# **Name: Abdurrahman Qureshi**

# **Roll No: 210451**

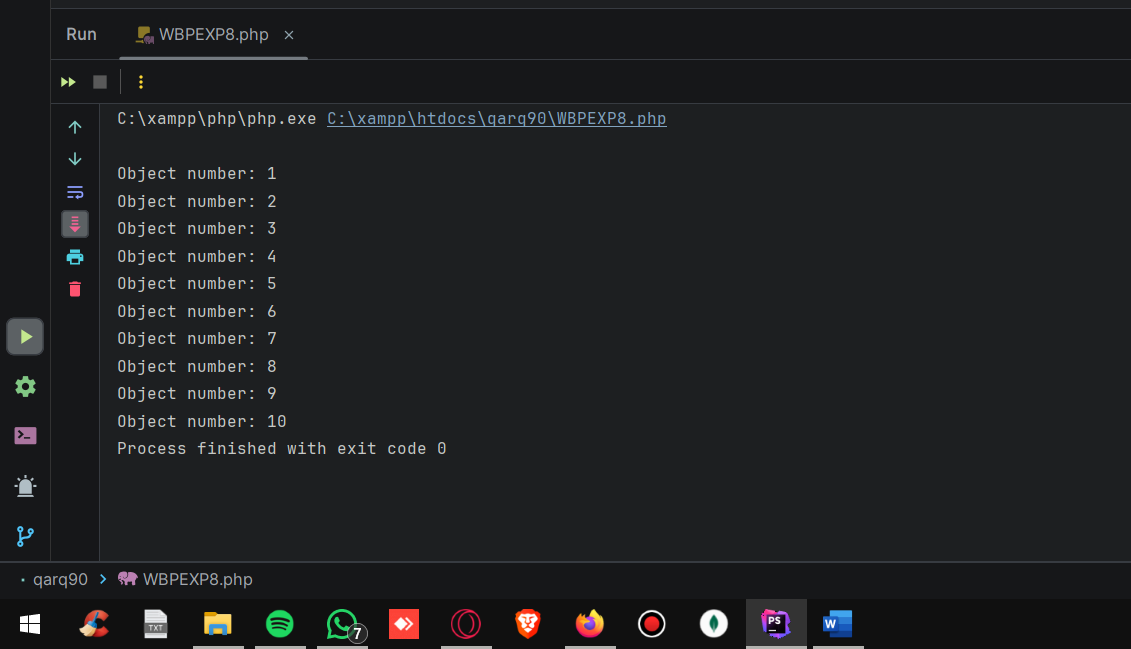
Practical No: 8

**1) Write a Program for Array of Objects**

**CODE:**

class arr\_obj  
{  
 function \_\_construct($num)  
 {  
 echo "\nObject number: " . $num;  
 }  
}  
  
for ($i = 1; $i <= 10; $i++) {  
 $obj[$i] = new arr\_obj($i);  
}

**OUTPUT:**

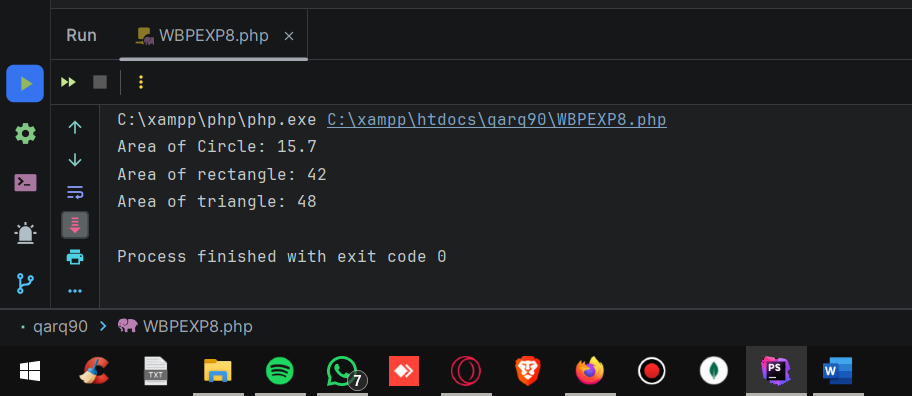
****

**2) Write a Program to demonstrate method overloading**

**CODE:**

class Shape  
{  
 function \_\_call($name\_of\_function, $arguments)  
 {  
 if ($name\_of\_function == "area") {  
 switch ($arguments[0]) {  
 case "circle":  
 return 3.14 \* $arguments[1];  
 case "rect":  
 return $arguments[1] \* $arguments[2];  
 case "tri":  
 return 0.5 \* $arguments[1] \* $arguments[2];  
 }  
 }  
 return "Something went wrong...";  
 }  
}  
  
