenti('202147112', 'Arton Gashi',8.6);
echo $studenti1;
```

Përdorimi i metodes __toString()



metodat dhe vetitë statike në PHP

- Funksionet **statike** dhe klasat **statike** mund të *qasemi lehtësisht* pa *krijuar* **instance** të asaj **klase**.
- **Vetitë** dhe **metodat statike** *asocohen* me klasën, ndërsa *vetitë* dhe *metodat* **jostatike** *asocohen* me instanc.
  - Është e pamundur që një **metodë statike** të *qaset* një **variabël jostatike** ose **metoda jostatike** sepse këto *kërkojnë* që të *krijohet* një **instance i klasës**.
  - Megjithatë, një **metode statike** mund të *qasemi* nga një *metodë* **jostatike** brenda së **njëjtës klasë** duke përdorur **self::** ose nga nje **klas tjeter**.

Pra, në vend që të shkruajm:

```
$objekti = new MyClass();
$objekti.someMethod();
```

Mund të shkruajmë:

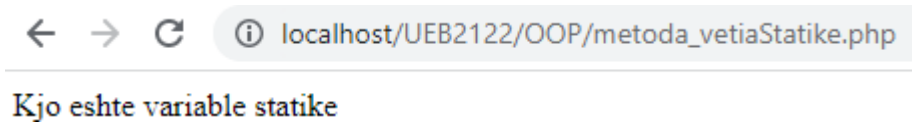
```
MyClass::myStaticMethod();
```

Rastet kryesore në të cilat marrim *parasysh përdorimin* e **metodave** dhe **vetive statike** *janë kur na duhen* si **numërues** dhe për klasa të **shërbimeve**.

metodat dhe vetitë statike në PHP...

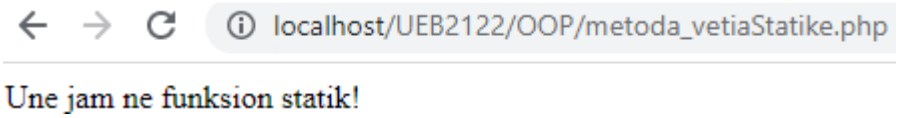
Deklarimi i një *variable* dhe *metode* statike

```php
<?php
class myKlas {
    //Deklarimi i një variabli statike
    public static $variablaStatike=
    'Kjo eshte variable statike';

    //Deklarimi i një metode statik
    public static function newFunction() {
    echo self::$variablaStatike;
    }
}

myKlas::newFunction()
?>
```



Qasja në *metoden statike* nga një *metode jostatike konstruktorit* brenda se njëjtës klasë

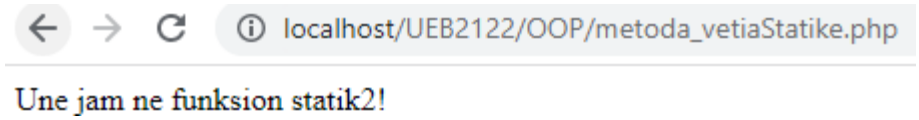
```php
class klasa2 {
    public static function myMethod(){
        echo 'Une jam ne funksion statik!';
    }
    public function __construct(){
    /*qasja në funksionin statik të përshëndetjeve
    përmes metodës jostatike construct().*/
       self::myMethod();
    }
}

new klasa2();
```



metodat dhe vetitë statike në PHP...

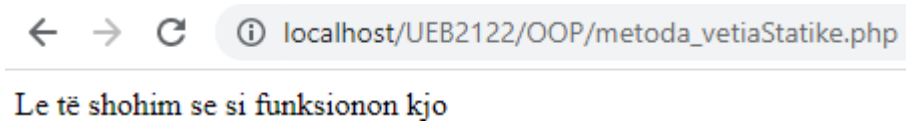
Qasja në **metoden statike** nga një *metode jostatike konstruktorit* brenda se njëjtës klasë

