Pitanje 1:

What is the output of the following code?

```
$first = "second";
$second = $first;
echo $$$first;
```

Ponuđeni odgovori:

first

second

an empty string

an error

Pitanje 1:

What is the output of the following code?

```
$first = "second";
$second = $first;
echo $$$first;
```

Ponuđeni odgovori:

first

second

an empty string

an error

Pitanje 2

What is the output of the following code?

```
function increment ($val)
{
    $GET['m'] = (int)$_GET['m']+1;
}
$_GET['m'] = 1;
echo $_GET['m'];
```

Pitanje 2

What is the output of the following code?

```
function increment ($val)
{
    $GET['m'] = (int)$_GET['m']+1;
}
$_GET['m'] = 1;
echo $_GET['m'];
```

1

Pitanje 3

Which of the following are valid identifiers?
(Choose 3)

function 4You() {}

function _4You() {}

function object() {}

$1="Hello";

$_1="Hello World";

Pitanje 3

Which of the following are valid identifiers? (Choose 3)

function 4You() {}

function _4You() {}

function object() {}

$1="Hello";

$_1="Hello World";

Pitanje 4

What is the output of the following code?

```
function increment($val)
{
++$val;
}
$val=1;
increment($val);
echo $val;
```

Pitanje 4

What is the output of the following code?

```
function increment($val)
{
++$val;
}
$val=1;
increment($val);
echo $val;
```

1

Pitanje 5

How often will the function counter() be executed in the following code?

```
function counter($start,&$stop)
{
        if($stop>$start)
        {
            return;
        }
        counter($start--,++$stop);
}
$start = 5;
$stop = 2;
counter($start,$stop);
```

Ponuđeni odgovori:

3

4

5

6

Pitanje 5

How often will the function counter() be executed in the following code?

```
function counter($start,&$stop)
{
        if($stop>$start)
        {
            return;
        }
        counter($start--,++$stop);
}
$start = 5;
$stop = 2;
counter($start,$stop);
```

Ponuđeni odgovori:

3

4

5

6

Pitanje 6

What is the output of the following code?

```
function fibonacci(&$x1=0,&$x2=1)
{
$result = $x1+$x2;
$x1 = $x2;
$x2 = $result;
return $result;
}
for($i=0;$i<10;$i++) {
echo fibonacci() . ‘,’ ;
}
```

Ponuđeni odgovori:

- An error
- 1,1,1,1,1,1,1,1,1,1,
- 1,1,2,3,5,8,13,21,34,55,
- Nothing

Pitanje 6

What is the output of the following code?

```
function fibonacci(&$x1=0,&$x2=1)
{
$result = $x1+$x2;
$x1 = $x2;
$x2 = $result;
return $result;
}
for($i=0;$i<10;$i++) {
echo fibonacci() . ‘,’ ;
}
```

Ponuđeni odgovori:

- An error
- **1,1,1,1,1,1,1,1,1,1,**
- 1,1,2,3,5,8,13,21,34,55,
- Nothing

Agenda

- **Klase i objekti**
  - Uvod u objektno orjentisani koncept
  - Pojam klase i objekta
  - Konstrukcija klase
  - Prava pristupa
  - Zaštita atributa
  - Nasleđivanje klasa
  - Instanciranje klasa
  - Apstraktne klase
  - Prostori imena

Php Core

Klase i objekti

ITAcademy

Klase u php-u

- Klase su šabloni za kreiranje objekata
- Klasa se takođe smatra i korisnički definisanim tipom podatka
- Klase su nam potrebne radi:

  - boljeg struktuiranja projekta

  - višestruke upotrebe njegovih komponenti

  - jednostavnije raspodele odgovornosti (u developerskom i sistemskom smislu

Gde možemo videti klase

- Klase ne možemo videti u grafičkom smislu, ali ih možemo lako prepoznati
- Na primer:
  - Facebook **korisnik** je u sistemu prezentovan kao klasa
  - **Tenk** je u igri world of tanks predstavljen kao klasa
  - **Automobil** je u poslovnoj aplikaciji za prodaju automobila predstavljen kao klasa
- Često, klasa se u sistemu može identifikovati kao slika tabele iz baze podataka

Bitna osobina klasa je da mogu da nasleđuju jedna drugu

Bazni objekat

šta može (metode):

prikaži

sakrij

Šta ima (atributi):

X

Y

oblik

Pokretni objekat

šta može (metode):

kreni

stani

Šta ima (atributi):

brzina

Tenk

(metode):

pucaj

eksplodiraj

...

