APLIKACJE INTERNETOWE PHP		
TEMAT 4-01: Klasy i obiekty.		
Autor dokumentu: Wojciech Galiński	środa, 16 listopada 2016 r.	351203 Technik informatyk
ŹRÓDŁA WIEDZY:	http://www.php.net/manual/pl, http://webmaster.helion.pl/index.php/kurs-php, http://pl.wikipedia.org/, http://pl.wikibooks.org/wiki/PHP, http://phpkurs.pl/, http://kursphp.com/.	

Zagadnienia obowiązkowe

1. **Programowanie obiektowe** – sposób programowania, w którym programy zawierają obiekty. Obiekty te złożone są z ***pól*** (zmiennych i obiektów składowych) oraz ***metod*** (funkcji składowych).
   Największym atutem programowania, projektowania oraz analizy obiektowej jest zgodność takiego podejścia z rzeczywistością – właśnie w taki sposób mózg ludzki widzi rzeczywistość.
2. **Podstawowe pojęcia programowania obiektowego** – wyróżniamy:
   - ➔ ***klasa*** – wzorzec grupy obiektów o wspólnych cechach (odpowiednikiem w programowaniu tradycyjnym będzie typ zmiennej), np. samochody, towary, uczniowie;

     PRZYKŁAD 1 (pusta klasa):
     ```
     class TPustaKlasa
     {

     }
     ```

     PRZYKŁAD 2 (klasa zawierająca 2 pola i 1 metodę):
     ```
     class TAuto
     {
        private $marka, $model;   // definicja 2 pól
        public function dane() {}   // definicja 1 metody
     }
     ```
   - ➔ ***obiekt*** – konkretny egzemplarz zdefiniowanej wcześniej klasy (odpowiednikiem w programowaniu tradycyjnym jest zmienna),
     np. samochód Opel Astra nr rej. ONY 1111, uczeń o peselu 2000010100001;
     PRZYKŁAD: `$obiekt_klasy_TPustaKlasa = new TPustaKlasa; $auto = new TAuto;`
   - ➔ ***hermetyzacja (enkapsulacja)*** – oznacza, że niektóre elementy składowe klasy są dostępne tylko dla projektanta klasy (tak samo, jak np. wnętrze silnika w samochodzie). Zabezpiecza nas to przed przypadkowym uszkodzeniem kodu, w który nie powinniśmy ingerować, chyba, że wiemy, co robimy;
     PRZYKŁAD:
     ```
     class TLicznik
     {
        private $wartosc = 0;                                    // definicja prywatnego pola
        public function wyswietl()  {  echo 'Licznik: <br />';  } // definicja publicznej metody
     }
     $licznik = new TLicznik;   $licznik->wyswietl();  // echo $licznik->wartosc; // niedostępne
     ```
   - ➔ ***dziedziczenie*** – oznacza, że definicje zdefiniowanych wcześniej klas mogą być wykorzystane do definicji nowych klas, np. klasa „**TPojazd**” zawiera cechy wspólne wszystkich pojazdów. Gdy klasa „**TAutobus**” dziedziczy po klasie „**TPojazd**”, to trzeba zdefiniować tylko cechy szczególne dla autobusów, a nie wszystko od początku (patrz: temat o dziedziczeniu i polimorfizmie);
   - ➔ ***polimorfizm*** – oznacza, że kompilator potrafi odróżnić czynności wykonywane przez obiekty różnych klas, które z nazwy są takie same, ale różne są z działania, np. samochód / pociąg rusza z miejsca, włączam pralkę / komputer, zmieniam dane ucznia / nauczyciela / rodzica (patrz: temat o dziedziczeniu i polimorfizmie).

ĆWICZENIE 1: *Przepisz i uruchom kod powyższych przykładów dotyczących klas „**TPustaKlasa**” i „**TPojazd**” do 1 skryptu w taki sposób, żeby nie wyświetlały się błędy. Przeanalizuj ten kod z nauczycielem.*

3. **Tworzenie obiektów danych klas** – to powołanie obiektu danej klasy do życia, czyli to samo, co utworzenie zmiennej.
   PRZYKŁAD: `$licznik = new TLicznik;`
4. **Obiekt „$this”** – to, dostępna tylko wewnątrz definicji klasy referencja do obiektu, który wywołał funkcję składową klasy. Jej istnienie jest spowodowane faktem, iż w momencie definiowania danej klasy nie ma (i nie może być) jeszcze żadnych obiektów tej klasy.
   PRZYKŁAD: `$this->wartosc = 3;`

5. **Odwoływanie się do pól oraz metod obiektu** – wewnątrz definicji klasy odwołujemy się do nich za pośrednictwem obiektu „**$this**”, natomiast poza nią – za pomocą **nazwy**, której użyliśmy podczas tworzenia obiektu tej klasy.