```php
class klasa2 {
    public static function myMethod(){
        echo 'Une jam ne funksion statik!';
    }
    public function getME(){
    /*qasja në funksionin statik të përshëndetjeve
    përmes metodës jostatike.*/
       self::myMethod();
    }
}
$x = new klasa2();
$x->getME();
```

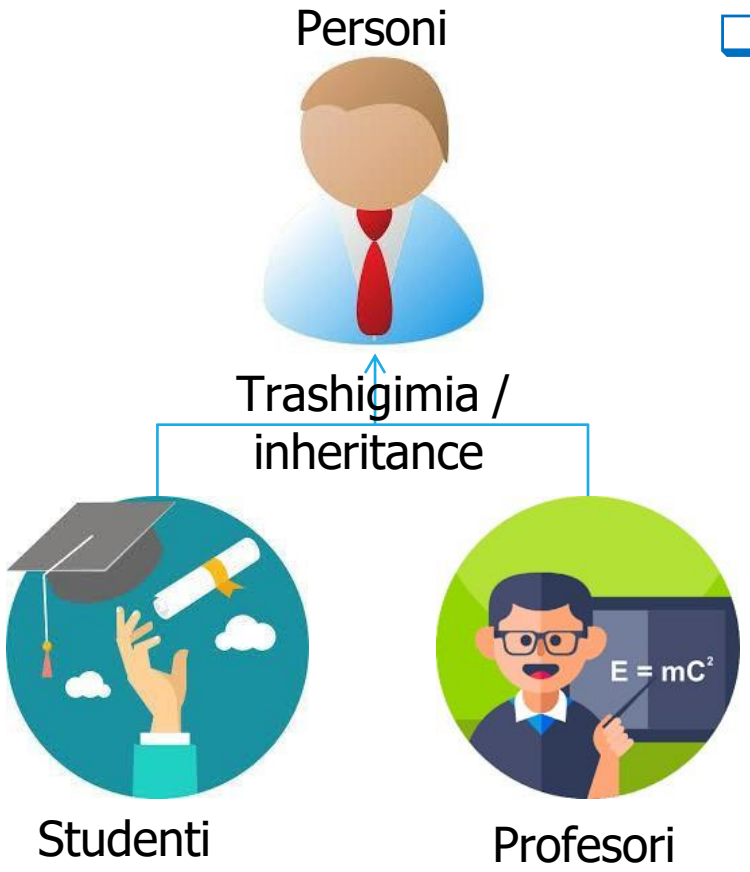


Qasja e një metodë statike në një klasë nga një klasë tjetër duke e bërë publike metodën statike.

```php
class klasaPare {
    public static function metodaPare(){
      echo "Le të shohim se si funksionon kjo";
    }
}
class klasaDyte{
    public function metodaDyte(){
        klasaPare::metodaPare();
    }
}

$klasa2 = new klasaDyte;
echo $klasa2 -> metodaDyte();
```



Trashëgimia (inheritance) në PHP



- Çfarë është **trashëgimia**
  - **Trashëgimia** është një *relacion* midis dy ose më shumë klasave ku klasa e *trashigues* **trashëgon vetitë** e klasave *ekzistuese* (bazë).
  - **Trashëgimia** lejon një klasë të *ripërdorë* kodin nga një **klasë tjetër** pa e **dubluar** atë.
  - Kodin e shkruar **një herë** në klasën prind *përdorni* kodin si në **klasën prind** ashtu edhe në **klasën e fëmijës**.
  - Për të definuar një klasë që **trashëgohet** nga një *klasë tjetër* *përdorni* fjalën kyçe **extends**.

```
class Prindi {
 // Kodi i klasës së prindit
    }
class Femija extends Prindi {
 // Fëmija mund të përdorë kodin e klasës së prindit
}
```

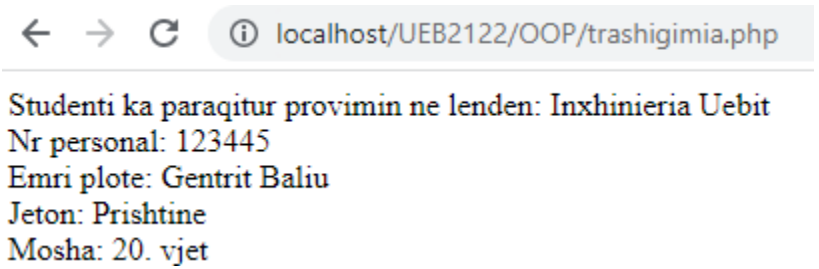
Shembull1: implementimi i trashigimis në PHP

Krijimit të klases Personi: në këtë rast është superKlasa/prinidi

