

Osnove izrade PHP aplikacija

Uvod u OO PHP

Izradio: Davor Cihlar

Uredili: Goran Žuri; Savjetovali: demosi

Plan

- ▶ Što ćemo naučiti
 - ▶ Klase
 - ▶ Metode
 - ▶ Instance
 - ▶ Vidljivost varijabli i metoda
 - ▶ Konstante
 - ▶ Pristup varijablama i metodama
 - ▶ Čarobne metode

Što je OOP

- ▶ **Objektno orijentirano programiranje**
 - ▶ Način programiranja koji je orijentiran prema objektu (entitetu), njegovom ponašanju i vezama
 - ▶ Prirodnije za čovjeka
- ▶ **Problemi kod proceduralnog programiranja**
 - ▶ Odvojenost ponašanja od podataka
 - ▶ Strukture se koriste za sadržavanje podataka, a procedure za rad nad tim podacima
 - ▶ Zbog te odvojenosti iznimno je teško održavati i graditi velike sustave

Što su klase?

- ▶ Klase su opis strukture objekta
 - ▶ Objekt je instanca (primjerak) klase
 - ▶ Npr. riječ "ovca" je klasa, a prava ovca je primjerak klase ovca
- ▶ Struktura klase
 - ▶ Podaci
 - ▶ Varijable i konstante
 - ▶ Svaka instanca ima vlastite vrijednosti
 - promjena varijable u jednoj instanci ne uzrokuje promjenu u drugoj
 - ▶ Ponašanje
 - ▶ Operacije se izvode nad instancom
 - ▶ Mogu utjecati na podatke

Brzi uvod u OOP (1)

► Najjednostavniji primjer u C-u bi bilo:

```
► typedef struct {  
    char ime[64], prezime[64];  
    int naRacunu;  
} Kupac; // opisnik kupca  
  
► typedef struct {  
    int cijena;  
} Artikl; // opisnik artikla  
  
► typedef struct {  
    Artikl artikli[128];  
    int nArtikala;  
    Kupac *kupac;  
} Kosarica; // opisnik kosarice
```

Brzi uvod u OOP (2)

- ▶ Kako bi u C-u ispisali ukupnu cijenu košarice?
- ▶ Podaci su odvojeni od ponašanja

```
▶ int izracunajCijenuKosarice(  
    Kosarica *kosarica);
```

```
Kosarica a;
```

```
... // ispuna kosarice
```

```
printf("Cijena kosarice A: %d\n",  
    izracunajCijenuKosarice(&a));
```

Brzi uvod u OOP (3)

- ▶ Strukture podataka želimo koristiti na što jednostavniji način
 - ▶ `Kosarica a, b; // instanciranje`

`... // ispuna košarica a i b`

`printf("Cijena kosarice A: %d\n",
 a.ukupnaCijena());`
`printf("Cijena kosarice B: %d\n",
 b.ukupnaCijena());`
- ▶ To omogućuju metode unutar same strukture podataka

Brzi uvod u OOP (4)

► Dodavanje metoda u C++:

```
class Kosarica {  
public:  
    Artikl artikli[128];  
    int nArtikala;  
    Kupac *kupac;  
  
    int ukupnaCijena() {  
        int i = 0, sum = 0;  
        for (; i < nArtikala; ++i)  
            sum += artikli[i].cijena;  
        return sum;  
    }  
};
```

```
typedef struct {  
  
    Artikl artikli[128];  
    int nArtikala;  
    Kupac *kupac;  
  
} Kosarica;
```

Brzi uvod u OOP (5)

- ▶ Isti primjer u PHP-u (definicije klasa):

```
class Kosarica {  
    var $artikli = array();  
    var $kupac;
```

```
    function ukupnaCijena() {  
        $sum = 0;  
        foreach ($this->artikli as &$a)  
            $sum += $a->cijena;  
        return $sum;  
    }
```

```
    function dodajArtikl($a) {  
        $this->artikli[] = $a;  
    }
```

```
}
```

```
class Kupac {  
    var $ime, $prezime;  
    var $naRacunu;  
}
```

Brzi uvod u OOP (6)

