Web2, curs 9, Aplicatii

Nota: Notiunile teoretice se regasesc in cursurile 11 si 12 de pe www.invata-online.ro

ex01 – OOP. Notiuni

1. Sa se creeze clasa Cursant avand atributele prenume, nume, email. Sa se creeze apoi doua obiecte $p1 si $p2 ale clasei Cursant. Sa se atribuie valori celor doua obiecte si sa se afiseze cu litere mari numele si prenumele si cu litere mici emailul fiecarui cursant.

Indicatii:

1. Sintaxa de declarare a unei clase:

```
class ClasaMea{
        <modificator_acces> atribut_1 [=valoare];
        <modificator_acces> atribut_2 [=valoare];
        ...
        <modificator_acces> atribut_n [=valoare];

        [<modificator_acces>] function metoda_1([lista_parametri]){
        //instructiuni
        }
        ...
        [<modificator_acces>] function metoda_m([lista_parametri]){
        //instructiuni
        }
}
```

- modificator_acces poate fi
  - **public** – atributul/metoda este vizibil complet in afara clasei
  - **protected** - atributul/metoda este vizibil doar in clasele derivate (ce mostenesc) din aceasta clasa
  - **private** - atributul/metoda nu este vizibil decat in alte metode din cadrul aceleiasi clase. Acest membru este "ascuns" in interiorul clasei, nu poate fi accesat din exterior.
  - daca modificatorul de acces pentru o metoda nu este specificat, se considera implicit public.
- Crearea de obiecte ce apartin unei clase (instantierea clasei) se realizeaza cu ajutorul operatorului **new** urmat de numele clasei sub forma unei functii. Ex: **$p1=new Cursant()**;
- Accesul la atributele unui obiect se realizeaza cu ajutorul operatorului **->**. Numele varibilei ce reprezinta obiectul precede **->**, urmeaza apoi atributul ce este citit sau modificat. Exemplu prin codul **$p->nume** accesam atributul nume al obiectului $p.

2. Schimbati modificatorul de acces al atributului $email din *public* in *private*. Rulati scriptul si observati ca atributul $email nu mai poate fi accesat de codul client (codul din afara clasei).Creati o metoda de tip setter si getter pentru atributele clasei si afisati aceleasi date ca la scriptul anterior.
3. Creati obiectele $p1 si $p2 ale clasei Cursant folsind **metoda contructor** cu parametrii $prenume, $nume, $email.

   Indicatii:

   Metoda constructor se defineste in interiorul unei clase si in general contine cod de initializare a obiectelor nou create

   trebuie denumita **__construct** (precedata de doua caractere underline)

   Ex:

```
class Cursant{
...
function __construct($prenume,$nume,$email){

        $this->prenume=$prenume;

        $this->nume=$nume;

        $this->email=$email;

        }
```

..
$p1=new Cursant('Mihai','Popa','mpopa@gmail.com');

4. Adaugati clasei Cursant constanta curs cu valoarea 'web2'. Pe langa nume, prenume si email, afisati si cursul la care participa $p1 si $p2.

Indicatii:

- Constantele unei clase
- similar cu constantele de sine statatoare, contin valori ce nu se schimba.
- se declara astfel: *const numeConstanta; Numele constantei nu este precedat de $!*
- sunt initializate obligatoriu in momentul cand sunt definite, cu o valoare constanta (nu expresie, variabila, etc) si de un tip scalar (deci nu array)
- nu sunt precedate de modificatorii de acces, sunt automat publice
- constantele clasei exista intr-un singur exemplar, avand aceeasi valoare (si loc de memorie) pentru fiecare obiect al clasei.
- pot fi accesate cu operatorul ::
  - Din afara clasei prin *NumeClasa::numeConstanta*.
  - Din interiorul clasei cu *self::numeConstanta*
    self este un cuvant cheie ce reprezinta clasa curenta

- Adaugati clasei Cursant atributul *$nr* de tip *static private* care va numara cursantii existenti.
- Reveniti asupra metodei *__construct* si incrementati *$nr* (pentru apelul metodelor din interiorul clasei, se va folosi operatorul :: precedat de **self**; ex: *self::$nr++)*
- Creati functia statica *getNr()* care va returna numarul de cursanti ($nr este private si nu poate fi accesat dinafara clasei decat prin intermediul unei metode)
- Creati 3 obiecte dupa care afisati numarul de cursanti. Pentru a referi metodele statice ale clasei, se va folosi operatorul :: precedat de numele clasei; ex: *Cursant::getNr()*
- Distrugeti obiectele cu unset() si apelati din nou *Cursant::getNr()*. Observati rezultatul. Pentru remediere, se va crea functia *__destruct()* care decrementeaza $nr. Metoda destructor sa apeleaza automat la distrugerea obiectelor.

Indicatii:

1. **Atributele statice** apartin clasei, nu obiectelor create, si sunt deci apelate prin intermediul numelui clasei, astfel:
   - ***NumeClasa::$numeAtribut*** - din afara clasei
   - ***self::$numeAtribut*** - din interiorul clasei. **self** fiind un cuvant cheie ce desemneaza clasa curenta
2. **Metodele statice** pot fi apelate atat prin sintaxa asemanatoare atributelor statice, (***NumeClasa::metoda()*** sau ***self::metoda()*** din interiorul clasei) , insa pot fi apelate si prin intermediul unui obiect. Metodele statice nu au acces la atribute si metode ce apartin obiectului curent (folosind sintaxa $this), dar au acces la alte atribute si metode statice.