```php
<?php
 class Personi{
     private $nrLeternjoftimit;
     public $emri;
     public $adresa;
     protected $mosha;

     public function __construct($newNrLetnj,
                                 $newEmri,
                                 $newAdresa,
                                 $newMosha)
     {
         $this->nrLeternjoftimit=$newNrLetnj;
         $this->emri=$newEmri;
         $this->adresa=$newAdresa;
         $this->mosha=$newMosha;
     }
     public function shfaqDhenat(){
         echo "Nr personal: {$this->nrLeternjoftimit}<br>
                Emri plote: {$this->emri}<br> Jeton:
                            {$this->adresa}<br>
                Mosha: $this->mosha. vjet";
     }
 }
```

```php
class Studenti extends Personi {

    protected $provimi;

    public function setProvimi($provimi){
        $this->provimi=$provimi;
    }
    public function getProvimi(){
         $provimi=$this->provimi;
         return  $provimi;
    }

 }

 $studenti1= new Studenti('123445','Gentrit Baliu','Prishtine',20);
 $studenti1->setProvimi('Inxhinieria Uebit');

 echo "Studenti ka paraqitur provimin ne lenden: "
                    .$studenti1->getProvimi()."<br>";

echo $studenti1->shfaqDhenat()."</br>";
?>
```

Klasa **Student** (femije) trashigon vetit dhe metodat e klases prind **Personi**



Shembull2: implementimi i trashigimis në PHP

```php
class User {

    public $name;
    public $email;

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

    public function getType() {
        return $this->type;
    }

}

class Admin extends User {

    public $permissionLevel;
    public $type = 'Admin';

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

}
```

```php
class Member extends User {

    public $dateAdded;
    public $type = 'Member';

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

}
```

```php
<?php

include('user.php');

$user1 = new Member('Brad', 'brad@azpixels.com', 'Nov 14');
echo $user1->getType();


$user2 = new Admin('Danny', 'danny@danny.com', 3);
echo $user2->getType();
```

Klasat dhe metodat abstrakte të PHP OOP

- Ne përdorim **klasa abstrakte:**
  - kur duam *të shkruajë një metodë të* caktuar të klases por jemi të sigurt **vetëm për emrin** e metodës, dhe **jo detajet** se si *duhet t*ë **shkruhet**.
  - përdoret për të **zgjeruar** klasat e tjera, kur **më shumë se një klasë** *përdor* të **njëjtën logjikë**.
- **Klasat abstrakte** është kur **klasa prind** *emron një* **metodë**, por kjo **detyre** duhët *ekzekutua*mga **klasa fëmijit** i saj.
- **Karakteristikat** e një **klase abstrakte**
  - **klasa abstrakte** *nuk mund të instancohet* drejtpërdrejt do të thotë se ne *nuk mund të krijojmë* **objekte** të një klase abstrakte, ato *mund* **të trashëgohen** vetëm.
  - **anëtarët abstraktë** dhe **joabstraktë** *mund të jenë pjesë* e **klasës abstrakte**.
  - **min një** *metodë abstrakte* duhet të *deklarohet* në **klasën abstrakte**.
  - klasat abstrakte mbeten *gjithmonë* **publike**.

Shembull: Rrastiti i përdorimit të klaseve abstrakte në OOP-PHP

```php
<?php
    // abstract class
    abstract class Vehicle {

        // protected variable
        protected $name;

        // non-abstract public function start
        public function start() {
            echo $this->name. " - Engine start...<br/>";
        }

        // non-abstract public function stop
        public function stop() {
            echo $this->name. " - Engine stop...<br/>";
        }

