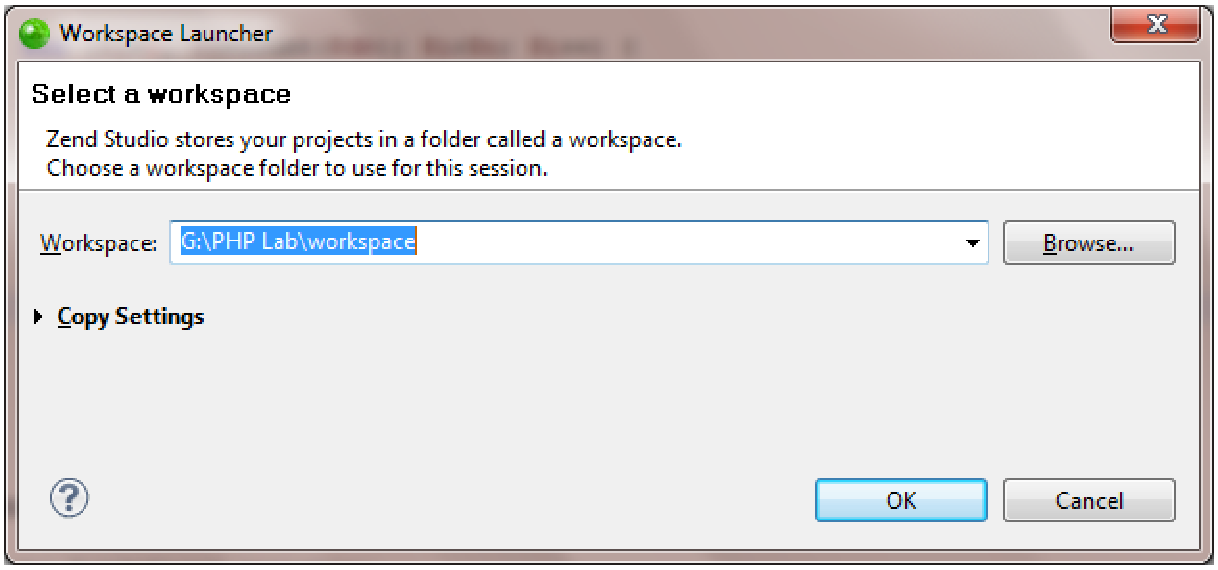
Lab 5. OOP in PHP

Prepared: TrangNTT

5.1. Open workspace and Create Folder

Step 1. Open Zend Studio 7.0

- Select Start → All Programs → Zend Studio – 7.0.0 → Zend Studio – 7.0.0
- Choose OK to confirm the workspace.



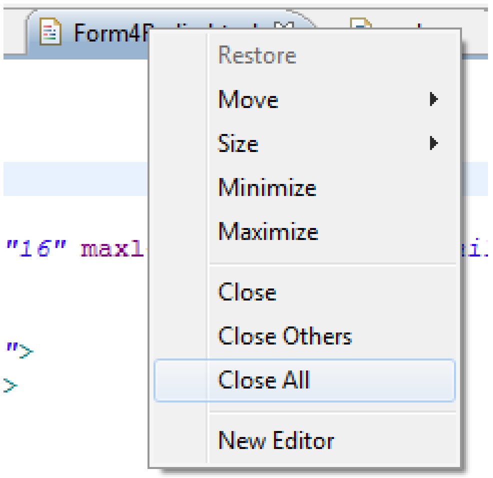
Workspace Launcher

Select a workspace

Zend Studio stores your projects in a folder called a workspace.
Choose a workspace folder to use for this session.

