Classi astratte e Traits

Php offre la possibilità di definire una classe astratta, ossia una classe che non può essere istanziata direttamente ma che funge da indirizzo per le altre classi.

Ma perchè si usano?

```
Persona
|
|
Studente - Docente

Carlo mangia un panno, Carlo aveva fame.
Carlo mangia un panno, aveva fame.
```

Come fare a raggruppare tutti questi? Sotto una classe **Person**.

```
<?php

abstract class Person
{
  public $name;


  public function __construct($name)
  {
    $this→name = $name;
  }

}

class Student extends Person
{
}
```

```
class Teacher extends Person
{
}

//$person1 = new Person('Francesco');
$student1 = new Student('Francesco');
print_r($student1);
//print_r($person1);
```

```
<?php
abstract class Person

Errore

Fatal error: Uncaught Error: Cannot instantiate abstract class Person in /Us
ers/francescomansi/Desktop/index.php:25
Stack trace:
```

Abstract Method

Generalizzando le classi, possiamo generalizzare anche i metodi.

```
<?php

abstract class Person
{
  public $name;


  public function __construct($name)
  {
    $this→name = $name;
  }

  public function hello()
  {
```

```
    echo "$this->name \n";
  }
}

class Student extends Person
{
}

class Teacher extends Person
{
}

$stundet1 = new Student('Mansi');
$teacher1 = new Teacher('Francesco');

$stundet1->hello();
$teacher1->hello();
```

Metodo corretto

```
<?php

abstract class Person
{
  public $name;


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

  abstract function hello();
}
```

```
class Student extends Person
{
  public function hello()
  {
    echo "Buongiorno Professor $this->name\n";
  }
}


class Teacher extends Person
{
  public function hello()
  {
    echo "Fermati $this->name, altrimenti ti metto una nota \n";
  }
}

class RemoteTeacher extends Teacher
{
}

$stundet1 = new Student('Mansi');
$teacher1 = new Teacher('Francesco');

$stundet1->hello();
$teacher1->hello();
```

Trait

Vi ricordate che nella lezione iniziale della OOP abbiamo detto che PHP non supporta la multiereditarietà? Bene, Trait sono una pezza che risolvono quel problema.

```
<?php
```

```
abstract class Animal
{
  use Life;
  public $pedigree;

  public function __construct($pedigree)
  {
    $this→pedigree = $pedigree;
  }
}
abstract class Person
{
  use Life;
  public $name;


  public function __construct($name)
  {
    $this→name = $name;
  }



  abstract function hello();
}

class Student extends Person
{
  public function hello()
  {
    echo "Buongiorno Professor $this→name\n";
  }
}




class Teacher extends Person
```

```
{
  public function hello()
  {
    echo "Fermati $this→name, altrimenti ti metto una nota \n";
  }
}

class RemoteTeacher extends Teacher
{
}

trait Life
{

  public function breathe()
  {
    echo "Sto resipirando bro";
  }
}

$student1 = new Student('Mansi');
$teacher1 = new Teacher('Francesco');

$student1→hello();
$teacher1→hello();

$student1→breathe();
```

Dependency Injection e Compostion

Esempio Essere Umano

```
<?php

abstract class artiSuperiori
{
  abstract function muoviLeBraccia();
```

```
}

abstract class artiInferiori
{
  abstract function muoviLeGambe();
}

class Gesticolare extends artiSuperiori
{
  public function muoviLeBraccia()
  {
    echo "Sto muovendo le mani in maniera incontrollata\n";
  }
}

class Afferrare extends artiSuperiori
{
  public function muoviLeBraccia()
  {
    echo "Con le mie mani sto afferrrando qualcosa\n";
  }
}

class Correrre extends artiInferiori
{
  public function muoviLeGambe()
  {
    echo "Sto correndo piu veloce di Jacobs\n";
  }
}

class Saltare extends artiInferiori
{
  public function muoviLeGambe()
  {
    echo "Ho fatto una schacciata a canestro saltando 3 metri\n";
  }
}
```

```
class Francesco
{
  public $mani;
  public $gambe;

  public function __construct($mani, $gambe)
  {
    $this->mani = $mani;
    $this->gambe = $gambe;
  }
}


$francesco = new Francesco('Indica', 'Calcia');

print_r($francesco);
```

Non è definito. Procedo quindi con il richiamare gli oggetti nella maniera corretta.

```
class Francesco
{
  public $mani;
  public $gambe;

  public function __construct(ArtiSuperiori $mani, ArtiInferiori $gambe)
  {
    $this->mani = $mani;
    $this->gambe = $gambe;
  }


  public function up()
  {
    $this->mani->muoviLeBraccia();
```

```
  }
  public function down()
  {
    $this->gambe->muoviLeGambe();
  }
}


$francesco = new Francesco(new Gesticolare, new Correrre,);
$francesco->up();
$francesco->down();
```