► Isti primjer u PHP-u (korištenje):

```
► $a = new Kosarica(); // instanciranje  
   $b = new Kosarica();
```

```
$a->dodajArtikl(...); // ispunjena košarica
```

```
echo "Cijena kosarice A: ",  
     $a->ukupnaCijena(), "\n";
```

```
echo "Cijena kosarice B: ",  
     $b->ukupnaCijena(), "\n";
```

Brzi uvod u OOP (7)

- ▶ Proširimo malo klasu Kupac:

```
▶ class Kupac {  
    var $ime, $prezime;  
    var $naRacunu;  
  
    function plati($iznos) {  
        if ($this->naRacunu < $iznos)  
            return false;  
        $this->naRacunu -= $iznos;  
        return true;  
    }  
}
```

Brzi uvod u OOP (8)

► Proširimo i klasu Kosarica:

```
► class Kosarica {  
    var $artikli = array();  
    var $kupac;  
  
    function ukupnaCijena() {  
        $sum = 0;  
        foreach ($this->artikli as &$a)  
            $sum += $a->cijena;  
        return $sum;  
    }  
    function dodajArtikl($a) { $this->artikli[] = $a; }  
    function prodaj() {  
        if ($this->kupac->plati($this->ukupnaCijena()))  
            echo "Kupac je sretno platio! :)\n";  
        else  
            echo "Kupac nema para! :(\n";  
    }  
}
```

Zadatak (1) – nabavka.php



- ▶ Kopirajte prije opisane klase u nabavka.php
- ▶ Pokušajte sami napraviti klasu `Artikl`

Zadatak (1) – nastavak



- ▶ Napokon, pokušajmo nešto napraviti s tim klasama!

```
▶ // stvorimo novu košaricu
$kosarica = new Kosarica();

// stvorimo dva nova artikla
$a1 = new Artikl();   $a2 = new Artikl();
$a1->cijena = 42;     $a2->cijena = 4;

// dodamo artikle u košaricu
$kosarica->dodajArtikl($a1);
$kosarica->dodajArtikl($a2);

// pogledajmo dodane artikle
var_dump($kosarica->artikli);
```

Zadatak (1) – nastavak



▶ Prodajmo kupcu sadržaj košarice

```
▶ $kupac = new Kupac();  
$kupac->naRacunu = 1337;
```

```
$kosarica->kupac = $kupac;  
$kosarica->prodaj();
```

```
echo "Na racunu (1): ",  
      $kupac->naRacunu, "\n";
```

```
echo "Na racunu (2): ",  
      $kosarica->kupac->naRacunu, "\n";
```

▶ Koju vrijednost na kraju ima kupac na računu?

▶ Da li je to ispravna vrijednost? Zašto?

Definicija klase i instanciranje

▶ Definicija:

- ▶ `class Klasa {`
`}`

- ▶ Samo opis klase!

- ▶ Ne zauzima mjesto u memoriji

- ▶ Klasu je potrebno instancirati

- ▶ Istu klasu moguće je instancirati više puta

- ▶ Sve instance su međusobno nezavisne

▶ Instanciranje:

- ▶ `$instanca = new Klasa();`

Operator clone

- ▶ Instance klasa se pretpostavljeno ne kopiraju!
 - ▶ `$a = new Klasa();`
`$b = $a; // $b je referenca na $a!`
 - ▶ Suprotno nego s ostalim tipovima varijabli!
- ▶ Operatorom `clone` naglašava se da je instancu potrebno kopirati (klonirati)
 - ▶ `$a = new Klasa();`
`$b = clone $a;`
 - ▶ Isto vrijedi i kod poziva funkcija ili metoda:
 - ▶ `$a = new Artikl();`
`$kosarica->dodajArtikl(clone $a);`

Pristup varijablama i metodama

- ▶ Pomoću operatora `->`
 - ▶ Za pristup varijablama iza `->` ne dolazi znak `$`
 - ▶

```
class Klasa {  
    var $a;  
}
```
 - ```
$x = new Klasa();
```
  - ```
$x->a = 4;
```
 - ```
var_dump($x);
```

