CENTER FOR TEXTUAL STUDIES AND
DIGITAL HUMANITIES

DIGH 402 - Introduction to Digital Humanities Design and Programming

Spring Semester 2014

Week 6

Object Oriented Programming

Why Object Oriented Programming?

- it is possible to write complex and useful sites using procedural outline, functions…

- true value of OOP is in a concept known as ‘encapsulation’
    - this allows us to associate values and functions together in one unit : the **object**

- objects allows us to collect values together, add functionality to the unit…

Basic Vocab

- class = outline / blueprint / design for creating a given object

- object = something that encapsulates the design etc of the class…

- method = a function that belongs to an object

- property = a variable that belongs to an object

Object Oriented Programming

Intro to Object Oriented Programming

- a given ‘class’ is a blueprint, a set of instructions for how to create our object
    - effectively, it describes an object

- classes can represent all sorts of entities within our application, for example our framework

a simple class

- shows the class declaration, and we’ll save as user.class.php

- we have one property, $username

- two methods, __construct() and name()
    - each method receives a parameter, $username and $name
    - we ‘return’ a value from each method

- $this is a special variable that is always available within an object’s scope
    - it refers to the current object

```
class User {

    public $username;

    public function __construct($username) {
    $this->username = $username;
    return true;
    }

    public function name($name) {
    //return the user's name
    return true;
    }

}
```

Object Oriented Programming

Intro to Object Oriented Programming - Class Constructors

- the __construct() function has two underscores
    - it's a special function
    - it's called when we instantiate an object
    - it's known as the 'constructor'

- the constructor is always called when we instantiate an object

- used to set up and configure the object before it's returned for use in the main code

```
class User {

        public $username;

        public function __construct($username) {
        $this->username = $username;
        return true;
        }

        public function name($name) {
        //return the user's name
        return true;
        }

}
```

Object Oriented Programming

Intro to Object Oriented Programming - Instantiating an Object

- to instantiate, or create, an object
    - use the 'new' keyword
    - specify the name of the class for the given object
    - pass any required parameters expected by the constructor

eg: require 'user.class.php';
$user = new User('fulcanelli');

- first, we require the file that contains the class definition

- then we instantiate a new 'User' object passing the name parameter the constructor expects

- we store the result in an object called '$user'

EXAMPLE OUTPUT

```
class User {

        public $username;

        public function __construct($username) {
        $this->username = $username;
        return true;
        }

        public function name($name) {
        //return the user's name
        return true;
        }

}
```

Object Oriented Programming

Intro to Object Oriented Programming - Instantiating an Object

Example output for class User

- we can see that it is an object of class ‘User’

- there is currently one property

- we can see the name and value of each property

```
class User {

        public $username;

        public function __construct($username) {
        $this->username = $username;
        return true;
        }

        public function name($name) {
        //return the user’s name
        return true;
        }

}
```

Object Oriented Programming

Intro to Object Oriented Programming - Using Objects

- we've declared an object and instantiated an object

- we need to access a property, call a method…eg:

```
//require
require 'user.class.php';
//instantiate an object
$user = new User('fulcanelli');

//access a property
echo "User's name: ".$user->username;

//call a method
$user->name($name);
```

EXAMPLE OUTPUT

- object operator = ->
    - goes between the object and the property or method you want to access

Object Oriented Programming

Intro to Object Oriented Programming - Visibility

- visibility of a property or method can be set using keywords ‘public’, ‘protected’, ‘private’

- ‘public’ can be accessed everywhere

- ‘protected’ can be accessed only within the class itself and by inheritance from the parent

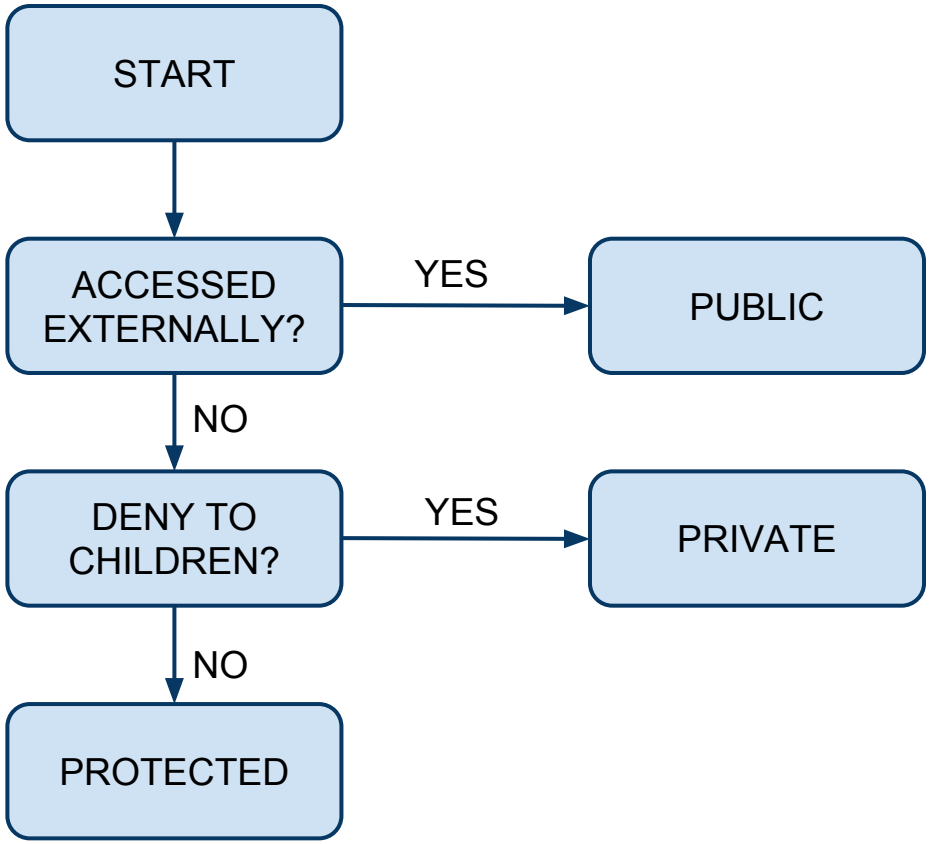
- ‘private’ can only be accessed by the class itself

eg:

```
class {
        public $ver1 = ‘ver 1’;
        protected $ver2 = ‘ver 2’;
        private $ver3 = ‘ver 3’;
}
```

