КАТЕДРА: КОМПЮТЪРНИ СИСТЕМИ И ТЕХНОЛОГИИ
ДИСЦИПЛИНА: ИНФОРМАЦИОННИ СИСТЕМИ

ЛАБОРАТОРНО УПРАЖНЕНИЕ № 7

ТЕМА: PHP като обектно ориентиран език

ЦЕЛ:

Целта на упражнението е студентите да получат практически знания и умения за търсене на информация от MySQL база данни чрез PHP скрипт След упражнението студентите би следвало и да могат да реализират елементарни справки към базата данни чрез PHP.

I. ТЕОРЕТИЧНА ЧАСТ

Обектно ориентираните страни на PHP запазват нещата прости и в синхрон с останалите аспекти на езика. Когато дефинира клас, програмистът определя кои ще бъдат член променливите и методите (променливите и функциите), включени в класа. Екземплярите на класовете в PHP се наричат обекти.

1. Създаване на класове

Клас се създава както в повечето програмни езици чрез ключова дума “class” и името на класа последвани от фигурни скоби в които се намира дефиницията му. Общият синтаксис е следния:

```
class Име_На_Класа {
}
```

В дефиницията на класа се съдържат член променливите и методите. Те трябва да започват с ключова дума, която отговаря областта на видимост на член променливата или метода. Това са съответно: public разрешаващ достъп от всякъде – както от вътре така и от вън класа, protected разрешаващ достъп само от наследниците и оригиналния клас и private – член променливата или метода е достъпен само от рамките на класа. Методите на класа се дефинират точно по същия начин както и функциите извън класовете, с ключова дума “function”. В следващия пример е показано как се дефинират член променливи и методи в класа:

```
Class Име_На_Класа {
    Private $име_променлива1;
    Private $име_променлива2;
    function Име_на_метод{
    }
}
```

Естествено ключовата дума за видимост на член на класа може да не е private, а някой останалите видове public или protected, зависимост от това каква искаме да е видимостта на дадения член. Ако на даден член на клас липсва информация за видимостта му то той се подразбира че е public. За това нашия метод в класа по подразбиране е public.

Функциите(методите) на класа могат да достъпват член променливите на класа по специфичен начин. Не биха могли да имат достъп до тях директно чрез техните имена понеже те не са дефинирани вътре в самите функции. За да могат да ги използват влиза в действие псевдопроменливата $this. Достъпът на функцията до член променлива от класа става по следния начин:

$this -> име_променлива;

Обърнете внимание, че символът $ е поставен пред „this“, а не непосредствено пред член променливата.

2. Създаване на обект от клас

Използването на клас става чрез създаването на екземпляр на класа, наричан още обект. Това става с ключова дума new. Всеки нов обект трябва да бъде създаден и присвоен на променлива:

$променлива = new Име_На_Класа ;

За един клас могат да се създават множество екземпляри. Извикването на член от класа (метод или променлива) става чрез знака „->“. Ето как се вика метод на клас:

$променлива->Име_на_метод();

Естествено за да можем да имаме достъп, по този начин, до член променливите или методите на класа, то те трябва да бъдат декларирани като public. В противен случай ще се генерира съобщение за грешка.

3. Конструктори и Деструктори

Както всеки програмен език и в PHP може да се използват конструктори и деструктори. Конструкторът е функция, която се извиква автоматично в момента, в който създадем обекта. Тя изглежда по следния начин:

```
Function __constructor(списък с параметри){
}
```

Извикването на конструктора казахме, че става при създаването на обект, т.е. по следния начин:

$променлива = new Име_На_Класа (списък параметри);

В PHP 5 е въведена концепцията деструктор, подобно на другите обектно ориентирани езици. Методът деструктор се извиква веднага щом бъдат премахнати всички препратки към определен обект или когато обектът бъде изрично унищожен. Пример за функция деструктор е:

```
Function __destructor () {
echo “<p>Destroying Class</p>”
}
```

Горния пример реално не прави нищо полезно, освен да ни каже в кой момент е изтрит обектът. Деструкторите могат да ни послужат, за да освободим памет , използвана от обектите.