Šta ima (atributi):

health

armour

PREVOZNO SREDSTVO

šta može (metode):
- stani
- kreni

atributi:
- stoji
- kreće se

AUTOMOBIL

šta može (metode):
- upali svetla
- ugasi svetla

koji su mu atributi:
- svetla (upaljena, ugašena)
- boja (crvena, zelena, plava)
- broj vrata (3 ili 5)
- tip (sportski, porodični, poslovni, terenski)

SPORTSKI AUTOMOBIL

koji su mu atributi:
- klasa (formula, reli)
- proizvođač

SUBARU IMPREZZA

AVION

šta može (metode):
- uvuče točkove
- izvuče točkove

koji su mu atributi:
- broj putnika
- tip (putnički, sportski, vojni)

F-18

šta može (metode):
- pucaj iz topa
- otpusti gorivo

koji su mu atributi:
- boja (sivi, smb)
- poletanje (nosač, zemlja)

Konstrukcija klase

```
<?php
    class MojaKlasa
        {

        }
?>
```

Polja klase (podaci)

```
<?php
    class MojaKlasa
    {
      var $x;
    }
?>
```

Ovako definisano polje je podrazumevano javno vidljivo

Moguće i:

```
<?php
    class MojaKlasa
    {
      var $x = 10;
    }
?>
```

Nije moguće:

```
<?php
    class MojaKlasa
    {
      var $x = 10 + 3;
    }
?>
```

Metode klase (funkcionalnosti klase)

```
<?php
class MojaKlasa
{
   public function Uradi() {
      echo "Dobar dan iz klase";
   }
}
?>
```

Konstrukcija klase

```
<?php
class klasa
    {
    function __construct()
            {
            echo "ovo je konstruktor";
            }
    function __destruct()
            {
            echo "ovo je destruktor";
            }
    function metoda()
            {
            return "Ovo je obicna metoda";
            }
    }
?>
```

Presecanje klase HTML kodom

```php
<?php
//moguce je
class a
{
    public function f()
    {
        ?>
        <div>Hello!</div>
        <?php
    }
}
?>
```

```php
<?php
//nije moguce
class b
{
    $a = 10;
?>
<?php
    $b = 20;
}
?>
```

Instanciranje klase i pristupanje članovima

```
<?php
$objekat = new klasa();
echo $objekat->a;
$objekat->a = 100;
?>
```

Ključne reči this, self i parent

- **this**
  - Predstavlja aktuelnu instancu klase

  $this->myField = "hello";
- **self**
  - Predstavlja aktuelnu klasu

  self::$myField = "hello";
- **parent**
  - Predstavlja aktuelnu instancu klase

  parent::setX(25);

  parent::getX();

Vidljivost varijabli i funkcija klase (polja i metoda)

- ***private***
  - vidljive samo u okviru klase
- ***public***
  - vidljive i van klase
- ***protected***
  - vidljive samo u okviru klase u kojoj su definisane ili klase koja je nasleđuje

private

```
<?php
class mojaKlasa
    {
        private $a=10;
        public function prikaziA()
        {
        return $this->a;
        }
    }
$mk = new mojaKlasa();
echo $mk->a; //NE MOZE
echo $mk->prikaziA(); //MOZE
?>
```

public

```
<?php
class mojaKlasa
{
public $a=10;
var $b=20;
}

$mk = new mojaKlasa();
echo $mk->a;
echo $mk->b;
?>
```

protected

```
class mojaKlasa
    {
        protected $a=10;
    }
class mojaDrugaKlasa extends mojaKlasa
    {
    public $b=20;
        public function __construct()
        {
        $this->b=$this->a;
        }
    }
$mk = new mojaDrugaKlasa();
echo $mk->b;
```

Primer: klasa Point

- Potrebno je kreirati klasu koja će čuvati podatke o tački.
- Klasa će imati polja x i y i metod Show.
- U poljima x i y će se nalaziti koordinate tačaka, a metod Show će prikazivati poruku:

  Position is x: 23, y: 32 (brojevi su proizvoljni)

