Internet programiranje

Objektno orijentisano programiranje u jeziku PHP

Predavač:
Dr Dražen Drašković, docent

8) Objektno orijentisano programiranje u programskom jeziku PHP

- Izrada klasa, objekata, atributa
- Polimorfizam, nasleđivanje, redefinisanje
- Modifikatori pristupa
- Interfejsi

Klase i objekti

- Klasa je osnovna jedinica programiranja na OO jezicima
- Klase sadrže:
  - Atribute (svojstva ili promenljive koje opisuju objekat)
  - Operacije (metode, radnje ili funkcije koje objekat može da izvršava)
  - Mehanizme za stvaranje objekata na osnovu definicije (konstruktore)
- Enkapsuliranje (skrivanje podataka)
  - Pristup podacima unutar datog objekta moguć samo pomoću operacija tog objekta
- Šta je objekat?
  - Jedan primerak (instanca, pojava) klase

Polimorfizam, nasleđivanje

- OO jezici podržavaju polimorfizam: svaki objekat izvedene klase izvršava operaciju onako kako je to definisano u njegovoj (izvedenoj) klasi
- Nasleđivanje: klasa + jedna ili više potklasa
- Potklasa (izvedena klasa, dete) nasleđuje atribute i operacije od svoje natklase (roditeljske klase)
- Nasleđivanje omogućava nadgradnju i proširenje postojećih klasa:

```
class Vozilo
class Auto extends Vozilo
class Kamion extends Vozilo
```

Struktura klase, konstruktori

```
class ime {
   var $atribut1;
   var $atribut2;
   function operacija1() {}
   function operacija2($par1, $par2) {}
}
```

- Konstruktor se poziva prilikom pravljenja objekata date klase
- Konstruktor se deklariše kao druge operacije, ali ima specijalno ime `construct()`

```
class ime {
   function __construct($par) {
   echo "pozvan je konstruktor sa parametrom $par"; }}
```

Pravljenje objekata

- Nov objekat se pravi pomoću rezervisane reči `new`
- Primer:

```
$a = new ime('Prvi');
$b = new ime('Drugi');
$c = new ime();
```

- Rezultat:

```
pozvan je konstruktor s parametrom Prvi
pozvan je konstruktor s parametrom Drugi
pozvan je konstruktor s parametrom
```

Upotreba atributa klase

- Pokazivač `$this` upućuje na tekući objekat
- Ako tekući objekat ima atribut `$atr`, možete da mu pristupite pomoću imena `$this->atr`
- Primer:

```
class ime {
    var $atr;
    function operacija($par) {
          $this->atr = $par;
          echo $this->atr;
    }
}
```

Pristupne funkcije

- Primer:

```
class ime {
    var $atr;
    function __get($imeatributa) {
          return $this->$imeatributa; }
    function __set($imeatributa, $nova_vred) {
          $this->$imeatributa=$nova_vred; }
}
```

- Ove 2 funkcije se pozivaju implicitno

```
$a = new ime();
$a->atr = 5; // poziva se funkcija __set
```

Modifikatori pristupa

- public (javni) - elementima koji se deklarišu kao javni može se pristupati i unutar i izvan klase;
- private (privatni) - elementima koji se deklarišu kao privatni može se pristupati samo unutar klase; ne mogu da se nasleđuju;
- protected (zaštićen) - označava da elementu klase može da se pristupi samo unutar klase, ali se taj element nasleđuje u svim potklasama;

```
class ime {
    public $atribut;
}
```

Pozivanje operacije klase

```
$a = new ime();

//pristup kao atributu tog objekta
$a->operacija1();
$a->operacija2(11,'proba');

//ako operacija vraca neku vrednost
$x = $a->operacija1();
$y = $a->operacija2(11,'proba');
```

Nasleđivanje - primer

```
class B extends A {
   var $atribut2;
   function operacija2() {}
}
class A {
   var $atribut1;
   function operacija1() {}
}
```

Klasa A nema funkciju `operacija2()`
i atribut `$atribut2` !!!

Redefinisanje (eng. *overriding*)

- Promena postojeće funkcionalnosti natklase u nasleđenoj klasi

```
class A {
    var $atribut = “staravrednost”;
    function operacija() {
          echo 'nesto<br/>';
          echo “vrednost je $this->atribut”; }
}
class B extends A {
    var $atribut = “novavrednost”;
    function operacija() {
          echo 'nesto drugo<br/>';
          echo “vrednost je $this->atribut”; }
}
```

Redefinisanje - Primer

```
   $a = new A();
   $a->operacija();
Izlaz:
   nesto
   vrednost je staravrednost

   $b = new B();
   $b->operacija();
Izlaz:
   nesto drugo                  //nesto
   vrednost je novavrednost     //vrednost je novavrednost
```

Poziv u potklasi B:

parent::operacija();

final

- Služi za sprečavanje nasleđivanja
- Kada se postavi ispred deklaracije funkcije, ta se funkcija ne može zameniti istoimenom ni u jednoj potklasi:

```
final function operacija()   { … }
```

- Upotrebom `final` se može sprečiti i nasleđivanje određene klase:

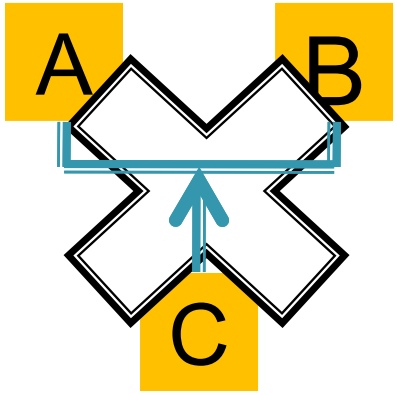
```
final class A()
{ … }
```

Interfejsi

- PHP ne podržava višestruko nasleđivanje.
- Interfejs deklariše određen broj funkcija koje moraju biti realizovane u svim klasama koje realizuje taj interfejs.

```
interface Prikazuje {
    function prikazi ();
}
```

- Interfejs se koristi kao “zaobilazni” način da se realizuje višestruko nasleđivanje



Naprednije OO funkcionalnosti

- const - konstanta klase, koja se koristi bez obaveze da se napravi objekat:

```
class Matematika {
    const pi = 3.14159;
}
echo 'Pi je `.Matematika::pi;
```

- static - kada je zadata ispred deklaracije metode, omogućava da se poziva metoda bez pravljenja instance klase:

```
static function kvadrat($broj) {
    return $broj*$broj;  }
echo 'Kvadrat od 8 = 'Matematika::kvadrat(8);
```

Primer sa statičkom funkcijom (1)

```
<?php                                                    //Student.php
class Student {
   public $firstName;
   public $lastName;
   public function __construct($firstName, $lastName = '') {
   //Optional parameter
        $this->firstName = $firstName;
        $this->lastName = $lastName;
   }
   public function greet() {
        return "Hello, my name is " . $this->firstName . " " .
       $this->lastName . ".";
   }
  public static function staticGreet($firstName, $lastName) {
        return "Hello, my name is " . $firstName . " " .
       $lastName . ".";
   }
}
?>
```

Primer sa statičkom funkcijom (2)

```php
<?php                                                    //index.php
    require("Student.php");
    $demon1 = new Student('Stefan', 'Kostic');
    $demon2 = new Student('Mina', 'Reljic');
    $demon3 = new Student('Filip Zivkovic');

    echo $demon1->greet();
    echo '<br />';
    echo $demon2->greet();
    echo '<br />';
    echo $demon3->greet();
    echo '<br />';
    echo Student::staticGreet('Balsa', 'Bojic');
?>
```

9) Obrada izuzetaka na PHP-u

- Koncepti obrade izuzetaka
- Klasa Exception
- Izuzeci koje korisnik definiše

Obrada izuzetaka

- Kod se izvršava unutar `try` *bloka*:

```
try {
    //kod
}
```

- Unutar `try` bloka izuzetak se izaziva:

```
throw new Exception('poruka', broj_greske);
```

- Svakom bloku `try` mora da sledi barem jedan blok `catch`:

```
catch (Exception $e) {
    //obrada izuzetka
}
```

- Objekat koji se prosleđuje bloku `catch` (da bi ga blok presreo) jeste objekat prosleđen iskazu `throw` koji je generisao izuzetak.

Obrada izuzetaka - Primer

```
<?php
    try
    {
      throw new Exception('Greska!!!', 42);
    }
    catch (Exception $e)
    {
       echo 'Exception '.$e->getCode().': '.
       $e-> getMessage().'<br />'.' u fajlu '. $e->getFile().
       'na liniji '. $e->getLine(). '<br />';
    }
?>
```

Klasa Exception

- Konstruktor te klase prihvata 2 parametra: tekst poruke o grešci i broj greške

getCode() - vraća broj greške koji je bio prosleđen konstruktoru

getMessage() - vraća tekst poruke koja je bila prosleđena konstruktoru

getFile() - pozivajućem kodu vraća punu putanju datoteke u kojoj je generisan izuzetak

getLine() - vraća broj reda koda u kome je generisan izuzetak

getTrace() - vraća niz s podacima o stablu pozivanja koji omogućavaju utvrđivanje mesta na kome je generisan izuzetak

getTraceAsString () - vraća iste podatke kao getTrace(), formatirane kao znakovni podaci

__toString() - omogućava da se iskazu echo direktno prosledi ceo sadržaj objekta Exception, sa svim podacima koje daju navedene metode

10) PDO klasa

- Služi za konekciju između PHP i baze, pomoću OO
- Sinopsis:

```
PDO {
   public __construct ( string $dsn [, string $username [, string $passwd
                                                          [, array $options ]]] )
   public beginTransaction ( void ) : bool
   public commit ( void ) : bool
   public errorCode ( void ) : string
   public errorInfo ( void ) : array
   public exec ( string $statement ) : int
   public getAttribute ( int $attribute ) : mixed
   public static getAvailableDrivers ( void ) : array
   public inTransaction ( void ) : bool
   public lastInsertId ([ string $name = NULL ] ) : string
   public prepare ( string $statement [, array $driver_options = array() ] ) : PDOStatement
   public query ( string $statement ) : PDOStatement
   public quote ( string $string [, int $parameter_type = PDO::PARAM_STR ] ) : string
   public rollBack ( void ) : bool
   public setAttribute ( int $attribute , mixed $value ) : bool
}
```

Objektno orijentisani PHP

Hvala!