4. Наследяване

Едно от основните предимства на обектно ориентираното програмиране е способността на класовете да се наследяват. Това спестява време на програмиста, опростява решението на задачите, и подобрява качеството, като ограничава повторението на код и съответно – възникването на грешки. Когато имаме наследяване, наследяващият клас има достъп до член променливите и методите на класа, от който

той наследява (естествено, ако класа го позволява) и същевременно може да добавя и модифицира свои собствени членове. За да посочим че даден клас наследява друг, използваме ключова дума „extends“ в дефиницията му:

Class Име_На_Клас extends Име_На_Другия_Клас

Достъпът до метод на родителския клас от наследяващия става чрез ключова дума parent и непосредствено след нея двойно двоеточие „::“. След което метода се достъпва с неговото име:

Parent::име_Метод();

5. Абстрактен клас

Понякога се нуждаем от клас, който да се използва само като градивен в структурата на наследяване и не искаме да позволим на програмистите да създават обекти от него. Създаването на обект от такъв клас би могло да доведе до редица проблеми. Решението е да обявим дадения клас за абстрактен:

```
Abstract class Име_На_Клас{
}
```

Това ще ограничи възможността да бъде създаден обект от този клас.

6. Членове и методи от тип static.

Ако член променлива или метод на клас бъдат декларирани от тип static, то те стават достъпни без да е необходимо създаването на обект от класа. Член променлива, обявена като static, не може да се използва от обектите на класа (докато статичните методи могат). Разгледайте следния пример:

```
class Име_На_Клас{
    Public static $име_променлива1= “статична променлива“;
    Public static function имеМетод(){
            Echo “<p>”. self::$име_променлива1 . „</p>“;
    }
}
Echo “<p>” . Име_На_Клас::$име_променлива1 .“</p>“
Име_На_Клас::имеМетод();
$обект=new Име_На_Клас;
$обект->имеМетод();
```

В горния пример е показано как се създава статична променлива и метод. Вижда се също че класа статичния метод може да бъде достъпен директно от името на класа както и от създадения обект, докато статичната член променлива се вика само от името на класа (тя не може от обекта). Обърнете внимание че когато достъпваме статичната променлива от метода се използва операция “self::”, а не $this. Псевдопроменливата не може да се използва за методи декларирани като статични, защото статичните методи могат да бъдат извиквани и без да бъде създаден обект от класа.

II. ПРАКТИЧЕСКА ЧАСТ

!ВАЖНО: За да можете да тествате упражнението ви е необходим следния софтуер. Web server, Php интерпретатор, MYSQL база данни. Всеки един от тези

софтуери ги има в пакетите WAMP или XAMP. Настоящото упражнение е тествано върху WAMP Version 3.2.0.

ЗАДАЧА 1: *Напишете PHP скрипт който създава клас Person. Нека той да съдържа две private член променливи отговарящи за Firstname и Surname на даден човек. Също така класа нека да съдържа методи извличащи информацията от член променливите както и методи които позволяват тяхната промяна.*

Код за решение на задача1:

```
<?php
//-----------------------file Person.php--------------------------
class Person {
	private $strFirstname = "Simon";
	private $strSurname = "Stobart";
	function getFirstname(){
		return $this->strFirstname;
	}
	function getSurname(){
		return $this->strSurname;
	}
	function setFirstname($strFirstname){
		$this->strFirstname=$strFirstname;
	}
	function setSurname($strSurname){
		$this->strSurname=$strSurname;
	}
}
?>


<?php
//---------------------------file script.php----------------------
function my_autoloader($class_name){
	require_once $class_name. '.php';
}
spl_autoload_register('my_autoloader');

$objSimon=new Person;
echo "<p>".$objSimon->getFirstname()."</p>";
echo "<p>".$objSimon->getSurname()."</p>";
$objSimon->setFirstname("Elizabeth");
$objSimon->setSurname("Hall");
echo "<p>".$objSimon->getFirstname()."</p>";
echo "<p>".$objSimon->getSurname()."</p>";
?>
```