Rešenje: Korak 1

Korak 1: Kreirati klasu ključnom rečju class

```
class Point
{
}
```

Rešenje: Korak 2

Korak 2: Dodati polja klase $x i $y:

```
class Point
{
   public $x;
   public $y;
}
```

Rešenje: Korak 3

Korak 3: Dodati metod Show:

```
class Point
{
   public $x;
   public $y;
   public function Show(){
   }
}
```

Rešenje: Korak 4

Korak 4: Dodati telo metodi Show:

```
class Point
{
   public $x;
   public $y;
   public function Show(){
     echo "Position is x: {$this->x} y: {$this->y} ";
  }
}
```

Rešenje: Korak 5

Korak 5: Instancirati klasu:

```
class Point
{
   public $x;
   public $y;
   public function Show(){
       echo "Position is x: {$this->x} y: {$this->y} ";
   }
}
$p = new Point();
```

Rešenje: Korak 6

Korak 6: Dodeliti vrednosti poljima:

```
class Point
{
   public $x;
   public $y;
   public function Show(){
      echo "Position is x: {$this->x} y: {$this->y} ";
   }
}
$p = new Point();
$p->x = 23;
$p->y = 32;
```

Rešenje: Korak 7

Korak 7: Aktivirati metod Show:

```
class Point
{
   public $x;
   public $y;
   public function Show(){
      echo "Position is x: {$this->x} y: {$this->y} ";
   }
}
$p = new Point();
$p->x = 23;
$p->y = 32;
$p->Show();
```

Statički elementi

- Može im se pristupiti bez instanciranja klase
- Označavaju se ključnom rečju static
- Pristupa im se uz pomoć scope resolution operatora ::
- :: je scope resolution operator (Paamayim Nekudotayim)
- Koristi se za pristupanje statičkim elementima klase ili konstantama
- Zašto bi ih koristili?
  - Zato što nam za određeni element nije potrebna instanca
  - Zato što hoćemo da neki element bude samo jedan, i globalno vidljiv u programu

```php
<?php

class MojaKlasa
{
public static $a=10;
}

echo MojaKlasa::$a;

?>
```

Zaštita atributa

```
class klasa
{
   private $a=0;
        function set($vrednost)
                { $this->a=$vrednost; }
        function get()
                { return $this->a; }
}
```

Get i set metode (getteri i setteri)

```
<?php
    class Mk
    {
       private $name;
       public function __get($name) {
            return $this->$name;
          }
       public function __set($name,$value) {
            $this->$name = $value;
          }
    }

    $mk = new Mk();
    $mk->ime="Pera";
    echo $mk->ime;
?>
```

Konstante klase

- Konstanta:

```
define("kljuc","ovo je neka konstanta");
echo (kljuc);
```

- Konstanta u klasi:

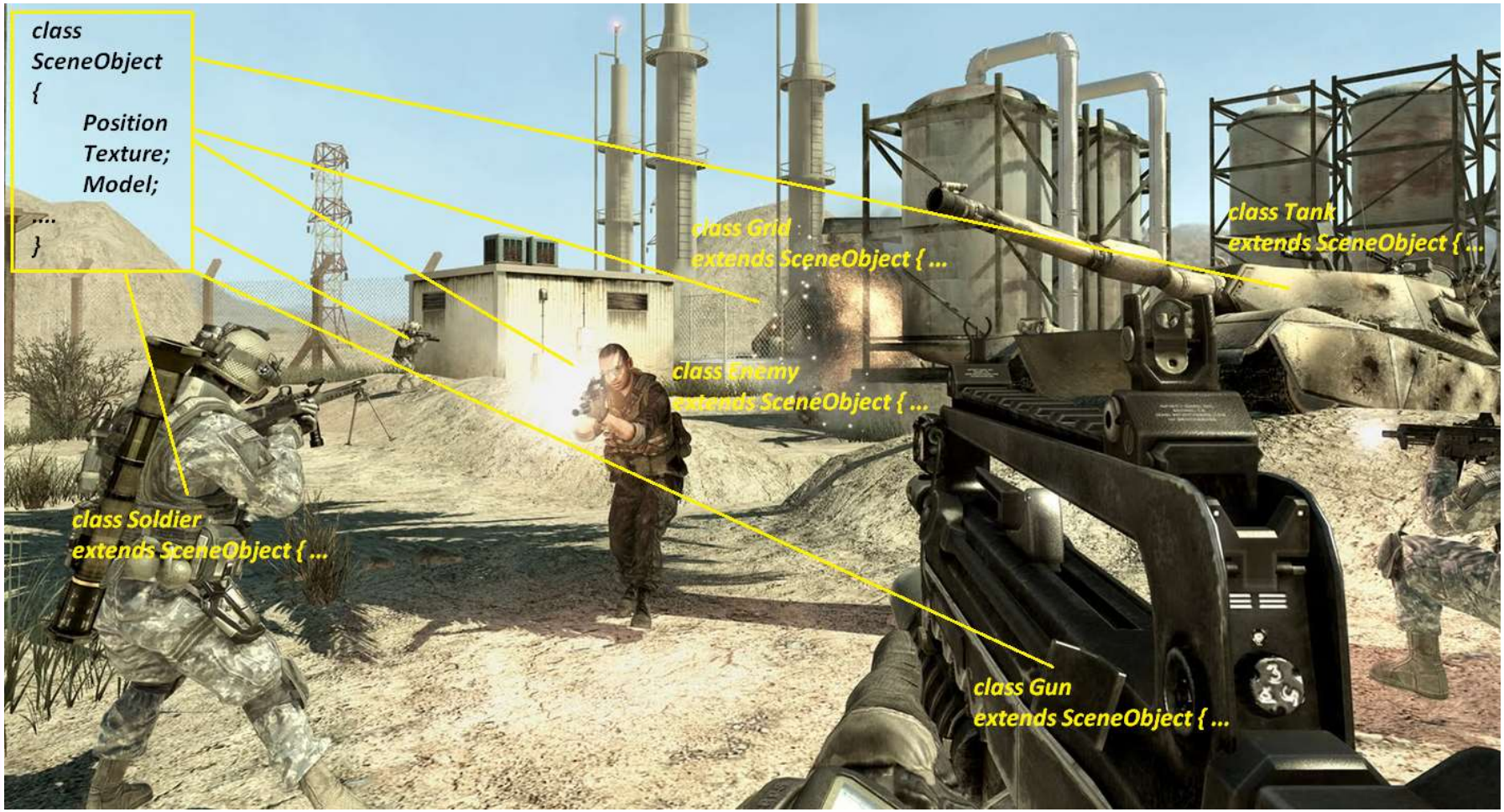
```
class mojaKlasa
{
const konstanta="moja konstanta";
}
```

Pristupanje konstanti klase

```
<?php
class Point
{
const konstanta="moja konstanta";
}
echo Point::konstanta;
$p = new Point();
//ne moze
//echo $p->konstanta;
?>
```

Nasleđivanje klasa

- Primer: Mogući objektni model FPS igre



- Napomena: Primer ne predstavlja pravi objektni model igre sa slike

Nasleđivanje klase

- Preuzimaju se sve dostupne karakteristike roditeljske klase
- Postiže se ključnom rečju extends

```
<?php
class Roditelj
{
   public $x = 10;
}
class Dete extends Roditelj
{

}
$d = new Dete();
echo $d->x;
?>
```

Prepisivanje članova

- Neki članovi definisani u dete klasi, možda već postoje u roditeljskoj klasi. Na primer, konstruktor.
- U sledećem primeru, konstruktor klase dete će pregaziti konstruktor roditelja:

```
class Roditelj
{
   function Roditelj()
   {
      echo "ovo je konstruktor roditelja";
   }
}
class Dete extends Roditelj
{
   function Dete()
   {
      echo "ovo ce pregaziti konstruktor roditelja kada instanciramo dete";
   }
}
$d = new Dete();
```

Pristup roditeljskim članovima

```
class Roditelj
{
   function Roditelj() {
      echo "ovo je konstruktor roditelja";
   }
}
class Dete extends Roditelj
{
   function Dete() {
      parent::Roditelj();
      echo "ovo ce pregaziti konstruktor roditelja kada
    instanciramo dete";
   }
}
$d = new Dete();
```

Prepisivanje toString metode

Pitanje:

- Šta bi odgovorili ako Vas neko pita šta je ovo?

