OOP Concepts

Session 24

Objectives

- *Explain the OOP Concepts*
- *Explain inheritance*
- *Explain the use of classes*
- *Describe the use of constructors*
- *Explain the use of objects*

Introduction

- Object-Oriented Programming (OOP) is:
  - Term used to characterize a programming language
  - Widely accepted paradigm for programming languages
- OOP uses three basic concepts are as follows:
  - Classes
  - Objects
  - Methods
- Additional concepts in object-oriented languages are as follows:
  - Inheritance
  - Abstraction
  - Polymorphism
  - Event Handling
  - Encapsulation

- Defines
  - Data type of the data structure
  - Type of functions to be performed on the data structure
- Unit of execution in an object-oriented system are objects
- Combines data and functions in a single unit

- The basic concepts of an object-oriented programming are as follows:
  - **Object:**
    - Consists of data structures and functions for manipulating the data
    - Data structure refers to the type of data while function refers to the operation applied to the data structures
    - An object is a self-contained run-time entity
    - An analysis of the programming problem is done in terms of objects and nature of communication between them

- **Class:**
  - Contains a collection of similar types of objects
  - Has its own properties
  - Once a class is specified, a number of objects can be created that belong to this category
- **Abstraction:**
  - Defines the process of selecting the common features from different functions and objects
  - The functions those perform same actions can be connected into a single function using abstraction

- **Encapsulation:**
  - Specifies the process of combining data and objects into a single unit
  - The data cannot be accessed directly; it can be accessed only through the functions present inside the unit
  - It enables data encryption
- **Polymorphism:**
  - Defines the use of a single function or an operator in different ways
  - The behavior of that function will depend on the type of the data used in it

- **Inheritance:**
  - Specifies the process of creating a new class from the existing one
  - The new class that is formed is called the derived class
  - The existing class is called the base class

Inheritance

- Inheritance means creating a new class with the help of a base class
- A base class is called the parent class and the derived class is called the child class
- A class is an object that contains variables and functions of different data types
- The properties, such as variables, functions, and methods of the base class are transferred to the newly derived class
- The `extends` keyword needs to be used for inheriting a new class from the base class
- A derived class can also have its own variables and functions

- The different types of inheritances are as follows:
  - **Single Inheritance:**
    - Contains only one base class and inherits the properties of the base class
    - In single inheritance, the derived class works with properties derived from the base class along with its own properties
  - **Multiple Inheritance:**
    - Contains more than one base class and inherits the properties of all base classes
    - A class derived from this inheritance works with the derived properties along with its own properties

- **Hierarchical Inheritance:**
  - Contains one base class and the properties of a single base class can be used multiple times in the sub-multiples
  - The derived class contains its own properties and also uses the derived properties of all the base classes

- **Multilevel Inheritance:**
  - Contains one base class and the base class of the derived class can be a class derived from another base class
  - The properties of a base class are inherited to another base class and the derived class can become a base class for another derived class
- **Hybrid Inheritance:**
  - Uses the combination of two or more inheritances
  - This inheritance is normally a combination of multiple and multilevel inheritance

Creating a Class

- A class is a collection of variables and functions that operate on data
- A class can be inherited from a base class to create a new derived class
- The new derived class uses all the properties of the base class including its own properties
- The new derived class uses all the properties of the base class including its own properties

- Declaring a class

Syntax

```
class class_name
{
 function_name($var1, $var2)
 {
 return $var1 + $var2;
 }
}
```

Where,

class - defines the keyword used to declare a class

class_name - specifies the class name

function - specifies the keyword to define a function

var1, var2 - specifies the variable name

function_name() - specifies the function name

return - returns the value after performing the specified mathematical function

Creating a Class

- In the class syntax:
  - Definition for a class is included within curly braces
  - Variables defined in the class are local to the class
  - Variable inside a class is declared with the keyword `var`
  - Class can also use global variables
  - Functions inside a class may use its own local variables or may use the class variables

- Following is the code for a class named `empdetail`

Snippet

```
<?php
class empdetail  {
 var $empid;
 var $empname;
 var $empcity;
 var $empdept;
 var $empdesign;
 function enteremp($id, $name, $city)
 {
 $this->empid=$id;
$this->empname=$name;
$this->empcity=$city;
```

```
}
 function enterdet($dept, $design)
 {
 $this->empdept=dept;
 $this->empdesign=design;
 }
}
?>
```

