Open Source Week 5 Summary.

Contents

[Templates 3](#_Toc129679731)

[Tables 3](#_Toc129679732)

[Textboxes 3](#_Toc129679733)

# Templates

## Tables

|  |
| --- |
| Codenae.code |
| <form action="confirm.php" method ="post">  <input type ="text" name="txtaccount" />  <input type ="submit"/>  </form> |

|  |  |
| --- | --- |
| Notation | Representation |
|  | Boxes represent classes within our system |
|  | Arrows between classes represent associations |
|  | A diamond on one end of the association indicates a contained class |
| **<<stereotype name>>** | Stereotypes within gullets identify specific types of class or associations |

## Textboxes

The browser generates a request for confirm.php, enclosing the form within the request body

# Objective Oriented Programming – Basics

<html>  
 <head>  
 <title>HTMLFILE</title>  
 </head>  
 <body>  
 Hello I am an HTML file  
 </body>  
</html>

## What is OOP?

OOP stands or object-oriented programming. Previously, we have been using procedural programming which creates procedures or functions that will perform operations on the data in a procedural manner. Object Oriented Programming works by creating ***objects*** that contain both ***data***  and ***functions***.

Object-Oriented Programming has several advantages when compared to procedural programming. In its nature, OOP is faster and easier to execute whilst providing a clear structure for a program. By using ***objects*** and ***functions***, it eliminates the re-usage of code providing cleaner code and easy maintenance.

## What are Classes and Objects?

Within Object-Oriented Programming are to main aspects known as ***Classes*** and ***Objects***.

|  |
| --- |
| Classes |
| Gun |

|  |
| --- |
| Objects |
| AK47  M16  AR15 |

Here is a representation of both:

A ***class***is a template for ***objects***, and an ***object*** is an ***instance*** of a ***class.***

When the individual objects are created, they inherit both ***properties*** and ***behaviors*** from the class, only differing in value of the properties.

## Classes and Objects

As a scenario, we are given a class named Gun. A Gun in realistic terms has properties such as its name, its calibre and issued magazine capacity. These properties can bed defined using variables such as **$name, $calibre** and **$magCapacity** and can be used to store the values of these properties.

When individual objects such as AK47. M16, AR15.etc are created, they inherit all properties and behaviours from the class, but will vary in value of the properties.

### Defining a Class

A class is defined by using the ***‘class’*** keyword, followed by its desired name and a pair of ***{}’s***. Within the curly brackets, all its **properties** and **methods**(*functions*) should be stored.

Here is its syntax:

<?php

    class Gun {

        //code goes here

    }

?>

Now we will add two properties to Gun; **$name** and **$calibre** along with two methods being **set\_name()** and **get\_name()**. These will be used for setting and getting the **$name** property.

In a class, **variables** are referred to as **properties** and **functions** are referred to as **methods.**

<?php

    class gun {

        //Properties

        public $name;

        public $color;

<?php

    class gun {

        //Properties

        public $name;

        public $color;

        //Methods

        function set\_name($name) {

            $this->name = $name;

        }

        function get\_name() {

            return $this->name;

        }

    }

?>

        //Methods

        function set\_name($name) {

            $this->name = $name;

        }

        function get\_name() {

            return $this->name;

        }

    }

?>

### Defining an Object

**Classes** are useless without **objects.** Within a **class** we can create multiple **objects** with **the properties** defined in the **class**.

**Objects** of a **class** are created using the ***new*** keyword. We will now add two **objects**, being ***$ak47*** and ***$m16***.



We will then add two more methods to the **Gun** class, acquiring the **$calibre** property.



### $This keyword

The ***$this*** keyword refers to the current object and can only be used inside methods, for example:



If we wanted to change the value of ***$name***’s property, we can do this inside the class by introducing a ***set\_name();*** method and using ***$this*** like so:

To set the ***public $name*** property to the value passed in the ***set\_name*** method, ***$this*** says to set the ***name*** to whatever is passed through.



Or we can change the value of ***$name’s*** property outside the class by directly changing the property value, like so:

The ***->*** symbol is essentially looking within the ***$ak47*** object for the ***name*** property.



If you wish to check whether an object belongs to a specific class, you can use the ***instanceof*** keyword to verify, for example:

It is essentially asking ‘Is the ***$ak47*** object an instance of the ***Gun*** class?’



## Constructors

A **constructor** allows you prepare an **object’s properties** upon creation of the object. If you make a ***\_\_construct()*** function (two underscores), PHP will automatically call this function when you create an object from a class. It is essentially a template that can replace the repetitive calling of ***set\_name()*** method. For example:

As you can see, no ***set\_name*** is needed to apply the name, rather the name is passed during creation of an instance



If preferred, you are also able to pass through multiple properties using constructors by extending the argument in the ***\_\_construct()*** brackets. For example:

We have passed both **$name and $calibre** through the **\_\_construct()** and have defined a method for each property.



## Destructors

A **destructor** is called when the **object** is destructed or the **script** is **stopped** or **exited**.  
If you create a ***\_\_destruct()*** function, PHP will automatically call this **function** at the end of the script.

For example:  
This can also be used to push out two properties, similar to the ***\_\_construct method()***.

When creating new objects, it is important to use the syntax used in the ***\_\_construct()*** method.



As the **$ak47** object is created, when the script comes to end, the ***\_\_destruct()*** function runs, printing:  
*“The gun is AK-47.”*



# Structured Query Language

Data Definition, Data Manipulation and data Retrieval.

## Data Definition Languages

Data definition language is a syntax for creating and altering database content including tables indices and its users. Data Definition Languages(DDL) works similar to a computer programming language in which it requires a structure and uses a data to formulate a service

### Creating a Database

When you are working with databases, there are various commands you can execute in order to provide specific sets of information. A common instruction used is the ***SHOW DATABASES;*** instruction used to display the databases managed by the MySQL server.

To create a database, the command ***CREATE DATABASE*** followed by a desired name will need to be passed through. It is important to know that after creation, the database is not selected meaning you will need to use the ***USE***(x) command followed by the database name.

Now that a database is created, we now need to create a **table** to sit inside the database. This is performed with the following command:

The general syntax for creating a table is as follows:  
***CREATE TABLE table\_name (  
 column1 datatype,  
 column2 datatype,  
 column 3 datatype,***

***);***



If you wish to display all tables within a database, use the ***SHOW TABLES***;instruction.

### Column Types

Within the *Column*, there are three groups of data types that can be used: **Numeric, Date and Time and Character types.**

When referencing **Numeric** types, its common instances are **integers, floats and decimals**. **Integers** are essentially whole numbers, **float** being fractional numbers and **decimal** being precision numbers including a decimal point.

**Date types** consists of instance such as:

* **DATE** -*YYYY-MM-DD*
* **DATETIME -***YYYY-MM-DD HH:MM;SS*
* **TIMESTAMP –** *same as DATETIME but auto-inputs date of entry*
* **TIME – *HH:MM:SS***

Character types consist of three:

* **CHAR*(length)*** – Length specifies max number of characters, ranging from 1-255
* **VARCHAR*(length)*** – Length between 1 and 255, but only uses number of characters necessary to store the value + one for the length
* **TEXT** – Similar to BLOB (Binary Large object), there is no restriction on size of data.