PHP

CLASSES AND OBJECTS

What is Class?

- Class is a programmer-defined data type, which includes local functions as well as local data.
- It is a template for making many instances of the same kind (or class) of object.

How to create a Class?

- In order to create a **class**, we group the code that handles a certain topic into one place.
- For example, we can group all of the code that handles the users of a blog into one class, all of the code that is involved with the publication of the posts in the blog into a second class, and all the code that is devoted to comments into a third class.

Defining PHP Classes

- **Example 1**

The general form for defining a new class in PHP is as follows –

```
<?php
   class phpClass {
      var $var1;
      var $var2 = "constant string";

      function myfunc ($arg1, $arg2) {
         [..]
      }
      [..]
   }
?>
```

How to add properties to a class?

- Properties can accept values like strings, integers, and booleans (true/false values), like any other variable.
- For example,

```
1  class Car {
2    public $comp;
3    public $color = 'beige';
4    public $hasSunRoof = true;
5  }
```

What is Object?

- An individual instance of the data structure defined by a class.
- Class is defined once and then many objects can be made that belong to it.
- Objects are also known as instance.

How to create objects from a class?

- We can create several objects from the same class, with each object having its own set of properties.
- In order to work with a class, we need to create an **object** from it. In order to create an object, we use the new keyword. For example:

How to create objects from a class?

- For Example,

  $physics = new Books();

  $maths = new Books();

  $chemistry = new Books();

- We can create more than one object from the same class, and then give each object its own set of properties.

Calling Member Functions

- $physics->setTitle("Physics for High School");
- $chemistry->setTitle("Advanced Chemistry");
- $maths->setTitle("Algebra");
- $physics->setPrice(10);
- $chemistry->setPrice(15);
- $maths->setPrice(7);
- $physics->getTitle();
- $chemistry->getTitle();
- $maths->getTitle();
- $physics->getPrice();
- $chemistry->getPrice();
- $maths->getPrice();

Constructor Functions

- Constructor Functions are special type of functions which are called automatically whenever an object is created. So we take full advantage of this behaviour, by initializing many things through constructor functions.
- PHP provides a special function called **__construct()** to define a constructor. You can pass as many as arguments you like into the constructor function.
- **Example 2**

Constructor Functions

behaviour, by initializing many things through constructor functions.

PHP provides a special function called **__construct()** to define a constructor. You can pass as many as arguments you like into the constructor function.

Following example will create one constructor for Books class and it will initialize price and title for the book at the time of object creation.

```
function __construct( $par1, $par2 ) {
   $this->title = $par1;
   $this->price = $par2;
}
```

Now we don't need to call set function separately to set price and title. We can initialize these two member variables at the time of object creation only. Check following example below –

```
$physics = new Books( "Physics for High School", 10 );
$maths = new Books ( "Advanced Chemistry", 15 );
$chemistry = new Books ("Algebra", 7 );

/* Get those set values */
$physics->getTitle();
$chemistry->getTitle();
$maths->getTitle();

$physics->getPrice();
$chemistry->getPrice();
$maths->getPrice();
```

Destructor Functions

- Like a constructor function you can define a destructor function using function **__destruct()**. You can release all the resources with-in a destructor.
- **Example 2**

What is Interfaces?

- Interfaces are defined to provide a common function names to the implementers.
- Object interfaces allow you to create code which specifies which methods a class must implement, without having to define how these methods are handled.
- Interfaces are defined in the same way as a class, but with the *interface* keyword replacing the *class* keyword and without any of the methods having their contents defined.

What is Interfaces?

- All methods declared in an interface must be public; this is the nature of an interface.
- To implement an interface, the *implements* operator is used. All methods in the interface must be implemented within a class; failure to do so will result in a fatal error.
- Classes may implement more than one interface if desired by separating each interface with a comma.

What is Interfaces?

- Interfaces can be extended like classes using the extends operator.
- The class implementing the interface must use the exact same method signatures as are defined in the interface. Not doing so will result in a fatal error.

Example of Interface

- Interface can be declared as follows :

```
interface iTemplate
{
    public function setVariable($name, $var);
    public function getHtml($template);
}
```

- Interfaces can be implemented as follows

```
class Template implements iTemplate
{
```

How to get an object's properties?

- Once we create an object, we can get its properties. For example:

```
1 echo $bmw -> color;
2 echo $mercedes -> color;
```

Result:

beige
beige

How to set the object's properties?

