

Obecná deklarace třídy

```
[abstract|final] class Trida [extends JinaTrida] [implements Rozhrani[,Rozhrani2[, ...]]]
{
    const KONSTANTA = 'hodnota';

    public [static] $verejnaVlastnost;
    protected [static] $chrananaVlastnost;
    private [static] $soukromaVlastnost;

    [abstract|final] [public|protected|private] [static] function metoda($parametr)
    {

    }
}
```

Naming convention

- PascalCase – názvy tříd a rozhraní
- camelCase – názvy metod a proměnných
- UPPER_CASE – konstanty

Práce se třídou vně třídy

```
$trida = new Trida(); // konstrukce objektu
echo $trida->verejnaVlastnost; // přístup k veřejné vlastnosti (bez dolaru)
echo $trida->metoda('hodnota'); // zavolání veřejné metody + předání parametru
Trida::KONSTANTA;
Trida::$statickaVlastnost;
Trida::statickaMetoda();
```

Práce se třídou uvnitř třídy

```
public function metoda($parametr)
{
    echo $parametr;

    $this->jinaMetoda();
    $this->vlastnost // bez dolaru;

    self::KONSTANTA;
    self::$statickaVlastnost;
    self::statickaMetoda();

    parent::metodaRodice();

    return 'Návratová hodnota';
}
```

Magické metody

- __construct = konstruktor, volá se na začátku
- __destruct = destruktory, volá se na konci
- __get = magický getter, získání nedefinované nebo nedostupné vlastnosti
- __set = magický setter, nastavení hodnoty pro nedefinovanou nebo nedostupnou vlastnost
- __call = volání nedefinované nebo nedostupné metody
- __callStatic = statické volání nedefinované nebo nedostupné metody, od PHP 5.3
- __clone = volá se při klonování objektu
- __toString = při potřebě přetypovat na string např. echo \$obj;

Obecná deklarace rozhraní

```
interface Rozhrani [extends JineRozhrani]
{
    [public|protected|private] [static] function VypisuRozhrani();
    [public|protected|private] [static] function VypisParametr($parametr);
}
```

Klíčová slova

class

deklarace třídy

interface

deklarace rozhraní

static

Metoda či vlastnost nepatří objektu, ale třídě (nelze použít `$this->`, musíte použít `self::`). Proto není potřeba sestavovat objekt (`$obj = new Obj();`) a lze je používat přímo. Statické vlastnosti jsou společné pro všechny objekty od dané třídy.

abstract

Abstraktní třídu nelze přímo zkonstruovat. Slouží pouze pro zapouzdření obecné logiky

Abstraktní metoda vynucuje definice této metody v dceřiné třídě.

final

Finální třídu nelze dědit. Finální metody nelze přepsat.

const

deklarace konstanty

public, protected, private

- public = veřejný - dostupný všem
- protected = chráněný - dostupný třídě a potomkům třídy
- private = soukromý - dostupný pouze dané třídě

extends

označuje dědění od dané třídy

implements

označuje implementaci rozhraní

```
/**
 * Abstraktní třída představující právě jedno zvíře.
 *
 * Tato třída je abstraktní (klíčové slovo 'abstract'). Tzn., že nelze vytvořit
 * přímo její instanci = objekt. Volání $obj = new Zvire('Míca'); skončí chybou.
 * Slouží k tomu, že shromažďuje obecnou logiku všech zvířat.
 */
abstract class Zvire
{
    /**
     * Jméno zvířete.
     *
     * Jméno je vlastnost zvířete. Její viditelnost je označena jako
     * 'protected' = chráněný.
     *
     * Význam viditelností:
     * a) public = veřejný - dostupný všem
     * b) protected = chráněný - dostupný třídě a potomkům třídy
     * c) private = soukromý - dostupný pouze dané třídě
     *
     * @var string Jméno zvířete
     */
    protected $jmeno;

    /**
     * Konstruktor zvířete
     *
     * Konstruktor se jedná z "magických metod". Musí se jmenovat '__construct'.
     * Je automaticky zavolán, pokud někdo použije new Třída($parametr)
     * a zároveň jsou mu předány parametry toho volání (zde $jmeno)
     *
     * @param string Jméno zvířete
     */
    public function __construct($jmeno)
    {
        /**
         * $this-> odkazuje na aktuální instanci = objekt
         * self:: odkazuje na aktuální třídu
         * parent:: odkazuje na nadřazenou třídu
         *
         * U $this se proměnné píše bez dolaru ($), u ostatních s dolarem
         */
        $this->jmeno = $jmeno;
    }

    /**
     * Magický getter
     *
     * Toto je další magická metoda. Je volána, pokud se někdo snaží získat vlastnost
     * objektu, která neexistuje, nebo není dostupná (private nebo protected)
     * Jako parametrem dostane název vlastnosti, ke které se snaží dostat.
     * Umožňuje přístup echo $obj->jmeno místo echo $obj->getJmeno()
     *
     * @param string Jméno vlastnosti
     * @return mixed
     * @throws MemberAccessException
     */
    public function __get($co)
    {
        switch ($co) {
            case 'jmeno':
                return $this->getJmeno();
            default:
                throw new MemberAccessException("Nelze získat hodnotu nedefinované vlastnosti [$co]");
        }
    }
}
```

```
/**
 * Magický setter
 *
 * Další "magická metoda". Volána, pokud někdo nastavuje hodnotu nedefinované
 * nebo nepřístupné proměnné. První parametr je jméno vlastnosti, druhý nastavovaná hodnota
 *
 * Umožňuje psát $obj->jmeno = "Nové jméno" místo $obj->setJmeno("Nové jméno")
 *
 * @param string   Jméno vlastnosti
 * @param mixed    Nová hodnota
 * @throws MemberAccessException
 */
public function __set($co, $hodnota)
{
    switch ($co) {
        case 'jmeno':
            $this->setJmeno($hodnota);
            break;
        default:
            throw new MemberAccessException("Nelze změnit hodnotu nedefinované vlastnosti [$co] na [$
hodnota]");
    }
}

/**
 * Setter pro vlastnost 'jmeno'
 *
 * Umožňuje nastavit hodnotu vlastnosti 'jmeno', která je 'protected'.
 *
 * @param string   Nové jméno
 * @return void    Nic nevrací
 */
public function setJmeno($jmeno)
{
    if (!is_string($jmeno)) {
        throw new InvalidArgumentException('Jméno musí být řetězec');
    }

    $this->jmeno = $jmeno;
}

/**
 * Getter pro vlastnost 'jmeno'
 *
 * Umožňuje získat hodnotu chráněné (= protected) vlastnosti
 *
 * @return string   Jméno zvířete
 */
public function getJmeno()
{
    return $this->jmeno;
}

/**
 * Napodobí zvuk zvířete
 *
 * Abstraktní metoda (klíčové slovo 'abstract').
 * Pokud třída obsahuje alespoň jednu abstraktní metodu,
 * tak musí být sama označena jako abstraktní.
 *
 * Všechny třídy, které jsou potomkem této třídy musí obsahovat
 * definice této metody.
 * Využití? U všech tříd, které jsou instancí třídy 'Zvíře'
 * si můžeme být jisti, že mají metodu pro napodobení zvuku.
 */
abstract public function napodobZvuk();
}
```