TITLE: OOP with PHP -2

PRACTICAL -8

OOP with PHP -2

Q-1. Create a HTML form which takes a bank account number, current balance, Create a class Course having property course name, no_of_year with a constructor, and display () method for the display course information. Create a class named Student which inherits a Course having stud_id, stud_name, array of marks (3 subject). Class containsthe constructor and is calling the parent constructor. Class contains method caltotal () which returns the total of 3 subject marks. It contains display method which shows all the information about the students: Name, id, marks, total, course_name, no_of_year

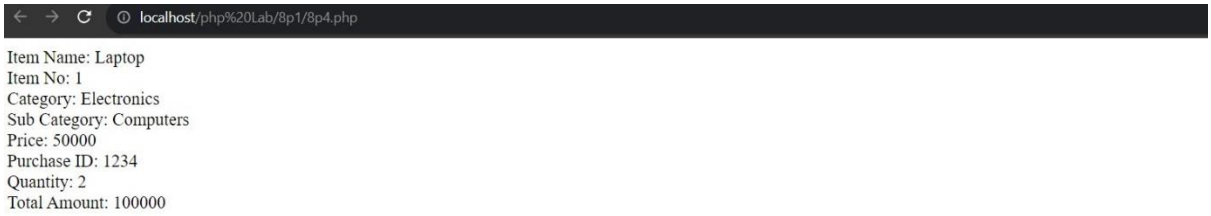
Code:

P1.php

```
<?php

class Course {

   public $coursename;

   public $no_of_year;

   public function __construct($coursename,$no_of_year)

   {

      $this->coursename=$coursename;

      $this->no_of_year=$no_of_year;

   }

   public function display()

   {

       echo "course name is:".$this->coursename."<br>"."no of course duration:".$this-
no_of_year."<br>";

   }

}

class Student extends Course{

   public $stud_id;

   public $stud_name;

   public $marks = array();


   public function __construct($id,$name,$mark,$c_name,$no_of_year){

      parent::__construct($c_name,$no_of_year);
```

```
        $this->stud_id = $id;
        $this->stud_name = $name;
        for ($i=0; $i < 3; $i++) {
            $this->marks[$i] = $mark[$i];
        }
    }
    public function caltotal(){
        $sum=0;
        for ($i=0; $i <3 ; $i++) {
            $sum+= $this->marks[$i];
        }
        echo "total marks are:".$sum;
    }
    public function dispaly(){
        echo "student infomation"."<br>";
        parent::display();
        echo "student ID:".$this->stud_id."<br>";
        echo "student name".$this->stud_name."<br>";
    }
}
$marks=array(70,80,90);
$obj=new Student(18,"jay",$marks,"mca",2);
$obj->dispaly();
$obj->caltotal();
?>
```

OUTPUT:



Q-2) Create a class named shape having a method area () which calculates the area of the shape and returns the area.
a. Create a class circle which inherits Shape and overrides the area () method and returns the area of the circle.
b. Create a class square which inherits the Shape class and have an area () method to return the area of the square.
c. Create a class Rectangle which inherits the Shape class and have an area () method which returns the area of the Rectangle.
d. Create the object of class shape, circle, square and rectangle class and display area of each. [override the area () method]

Code:

P2.php

```
<?php
class Shape {
   public function area () {
   }
   }
   class Circle extends Shape {
      private $radius;
      public static $pi=3.14;
      public function __construct($radius) {
      $this->radius = $radius;
      }
      public function area() {
      return static::$pi *$this->radius*$this->radius;
      }
   }
```

```
class Square extends Shape {
    private $length;
        public function __construct($length){
            $this->length=$length;
        }
        public function area(){
             if (is_numeric($this->length))
            {
                return $this->length * $this->length;
            }
            }
    }
class Rectangle extends Shape {
    private $length,$width;
    public function __construct($length,$width){
        $this->length=$length;
        $this->width=$width;
    }
    public function area(){
        return $this->length * $this->width;
    }
}

$circle = new Circle(10);
$ans=$circle->area();
echo "area of cicrcle is:".$ans."<br>";
$square=new square(20);
$ans=$square->area();
echo "area of suqare is:".$ans."<br>";
$r1=new Rectangle(5,10);
$ans=$r1->area();
```

```
echo "area of rectangle is:".$ans."<br>";
?>
```

OUTPUT

Q-3). Create an abstract class Employee having a constructor for setting name, year of joining, date of birth and department of employee. Create an appropriate method to display the details of the Employee in well-designed HTML format. Create an abstract method calculate salary () in Employee class.
a. Create a class Manager inherits the Employee class. It should include properties like basic salary, DA, tax amount, HRA etc. property. Create the constructor for setting the values. Implement the calculate salary () (basic+DA+HRA – tax amount).
b. Create a class worker which inherits the Employee class. Create a constructor which sets the property wages per hour, worked hour.
Implement calsal() method (wages per hour*worked hour) .
c. Create appropriate methods in each class to display the well formatted details in HTML.
d. Write a php program which create multiple objects of Manager and Worker class, and display the name, designation and salary of each

