Programación orientada a objetos

Encapsulamiento

Este concepto consiste en la ocultación del estado o de los datos propios de un objeto, de forma que sólo es posible modificar los mismos mediante los métodos definidos para dicho objeto (como se verá más adelante set/get).

Hay cosas que se han de hacer desde la propia clase y no desde cualquier clase.

Sonia Sánchez Sierra

Métodos Getter / Setter:

- Set: establece el valor de una propiedad
- Get: devuelve el valor de una propiedad
- Uno de cada para cada propiedad

Sonia Sánchez Sierra

Ámbito de las variables y métodos

- public: se puede acceder desde fuera de la clase, es decir desde la instancia de la clase.
- private: sólo se puede acceder desde dentro de la propia clase, por un método de la clase. No se puede acceder desde una instancia.
- protected: sólo se puede acceder por la clase y las subclases (herencia). No se puede acceder desde una instancia.

Sonia Sánchez Sierra

facebook0.php

```php
<?php
 class Facebook{
  public $nombre;
  public $edad;
  private $pass;
   public function __construct($nombre,$edad,$pass){
    $this->nombre=$nombre;
    $this->edad=$edad;
    $this->pass=$pass;

    //accedemos desde la propia clase
   }
   public function verInformacion(){

    echo "Nombre:".$this->nombre."<br/>";
    echo "Edad:".$this->edad."<br/>";
    echo  "Password".$this->pass."<br/>";
    }
 }
$face=new Facebook("Sonia",25,"1234");
$face->nombre="Laura";
$face->verInformacion();

 ?>
```

```php
$face=new Facebook("Sonia",25,"1234");
$face->nombre="Laura"; //se puede cambi

$face->verInformacion();
$face->pass="123";
```

Fatal error: Uncaught Error: Cannot access private property Facebook::$

Sonia Sánchez Sierra

Modificadores de acceso

Ejemplo: cuenta de Facebook . Datos personales visibles para todos pero la contraseña no se ve, sería private.

Se pueden añadir a nuestros métodos y atributos, para poder acceder a ellos.

Son los ’ getters y setters ‘

Ejemplo: facebook1.php

Sonia Sánchez Sierra

```php
<?php
 class Facebook{
    public $nombre;
    public $edad;
    private $pass;

   public function __construct($nombre,$edad){
    $this->nombre=$nombre;
    $this->edad=$edad;
    //accedemos desde la propia clase
   }
   public function verInformacion(){

    echo "Nombre:".$this->nombre."<br/>";
    echo "Edad:".$this->edad."<br/>";
    echo "password". $this->getPass();
    }
 //establecemos set y get para ese atributo

  public function setPass($passn){
    $this->pass=$passn;
   }
   public function getPass(){
    return $this->pass;
   }
 }
$face=new Facebook("Sonia",25);
$face->setPass("1234");
$face->verInformacion();

 ?>
```

Definimos set y get
Métodos modificadores de acceso

facebook1.php

Sonia Sánchez Sierra

Crear la clase triángulo teniendo en cuenta los principios de la POO

Ejercicio

Pasos:

1. Pensar en la clase (propiedades y métodos)
2. Código en PHP (encapsulamiento)

3. Crear una instancia de la clase con parámetros pasados desde un cuadro de diálogo.

Teniendo en cuenta la encapsulación y la modularización.

Sonia Sánchez Sierra

Características de un triángulo:

- Altura
- Base

Posibles métodos:

- Calcular el área
- Calcular perímetro

Definir constructor

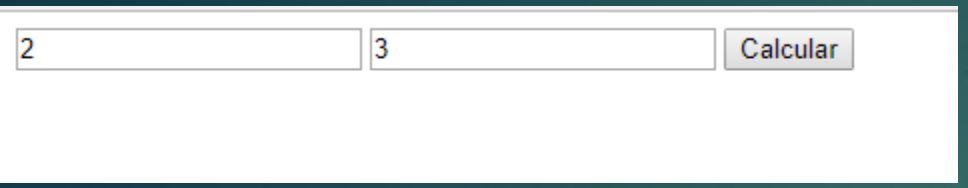
Definir set? Get?