Пояснение за задача 1

В PHP е прието всеки клас да се прави в отделен файл наименуван с името на класа. То помага многократната употреба на класове и допринася за по изчистен вид на кода. Поради тази причина класа Person е създаден в отделен файл „Person.php“, а обектите са създадени във файл “script.php”. За целта във файла script.php е необходимо да се използват няколко оператори include или require които зареждат всички класове които ще се използват. В нашия случай това е само един клас, но ако

се налага да се ползват доста класове както е и в повечето случаи в практиката то ще трябва да се напише един дълъг списък от тези оператори. За целта в поновите версии на php се използва функция която се извиква автоматично в случай че извикваме клас който не е бил дефиниран. Във файла това е показано с първите 4ри реда код във файла script.php. На следващия ред следващите редове от този файл следва създаването на обект от клас Person и извличането на информацията от член променливите чрез функциите getFirstname и getSurname. Със функциите setFirstname и setSurname се променя информацията в член променливите $strFirstname и $strSurname.

Във файла Person.php е създаден клас Person. Обърнете внимание на методите на класа, как достъпват член променливите. Те използват псевдо променливата $this, понеже член променливите са от тип privete и са декларирани извън метода.

ЗАДАЧА2: *Реализирайте един клас ключалка описващ състоянието на ключалката (отключено и заключено). Реализирайте и друг клас врата описващ състоянието на врата (отворена и затворена), както и в какво състояние е ключалката на вратата.*

Код за решение на задача2.

```
<?php
//-----------------------file Lock.php--------------------------
   class Lock{
      private $strLockStatus;
      function __construct($strLockStatus){
            $this->strLockStatus=$strLockStatus;

      }

      function display(){
            echo " and the lock is ". $this->strLockStatus . "</p>";
      }
}
?>

<?php
//-----------------------------door.php-------------------------
class Door {
      private $strDoorStatus;
      private $objLock;
      function __construct($strDoorStatus, $strLockStatus){
            $this->strDoorStatus=$strDoorStatus;
            $this->objLock=new Lock ($strLockStatus);
            $this->display();
      }

      function display(){
            echo "<p> The door is ".$this->strDoorStatus;
            $this->objLock->display();
      }
}
?>

<?php
```

```php
//-------------------------file script.php----------------------
function my_autoloader($class_name){
      require_once $class_name. '.php';
}
spl_autoload_register('my_autoloader');

$objDoor=new Door ("Closed","Locked");
$objDoor=new Door ("Open","Unlocked");
?>
```

Пояснение за задача 2

Таза задача визуализира как се използват обекти в обекти. Във файла "script.php" са създадени два обекта отговарящи за две врати. Вратите са описани чрез класа Door във файл "door.php". Той се състои от две член променливи едната съдържа статуса на вратата (дали е отворена или затворена), а другата $objLock представлява обект от тип lock. Конструктура за клас door конструира член променливата, която определя статуса на вратата, както и създава нов обект ключалка. Той извиква и метода display на клас door. Метод display() изписва на екрана състоянието на вратата. Също така извиква метода display на обекта Lock. Обърнете внимание как се случва това:

```php
$this->objLock->display();
```

Във файл lock.php си е създаден клас Lock, чрез който се създава обект за ключалка. Той описва състоянието на тази ключалка отключена или заключена.

Резултат от изпълнението на тази задача ще визуализира две съобщения първото ще гласи, че врата е затворена и заключенам, а второто- вратата е отворена и отключена.

ЗАДАЧА3: *Използвайки задача1 съсдайте клас Student който да наследява клас Person. В клас Students да се съхранява следната информация за студента: фак. Номер и специалност в която се обучава, докато в клас Person се съхранява информация само за имената на студента. Създайте обект за студент и разпечатайте имената, фак. номер и от коя специалност се обучава. Направете така че да не може да се създава обект от клас Person.*