# Vidljivost

---

- ▶ Tri tipa vidljivosti (za varijable i metode):
  - ▶ `public` – podrazumijevano
  - ▶ `private` – pristup moguć samo unutar klase
  - ▶ `protected` – pristup moguć unutar naslijeđenih klasa
- ▶ Primjer određivanja vidljivosti varijabli:
  - ▶ 

```
class Klasa {
 var $a; // podrazumijeva se public
 public $b;
 private $c;
 protected $d;
}
```

# Početna vrijednost varijable

---

- ▶ Moguće zadati početnu vrijednost varijable
  - ▶ Vrijednost se automatski postavlja pri instanciranju
  - ▶ Nije moguće koristiti instanciranje neke klase kao početnu vrijednost
- ▶ Primjer:

```
class Klasa {
 var $a = 42;
}
```

```
$x = new Klasa();
```

```
$x->a = 4;
```

```
$y = new Klasa();
```

```
var_dump($x); // a = 4
```

```
var_dump($y); // a = 42 (nepromijenjen)
```

# Naknadne varijable

---

- ▶ Varijable nije nužno definirati
  - ▶ Ali je često poželjno zbog početne vrijednosti
  - ▶ Preporučeno je definirati varijable
- ▶ Primjer:

```
▶ class Klasa {
 }
```

```
$x = new Klasa();
$x->a = 4;
var_dump($x);
```

# Konstante

---

- ▶ Klase mogu imati i konstante
  - ▶ Konstanta ne može biti polje ili instanca!
- ▶ Ispred imena konstante ne dolazi znak \$
- ▶ Pristup operatorom ::
  - ▶ Unutar klase konstanti je moguće pristupiti i sa *self::imeKonstante*
- ▶ Primjer:
  - ▶ 

```
class Klasa {
 const konstanta = 'vrijednost';
}
```

  

```
echo Klasa::konstanta, "\n";
```

# Metode (1)

---

- ▶ Sintaksa definicije metoda je ista kao i za definiciju funkcija
  - ▶ Metode su unutar klase
    - ▶ Ne smije biti istoimenih metoda!
  - ▶ Sva pravila kao i kod funkcija su ista
  - ▶ Metode mogu imati definiranu vidljivost
    - ▶ Ispred ključne riječi `function`
    - ▶ Pretpostavljena vidljivost je `public`
- ▶ Pristup operatorom `->`
  - ▶ Kao i kod varijabli

# Metode (2)

---

## ► Primjer:

```
► class Klasa {
 public function javna() {
 echo "Ovdje sam!\n";
 }
 private function privatna() {
 echo "Ne dam se!\n";
 }
}
```

```
$x = new Klasa();
$x->javna();
$x->privatna(); // greska!
```

# \$this (1)

---

- ▶ Posebna varijabla koja označava "ovu" instancu
  - ▶ Instanca nad kojom se metoda poziva
  - ▶ Omogućuje korištenje varijabli instance
    - ▶ Bez \$this se definiraju lokalne varijable
  - ▶ Omogućuje poziv metoda instance
    - ▶ Bez \$this se pozivaju globalne **funkcije**

# \$this (2)

---

## ► Primjer 1:

```
► class Klasa {
 private $priv;

 private function brisi() {
 $this->priv = 0; // varijabla instance
 }

 public function napraviNesto() {
 $a = 42; // lokalna varijabla
 $this->brisi(); // poziv na funkciju instance
 gotov(); // poziv na globalnu funkciju
 }
}
```

# \$this (3)

---

## ► Primjer 2:

```
► class Klasa {
 private $priv = 42;
 public function postavi($a) {
 $this->priv = $a;
 }
}
```

```
$x = new Klasa();
$x->postavi(4);
$y = new Klasa();
var_dump($x);
var_dump($y);
```

# Pristup varijablama preko metoda

---

- ▶ Tzv. *geteri* i *seteri*
- ▶ Omogućuju izvođenje neke akcije pri izmijeni varijable
  - ▶ Izmjena neke varijable možda traži izmjenu drugih
  - ▶ Moguće promijeniti tip ili sam podatak prije izmjene varijable
    - ▶ Npr. `get/setCelsius`, `get/setKelvin` koji koriste samo jednu varijablu
- ▶ Omogućuju zabranu izmjene varijabli
  - ▶ Varijabla se definira kao `private` (ili `protected`)
  - ▶ Implementira se samo *getter*
- ▶ Preporučena praksa
- ▶ Primjer:
  - ▶ 

```
function getBoja() {
 return '#' . $this->boja;
}
```