Odgovor 1:

- **Engine**
- Horsepower 138 @ 6300 Size 1.8L/110 Torque 123 @ 3800 Type Gas I4
  Fuel Data
- EPA Fuel Economy Est - City 25 MPG EPA Fuel Economy Est - Hwy 36 MPG Fuel Tank Capacity, Approx 15.6 gal
  Drivetrain Specifications
- Drivetrain Front Wheel Drive
  Interior Specifications
- Front Headroom 39.3 in Front Hiproom 53.0 in Front Legroom 42.3 in Front Shoulderroom 54.8 in Passenger Capacity 5 Second Headroom 37.9 in Second Hiproom 52.4 in Second Legroom 35.4 in Second Shoulderroom 53.9 in
  Exterior Specifications
- Curb Weight N/A lbs Height, Overall 58.1 in Length, Overall 181.0 in Min Ground Clearance N/A in Tread Width, Front 60.7 in Tread Width, Rear 61.3 in Turning Diameter - Curb to Curb 35.7 ft Turning Diameter - Wall to Wall N/A ft Wheelbase 105.7 in Width, Max w/o mirrors 70.7 in
  Wheels
- Front Wheel Material Steel Rear Wheel Material Steel
  Tires
- Front Tire Size P215/60R16 Rear Tire Size P215/60R16
- .....

I još mnogo informacija

Odgovor 2:

- **Chevy Cruze**

Pitanje:

```
<?php
class Person
{
    public $firstname;
    public $lastname;
    public $personalid;
    public $birthdate;
    public $whatever;
}
$p = new Person();
$p->firstname = "Luke";
$p->lastname = "Skywalker";
$p->personalid = "12345";
$p->birthdate = "19 BBY";
?>
```

Kako bi opisali objekat $p?

- **Odgovor 1:**

  Ime Luke, prezime Skywalker, lični broj 12345, datum rođenja 19 BBY

- **Odgovor 2:**

  Luke Skywalker

- Odgovor 2 će nam u velikom broju slučajeva biti dovoljan, ali klasa Person ni na koji način ne može da zna da su baš te informacije one koje su nam potrebne. Ali joj mi možemo to reći, metodom ***__toString***.

Prepisivanje toString metode

```
class Mk
{
   public function __toString()  {
      return "This is from my class";
      }
}
```

Metod __toString mora vratiti tekst

Metod __toString ne bi trebalo da ispisuje sadržaj na izlaz već samo da vraća string

U prethodnom primeru izmena bi bila sledeća:

```
class Person
{
   public $firstname;
   public $lastname;
   public $personalid;
   public $birthdate;
   public $whatever;

   public function __toString()
   {
      return $this->firstname . " " . $this->lastname;
   }
}
```

Postavljanje vrednosti poljima putem konstruktora

```
<?php
class Person
{
   public $firstname;
   public $lastname;
   public $personalid;
   public $birthdate;
   public $whatever;
   public function
     __construct($firstname,$lastname,$personalid,$birthdate,$whatever="whatever") {
      $this->firstname = $firstname;
      $this->lastname = $lastname;
      $this->personalid = $personalid;
      $this->birthdate = $birthdate;
      $this->whatever = $whatever;
   }
public function __toString()  {
      return $this->firstname . " " . $this->lastname;
   }
}
$p = new Person("Luke","Skywalker","12345","19 BBY");
echo $p;
?>
```

Preuzimanje tipa:

```
<?php
class MojaKlasa { }
$mk = new MojaKlasa();
echo get_class($mk);
?>
```

Kloniranje i kopiranje

- Objekti se mogu preneti jedino po referenci:

```
<?php
class MojaKlasa { public $x; }
$mk = new MojaKlasa();
$mk->x = 10;
$mk1 = $mk;
$mk1->x = 25;
echo $mk->x;
?>
```

Rezultat na izlazu je 25

Kloniranje i kopiranje

- Objekti se mogu preneti jedino po referenci:

```
<?php
function f($kl)
{
   $kl->x = 15;
}
class MojaKlasa { public $x; }
$mk = new MojaKlasa();
$mk->x = 10;
f($mk);
echo $mk->x;
?>
```

Rezultat na izlazu je 15 iako nije korišćen operator &

Kloniranje i kopiranje

```
<?php
class MojaKlasa { public $x; }
$mk = new MojaKlasa();
$mk->x = 10;
$mk1 = clone $mk;
$mk1->x = 25;
echo $mk->x;
?>
```

- Rezultat na izlazu će biti 10, jer je objekat kloniran

Uništavanje objekata

unset($objekat);

Zadatak:

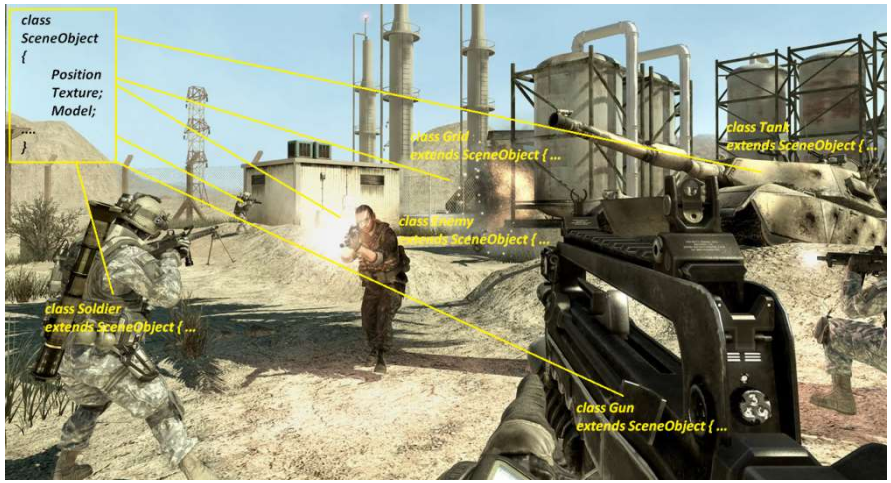
- Kreirati klasu kalkulator
- Klasa treba da ima dva polja za operande i jedno polje za rezultat
- Klasa mora da ima metode add i sub koje će da vrše sabiranje ili oduzimanje operanada i da smeštaju rezultat u polje sa rezultatom
- Klasa mora imati metod toString koji će prikazati rezultat u formi:
  - Trenutna vrednost promenljive rezultat je: 123
- Klasa mora imati parametrizovani konstruktor za postavljanje inicijalnih vrednosti operanada
- Potrebno je instancirati i isprobati klasu

Rešenje:

```
<?php
class Calculator
{
   public $opa;
   public $opb;
   public $res;
   public function __construct($opa,$opb){
      $this->opa = $opa;
      $this->opb = $opb;
   }
   public function Add(){
      $this->res = $this->opa + $this->opb;
   }
   public function Sub(){
      $this->res = $this->opa - $this->opb;
   }
   public function __toString(){
      return "Trenutna vrednost promenljive rezultat je: $this->res";
   }
}
$c = new Calculator(2,3);
$c->Add();
echo $c . "<br>";
$c->Sub();
echo $c;
?>
```

Apstraktne klase

- Nije ih moguće instancirati
- Metodi koji su obeleženi kao abstract moraju postojati u dete klasi i ne smeju imati telo
- Zbog čega bi koristili apstraktnu klasu?
  - Hoćemo da napravimo šablon za nasleđivanje
  - Hoćemo da obavežemo klase koje nasleđuju da će rešiti neki problem, ali ne želimo da ulazimo u to na koji način će ga rešiti



```php
<?php
abstract class proba
{
   protected $rez;
   protected function plus($a,$b)
   {
      $this->rez = $a+$b;
   }
   abstract function stampaj();
}

class p extends proba
{
   public function __construct($a,$b)
   {
    $this->plus($a,$b);
   }
   public function stampaj()
   {
    echo $this->rez;
   }
}
$p=new p(3,2);
$p->stampaj();
?>
```

Interfejsi

- Implementiraju se u klase
- Sve metode moraju biti ispoštovane u klasi koja implementira
- Zbog čega bi koristili interfejs?
  - Hoćemo da damo do znanja klasi da mora da implementira neku funkcionalnost, ali nas ne interesuje na koji način će to uraditi
    - Zašto bi ovo hteli?
      - Zato što hoćemo da pozovemo kreiranu funkcionalnost iz nekog dela programa
  - U primeru sa slike (FPS igra)
    - Svaka od klasa na sceni ima metod Draw (ili sličan)
    - Klasa koja rukuje scenom, iscrtava objekte tako što aktivira metod Draw svakog od njih

Interfejsi – kreiranje i implementacija

```
<?php
interface interfejs
    {
        public function saberi($a,$b);
    }
class proba implements interfejs
    {
        public function saberi($a,$b)
            {
                echo $a+$b;
            }
    }
?>
```