Code:
p3.php

```
<?php
abstract class Employee {
   protected $name;
   protected $yearOfJoining;
   protected $dob;
   protected $department;
```

```
    public function __construct($name, $yearOfJoining, $dob, $department) {
        $this->name = $name;
        $this->yearOfJoining = $yearOfJoining;
        $this->dob = $dob;
        $this->department = $department;
    }
    public function display() {
        echo "<h2>Employee Details</h2>";
        echo "Name: ".$this->name."<br>";
        echo "Year of Joining: ".$this->yearOfJoining."<br>";
        echo "Date of Birth: ".$this->dob."<br>";
        echo "Department: ".$this->department."<br>";
    }
    abstract public function calculate_salary();
}
class Manager extends Employee {
    private $basicSalary;
    private $da;
    private $taxAmount;
    private $hra;
    public function __construct($name, $yearOfJoining, $dob,
                        $department, $basicSalary,
                        $da, $taxAmount, $hra)
        {
        parent::__construct($name, $yearOfJoining,
                    $dob, $department);
        $this->basicSalary = $basicSalary;
        $this->da = $da;
        $this->taxAmount = $taxAmount;
        $this->hra = $hra;
    }
```

```
    public function calculate_salary() {
        return ($this->basicSalary +
                $this->da +
                $this->hra -
                $this->taxAmount);
    }
    public function display () {
        parent::display();
        echo "Designation: Manager<br>";
        echo "Salary: ".$this->calculate_salary()."<br>";
    }
}
class Worker extends Employee {
    private $wagesPerHour;
    private $workedHours;
    public function __construct($name,
                        $yearOfJoining,
                        $dob,
                        $department,
                        $wagesPerHour,
                        $workedHours) {
        parent:: __construct($name,
                    $yearOfJoining,
                    $dob,
                    $department);
        $this->wagesPerHour =
            (float)$wagesPerHour;
        $this->workedHours =
            (int)$workedHours;
    }
    public function calculate_salary() {
```

```
        return ($this->wagesPerHour *
                (float)$this->workedHours);
    }
    public function display () {
        parent:display();
        echo "Designation: Worker<br>";
        echo "Salary: ".$this->calculate_salary()."<br>";
    }
}
$manager = new Manager("Jay", 2023, "2001-01-01",
                    "Sales", 50000.00, 10000.00, 5000.00,
                    20000.00);
$worker = new Worker("ram", 2023, "2001-01-01",
                   "Production", 500.00, 160);
$manager->display();
$worker->display();
?>
```

OUTPUT

Q-4) Create a class Item with
Property: Item name, Item no

Method: display -> display item name and ion
a. Create a class Category which inherits Item
Property: category name [e.g., cloth/electronics/kids' toys. etc], subcategory [e.g. mobile, laptop, jeans shirt] , price
Method: display Item () to display category, subcategory and price and get price() to return the price of item
b. Create a class purchase which inherits Item
Property: purchase id, total amount, quantity
Method:
● calculate_order_amout (): calculate and return the total amount (qty*price)
● display purchase () : display the all details of Item and category.
c. Write a Php script which creates the necessary constructor and creates an object of purchase class. Calculate the total order amount. It should also display the details purchase

P4.php

```
<?php

class Item {

   protected $itemName;

   protected $itemNo;

   public function __construct($itemName, $itemNo) {

      $this->itemName = $itemName;

      $this->itemNo = $itemNo;

   }

   public function display() {

      echo "Item Name: ".$this->itemName."<br>";

      echo "Item No: ".$this->itemNo."<br>";

   }

}

class Category extends Item {

   protected $categoryName;

   protected $subCategory;

   protected $price;

   public function __construct($itemName, $itemNo, $categoryName, $subCategory, $price) {

      parent::__construct($itemName, $itemNo);

      $this->categoryName = $categoryName;

      $this->subCategory = $subCategory;
```

```
        $this->price = $price;
    }
    public function displayItem() {
        parent::display();
        echo "Category: ".$this->categoryName."<br>";
        echo "Sub Category: ".$this->subCategory."<br>";
        echo "Price: ".$this->price."<br>";
    }
    public function getPrice() {
        return $this->price;
    }
}
class Purchase extends Category {
    private $purchaseId;
    private $totalAmount;
    private $quantity;
    public function __construct($itemName, $itemNo, $categoryName, $subCategory,
                            $price, $purchaseId, $quantity) {
        parent::__construct($itemName, $itemNo, $categoryName,
                        $subCategory, $price);
        $this->purchaseId = $purchaseId;
        $this->quantity = $quantity;
        $this->totalAmount = self::calculate_order_amount();
    }
    private function calculate_order_amount() {
        return ($this->quantity * parent::getPrice());
    }
    public function display_purchase() {
        parent::displayItem();
        echo "Purchase ID: ".$this->purchaseId."<br>";
        echo "Quantity: ".$this->quantity."<br>";
```

```
        echo "Total Amount: ".$this->totalAmount."<br>";
    }
}
$purchase = new Purchase ("Laptop", 1, "Electronics", "Computers", 50000, 1234, 2);
$purchase->display purchase ();
?>
```