# Čarobne (engl. *magic*) metode

---

- ▶ Postoji nekoliko rezerviranih metoda
- ▶ Koriste dva znaka `_` kao prefiks
  - ▶ `__construct` - konstruktor
  - ▶ `__destruct` - destruktork
  - ▶ `__call` - pozvana pri pozivu na nepostojeću metodu
  - ▶ `__callStatic` - kao `__call`, samo za statičke metode
  - ▶ `__get`, `__set`, `__isset`, `__unset` - pozvane pri pristupu na nepostojeću varijablu
  - ▶ `__toString` - pozvana pri pretvorbi instance u string
  - ▶ `__clone` - pozvana pri kopiranju instance
  - ▶ ...
- ▶ <http://www.php.net/manual/en/language.oop5.magic.php>

# Konstruktor

---

- ▶ Poziva se pri samom instanciranju
- ▶ Omogućuje "inteligentnu" inicijalizaciju
  - ▶ Moguće instancirati klasu i postaviti ju kao inicijalnu vrijednost neke varijable
- ▶ Omogućuje parametriranje pri instanciranju
  - ▶ Parametri se nalaze unutar zagrada iza imena klase koja se instancira operatorom `new`

# Konstruktor – primjer

---

```
▶ class Klasa {
 private $a;
 function __construct($a) {
 $this->a = $a;
 echo "Konstruirana! ($a)\n";
 }
}
```

```
$x = new Klasa(42);
var_dump($x);
```

```
▶ Konstruirana! (42)
object(Klasa)#1 (1) {
 ["a": "Klasa":private] =>
 int(42)
}
```

# Pretvorba instance u string

---

- ▶ Pomoću čarobne funkcije `__toString`
  - ▶ Za tekstualnu reprezentaciju instance
- ▶ Primjer:

```
▶ class Cijena {
 var $iznos, $valuta = 'kn';

 function __construct($iznos) {$this->iznos = $iznos;}

 function __toString() {
 return sprintf("%.4f$this->valuta",
 $this->iznos);
 }
}

$racun = new Cijena(42);
echo $racun, "\n";
```

# Statičke metode i varijable

---

- ▶ Statičke metode i varijable su nevezane uz instancu
  - ▶ Moguće im pristupiti bez instanciranja
  - ▶ `$this` ne postoji unutar statičkih metoda
- ▶ Pristupa im se operatorom `::`
- ▶ Primjer:

```
class Klasa {
 static $staticna;
 static function stvori() {
 return new Klasa();
 }
}
```

```
$x = Klasa::stvori();
var_dump($x); var_dump(Klasa::$staticna);
```

# Zadatak (2) – kartas.php



## ► Definirati dvije klase: Karta i Spil

### ► Karta

► Mora imati metode: `getBoja`, `getJacina` i `__toString`

► Boja može biti: "Žir", "Zelje", "Srce", "Bundeva"

□ `static $boja = array("Žir", "Zelje", "Srce", "Bundeva");`

► Jačina može biti: "7", "8", "9", "10", "Dečko", "Dama", "Kralj", "As"

□ `static $jacina = array("7", ..., "As");`

### ► Spil

► u konstruktoru ispuniti sa svim mogućim kartama

□ Polje s kartama je privatna varijabla

► mora imati metode:

□ `promijesaj` – promiješa karte (koristiti funkciju `shuffle`)

□ `izvuciKartu` – vraća prvu kartu i briše ju iz polja

# Zadatak (2) – nastavak

---



- ▶ Podijeliti 6 karata svakom od 4 igrača
  - ▶ Koristiti globalnu funkciju (ne metodu!) koja vraća polje sa setovima karata
    - ▶ Svaki element polja je set (polje) karata za svakog pojedinog igrača
  - ▶ Rezultat podjele jasno prikazati
    - ▶ Koristiti `__toString` za prikaz karata