Workspace: G:\PHP Lab\workspace

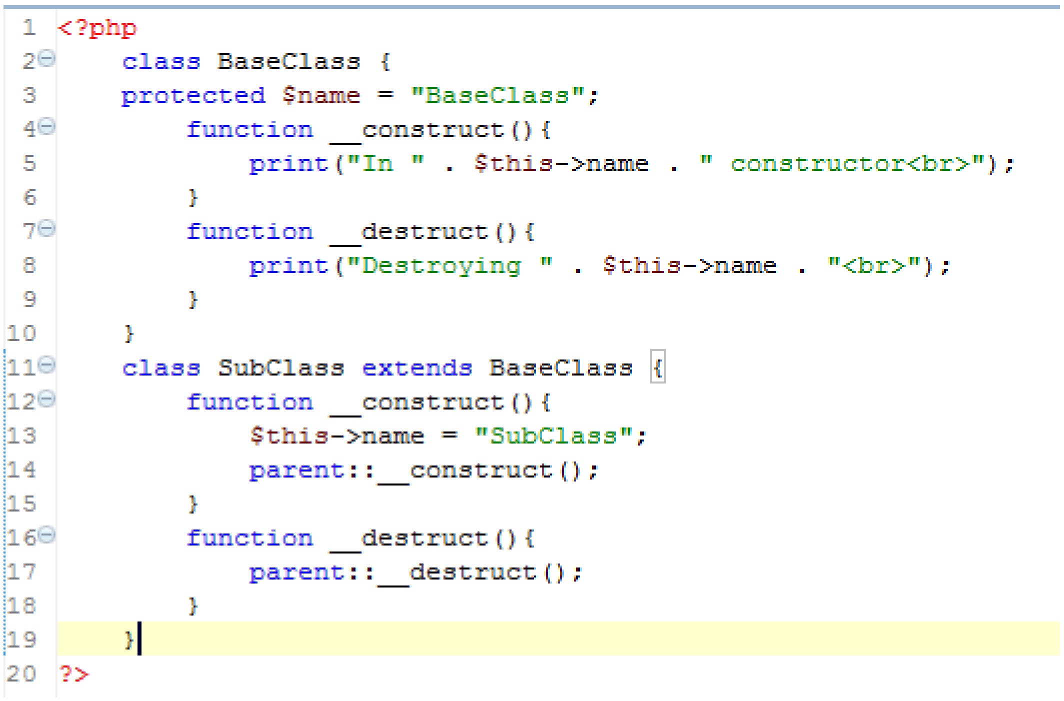
- Close all the opening file in the **PHP Editor** view by right click to a file and choose **Close All**.



Step 2. Right click on LabProject, choose New Folder. Enter Lab5 as folder name.

5.2. Constructor & Destructor

Step 1. Con_des.php



```php
<?php
    class BaseClass {
    protected $name = "BaseClass";
        function __construct(){
            print("In " . $this->name . " constructor<br>");
        }
        function __destruct(){
            print("Destroying " . $this->name . "<br>");
        }
    }
    class SubClass extends BaseClass {
        function __construct(){
            $this->name = "SubClass";
            parent::__construct();
        }
        function __destruct(){
            parent::__destruct();
        }
    }
?>
```

Step 2. Add the following code to the end of con_des.php file:

$obj1 = new SubClass();

Run and give comments.

