CHT2520 Advanced Web Programming

Matthew Mantle m.e.mantle@hud.ac.uk

Today's session - OOP PHP - Inheritance

Inheritance

- Inheritance lets one class make use of code in another class.
    - One of the 'four pillars of OOP'
- We can think of it as an easy way to for us to re-use code...

A Simple Inheritance Example

```php
class Animal {
    protected $name;

    function __construct($name){
         $this->name = $name;
    }
    public function sleep(){
        return "{$this->name} is sleeping.";
    }

}
```

```php
class Cat extends Animal{
     //nothing in here
}
$myCat = new Cat("Mackeral");
echo "<p>{$myCat->sleep()}</p>"; // Displays Mackeral is sleeping.
```

Inheritance

- Cat **extends** Animal.
- It inherits all of the Animal class’s properties and methods
  - e.g. I can call the `sleep()` method on a Cat object.
- Note the use of the **protected** access modifier
  - The property can be accessed by child classes i.e. Cat but isn't publicly accessible.

Inheritance

```
class Cat extends Animal{
     public function scratch(){
        return "{$this->name} scratched you";
    }
}
$myCat = new Cat("Mackeral");
echo "<p>{$myCat->sleep()}</p>"; // Displays Mackeral is sleeping.
echo "<p>{$myCat->scratch()}</p>"; // Displays Mackeral scratched.
```

- Using Animal as a starting point, we can add methods and properties to Cat
- We can have lots of other classes that also inherit from Animal e.g. Dog with a `fetch()` method.

Abstract Classes

- Often we want the parent class to simply be a template for creating other classes.
- Note the keyword `abstract`

```
abstract class Animal {
    protected $name;
    function __construct($name){
         $this->name=$name;
    }
    public function sleep(){
        return "{$this->name} is sleeping.";
    }
    abstract public function talk();
}
```

- abstract methods are like rules
  - All child classes must implement a `talk()` method

Abstract Classes

```
class Dog extends Animal{
    public function fetch($item)
    {
        return "{$this->name} has picked up the {$item}";
    }
    public function talk() //we have to implement a talk method
    {
        return "{$this->name} says woof";
    }
}
```

```
$myDog = new Dog("Fido");
echo "<p>{$myAnimal->sleep()</p>"; // Displays Dave is sleeping.
echo "<p>{$myDog->fetch("ball")}</p>"; // Displays Fido has picked up the ball.
echo "<p>{$myDog->talk()}</p>"; // Fido says woof
```

Polymorphism

- Means 'many different forms'
  - The last of the 'four pillars'
- Allows objects of different types to respond to method calls of the same name e.g. `talk()` .

```php
$animals = [];
$animals[] = new Cat("Paws");
$animals[] = new Dog("Lick");
$animals[] = new Cat("Mog");
$animals[] = new Snake("Sammy");

foreach($animals as $animal){
    echo $animal->talk();
}
```

- It makes our code flexible. We can add new types of Animal (e.g. a Snake) and the code will still work.

Practical Work

- Inheritance in OOP
- There is also a second practical on Namespacing
  - How we organise applications that use multiple classes
- Again this is all largely theory, we'll do something useful with OOP next week.
- There's more to OOP e.g. related ideas to inheritance - interfaces, traits.