Chapter 11

Developing Object-Oriented PHP

PHP Programming with MySQL

Revised by A. Philipp – Spring 2010 (Rev SP'11)

2

Objectives

- Study object-oriented programming concepts
- Use objects in PHP scripts
- Declare data members in classes
- Work with class member functions

3

Object-Oriented Programming

- Refers to the creation of reusable software objects that can be easily incorporated into multiple programs
- **Object** (AKA **Components**)refers to
  - programming code
  - data
  - treated as an individual unit or component

4

Object-Oriented Programming

- **Data**
  - refers to information contained within variables or other types of storage structures
- **Methods**
  - functions associated with an object
- **Properties** or **Attributes**
  - variables that are associated with an object

Note: Popular object-oriented programming languages include C++, Java, and Visual Basic

5

Object-Oriented Programming

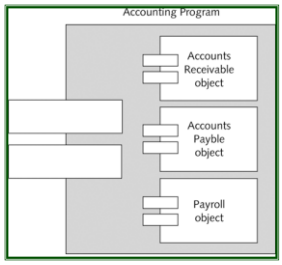

Figure 11-1 Accounting program

6

Understanding Encapsulation

- **Encapsulation**
  - all code and required data are contained within the object itself
  - hides all internal code and data
  - allow users to see only the methods and properties of the object that you allow them to see
  - reduces the complexity of the code
  - prevents other programmers from accidentally introducing a bug into a program, or stealing code
- **Interface**
  - methods and properties that are required for a source program to communicate with an object

7

Classes

- **Classes**
  - Organization for code, methods, attributes, and other information that make up an object
- **Instance**
  - object that has been created from an existing class
- **Instantiating**
  - creating an object from an existing class
- **Inheritance**
  - objects takes on the characteristics of the class on which it is based

8

Using Objects

- Declare an Object
  - **new** operator with a class constructor
- Syntax for Instantiation

```
$ObjectName = new ClassName();
```

- **Identifiers**
  - Must begin with a dollar sign
  - Can include numbers or an underscore
  - Cannot include spaces
  - Are case sensitive

```
$Checking = new BankAccount();
```

9

USING OBJECTS *(CONTINUED)*

Member Selection Notation

->

Access the methods and properties contained in the object

10

USING OBJECTS *(CONTINUED)*

- **Properties**
  - `$Checking -> Balance`
- **Methods**
  - include a set of parentheses at the end of the method name
  - methods can also accept arguments

```
$Checking -> getBalance();
$CheckNumber = 1022;
$Checking -> getCheckAmount($ChkNum);
```

11

DATABASE CONNECTIONS AS OBJECTS

- MySQL database connections as objects
  - instantiating an object from the **mysqli** class
- **Traditional**

```
$DBConnect = mysqli_connect("localhost","root","");
mysqli_select_db($DBConnect,"RealEstate");
……
mysqli_close($DBConnect);
```

- **Object-Oriented**

```
$DBConnectObj = new mysqli("localhost", "root", "", "Real Estate");
……
$DBConnectObj->close();
```

Note: Can select another database
$DBConnectObj->select_db("New Listings");

12

HANDLING MYSQL ERRORS - OOP

- **mysqli constructor**
  - always creates a new object even if connection fails
  - must test for errors
- Must check the values assigned to the `mysqli_connect_errno()` or `mysqli_connect_error()`
- Use error codes/statements with a selection structure

```
$DBConnectObj = @new mysqli("localhost", "root", "", "Real Estate");
if (mysqli_connect_errno()) {
   echo ("<p>Error Code".
   mysqli_connect_errno().":".
   mysqli_connect_error())."</p>";
  }
```

13

Executing SQL Statements

MySQL DataBase Query

Traditional

```
$QueryResult = mysqli_query($DBConnect, $SQLString)
…
$Row = mysqli_fetch_row($QueryResult)
$Row = mysqli_fetch_assoc($QueryResult);
```

Object-Oriented

```
$QueryResultObj=$DBConnectObj->query($SQLstring)
….
$Row = $QueryResultObj->fetch_row();
$Row = $QueryResultObj->fetch_assoc();
```