   PRZYKŁADY:

```
class TLicznik
{  public $wartosc = 0;                              // definicja pola (zmiennej składowej)
   public function wyswietl()                        // definicja metody (funkcji składowej)
   {
      echo  'Licznik: '. $this->wartosc. '<br />';  // treść (ciało) metody (funkcji składowej)
   }
}
$licznik = new TLicznik;              // utworzenie obiektu klasy TLicznik
$licznik->wartosc = 3;                // zapisanie wartości do pola klasy TLicznik
do   $licznik->wyswietl();            // wywołanie metody klasy TLicznik
   while ($licznik->wartosc-- > 0);   // zmniejszanie o 1 wartości w polu klasy TLicznik
```

ĆWICZENIE 2: *Przepisz i uruchom powyższy kod w taki sposób, żeby nie wyświetlały się błędy. Przeanalizuj ten kod z nauczycielem.*

6. **Stałe składniki klasy** – to stałe należące do klasy. Przypominają one pola statyczne, ale ich wartość ustawia się tylko raz, a potem są one tylko do odczytu.

   Do stałych należących do klasy odwołujemy się używając operatora „**::**”:

   ➔ ***wewnątrz definicji klasy*** - za pomocą odwołania „**self**”;

   ➔ ***poza definicją klasy*** – za pomocą obiektu tej klasy albo za pomocą nazwy klasy.

   PRZYKŁAD (przykładowa statyczna funkcja składowa klasy „Klasa_z_polem_static”):

```
class Klasa_z_const
{  const  STALA_LICZBA = 3.14,  STALY_TEKST = 'PI';
   public function wyswietl_stale() { echo self::STALY_TEKST.'='.self::STALA_LICZBA.'<br />'; }
}
echo Klasa_z_const::STALY_TEKST. ' równe jest '. Klasa_z_const::STALA_LICZBA. '<br />';
$kzc = new Klasa_z_const();   $kzc->wyswietl_stale();
echo $kzc::STALY_TEKST. ' to '. $kzc::STALA_LICZBA. '<br />';
```

ĆWICZENIE 3: *Przepisz i uruchom powyższy kod w taki sposób, żeby nie wyświetlały się błędy. Czy do stałych należących do klas możemy odwoływać się przed powstaniem pierwszego obiektu tej klasy?*

7. **Poziomy dostępu do składników klasy (hermetyzacja)** – wyróżniamy:

   ➔ ***public*** – takie elementy dostępne są wszędzie: wewnątrz definicji klasy i poza definicją klasy;

   ➔ ***protected*** – takie elementy dostępne są w klasach pochodnych (patrz: temat o dziedziczeniu);

   ➔ ***private*** – takie elementy dostępne są tylko wewnątrz definicji klasy.

   PRZYKŁAD:

```
class TLicznik
{  const PREFIKS = 'Licznik: '; // stałe składniki klasy są dostępne zawsze(?) publicznie
   private $wartosc = 0;        // dostęp do prywatnych pól uzyskujemy poprzez metody publiczne
   private function ustaw($w)    { $this->wartosc = $w; }
   private function wyswietl()   { echo self::PREFIKS. $this->wartosc. '<br />'; }
   public function odliczaj($w = null)
   {  if ($w != null)  $this->ustaw($w);
      do  $this->wyswietl();  while ($this->wartosc-- > 0);
   }
}
echo '<p>'. TLicznik::PREFIKS. '</p>';
$licznik = new TLicznik;   $licznik->odliczaj(3);
echo '<p>'. $licznik::PREFIKS. '</p>';
```

ĆWICZENIE 4: *Przepisz i uruchom powyższy kod w taki sposób, żeby nie wyświetlały się błędy. Spraw, żeby odliczanie następowało do 0, zamiast do 1.*

Zadania

1. Zaprojektuj prostą klasę generującą prostą tabelę HTML na podstawie tablicy 2-wymiarowej.
2. Zaprojektuj klasę generującą: formularz logowania, formularz edycji danych użytkownika.
3. Zaprojektuj prostą klasę generującą dokument HTML. Dodaj do tej klasy obsługę następujących modułów dokumentu HTML: nagłówek, główne menu, treść i stopka dokumentu. W tym celu należy zdefiniować odpowiednie pola i metody, które należy wywołać w funkcji składowej generującej kod HTML dokumentu.
4. Utwórz klasę implementującą najprostsze cechy kontrolki „input” typy kontrolki ustaw jako tablicę stałych (text, password, …).