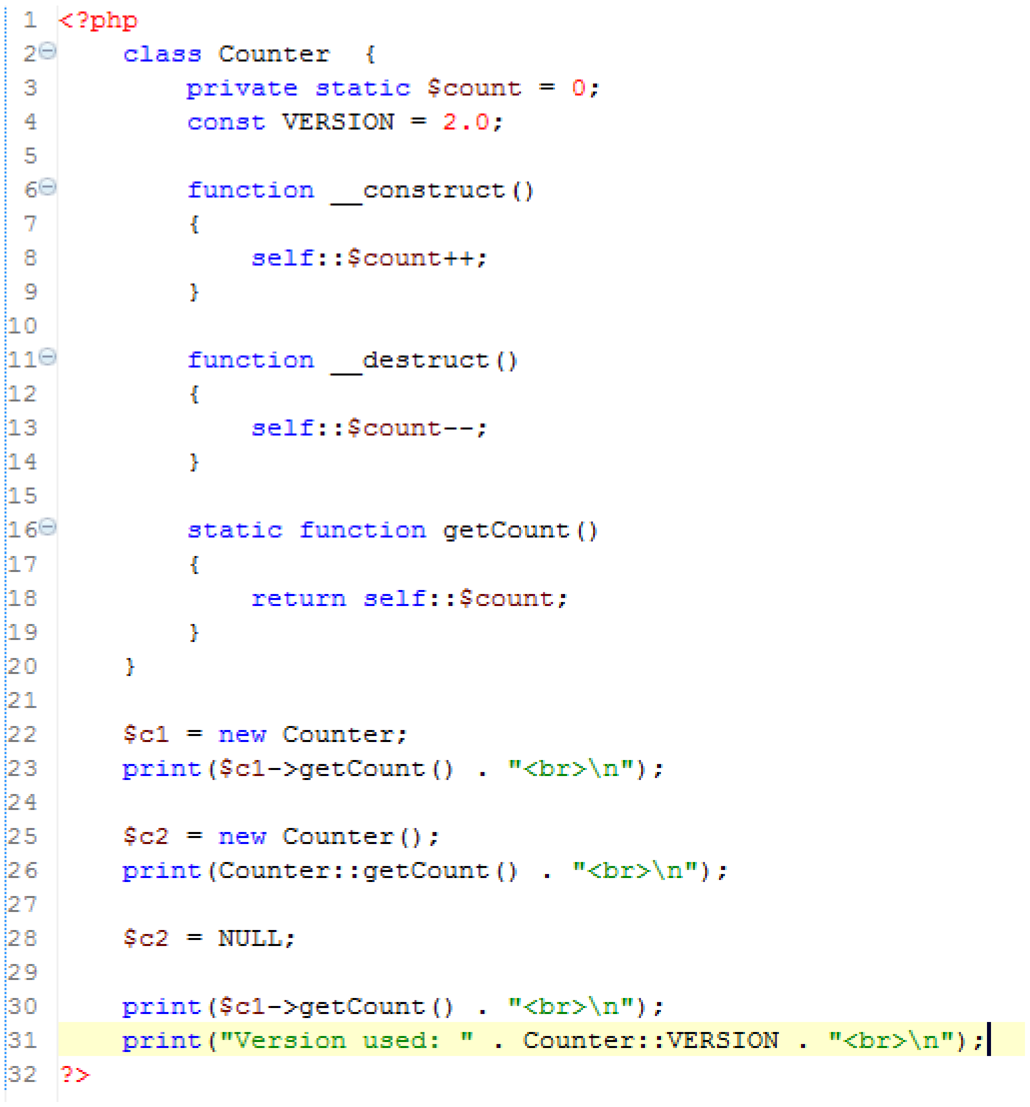
Step 3. Add the following code to the end of con_des.php file:

$obj2 = new BaseClass();

Run and give comments.

5.3. Counter – Static member

Step 1. Counter.php



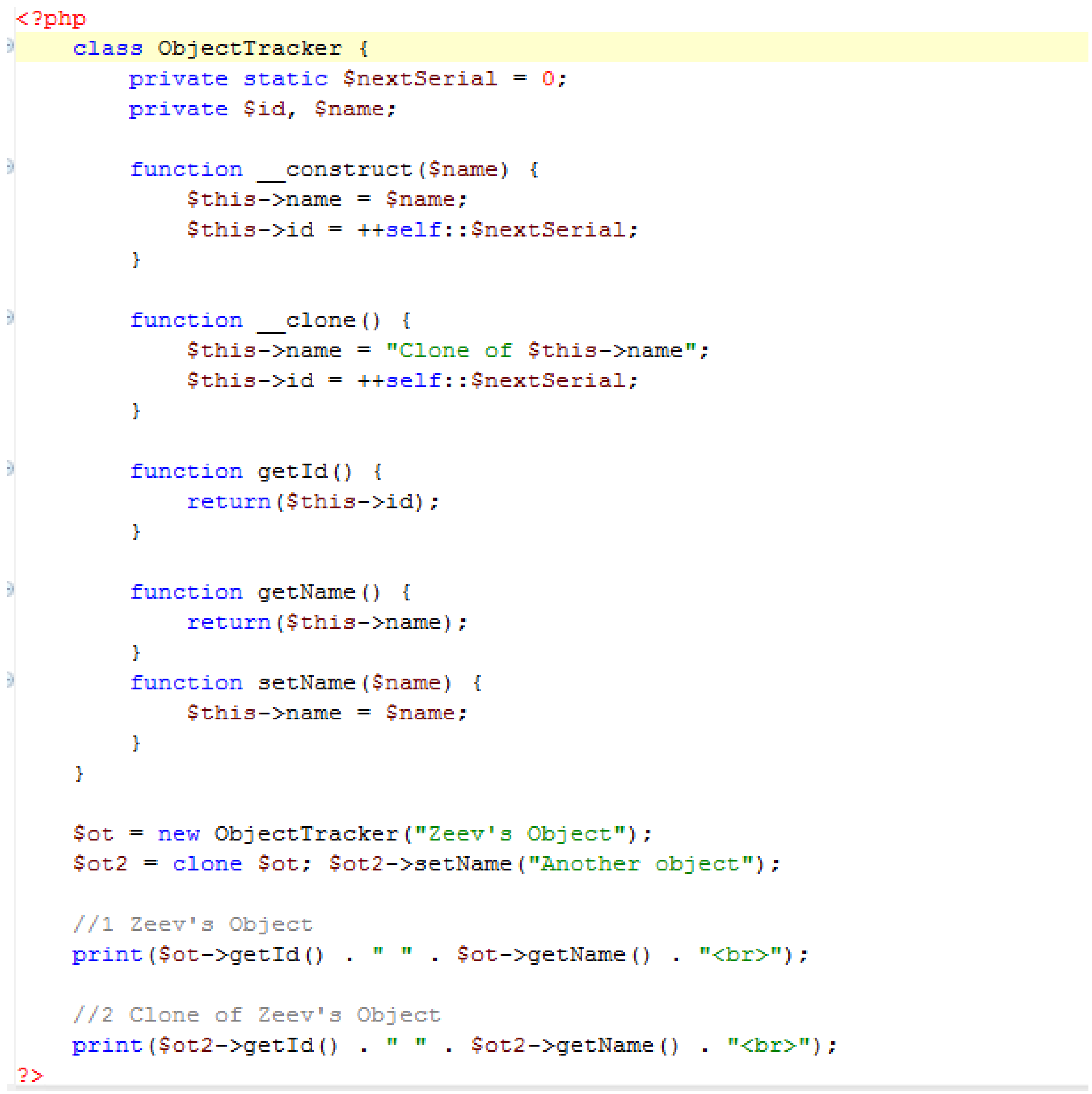
```php
<?php
    class Counter  {
        private static $count = 0;
        const VERSION = 2.0;

        function __construct()
        {
            self::$count++;
        }

        function __destruct()
        {
            self::$count--;
        }

        static function getCount()
        {
            return self::$count;
        }
    }

    $c1 = new Counter;
    print($c1->getCount() . "<br>\n");

    $c2 = new Counter();
    print(Counter::getCount() . "<br>\n");

    $c2 = NULL;

    print($c1->getCount() . "<br>\n");
    print("Version used: " . Counter::VERSION . "<br>\n");
?>
```