$s = new Shape;  
echo "Area of Circle: " . $s->area("circle", 5) . "\n";  
echo "Area of rectangle: " . $s->area("rect", 7, 6) . "\n";  
echo "Area of triangle: " . $s->area("tri", 12, 8) . "\n";

**OUTPUT:**

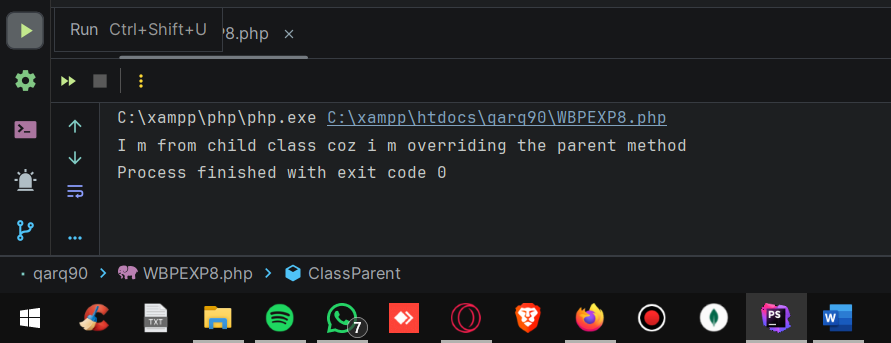
****

**3) Write a Program to demonstrate method overriding**

**CODE:**

class ClassParent  
{  
 public function showMsg()  
 {  
 echo "I m from parent class";  
 }  
}  
  
class ClassChild extends ClassParent  
{  
 function showMsg()  
 {  
 echo "I m from child class coz i m overriding the parent method";  
 }  
}  
$overriding = new ClassChild();  
$overriding->showMsg();

**OUTPUT:**

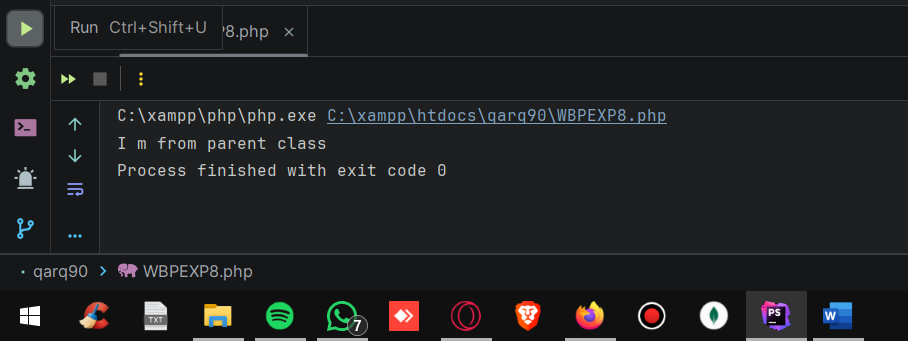


**4) Write a Program for Single level Inheritance**

**CODE:**

class ClassParent{  
 public function showMsg(){  
 echo "I m from parent class";  
 }  
}  
  
class ClassChild extends ClassParent{}  
$inherit = new ClassChild();  
$inherit->showMsg();