- This file need to be saved with an extension `.inc`.
- In the code, the `enteremp()` and `enterdet()` functions are user defined
- These functions are defined in the `empdetail` class, and are used for entering data of an employee

Creating a Class

- The `enteremp()` function accepts the employee id, employee name, and city
- The `enterdet()` function accepts the employee department and the designation
- The `filename.inc` file is included in the `filename.php` file with the help of the include keyword
- The `include` keyword enables to incorporate any type of file with the main file

- The syntax to include a file in the program is as follows:

Syntax

```
include "filename.ext";
```

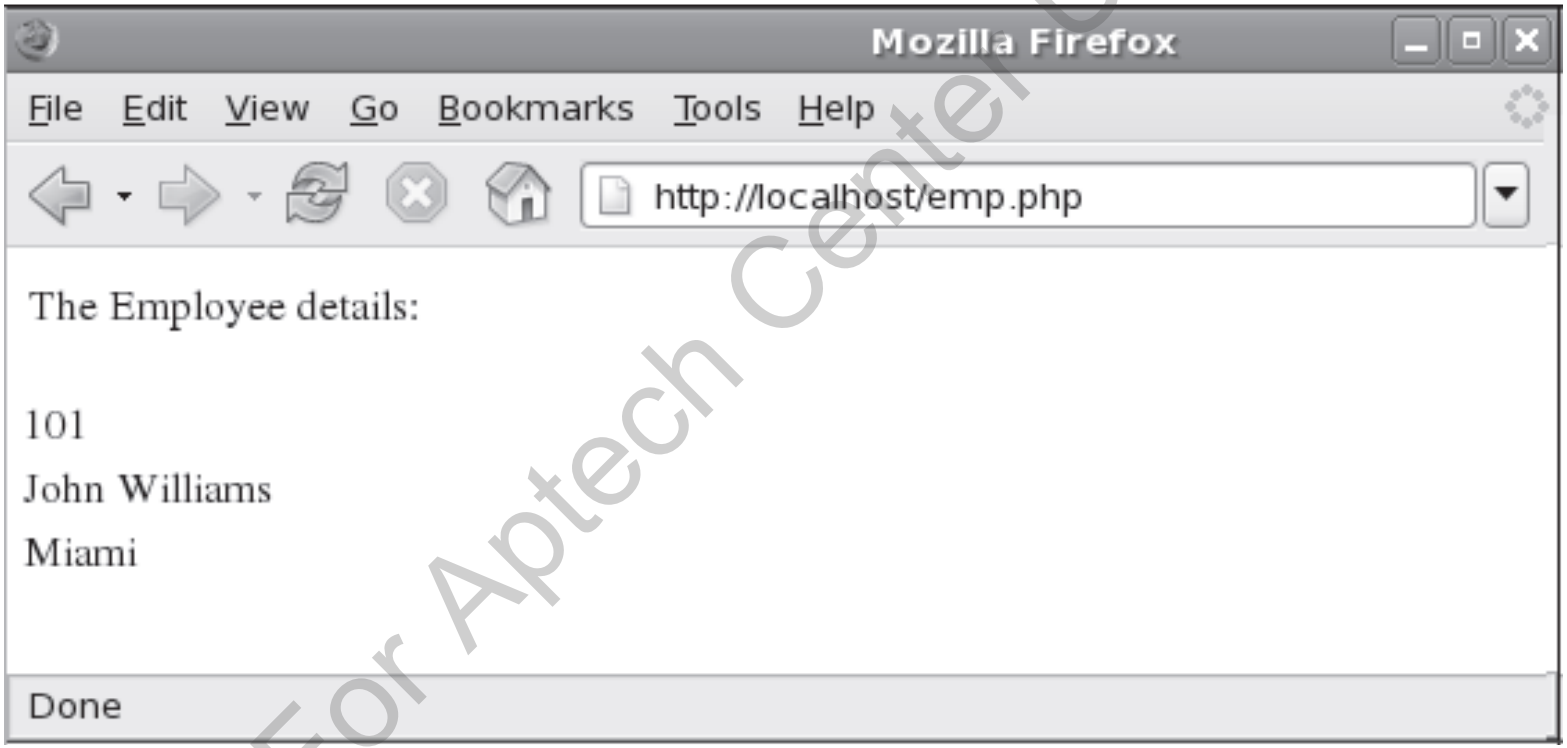
- Assuming that the code `empdetail.inc` is saved, it will be used in the next code