14

Executing SQL Statements

Use **`query()`** method of **`mysqli`** class

- Must check for NULL not FALSE
- FALSE will result in infinite loop

```
$QueryResultObj=$DBConnectObj->query($SQLstring)
  …
while ((($Row = $QueryResultObj->fetch_row())!=NULL);
  {
  //do something
  }
```

15

Defining Custom PHP Classes

- **Data Structure**
  - system for organizing data
- **Class Members**
  - functions and variables defined in a class
- **Data Members or Member Variables**
  - Class variables
- **Member Functions or Function Members**
  - Class functions

16

Creating a Class Definition

- **Class**
  - Keyword used to write a class definition
- **Class definition**
  - The data members and member functions that make up the class
- Syntax for defining a class

```
class ClassName {
 data/function member definitions
 }
```

17

Creating a Class Definition (Continued)

- **ClassName**
  - Name of the new class
  - Usually begin with an uppercase letter to distinguish them from other identifiers
  - Within the class's curly braces, declare the data type / field names for all information stored in the structure

```
class BankAccount {
  data member definitions
  member function definitions
  }
$Checking = new BankAccount();
```

18

Creating a Class Definition (Continued)

Get_class() returns name of that instantiated it

```
$Checking = new BankAccount();
echo "$Checking object is instantiated from the ".
    get_class($Checking).
    " class.";
```

instanceof determines if an object is instantiated from a given class

```
if ($Checking instanceof BankAccount){
Echo "The \$Checking object is from BankAccount class";
```

19

Storing Classes in External Files

- Class files should begin with **class_**
  - class_BankAccount.php
  - Similar to inc_SomeIncludeFile.php
- Functions for external files in PHP scripts:
  - **include()**
  - **require()**
  - **include_once()**
  - **require_once()**
- Pass to each function the name and path of the external file

20

Collecting Garbage

- Cleaning up or **reclaiming memory** that is reserved by a program
- PHP knows when your program no longer needs a variable or object and **automatically cleans up the memory** when an object of variable is no longer needed
- **Exception: open database connections**

21

Information Hiding

- Any class members that other programmers/clients do not need to access or know about should be **hidden**
- Helps **minimize** the amount of information that needs to **pass in and out** of an object
- **Reduces the complexity** of the code that clients see
- **Prevents other programmers** from accidentally introducing a bug into a program by modifying a class's internal workings

22

Using Access Specifiers

- **`Access Specifiers`**
  - controls a client's access to individual data members and member functions
  - **`Public`**
    - items can be accessed everywhere
  - **`Private`**
    - limits visibility only to the class that defines the item
  - **`Protected`**
    - limits access to inherited and parent classes (and to the class that defines the item)

23

Using Access Specifiers *(Continued)*

- Include an access specifier at the beginning of a data member declaration statement
- Always assign an initial value to a data member when you first declare it

```
class BankAccount {
    public $Balance = 0;
    }
```

24

Working with Member Functions

- Create **`Public`** member functions for any functions that clients need to access
- Create **`Private`** member functions for any functions that clients do not need to access

```
class BankAccount {
    private $Balance = 958.20;
    public function withdrawal($Amount) {
        $this->Balance -= $Amount;
    }
}
```

Note: **`$this`** – psuedo variable that is reference to the calling object

25

Constructors and Destructors

- **Constructor**
  - function is called when a class object is first instantiated
- **Destructor**
  - function is called when the object is destroyed
  - cleans up any resources allocated to an object after the object is destroyed

26

Constructor Functions

Constructor

- Special function - called automatically when an object from a class is instantiated
- The `__construct()` function takes precedence over a function with the same name as the class - *BankAccount in this example*
- Commonly used to handle database connections

```
class BankAccount {
        private $AccountNumber;
        private $CustomerName;
        private $Balance;
        function __construct() { //could be BankAccount()
                $this->AccountNumber = 0;
                $this->Balance = 0;
                $this->CustomerName = "";

        }
```