Sonia Sánchez Sierra

```php
<?php
class Triangulo{
    private $base;
    private $altura;
    public function __construct($b,$a){
        $this->base=$b;
        $this->altura=$a;
    }
    public function  getBase(){
        return $this->base;
    }
    public  function getAltura(){
       return $this->altura;
    }
    public function calcularArea(){
        return ($this->base*$this->altura)/2;
    }
}
 $triangulo1=new Triangulo(5,3);
 echo "El area es".$triangulo1->calcularArea();
 ?>
```


Encapsulamiento
Métodos modificadores
de acceso:definimos get

claseTriangulo.php



FormTriangulo.php
TestTriangulo.php
Triangulo.php

```php
<?php

include("Triangulo.php");
 $triangulo1=new Triangulo($_REQUEST['fbase'],$_REQUEST['faltura']);
 echo "El area es".$triangulo1->calcularArea();

  ?>
```

```php
<?php
class Triangulo{
    private $base;
    private $altura;
    public function __construct($b,$a){
        $this->base=$b;
        $this->altura=$a;
    }
    public function  getBase(){
        return $this->base;
    }
    public  function getAltura(){
       return $this->altura;
    }
    public function calcularArea(){
        return ($this->base*$this->altura)/2;
    }
}

 ?>
```

Ejercicio: Añadir el código necesario para obtener el resultado:

```
El area es3
El area de 2 y altura: 3 es: 3
```

Sonia Sánchez Sierra

```php
echo "El area es".$triangulo1->calcularArea()."<br/>";
echo "El area de ".$triangulo1->getBase()." y altura: ".$triangulo1->
    getAltura()." es: ".$triangulo1->calcularArea();
```

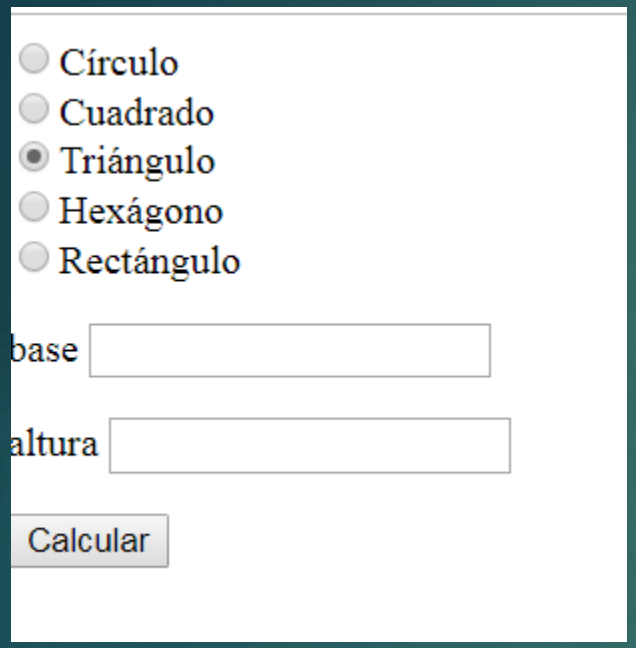
Hacer el mismo proceso para el círculo y el rectángulo .

1. Pensar en la clase (propiedades)
2. Código en PHP (encapsulamiento)

3. Crear una instancia de la clase con parámetros pasados desde un cuadro de diálogo.

Sonia Sánchez Sierra

Exercici



- Creación de un triángulo
- Cálculo de su área

El área del cuadrado: 16

Sonia Sánchez Sierra

1. Public: la **propiedad o método** podrá usarse en cualquier parte del script
2. Private: la **propiedad o método** sólo podrá usarse en la clase a la que pertenece
3. Protected: la **propiedad o método** se podrá usar por la clase a la que pertenece y por sus descendientes.
4. Final: la **clase o método** no puede ser sobreescrito en clases descendientes.
5. Abstract: la **clase o método** no puede ser usado directamente, ha de ser heredado primero para usarse.

Sonia Sánchez Sierra

```
class padre
{

    public function saluda() {

    echo "Hola Mundo";

    }

}
```

Y luego la extendemos a otra:

```
class hija extends padre
{

}
```

Podríamos usar todos los métodos y propiedades de la clase padre, pues han sido «heredados» por la hija. Así, esto...

```
$mi_hija = new hija();

$mi_hija->saluda();
```

Sonia Sánchez Sierra

```php
class Perro
{
    public $nombre = "Rudolf";
    public function ladrar(){
        print "Guau!";
    }
}
class Bulldog extends Perro {
    public function ladrar(){
        print "Woof!";
    }
}
$cachorro = new Bulldog(); // Instancia de la clase hija
$cachorro->nombre = "Jeffrey"; // Heredamos la propiedad padre $nombre y le asignamos Jeffrey
echo $cachorro->nombre; // Devuelve Jeffrey
$cachorro->ladrar(); // Devuelve Woof! ya que ha sobreescrito la función padre ladrar().
```

Sobrescribir métodos

Sonia Sánchez Sierra

```php
class Perro
{
    private $nombre;
    private function ladrar(){
        print "Guau!";
    }
}
class Bulldog extends Perro {}
$cachorro = new Bulldog();
$cachorro->nombre = "BunBuns";
var_dump($cachorro);
/*
Devuelve:
object(Bulldog)[1]
  private 'nombre' (Perro) => null
  public 'nombre' => string 'BunBuns' (length=7)
*/
// Los métodos en cambio no se pueden usar:
$cachorro->ladrar(); // Fatal error: Call to private method
```

Sonia Sánchez Sierra