**OUTPUT:**

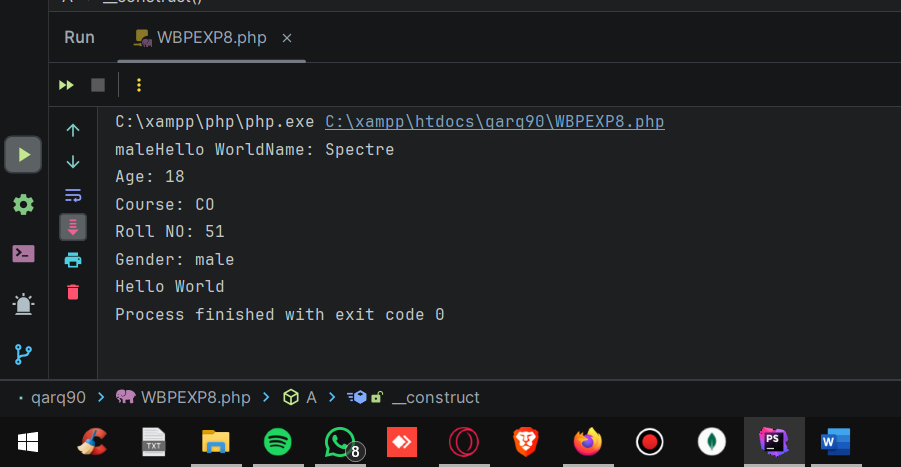
****

**5) Write a Program to demonstrate multi-level inheritance**

**CODE:**

abstract class A{  
 public string $name;  
 public int $age;  
 public string $course;  
 public static string *$gender* = "male";  
 public function \_\_construct($name, $age, $course) {  
 $this->name = $name;  
 $this->course = $course;  
 $this->age = $age;  
 echo self::*$gender*;  
 self::*helloWorld*();  
 }  
 public static function helloWorld(): void  
 {  
 echo "Hello World";  
 }  
 public abstract function showData(): void;  
}  
abstract class B extends A{  
 public int $roll\_no = 51;  
}  
  
class C extends B{  
 public function showData(): void {  
 echo "Name: " . $this->name;  
 echo "\nAge: " . $this->age;  
 echo "\nCourse: " . $this->course;  
 echo "\nRoll NO: " . $this->roll\_no;  
 echo "\nGender: " . parent::$gender;  
 echo "\n";  
 parent::*helloWorld*();  
 }  
}  
$c = new C("Spectre", 18, "CO");  
$c->showData();

**OUTPUT:**

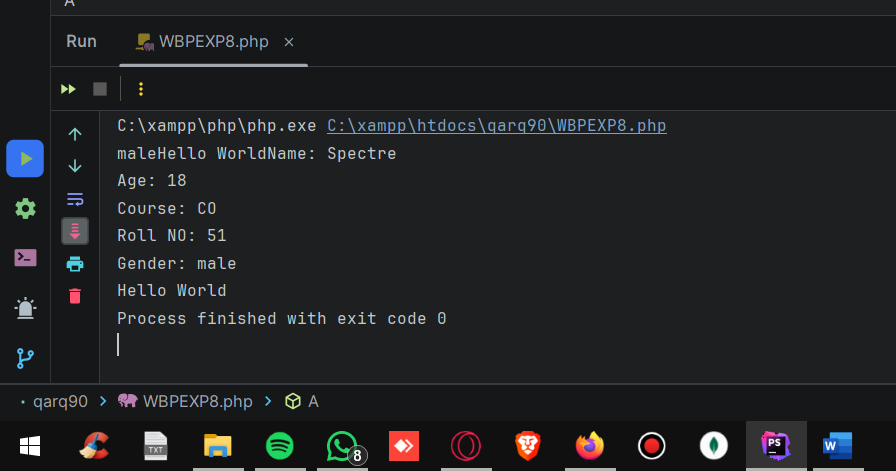


**6) Write a Program for Multiple Inheritance. (make use of Trait)**

**CODE:**

abstract class A{  
 public string $name;  
 public int $age;  
 public string $course;   
 public static string *$gender* = "male";  
 public function \_\_construct($name, $age, $course) {  
 $this->name = $name;  
 $this->course = $course;  
 $this->age = $age;  
 echo self::*$gender*;  
 self::*helloWorld*();  
 }  
 public static function helloWorld(): void{  
 echo "Hello World";  
 }  
 public abstract function showData(): void;  
}  
trait B{  
 public int $roll\_no = 51;  
}  
class C extends A{  
 use B;   
 public function showData(): void{  
 echo "Name: " . $this->name;  
 echo "\nAge: " . $this->age;  
 echo "\nCourse: " . $this->course;  
 echo "\nRoll NO: " . $this->roll\_no;  
 echo "\nGender: " . parent::$gender;  
 echo "\n";  
 parent::*helloWorld*();  
 }  
}  
$c = new C("Spectre", 18, "CO");  
$c->showData();