Snippet

```
<?php
include "empdetail.inc";
echo "The Employee details: <BR><BR>";
$empdet = new empdetail();
$empdet->enteremp(101, "John Williams", "Miami");
echo $empdet->empid,"<BR>";
echo $empdet->empname,"<BR>";
echo $empdet->empcity;
?>
```

- Save this file as **`emp.php`** and open it in Mozilla Firefox Web Browser

The following output is displayed:



Creating a Class

- A new class is inherited from an existing class
- The new class uses the properties of the parent class along with its own properties
- The syntax for inheriting a new class is as follows:

Syntax

```
class new_class extends class_name
{
 var class_variable;
 function function_name()
 {
 . . .
 }
}
```

Where,

new_class - specifies the name of the derived class

extends - defines the keyword used to derive a new class from the base class

class_name - specifies the name of the class from which the new class is to be derived

- **salary.inc -** Deriving the `net_salary` class from the `salary` class in PHP

Snippet

```
<?php
class salary
{
 public $hra;
 public $ta;
 public $tax;
public function hra_calc($basic)
{
 $hra = $basic * 0.25;
 return $hra;
}
```

Creating a Class

```
public function travelallow_calc($basic)
{
 $ta = $basic * 0.08;
 return $ta;
}
public function tax_calc($basic)
{
 $tax = $basic * 0.05;
 return $tax;
}
}
```

Creating a Class

```
class net_salary extends salary
{
function net($basic)
{
 $hra = $this->hra_calc($basic);
 $ta = $this->travelallow_calc($basic);
 $tax = $this->tax_calc($basic);
 return $basic + ($hra + $ta)-$tax;
}
}
?>
```

Creating a Class

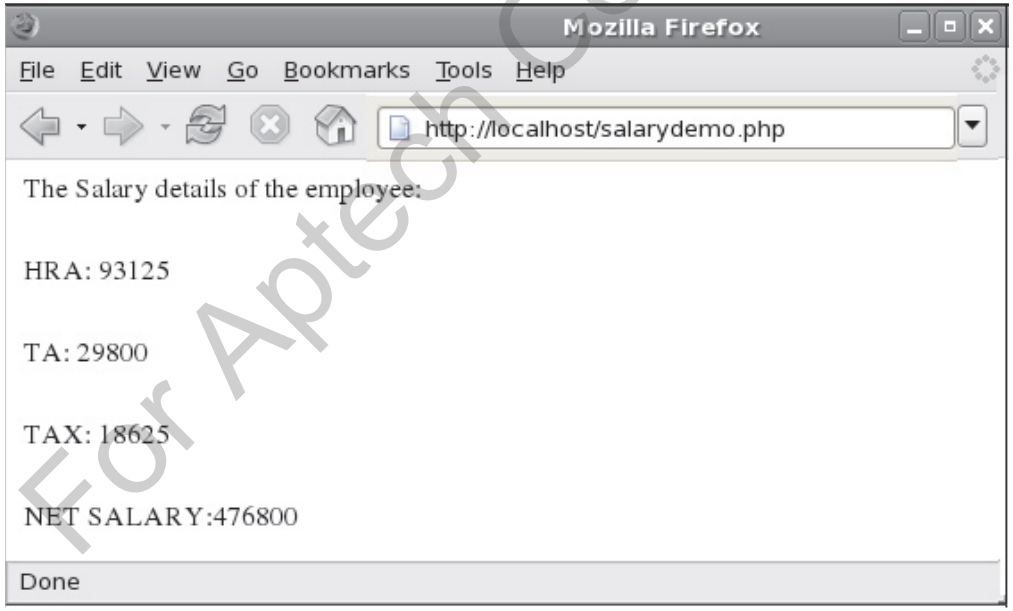
- In the code, the `net_salary` class is derived from the base class, salary
- The `net_salary` class uses the functions defined in the salary class along with its own functions
- The `hra_calc()`, `travelallow_calc()`, and `tax_calc()` functions are the user-defined functions
- The `hra_calc()` function calculates HRA from the basic salary of the employee
- The `travelallow_calc()` function calculates traveling allowance from the basic salary of the employee
- The `tax_calc()` function calculates the tax from the basic salary of the employee

- This class is used in a file named **salarydemo.php**

Snippet

```
<?php
include "salary.inc";
echo "The Salary details of the employee: <BR><BR>";
$sal = new net_salary();
echo $sal->net(372500);
?>
```

The following output is displayed:



Creating a Constructor

- Constructor
  - Is a special function that is a member of the class
  - Is called in the main program by using the new operator
  - Is invoked whenever an object of the class is created
  - Is declared, it provides a value to all the objects that are created in the class and this process is known as initialization
  - Has the same name as that of its class name

Creating a Constructor

Syntax

```
class class_name
{
 var class_variable;
 Function constructor_name()
}
```

Where,

constructor_name - specifies the name of the constructor

- **emp_constructor.inc -** Creating a constructor for the class named `empdetail`

Snippet

```
class empdetail
{
 var $empid;
 var $empname;
 var $empcity;
 var $empdept;
 function empdetail($id, $name, $city, $dept)
  {
 $this->empid=$id;
```

Creating a Constructor

```
   $this->empname=$name;
 $this->empcity=$city;
 $this->empdept=$dept;
}
}
?>
```

Creating a Constructor

- Once the constructor is defined, call the constructor of the class using the `new` operator
- Before calling the constructor of the class, include the file that contains the class
- **`emp_constructor.php`** - Calling the constructor of the **`empdetail`** class

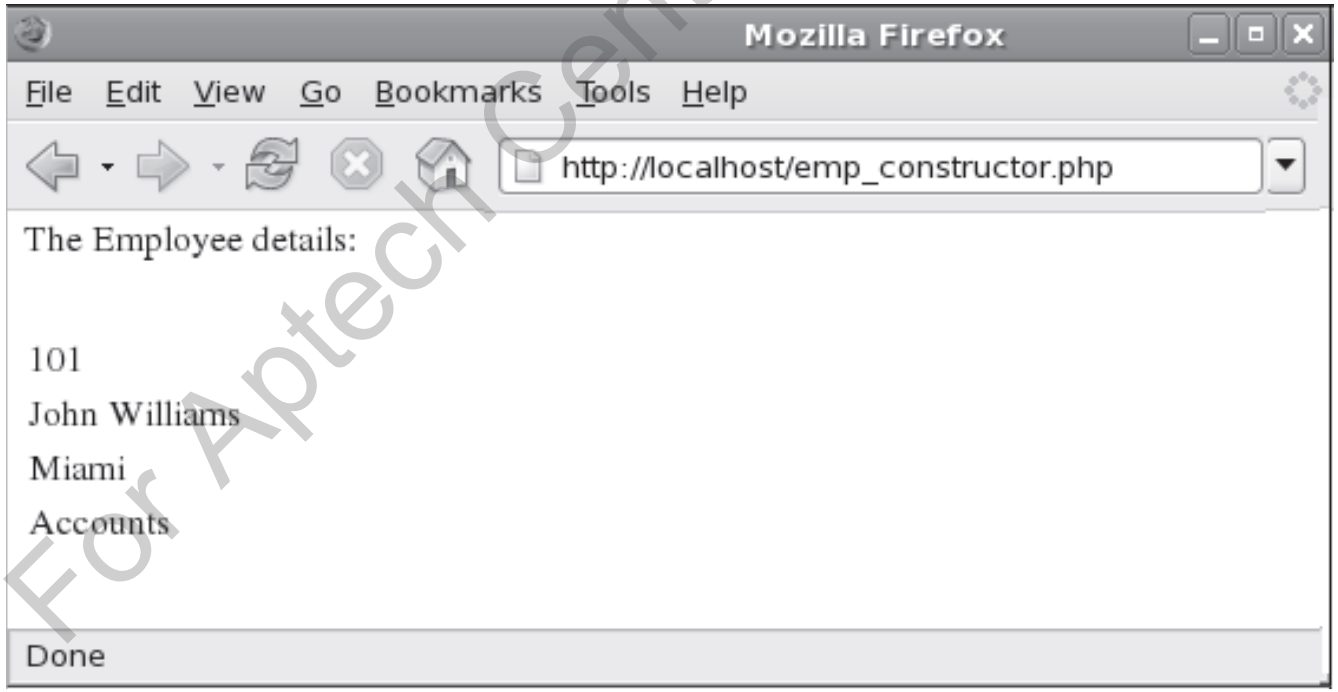
Snippet

```
<?php
include('emp_constructor.inc');
echo "The Employee details: <BR><BR>";
$empdet = new empdetail(101, "John Williams", "Miami", "Accounts");
echo $empdet->empid,"<BR>";
echo $empdet->empname,"<BR>";
echo $empdet->empcity,"<BR>";
echo $empdet->empdept;
?>
```