Object Oriented Programming

Intro to Object Oriented Programming - Choosing the Correct Visibility



Object Oriented Programming

Intro to Object Oriented Programming - Using Static Properties and Methods

- we can define class properties and methods that are ‘static’
    - a static method or property can be used without instantiating the object first

- mark as static by putting the ‘static’ keyword after ‘public’ etc

- ‘scope resolution operator’ :: is used to access ‘static’ properties or methods
    - eg: $user = User::get_instance();

- ‘static’ property is a variable that belongs to the class only, not any object
    - isolated from any other property, even another of the same name in an object of this class

- ‘static’ method has no need to access any other part of the class
    - you can’t refer to $this inside a static method (because no object has been created to refer to)

- NB: often used in PHP libraries, references etc where the functionality is independent of any object properties

Object Oriented Programming

Intro to Object Oriented Programming - Object Inheritance

- ‘Inheritance’ is effectively how classes relate to each other
    - a class can inherit from another class
    - parent to child…

- classes can inherit or extend a parent class

- classes are unaware of other classes inheriting from them
    - therefore, no limits on how many child classes per parent

- child class has all the parent’s characteristics
    - we can add or change any elements that need to be different for the child

- using ‘User’ class as an example we can create child classes
    - user type etc
    - registration
    - login
    - user details...

Object Oriented Programming

Intro to Object Oriented Programming - Type Hinting in PHP

- type hinting
    - allows methods to only accept parameters from objects of a specified class

    eg: public function delete(User $user) {
        //delete the user from the DB
        return true;
        }

- you can type hint any object name
    - PHP is a dynamic and weakly type language
    - no type hinting for simple types such as strings, number types etc

- type hinting allows us to be sure about the type of object passed to a given method/function
(Method is actually a function used in the context of a class/object - an object can have methods and properties...)

- this allows us to make assumptions in our code
    - eg: properties and methods available as a result

Object Oriented Programming

Intro to Object Oriented Programming - Objects and References in PHP

```
$user1 = $user2
```

- variable $user1 will contain the same value as $user2
  - we end up with two independent variables, but same value

- objects work in a different way

```
$user1 = new User();
$user1->username = ‘fulcanelli’;
$user2 = $user1;
$user2->username = ‘amelie’;
```

- $user2 is not a copy of $user1, but another reference to the same object for the class User
  - this is called a ‘reference’

Object Oriented Programming

Intro to Object Oriented Programming - Objects and References in PHP (cont'd)

- objects are always passed by reference
    - when you pass an object into a function, the function operates on the same object
    - any change inside the function is reflected outside the function
    - objects provide a reference to the original object rather than produce a copy

- this means that if a function operates on an object passed in
    - there's no need to return the object from the function
    - any change will be reflected in the original object

- 'clone' keyword allows us to create a separate copy of an object
    - cloned object will have same properties etc as original object
    - changes made to cloned object will not change original object
    - original object will maintain its own values etc...

Object Oriented Programming

Intro to Object Oriented Programming - Getters and Setters in PHP

- public, protected, or private to control visibility for a property or method

- another option is to mark all properties as protected
    - we can then use ‘getter’ and ‘setter’ method to access them
    - they basically allows us to ‘get’ and ‘set’ the values

eg:

```
class User {
        protected $username;

        function getName() {
        return $this->username;
        }

        function setName($value) {
        $this->username = $value;
        return true;
        }
}
```

Object Oriented Programming

Intro to Object Oriented Programming - Getters and Setters in PHP (cont'd)

- very useful for tracing object code that accesses properties

- the getter and setter methods offer an access point each time we need a property
    - this provides a 'hook' or 'intercept' point
    - we might use these methods to log what information was updated
    - or perhaps to add some access control logic…

- often a personal choice whether to use 'getters' and 'setters' or access properties directly

Object Oriented Programming

Intro to Object Oriented Programming - Exceptions

- an object oriented approach to handling errors

- exceptions themselves are ‘objects’ and ‘Exception’ is a built-in class in PHP

- an ‘exception’ object contains information about
    - where the error occurred (filename and line number)
    - an error message
    - and optionally an error code

- exceptions are considered a more elegant way of handling errors

- they allow us to react to exceptions in the course of execution, dependent upon the severity of the problem

- we can assess the code situation and then react by either recovering or bailing out gracefully

Object Oriented Programming

Intro to Object Oriented Programming - Exceptions (cont'd)

- we can also extend exception objects, customise their data and behaviour…

eg: Try/Catch block

```
try {
        $db = new PDO('mysql:host=notthere');
        echo "Connected to Database";
} catch (Exception $e) {
        echo 'Oh no! '.$e->getMessage();
}
```

- we can also throw our own exceptions

eg:

```
throw new Exception ('a useful error message string!');
```

Object Oriented Programming

Intro to Object Oriented Programming

That’s enough for now - let’s consider OOP code for our framework