**OUTPUT:**

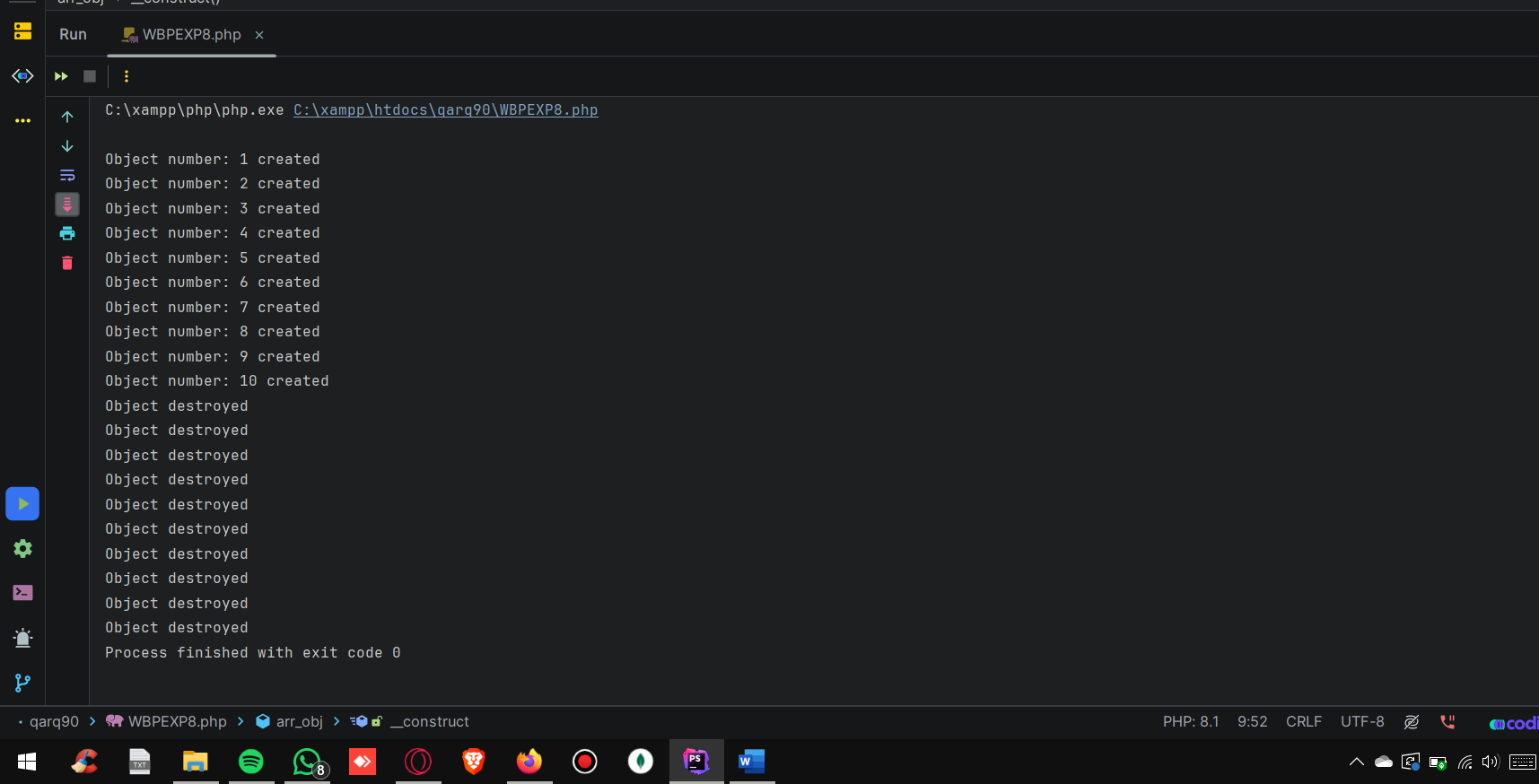


**7) Write a Program for Constructor and Destructor.**

**CODE:**

class arr\_obj{  
 function \_\_construct($num) {  
 echo "\nObject number: " . $num . " created";  
 }  
 function \_\_destruct(){  
 echo "\nObject destroyed";  
 }  
}  
for ($i = 1; $i <= 10; $i++) {  
 $obj[$i] = new arr\_obj($i);  
}