Step 2. Run

```
1
2
1
Version used: 2
```

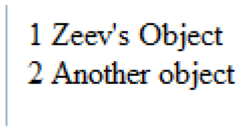
5.4. *Object Cloning*

Step 1. ObjectCloning.php



```php
<?php
    class ObjectTracker {
        private static $nextSerial = 0;
        private $id, $name;

        function __construct($name) {
            $this->name = $name;
            $this->id = ++self::$nextSerial;
        }

        function __clone() {
            $this->name = "Clone of $this->name";
            $this->id = ++self::$nextSerial;
        }

        function getId() {
            return($this->id);
        }

        function getName() {
            return($this->name);
        }
        function setName($name) {
            $this->name = $name;
        }
    }

    $ot = new ObjectTracker("Zeev's Object");
    $ot2 = clone $ot; $ot2->setName("Another object");

    //1 Zeev's Object
    print($ot->getId() . " " . $ot->getName() . "<br>");

    //2 Clone of Zeev's Object
    print($ot2->getId() . " " . $ot2->getName() . "<br>");
?>
```

Step 2. Run



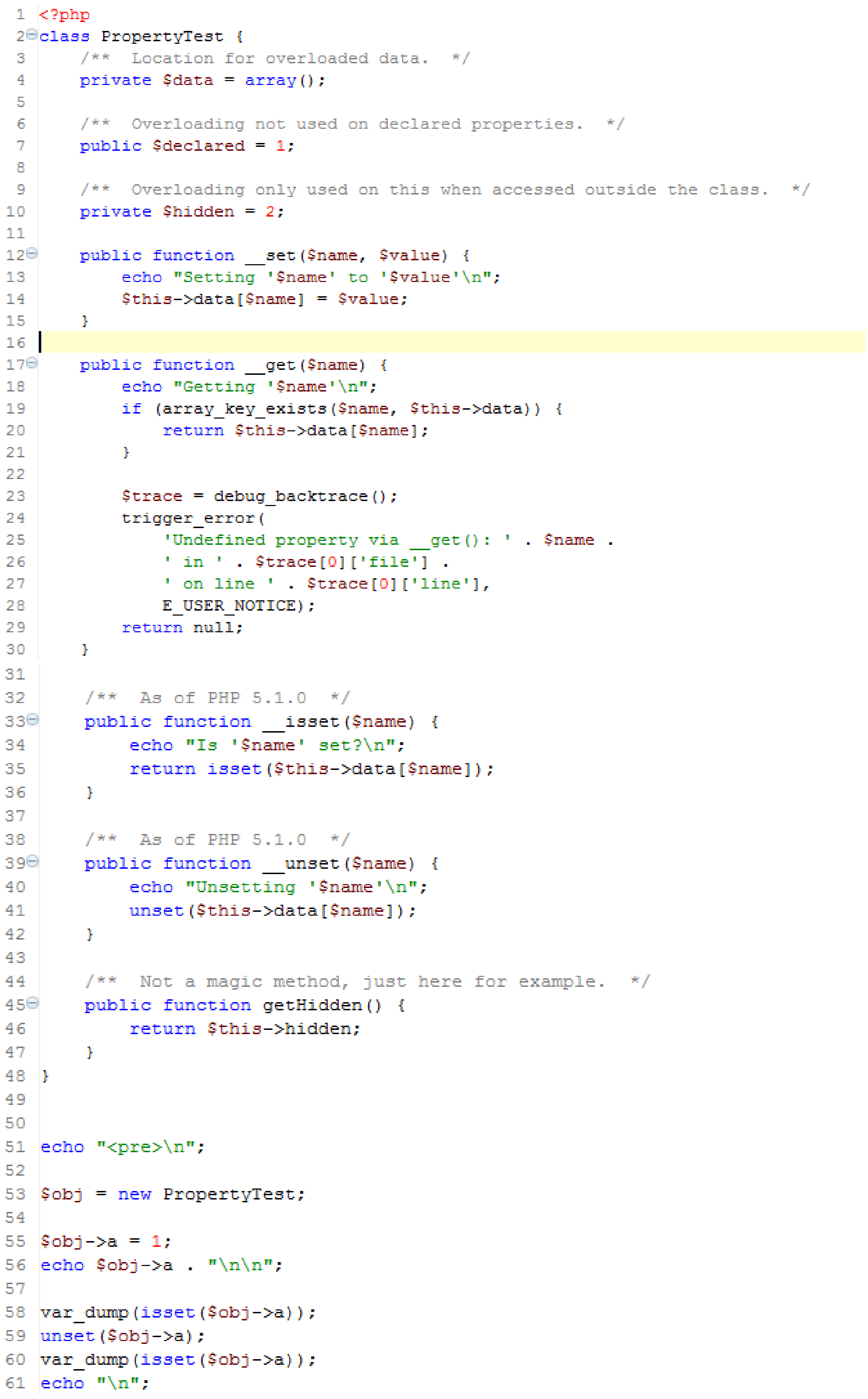
```
1 Zeev's Object
2 Another object
```

Step 3. Modify ObjectCloning.php

Do not use clone but using = operator to assign ot to ot2. Observe the result and give comments.

5.5. *Overloading*

Step 1. AttributeOverloading.php



```php
<?php
class PropertyTest {
    /**  Location for overloaded data.  */
    private $data = array();

    /**  Overloading not used on declared properties.  */
    public $declared = 1;

    /**  Overloading only used on this when accessed outside the class.  */
    private $hidden = 2;

    public function __set($name, $value) {
        echo "Setting '$name' to '$value'\n";
        $this->data[$name] = $value;
    }

    public function __get($name) {
        echo "Getting '$name'\n";
        if (array_key_exists($name, $this->data)) {
            return $this->data[$name];
        }

        $trace = debug_backtrace();
        trigger_error(
            'Undefined property via __get(): ' . $name .
            ' in ' . $trace[0]['file'] .
            ' on line ' . $trace[0]['line'],
            E_USER_NOTICE);
        return null;
    }

    /**  As of PHP 5.1.0  */
    public function __isset($name) {
        echo "Is '$name' set?\n";
        return isset($this->data[$name]);
    }

    /**  As of PHP 5.1.0  */
    public function __unset($name) {
        echo "Unsetting '$name'\n";
        unset($this->data[$name]);
    }

    /**  Not a magic method, just here for example.  */
    public function getHidden() {
        return $this->hidden;
    }
}


echo "<pre>\n";

$obj = new PropertyTest;

$obj->a = 1;
echo $obj->a . "\n\n";

var_dump(isset($obj->a));
unset($obj->a);
var_dump(isset($obj->a));
echo "\n";

echo $obj->declared . "\n\n";

echo "Let's experiment with the private property named 'hidden':\n";
echo "Privates are visible inside the class, so __get() not used...\n";
echo $obj->getHidden() . "\n";
echo "Privates not visible outside of class, so __get() is used...\n";
echo $obj->hidden . "\n";
?>
```