Код за решение на задача3.

```php
<?php
//-----------------------file Person.php--------------------------
Abstract class Person {
      private $strFirstname;
      private $strSurname;
      function __construct($strFirstname,$strSurname){
            $this->strFirstname=$strFirstname;
            $this->strSurname=$strSurname;
      }
      function display(){
            echo " Името на студента е: ". $this->strFirstname . " "
      .$this->strSurname. ".";
      }
}
?>
```

```php
//----------------------file Student.php-------------------------
class Student extends Person{
	private $strFack;
	private $strSpec;
	function __construct($strFirstname,$strSurname,$strFack,$strSpec)
	{
		parent::__construct($strFirstname,$strSurname);
		$this->strFack=$strFack;
		$this->strSpec=$strSpec;
		parent::display();
		$this->display();
	}
	function display(){
		echo " Неговия факултетен номер е: ". $this->strFack . ", а неговата специалност: " .$this->strSpec. ".";
	}
 }?>
```

```php
<?php
//--------------------------file script.php----------------------
function my_autoloader($class_name){
	require_once $class_name. '.php';
}
spl_autoload_register('my_autoloader');

$objStudent=new Student("Иван","Иванов","201911111","КСТ");
//objPerson=new Person  - това е код който следва да генерира грешка
?>
```

<u>Пояснение за задача 3</u>

В скриптовия файл е създаден обект от тип Students чрез неговия конструктор който се състои от 4 параметъра. Във файла Student.php е създаден клас Student, който наследява клас Person. Обърнете внимание в конструктора на този клас какво е направено. Освен че са достъпени член променливите на класа се достъпва и конструктура на клас Person и метода му display(). Това е направено чрез оператора “parent::” Във файла Person.php е създаден клас Person който е абстрактен. Ако се опитате да създадете обект от този клас то следва да генерира грешка (това може да го тествате като махнете коментара на последния ред на скриптовия файл).

Резултата от изпълнението на задачата ще бъде: „Името на студента е Иван Иванов. Неговият факултетен номер е:201911111, а неговата специалност: КСТ. “

III. Задача за самостоятелна работа.

1. Напишете скрипт, който разширява функционалността на класовете door и lock в задача 2:
   - Вратата може да бъде отворена само ако ключалката е отключена

- Когато вратата е отворена, ключалката може да бъде в отключено или заключено състояние. Вратата обаче не може да бъде затворена ако ключалката е заключена.

Можете да приемете, че в момента на създаването на вратата и ключалката и вратата е затворена, но отключена.

2. Напишете скрипт, който създава класа „Electriccar“ (електрическа кола). Този клас трябва да включва два други, наречени Engine(мотор) и Gearbox(скоростна кутия):
   - Моторът може да бъде пуснат или спрян
   - Лостът на скоростната кутия може да бъде в положение напред, неутрално и назад.
   - Класът Electriccar има функция speedup, която всеки път, когато бъде извикана, увеличава скоростта на колата с 5 мили/час, ако моторът е пуснат и скоростният лост не е в неутрално положение.
   - Функцията showdown() намаля скоростта на колата с 5 мили/час всеки път когато бъде извикана
   - Ако скоростният лост е в неутрално положение, моторът само форсира на място (ако приемем че колата е запалена!)

3. Напишете структура на наследяване на клас, която реализира следните класове:
   - Vehicle (превозно средство);
   - Car (кола);
   - Hybrid Car (хибридна кола);
   - Boat (лодка);
   - Bicycle (велосипед);

   Класът Vehicle трябва да бъде абстрактен. Класовете за кола, лодка и велосипед трябва да го наследяват. Hybrid Car наследява клас Car. Хибридната кола е кола която може да се движи с електричество или бензин. Списъкът по долу съдържа различни членове на класове. Помислете къде трябва да бъдат разположени в йерархията.
   - IntWeight (тегло);
   - IntLength(дължина);
   - IntTopSpeed(максимална скорост);
   - intDisplacement(водоизместимост);
   - intRemainingFuelLitres(оставащо гориво в литри);
   - intRemainingBatterylife(оставащо електричество в акумулатора);
   - strDescription(описание);

   Трябва да създадете метод display който да показва всички данни на конкретното превозно средство. Ще се наложи също да създадете и други помощни методи, като например контруктор и някои методи за извличане на член променливите.

4. Опитайте се да решите задачата в практическата част на предното лабораторно упражнение като използвате класове. Да се създадат класове извличащи информацията от базата данни. Имената на класовете да бъдат спрямо извличаната информация от базата.