Creating a Constructor

- The `new` operator creates a new object of the **`empdetail`** class
- It calls the functions defined in the **`empdetail`** class

The following output is displayed:



- **stringtest.php** - Calling a constructor from the derived class

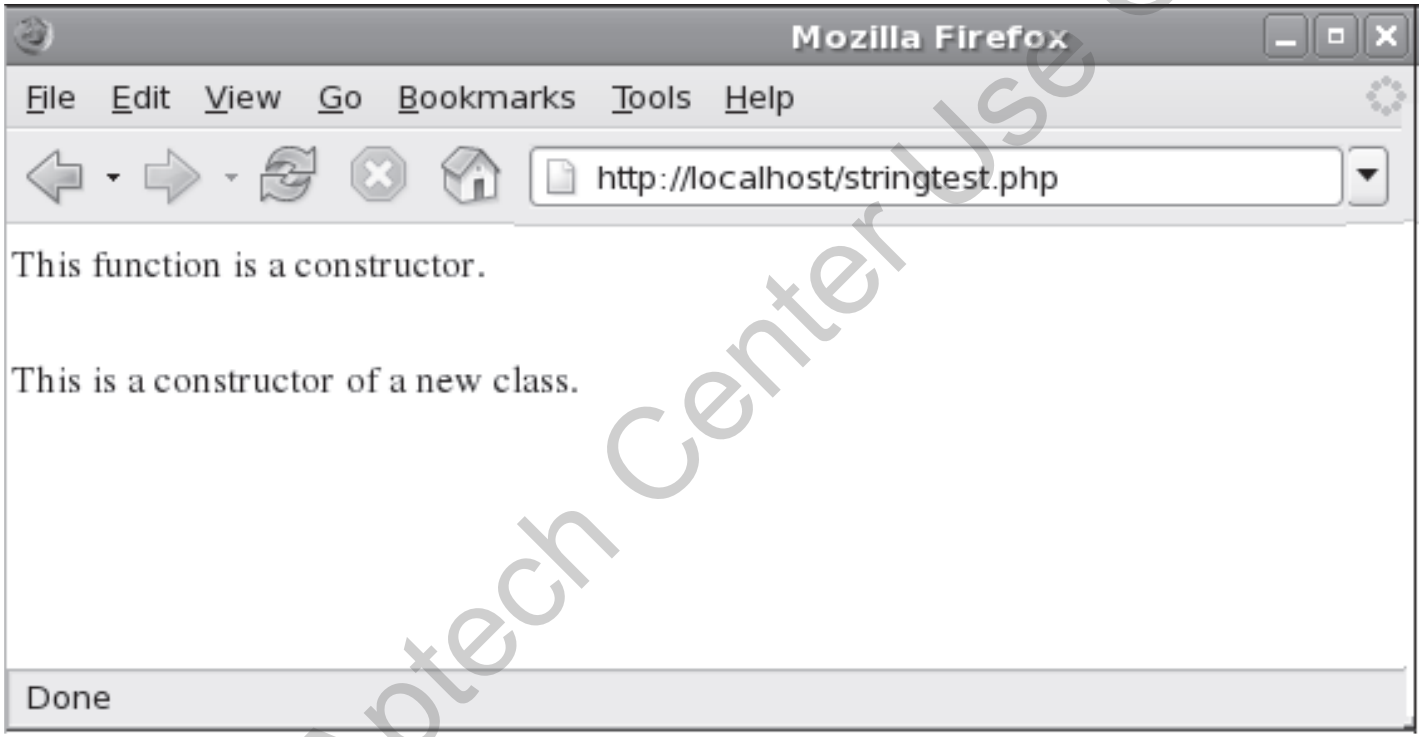
Snippet

```
<?php
class string
{
function string()
{
 echo "This function is a constructor.";
}
function stringdisp()
{
 echo "This is function.";
}
}
```

Creating a Constructor

```
class display extends string

{

function display()

{

 echo "This is a constructor of a new class.";


}

}

$disp1 = new string;

echo "<BR><BR>";

$disp = new display;

?>
```

The following output is displayed:



Creating and Using Objects

- An object is a self-contained entity that includes both data and procedures to manipulate the data
- It maintains the codes of the program
- An object is an instance of a class
- It provides a reference to the class
- An object can be manipulated as required
- The new operator can be used to initialize the object

- The syntax for creating an object is as follows:

Syntax

```
$object_name = new class_name;
```

Where,

object_name - specifies the name of the object

new - initializes the object

class_name - specifies the class name

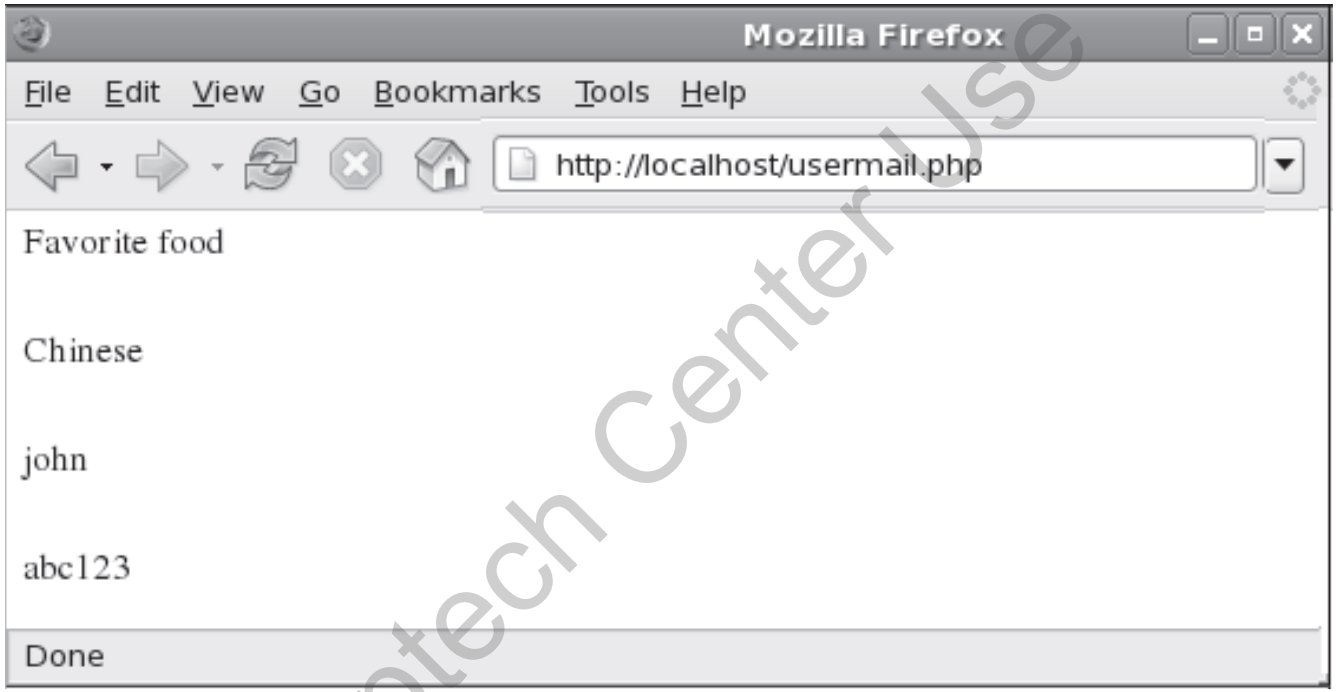
- **usermail.php -** Create an object for the class, **`usermail`**

Snippet

```
<?php
class usermail
{
 var $username = "john";
 var $password = "abc123";
 function dispuser ()
 {
      echo $this->username,"<BR><BR>";
      echo $this->password, "<BR><BR>";
      }
}
class userdetails extends usermail
{
 var $secretquery = "Favorite food";
      var $answer = "Chinese";
```

```
function dispdetail()
{
 echo "<BR><BR>";
 echo $this->secretquery, "<BR><BR>";
 echo $this->answer, "<BR><BR>";
}
}
$mail = new userdetails;
$mail1 = new usermail;
$disp1 = $mail->dispdetail();
$disp2 = $mail1->dispuser();
?>
```

The following output is displayed:



Summary

- Object-oriented programming defines the data type of the data structure and the type of functions that can be performed on the data structure
- The main concepts of an OOP are objects, class, abstraction, encapsulation, polymorphism, and inheritance
- In hierarchical inheritance, the properties of a single base class can be used multiple times in multiple subclasses
- A class is a collection of variables and functions that operate on that data.
- A class can be inherited from a base class to create a new derived class.

Summary

- A constructor is a special function that has the same name as that of its class name, and is called in the main program by using the new operator
- An object is a self-contained entity that consists of both data and procedures to manipulate the data. It is an instance of a class and can be manipulated as needed