Step 2. Run

```
Setting 'a' to '1'
Getting 'a'
1

Is 'a' set?
bool(true)
Unsetting 'a'
Is 'a' set?
bool(false)

1

Let's experiment with the private property named 'hidden':
Privates are visible inside the class, so __get() not used...
2
Privates not visible outside of class, so __get() is used...
Getting 'hidden'
```

Step 3. MethodOverloading.php

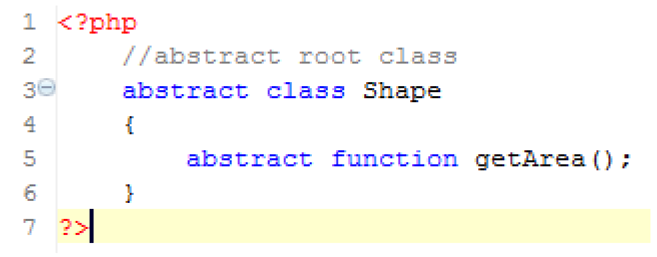
```php

<?php
class MethodTest {
    public function __call($name, $arguments) {
        // Note: value of $name is case sensitive.
        echo "Calling object method '$name' "
             . implode(', ', $arguments). "<br>";
    }

    /**  As of PHP 5.3.0  */
    public static function __callStatic($name, $arguments) {
        // Note: value of $name is case sensitive.
        echo "Calling static method '$name' "
             . implode(', ', $arguments). "<br>";
    }
}

$obj = new MethodTest;
$obj->runTest('in object context');

MethodTest::runTest('in static context');  // As of PHP 5.3.0
?>
```

Step 4. Run

```
Calling object method 'runTest' in object context
Calling static method 'runTest' in static context
```

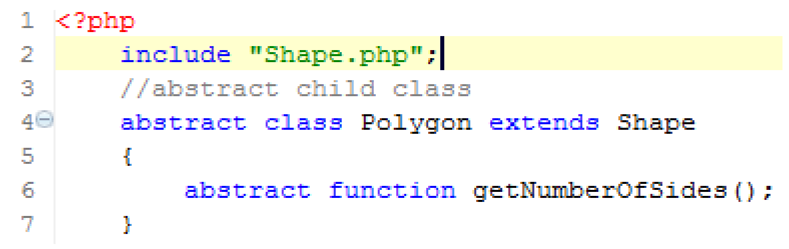
5.6. *Abstract methods and abstract classes*

Step 1. Shape.php (New Class)



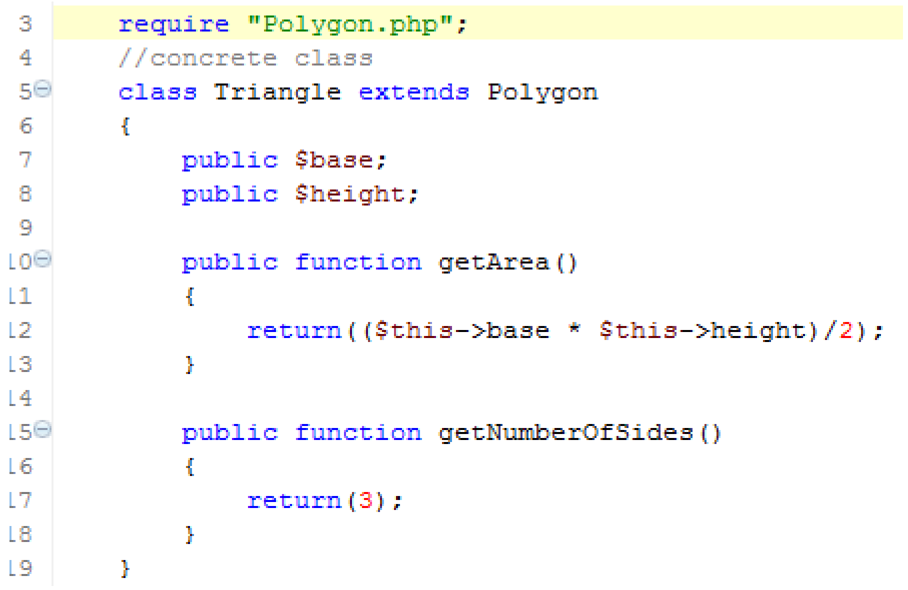
```php
<?php
    //abstract root class
    abstract class Shape
    {
        abstract function getArea();
    }
?>
```

