

PHP

```
break; // 'Ciepto');  
case 35 :  
    print('Gorąco');  
    break;  
default :  
    print('Nie zdefiniowa  
do?>  
print'
```

PROGRAMOWANIE OBIEKTOWE PHP 5



Definicja programowania obiektowego

- Klasa
- obiekt
- właściwość
- metoda

Tworzenie obiektu klasy i praca na nim

klasa telewizor

właściwości:

aktualny_program
glosnosc

metody:

ustaw_program(\$numer)
podglosn()
scisz()

```
$moj_tv = new telewizor();
```

```
$moj_tv->ustaw_program(15);
```

```
while (przycisk_podglasniania()) {  
    $moj_tv->podglosn();  
}
```

Tworzenie klas

metody

Tworzenie klas: metody

```
class Nazwa_klasy {  
    function nazwa_metody() {  
        // zawartosc metody  
    }  
  
    function nazwa_innej_metody() {  
        // zawartosc innej metody  
    }  
}
```

Tworzenie klas

właściwości

Tworzenie klas: właściwości

```
class Nazwa_klasy {  
    var $wlasciwosc1;  
    var $wlasciwosc2 = 10;  
    function nazwa_metody($argument1, $argument2) {  
        // zawartosc metody  
    }  
    function nazwa_innej_metody() {  
        // zawartosc innej metody  
    }  
}
```

właściwości:

```
class Nazwa_klasy {  
    var $wlasciwosc1;  
    var $wlasciwosc2 = 10;  
    function nazwa_metody($argument1, $argument2) {  
        // zawartosc metody  
    }  
    function nazwa_innej_metody() {  
        // zawartosc innej metody  
    }  
}
```

Tworzymy obiekt:

```
$obiekt_klasy = new Nazwa_klasy();
```

```
$obiekt_klasy->nazwa_metody("test", "metody");
```

```
$obiekt_klasy->wlasciwosc2 = 30;
```

```
print($obiekt_klasy->wlasciwosc2);
```

Odwołania do metod i atrybutów klasy

```
class Drukarka {  
    var $ciag_znakow;  
    function drukuj() {  
        print($this->ciag_znakow);  
    }  
    function drukuj_z_nowa_linia() {  
        $this->drukuj();  
        print("\n");  
    }  
}
```

```
$hp400 = new drukarka();
```

```
$hp400->ciag_znakow = "treść do wydrukowania na drukarce";
```

```
$hp400->drukuj_z_nowa_linia();
```

Hermetyzacja

private – właściwość lub metoda jest dostępna z zewnątrz klasy

public – właściwość lub metoda jest dostępna tylko z klasy w której się znajduje

protected – właściwość lub metoda jest dostępna z klasy w której się znajduje oraz z klas dziedziczących po tej klasie

```
class Drukarka {  
    private $ciag_znakow;  
}  
  
$hp400 = new Drukarka();  
  
$hp400->ciag_znakow = 100;
```

Dziedziczenie/rozszerzanie

```
class Drukarka {  
  
    public $ciag_znakow = "";  
  
    public function drukuj() {  
        print($this->ciag_znakow);  
    }  
}  
  
class Lepsza_drukarka extends Drukarka {  
  
    public function drukuj_z_nowa_linia() {  
  
        $this->drukuj();  
        print("\n");  
    }  
}  
  
$hp400 = new Lepsza_drukarka();  
  
$hp400->ciag_znakow = "test drukarki";  
$hp400->drukuj_z_nowa_linia();
```

Metody i właściwości statyczne

```
class klasa {  
    public static $wlasciwosc;  
    public static function metoda() {  
        print(self::$wlasciwosc);  
    }  
}
```

```
klasa::$wlasciwosc = 4;
```

```
klasa::metoda();
```

Metody specjalne

```
__construct() __destruct()
```

```
class Drukarka {  
    private $ciag_znakow;  
  
    public function __construct($ciag_znakow) {  
        $this->ciag_znakow = $ciag_znakow;  
        // tutaj np. jakas metoda wlaczajaca drukarke  
    }  
  
    public function __destruct() {  
        // tutaj metoda wylaczajaca drukarke  
    }  
  
    public function drukuj() {  
        print($this->ciag_znakow);  
    }  
}
```

```
hp400 = new Drukarka("test drukarki");  
hp400->drukuj();
```

Rozszerzenie klasy - konstruktor

```
class Drukarka {  
    private $ciag_znakow;  
    public function __construct($ciag_znakow) {  
        $this->ciag_znakow = $ciag_znakow;  
    }  
    public function drukuj() {  
        print($this->ciag_znakow);  
    }  
}
```

```
class Lepsza_drukarka extends Drukarka {  
    public function __construct($ciag_znakow="") {  
        if($ciag_znakow != "")  
            parent::__construct($ciag_znakow);  
        else  
            die('Ta drukarka nie drukuje pustych wierszy');  
    }  
    public function drukuj_z_nowa_linia() {  
        $this->drukuj();  
        print("\n");  
    }  
}
```

```
$lexmark = new Lepsza_drukarka(); // spowoduje wypisanie komunikatu błędu  
$lexmark = new Lepsza_drukarka(""); // tak samo jak wyżej  
$lexmark = new Lepsza_drukarka("drukuję test"); // nie spowoduje błędu  
$lexmark->drukuj_z_nowa_linia();
```

Metody `__get` `__set` i `__call`

```
class klasa {  
  
    public function __call($nazwa_metody , $tablica_argumentow) {  
  
        print("Wywołujesz metode: $nazwa_metody, argumenty:\n");  
        print_r($tablica_argumentow);  
  
    }  
}  
  
$obiekt = new klasa();  
  
$obiekt->metoda_ktorej_nie_ma('argument','innyargument', 34);
```

Wywołujesz metodę: metoda_ktorej_nie_ma, argumenty:

```
Array ( [0] => argument  
        [1] => innyargument  
        [2] => 34 )
```

Nieistniejące wł: __get() i __set()

```
class klasa {  
  
    private $tablica = array();  
  
    public function __get($nazwa) {  
        return $this->tablica[$nazwa];  
    }  
  
    public function __set($nazwa, $wartosc) {  
        $this->tablica[$nazwa] = $wartosc;  
    }  
}
```

```
$obiekt = new klasa();  
$obiekt->nowa_zmienna = "test";  
$obiekt->inna_zmienna = "blah";  
print($obiekt->nowa_zmienna); // test  
print($obiekt->inna_zmienna); // blah
```

Kolonowanie obiektów (__clone)

```
$obiekt = new Klasa();  
$kopia_objektu = $obiekt->__clone();
```

```
class klasa {  
  
    public static $licznik;  
  
    public function __destruct() {  
        self::$licznik--;  
    }  
  
    public function __clone() {  
        $this->id = self::$licznik++;  
        print("tworze " . $this->id . " klon obiektu");  
    }  
}
```

Funkcja __autoload

```
function __autoload($nazwa_klasy) {  
    print("stworzono obiekt klasy $nazwa_klasy,  
        taka klasa nie istnieje");  
}
```

```
function __autoload($nazwa_klasy) {  
    include_once("klasy/" . $nazwa_klasy . ".php");  
}
```

PEAR

<http://pear.php.net>

PHP

```
break; // 'Ciepto');  
case 35 :  
    print('Gorąco');  
    break;  
default :  
    print('Nie zdefiniowa
```

KONIEC