27

Destructor Functions *(Continued)*

Calling **Destructors**

- When a script ends
- Delete an object with **unset()**
- Adding a destructor function to a class, create a function named **__destruct()**

```
function __construct() {
      $DBConnectObj = new mysqli("localhost", "root", "", "RealEstate");
 }

function __destruct() {
        $DBConnectObj->close();
 }
```

28

Writing Accessor Functions

- **Accessor Functions**
  - Public member functions
  - Client can call to retrieve or modify the value of a data member
  - Begin with the words "set" or "get"
    - **Set** functions **modify** data member values
    - **Get** functions **retrieve** data member values

29

Writing Accessor Functions (Continued)

```
class  BankAccount {
       private $Balance = 0;
       public function setBalance($NewValue) {
          $this->Balance = $NewValue;
       }
       public function getBalance() {
         return $this->Balance;
       }
}

if (class_exists("BankAccount"))
      $Checking = new BankAccount();
else
      exit("<p>The BankAccount is not available!</p>");
$Checking->setBalance(100);
echo "<p>Checking balance: ". $Checking->getBalance() . "</p>";
```

30

Serialization

- Serialization
  - Convert an object into a string
  - Store for reuse
  - Stores data members and member functions into strings
  - Store the data in large arrays
- To serialize/unserialize an object, pass an object name

  $Saved = serialize($Checking);

  $Checking = unserialize($Saved);
- Assign a serialized object to a session variable

  session_start();

  $_SESSION['Saved']=serialize($Checking);

31

Serialization: sleep(), wakeup()

- **`serialize()`**
  - PHP looks in the object's class for **`__sleep()`**
  - specifies which data members of the class to serialize, otherwise serializes all members

    ```
    function __sleep(){
            $SerialVars = array('Balance');
            return $SerialVars;}
    ```
- **`unserialize()`**
  - PHP looks in the object's class for **`__wakeup()`**
  - used to restore DB connections lost during serialization

    ```
    function __wakeup(){  }
    ```

FYI: According to PHP.net any function names that begin with __ (Double _ _ not single _) *are referenced as magical.*

32

Object-Oriented Review

- The term (OOP)" refers to the creation of reusable software objects that can be easily incorporated into multiple programs.
- The term specifically refers to programming code and data that can be treated as an individual unit or component (object)
- The term refers to information contained within variables or other types of storage structures
- The functions associated with an object are called and the variables associated with an object are called or
- Objects are , which means that all code and required data are contained within the object itself
- An represents elements required for a source program to communicate with an object
- In object-oriented programming, the code, methods, attributes, and other information that make up an object are organized into
- An is an object that has been created from an existing class. When you create an object from an existing class, you are "*instantiating the object*"
- A particular instance of an object its methods and properties from a class—that is, it takes on the characteristics of the class on which it is based
- A is a special function with the same name as its class; it is called automatically when an object from the class is instantiated
- The term " refers to a system for organizing data
- The functions and variables defined in a class are called Class variables are referred to as **data members** or **member variables**, whereas class functions are referred to as **member functions** or function **members**
- A contains the data members and member functions that make up the class
- PHP provides the following functions that allow you to use external files in your PHP scripts: `include()`, `require()`, `include_once()`, and `require_once()`

33

Object-Oriented Review

- The principle of [illegible] states that class members should be hidden when other programmers do not need to access or know about them
- [illegible] control a client's access to individual data members and member functions
- [illegible] refers to the process of converting an object into a string that you can store for reuse
- A [illegible] is a special function that is called automatically when an object from a class is instantiated
- When you serialize an object with the `serialize()` function, PHP looks in the object's class for a special function named [illegible], which you can use to perform many of the same tasks as a destructor function
- When the `unserialize()` function executes, PHP looks in the object's class for a special function named [illegible], which you can use to perform many of the same tasks as a constructor function
- A [illegible] cleans up any resources allocated to an object after the object is destroyed
- [illegible] are public member functions that a client can call to retrieve the value of a data member
- [illegible] are public member functions that a client can call to modify the value of a data member