- In order to set an object property, we use a similar approach.
- For example, in order to set the color to 'blue' in the bmw object:

```
1 $bmw -> color = 'blue';
```

How to set the object's properties?

- and in order to set the value of the $comp property for both objects:

```
1 $bmw -> comp = "BMW";
2 $mercedes -> comp = "Mercedes Benz";
```

- Once we set the value of a property, we can get its value.

How to set the object's properties?

- In order to get the color of the $bmw object, we use the following line of code:

```
1 echo $bmw -> color;
```

Result:

blue

How to set the object's properties?

- We can also get the company name and the color of the second car object.

```
1 echo $mercedes -> color;
2 echo $mercedes -> comp;
```

Result:

beige

Mercedes Benz

Inheritance

- PHP class definitions can optionally inherit from a parent class definition by using the extends clause. The syntax is as follows

```
class Child extends Parent {
   <definition body>
}
```

- Example 5

Inheritance

- The effect of inheritance is that the child class (or subclass or derived class) has the following characteristics –
  - Automatically has all the member variable declarations of the parent class.
  - Automatically has all the same member functions as the parent, which (by default) will work the same way as those functions do in the parent.

Inheritance
Function Overriding

- Function definitions in child classes override definitions with the same name in parent classes.
- In a child class, we can modify the definition of a function inherited from parent class.

- Example 6

Inheritance
Public Members

- Unless you specify otherwise, properties and methods of a class are public.

Inheritance
Private members

- By designating a member private, you limit its accessibility to the class in which it is declared. The private member cannot be referred to from classes that inherit the class in which it is declared and cannot be accessed from outside the class.
- Example 7

Inheritance
Private members

- By designating a member private, you limit its accessibility to the class in which it is declared. The private member cannot be referred to from classes that inherit the class in which it is declared and cannot be accessed from outside the class.
- Example 7

Inheritance
Protected members

- A protected property or method is accessible in the class in which it is declared, as well as in classes that extend that class.
- Protected members are not available outside of those two kinds of classes. A class member can be made protected by using **protected** keyword in front of the member.
- **Example 8**

Inheritance
Calling parent constructors

- **Example 9**

Introspection Functions [5]

- Introspection is a common feature in any programming language which allows object classes to be manipulated by the programmer.
- **Introspection in PHP offers the useful ability to examine classes, interfaces, properties, and methods.**
- PHP offers a large number functions that you can use to accomplish the task.

Introspection Functions [5]

- **class_exists()** – checks whether a class has been defined
- **get_class()** – returns the class name of an object
- **get_parent_class()** – returns the class name of an object's parent class
- **is_subclass_of()** – checks whether an object has a given parent class

- Example 10

Introspection Functions [5]

- **class_exists()** method, which takes a string argument representing the name of the desired class to check, and an optional Boolean value whether or not to call the autoloader in the process.
- The **get_class()** and **get_parent_class()** methods return the class name of an object or its parent's class name respectively. Both takes as arguments an **object instance**.
- The **is_subclass_of()** methods takes an object instance as its first argument and a string, representing the parent class name, and returns whether the object belongs to a class which is a subclass of the class given as argument.

Introspection Functions [5]

- get_declared_classes() – returns a list of all declared classes
- get_class_methods() – returns the names of the class' methods
- get_class_vars() – returns the default properties of a class
- interface_exists() – checks whether the interface is defined
- method_exists() – checks whether an object defines a method

- Example 11

Introspection Functions [5]

- **interface_exists()** method is very similar to **class_exists()** discussed in the first example. It determines whether or not the given interface has been defined and takes a string argument for the interface name and an optional autoloader Boolean.
- The **get_declared_classes()** method returns an array with the names of all of the defined classes and takes no arguments. Depending on what libraries you have loaded (either complied into PHP or loaded with require/include), additional classes could be present.

Introspection Functions [5]

- The **get_class_methods()** takes either an object instance or a string as argument representing the desired class and returns an array of method names defined by the class.
- Notice that from all the properties defined in the ICurrencyConverter class and returned by the **get_class_vars()** method, only $name and $rates appeared in the output. The private and protected properties were skipped.

References

1. https://www.tutorialspoint.com
2. http://php.net/manual/en/language.oop5.interfaces.php
3. http://phpenthusiast.com/object-oriented-php-tutorials/create-classes-and-objects
4. https://www.tutorialspoint.com/php/php_object_oriented.htm
5. https://www.sitepoint.com/introspection-and-reflection-in-php/

THANK YOU