Tema de lucru:

1. Preluati numele, prenumele, emailul si nivelul de studii al cursantilor dintr-un fisier csv. Afisati in browser lista cursantilor si numarul acestora. (pentru lucru cu fisiere, a se vedea curs 7)

ex02 – OOP. Aplicatie. Exemplu de clasa pentru crearea unui tabel de date

1. Testati functionalitatea clasei DataTable. Permite adaugarea unei linii de date, setarea atributelor unei linii, setarea atributelor unei celule.
2. Folosind clasa DataTable, creati un tabel pe baza datelor din array-ul urmator:

```
array(
   array('id' => '1', 'desc' => 'Dictionar englez-roman',
   'pret' => 24.95),

  array('id' => '2', 'desc' => 'Songs of the Goldfish (set 2CD)',
   'pret' => 100),

  array('id' => '3', 'desc' => 'PHP pentru World Wide Web',
  'pret' => 35),
```

```
 array('id' => '4', 'desc' => 'Dictionar roman-englez',
  'pret' => 39.95));
```

3. Aplicati stilul zebra tabelului de mai sus
4. Adaugati clasei metoda *setColAttributes()* care seteaza atributele unei coloane. Testati setand la alegere un stil pe coloane pentru tabelul anterior.

ex03– OOP. Exemplu de clasa pentru interogarea unei baze de date

1. Creati baza de date cursanti cu tabelul curasanti(cursantId, prenume, nume, email, studii)
2. Creati scriptul **config.php** in care veti initializa parametrii de configugarea a bazei de date:

Ex:

```
define ( 'db_host', 'localhost' );
define ( 'db_user', 'root' );
define ( 'db_password', '' );
define ( 'db_name', 'cursanti' );
```

3. Creati scriptul **clase.php**
4. In **clase.php c**reati clasa QueryDb care va contine:
   a) atributele $db_host, $db_user,$db_pass, $db_name de tip private
   b) metoda __construct
   c) metoda dbConnect care va realiza conexiunea la baza de date si va returna rezultatul conexiunii ($link)
5. Creati scriptul index.php in care:
   a) veti include scripturile **config.php** si **clase.php**
   b) veti crea un obiect $db din clasa QueryDb care va referi baza de date 'cursanti' (*$db=new QueryDb('localhost','root','','cursanti');*)
   c) veti apela metoda dbConnect pentru obiectul $db pentru a realiza conexiunea la baza de date (*$link=$db->dbConnect();*)
6. Reveniti la clasa **QueryDb**
   a) creati metoda *query()* cu parametrul $sql care:
      1. va realiza conexiunea la baza de date prin apelul metodei dbConnect() (*$link=$this->dbConnect();*)
      2. va memora in $result rezultatul interogarii *($result=mysqli_query($link,$sql);)*
      3. daca exista rezultat al interogarii, se va construi si se va returna array-ul $results care va avea drept elemente inregistrarile rezultate in urma interogarii; altfel se va returna FALSE

```
$results=array();
if(!$result){
    return FALSE;
    }else{
        while($d=mysqli_fetch_array($result)){
            $results[]=$d;
            }//end while
        return $results;
                }
}//end function
```

7. Reveniti la **index.php** si testati metoda query()
   a) anulati linia *$link=$db->dbConnect();* pentru ca se va face conectarea prin metoda query()
   b) folosind metoda query() pentru obiectul $db
      1. afisati sub forma de lista numerotata numele si prenumele cursantilor

```
$sql="SELECT * FROM cursanti";
$cursanti=$db->query($sql);

echo '<ol> <b>Lista cursantilor:</b>';
if($cursanti){
    //afisez cursantii
    foreach($cursanti as $cursant){
        echo '<li>'.$cursant['prenume'].' '.$cursant['nume'].'</li>';
        }
    echo '</ol>';
    }else {
        echo 'Eroare la SELECT';
        }
```

      2. afisati statistica pe nivel de studii

```
$sql="SELECT studii, count(*) AS NR FROM cursanti GROUP BY studii";
$statistica=$db->query($sql);
```

```
echo '<ul><b>Statistica pe nivel de studii:</b>';
if($statistica){
    foreach($statistica as $k=>$stat)
        echo '<li>'.$stat['studii'].':'.$stat['NR'].'</li>';
    echo '</ul>';
    }else {
        echo 'Eroare la SELECT';
        }
```

8. **Tema de lucru**:

1. Afisati separat, mai intai lista cursantilor absolventi de facultate, apoi lista cursantilor absolventi de liceu
2. dezvoltati metoda query astfel incat, in functie de un al doilea parametru $type, sa poata realiza si operatii de INSERT, UPDATE sau DELETE asupra bazei de date.
3. Folosind OOP si baza de date cursanti creata anterior, creati o aplicatie care:
    - va afisa sub forma de tabel datele tuturor cursantilor
    - va sterge un cursant atunci cand se selecteaza link-ul Delete din dreptul acestuia
    - va insera datele unui nou cursant, datele fiind preluate prin intermediul unui formular