**OUTPUT:**

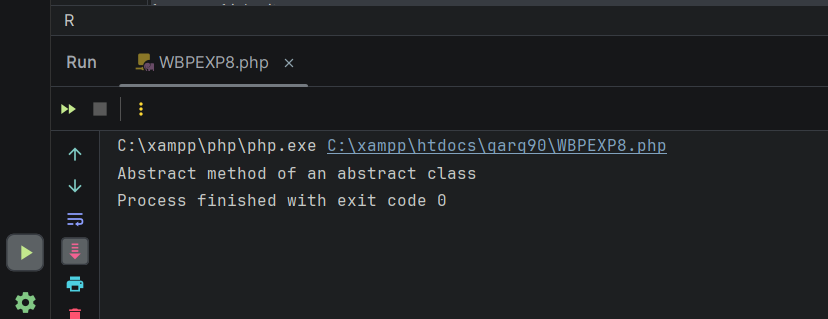


**8) Write a Program for abstract class and method**

**CODE:**

abstract class R {  
 public abstract function show();  
}  
abstract class Q extends R{  
 public function show()  
 {  
 echo "Abstract method of an abstract class";  
 }  
}  
class S extends Q{  
 public function show2(){  
 $this->show();  
 }  
}  
$q = new S();  
$q->show();

**OUTPUT:**

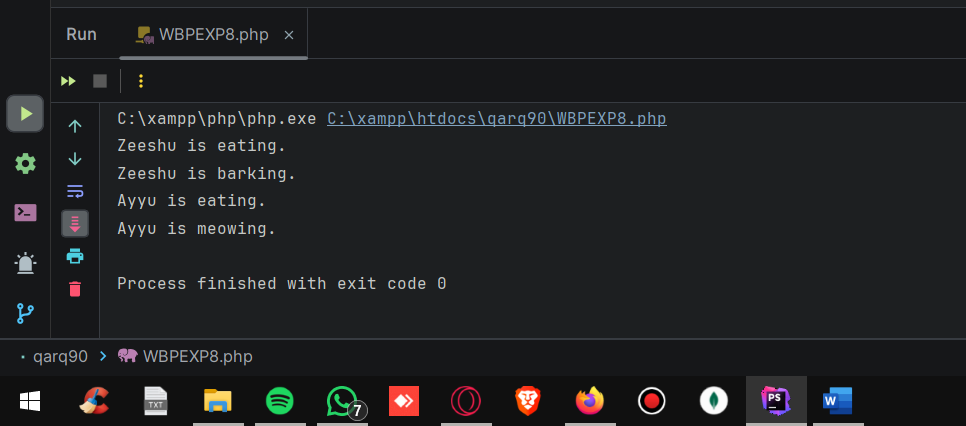


**9) Write a Program for hierarchical inheritance**

**CODE:**

class Animal  
{  
 public $name;  
  
 public function \_\_construct($name)  
 {  
 $this->name = $name;  
 }  
  
 public function eat()  
 {  
 echo $this->name . " is eating. \n";  
 }  
}  
  
class Dog extends Animal  
{  
 public function bark()  
 {  
 echo $this->name . " is barking. \n";  
 }  
}  
  
class Cat extends Animal  
{  
 public function meow()  
 {  
 echo $this->name . " is meowing. \n";  
 }  
}  
  
$dog = new Dog("Zeeshu");  
$cat = new Cat("Ayyu");  
  
$dog->eat();  
$dog->bark();  
  
$cat->eat();  
$cat->meow();