Autoload klasa

- Funkcija __autoload se startuje automatski svaki put kada instanciramo klasu koja nije dostupna unutar skripte

```
<?php
function __autoload($cname){
    echo "Nema klase " . $cname;
}
$mk = new MojaKlasa();
?>
```

Autoload klasa

- Ponašanje autoload klase možemo iskoristiti za automatsko učitavanje klase iz biblioteke klasa

```
function __autoload($class_name)
{
require_once $class_name . '.php';
}
```

Zadatak 1

- Kreirati klasu Page
- Klasa treba da ima polja head, body, scripts, styles i title
- Ovi podaci se unose kroz konstruktor klase
- Klasa treba da ima metod render koji prikazuje html kod strane
- Klasa treba da ima pregažen metod toString, koji vraća html kod strane

```php
<?php
class Page
{
   public $head;
   public $body;
   public $title;
   public $styles;
   public $scripts;

   public function __construct($head,$body,$title,$styles,$scripts)
   {
    $this->body=$body;
    $this->head=$head;
    $this->title=$title;
    $this->styles=$styles;
    $this->scripts=$scripts;
   }
   public function render()
   {
     echo $this;
   }
   public function __toString()
   {
      $res = "<head>";
      $res.= "<title>" . $this->title . "</title>";
      $res.= $this->head;
      $res.= "<script>". $this->scripts ."</script>";
      $res.= "<style>" . $this->styles . "</style>";
      $res.= "</head>";
      $res.= "<body>" . $this->body . "</body>";
      return $res;
   }
}
$pg = new Page("","<div>PAGE BODY</div>","MY PAGE","","");
$pg->render();
?>
```

Zadatak 2

- Kreirati klasu user
- Klasa mora sadržati podatke o korisniku u poljima (firstname, lastname, nickname, password, email i id)
- Podaci se unose kroz konstruktor klase
- Klasa mora sadržati metod show koji prikazuje sve podatke o korisniku
- Klasa mora sadržati metod toString, koja vraća sve podatke o korisniku
- Klasa mora sadržati statičke metode fromId i checkCredentials koji treba da vraćaju novog korisnika i true ili false vrednost ukoliko korisnik postoji (oba metoda samo treba da simuliraju rad. Ne treba stvarno proveravati korisnika)

```php
<?php
class User
{
   public $userId;
   public $firstName;
   public $lastName;
   public $nickName;
   public $email;
   public $password;
   public function __construct($userId,$firstName,$lastName,$nickName,$email,$password)
   {
      $this->userId = $userId;
      $this->firstName = $firstName;
      $this->lastName = $lastName;
      $this->nickName = $nickName;
      $this->email = $email;
      $this->password = $password;
   }
   public function showUser()
   {
      echo $this;
   }
   public function __toString()
   {
      return "id: " . $this->userId . "<br> first name: " . $this->firstName . "<br> last name: " . $this-
      >lastName . "<br> nick: " . $this->nickName . "<br> email: " . $this->email . "<br> password: " .
      $this->password;
   }
}
$u = new User(1,"Anakin","Skywalker","Darth Vader", "darthvader@galacticempire.universe","123");
$u->showUser();
?>
```

Zadatak 3

- Napraviti apstraktnu klasu Geom
- Klasa poseduje polja a i b
- Klasa mora da sadrži apstraktan metod getArea
- Naslediti klasu Geom klasama Rectangle i Circle i pravilno realizovati metod getArea za obe klase
- U konstruktoru klasa kreirati parametrizaciju tako da, u klasi Circle, jedan ulazni parametar se mapira na polje a i ovo se polje koristi kao r, dok se u klasi Rectangle oba polja mapiraju na a i b

```php
<?php
abstract class Geom
{
   const PI = 3.14;
   public $a;
   public $b;
   public abstract function getArea();
}
class Circle extends Geom
{
   public function __construct($r)
   {
      $this->a=$r;
   }
   public function getArea()
   {
      return self::PI * pow($this->a,2);
   }
}
class Rectangle extends Geom
{
      public function __construct($a,$b)
   {
      $this->a=$a;
      $this->b=$b;
   }
   public function getArea()
   {
      return $this->a*$this->b;
   }
}
$c=new Circle(2);
echo $c->getArea();
$r=new Rectangle(2,3);
echo $r->getArea();
?>
```