        // non-abstract public function setName
        public function setName($name) {
            $this->name = $name;
        }

        // abstract function mileage
        abstract public function mileage() {

        }
    }

?>
```

```php
<?php
    // child class
    class Car extends Vehicle {

        public function mileage() {
            echo "I am " . $this->name . "<br/>";
            echo "My mileage range is - 15 to 22 Km/L";
        }

    }

?>
```

```php
<?php

    // child class
    class Motorcycle extends Vehicle {

        public function mileage() {
            echo "I am " . $this->name . "<br/>";
            echo "My mileage range is - 35 to 47 Km/L";
        }

    }

?>
```

```php
<?php

    $car = new Car();
    $car->setName("BMW X1");
    $car->mileage();

?>
```

interface në PHP

- Me ndihmën e **interface**, zhvilluesit *mund të krijojnë* programe dhe të **specifikoj metodat publike** që *implementohen* nga një klasë, duke *fshehur kompleksitetin* dhe *detajet se si implementohen* **metodat** e **veçanta**.
- Ky zakonisht quhet **niveli tjetër i abstraksionit**.
  - Një **interface** i përckaton që klasat e saj fëmijë, **metodat abstrakte** që duhet të implemntohen.
- deklarimi dhe implemntimi një **interface** :

```
interface interfaceName {
    // metodat abstrakte
}
class Child implements interfaceName {
    // definon metodat e interfaces dhe mund te kete kodeni e vete
}
```

interface për klasat që trajtojnë Veturat, e cila i angazhon të gjitha klasat e saj fëmijë në metodat setModel() dhe getModel().

```php
interface Car {
    public function setModel($name);

    public function getModel();
  }

  Class miniCar implements Car {
    private $model;

    public function setModel($name)
    {
      $this -> model = $name;
    }

    public function getModel()
    {
      return $this -> model;
    }
  }
```

```php
interface Car {
    public function setModel($name);

    public function getModel();
  }

  interface Vehicle {
    public function setHasWheels($bool);

    public function getHasWheels();
  }
```

Ne mund të implementojmë një numër interface'ave në të njëjtën klasë.

```php
class miniCar implements Car, Vehicle {
  private $model;
  private $hasWheels;

  public function setModel($name)
  {
    $this -> model = $name;
  }

  public function getModel()
  {
    return $this -> model;
  }

  public function setHasWheels($bool)
  {
    $this -> hasWheels = $bool;
  }

  public function getHasWheels()
  {
  return ($this -> hasWheels)? "has
                wheels" : "no wheels";
  }
}
```

Shembull: Rrastiti i përdorimit të interface në PHP

```php
<?php
    // interface declaration
    interface WebApp {

        // methods declaration
        public function login($email, $password);

        public function register($email, $password, $username);

        public function logout();

    }

?>
```

```php
<?php
    // class declaration
    class Studytonight implements WebApp {

        // methods definition
        public function login($email, $password) {
            echo "Login the user with email: " . $email;
        }

        public function register($email, $password, $username) {
            echo "User registered: Email=".$email." and Username=".$username;
        }

        public function logout() {
            echo "User logged out!";
        }

    }

?>
```

Rasti i metodes **overloading** në PHP

```php
<?php

class Sum {

    public function getSum() {

        return 0;
    }

    public function getSum($x) {

        return $x;
    }

    public function getSum($x, $y) {

        return $x + $y;
    }
}

$s = new Sum();
echo $s->getSum() . "\n" ;
echo $s->getSum(5) . "\n" ;
echo $s->getSum(3, 4) . "\n" ;
```

```php
<?php
class Sum {

    public function getSum() {

        $sum = 0;
        $args = func_get_args();

        if (empty($args)) return 0;

        foreach ($args as $arg) {

            $sum += $arg;
        }

        return $sum;
    }
}
$s = new Sum();

echo $s->getSum() . "\n" ;
echo $s->getSum(5) . "\n" ;
echo $s->getSum(3, 4) . "\n" ;
echo $s->getSum(3, 4, 7) . "\n" ;
```

Pyetje