**OUTPUT:**

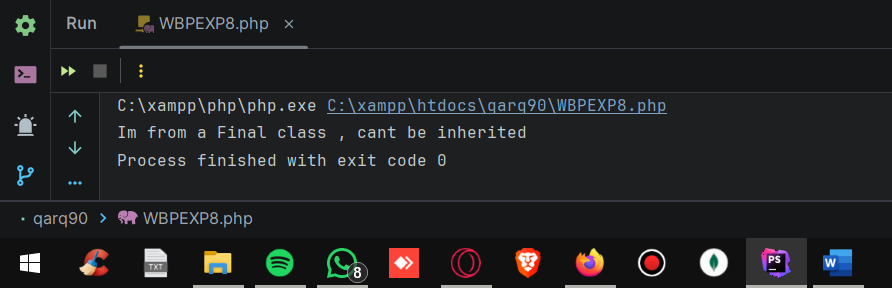
****

**10) Write a Program for demonstration of final keyword – (class)**

**CODE:**

final class x  
{  
 static public function show()  
 {  
 echo "Im from a Final class , cant be inherited";  
 }  
}  
  
x::*show*();

**OUTPUT:**

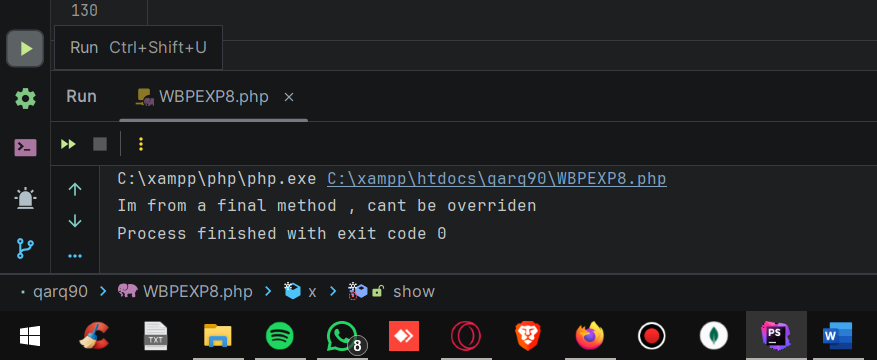
****

**11) Write a Program for demonstration of final keyword – (function)**

**CODE:**

final class x  
{  
 final static public function show()  
 {  
 echo "Im from a final method , cant be overriden";  
 }  
}  
  
x::*show*();

**OUTPUT:**

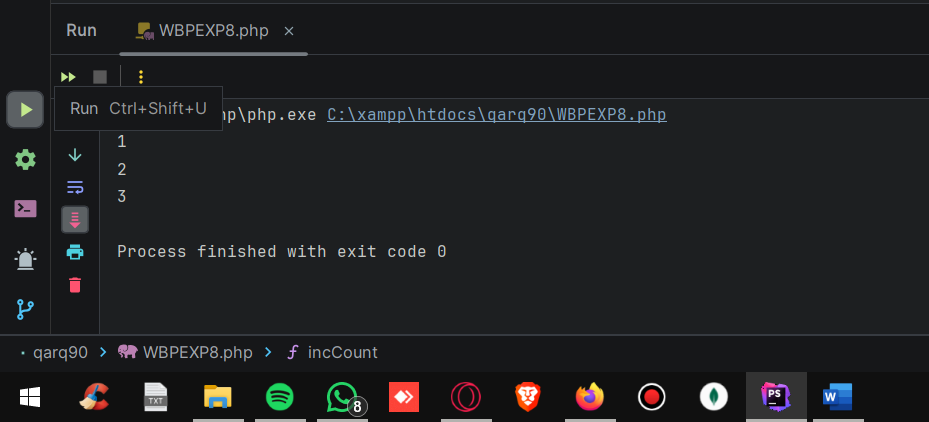


**12) Write a Program for demonstration of static keyword – (variables)**

**CODE:**

function incCount()  
{  
 static $myCount = 0;  
 $myCount++;  
 echo $myCount . "\n";  
}  
  
incCount();  
incCount();  
incCount();

**OUTPUT:**



**13) Write a Program for demonstration of static keyword – (properties and methods)**

**CODE:**

class P  
{  
 static public *$name* = "Simon Riley";  
  
 static public function show()  
 {  
 echo "i m a static method";  
 }  
}  
  
echo P::*$name* . " is a static property \n";  
echo P::*show*()

**OUTPUT:**