Zadatak 4:

- Kreirati klasu Form
- Klasa mora sadržati javna polja action, method, enctype i controls
- Klasa treba da sadrži tri konstante za tri enctype tipa
- Konstruktor klase prihvata tri parametra: action, method i enctype i, obezbeđuje podrazumevane vrednosti za ova tri parametra.
- Klasa sadrži metod addControls, koji omogućava postavljanje vrednosti controls polja. Ovaj metod prihvata kao parametar string koji predstavlja kontrole.
- Klasa sadrži metod render, koji prikazuje formu i kontrole unutar nje. Posle svih kontrola, prikazuje se i submit kontrola koja aktivira formu

```php
<?php
class Form
{
   const MULTIPART = "multipart/form-data";
   const APP = "application/x-www-form-urlencoded";
   const TEXT = "text/plain";

   public $method;
   public $action;
   public $enctype;
   public $controls;

   public function __construct($method="post",$action="",$enctype=self::MULTIPART)
   {
      $this->method = $method;
      $this->action = $action;
      $this->enctype = $enctype;
   }

   public function addControls($controls)
   {
      $this->controls = $controls;
   }

   public function render()
   {
      echo "<form method='".$this->method."' action='".$this->action."' enctype='".$this->enctype."' >";
      echo $this->controls;

      echo "<br><input type='submit' value='confirm' />";
      echo "</form>";
   }
}

$ctrls = <<<CTRLS
<input type="text" name="name" />
<input type="text" name="password" />
CTRLS;

$f=new Form();
$f->addControls($ctrls);
$f->render();
?>
```

Zadatak 5

- Potrebno je napraviti klasu Control koja predstavlja kontrolu html forme.
- Klasa mora sadržati tri polja: name, type i value. Ova polja predstavlja će odgovarajuće atribute kontrole.
- Klasa u konstruktoru mora prihvatati vrednosti za sva tri polja i dodeljivati te vrednosti poljima
- Klasa mora sadržati metod show, koja će prikazivati kontrolu

- Potreno je kreirati niz, i napuniti ga kontrolama, tako da sadrži dve tekst kontrole (firstname i lastname), jednu password kontrolu i jednu submit taster kontrolu.

- Potrebno je prolaskom kroz niz, prikazati sve kontrole iz niza na strani

Rešenje – zadatak 5

```php
<?php
class Control
{
   public $type;
   public $value;
   public $name;
   public function __construct($type,$value,$name)
   {
      $this->type=$type;
      $this->name=$name;
      $this->value=$value;
   }
   public function show()
   {
      echo "<input type='".$this->type."' name='".$this->name."' value='".$this->value."' />";
   }
}

$controls = array(
new Control("text","firstname","Peter"),
new Control("text","lastname","Parker"),
new Control("text","email","Parker"),
new Control("password","password","Peter"),
new Control("submit","submit","Confirm")
);

foreach($controls as $c) {
   $c->show();
   echo "<br>";
}
?>
```

Prostori imena

```
<?php
namespace MyNamespace;
class MyClass
{
	public $x=10;
}

namespace MyNamespace1;
class MyClass
{
	public $x=20;
}
?>

<?php
	include "index.php";
	$mk = new MyNamespace\MyClass();
	echo $mk->x;
	$mk = new MyNamespace1\MyClass();
	echo $mk->x;
?>
```

```php
<?php
namespace MyNamespace
{
class MyClass
{
public $x=10;
}
}
namespace MyNamespace1
{
class MyClass
{
public $x=20;
}
}
?>
```

autoload sa namespace-om

```
function __autoload($classname){
    require_once “classes/" .
    str_replace('\\','/',$classname) . '.class.php';
}
```

ili

```
function autoloader($classname){
    require_once "classes/" . Sys::CLASSES_DIR ."/" .
    str_replace('\\','/',$classname) . '.class.php';
}
spl_autoload_register("autoloader");
```

Trait-ovi

```php
<?php
    class Base {
        public function sayHello() {
            echo 'Hello ';
        }
    }

    trait SayWorld {
        public function sayHello() {
            parent::sayHello();
            echo 'World!';
        }
    }

    class MyHelloWorld extends Base {
        use SayWorld;
    }

    $o = new MyHelloWorld();
    $o->sayHello();
?>
```

vladimir.maric@link.co.rs