Step 2. Polygon.php (New Class)



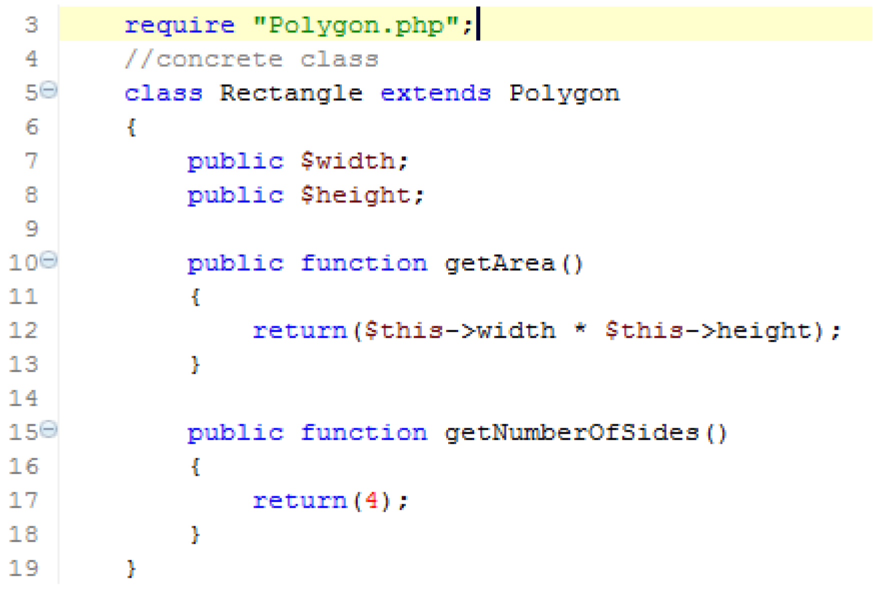
```php
<?php
    include "Shape.php";
    //abstract child class
    abstract class Polygon extends Shape
    {
        abstract function getNumberOfSides();
    }
```

Step 3. Triangle.php (New Class)



```php
    require "Polygon.php";
    //concrete class
    class Triangle extends Polygon
    {
        public $base;
        public $height;

        public function getArea()
        {
            return(($this->base * $this->height)/2);
        }

        public function getNumberOfSides()
        {
            return(3);
        }
    }
```

Step 4. Rectangle.php (New Class)



```php
    require "Polygon.php";
    //concrete class
    class Rectangle extends Polygon
    {
        public $width;
        public $height;

        public function getArea()
        {
            return($this->width * $this->height);
        }

        public function getNumberOfSides()
        {
            return(4);
        }
    }
```

Step 5. Circle.php (New Class)

```php
require "Shape.php";
//concrete class
class Circle extends Shape
{
    public $radius;

    public function getArea()
    {
        return(pi() * $this->radius * $this->radius);
    }
}
```

Step 6. Color.php

```php
//concrete root class
class Color
{
    public $name;
}
```

Step 7. Test_Shape.php

```php
<?php

    $myCollection = array();

    //make a rectangle
    $r = new Rectangle;
    $r->width = 5;
    $r->height = 7;
    $myCollection[] = $r;
    unset($r);

    //make a triangle
    $t = new Triangle;
    $t->base = 4;
    $t->height = 5;
    $myCollection[] = $t;
    unset($t);

    //make a circle
    $c = new Circle;
    $c->radius = 3;
    $myCollection[] = $c;
    unset($c);

    //make a color
    $c = new Color;
    $c->name = "blue";
    $myCollection[] = $c;
    unset($c);

    foreach($myCollection as $s)
    {
        if($s instanceof Shape)
        {
            print("Area: " . $s->getArea() .
                "<br>\n");
        }

        if($s instanceof Polygon)
        {
            print("Sides: " .
                $s->getNumberOfSides() .
                "<br>\n");
        }

        if($s instanceof Color)
        {
            print("Color: $s->name<br>\n");
        }

        print("<br>\n");
    }

?>
```

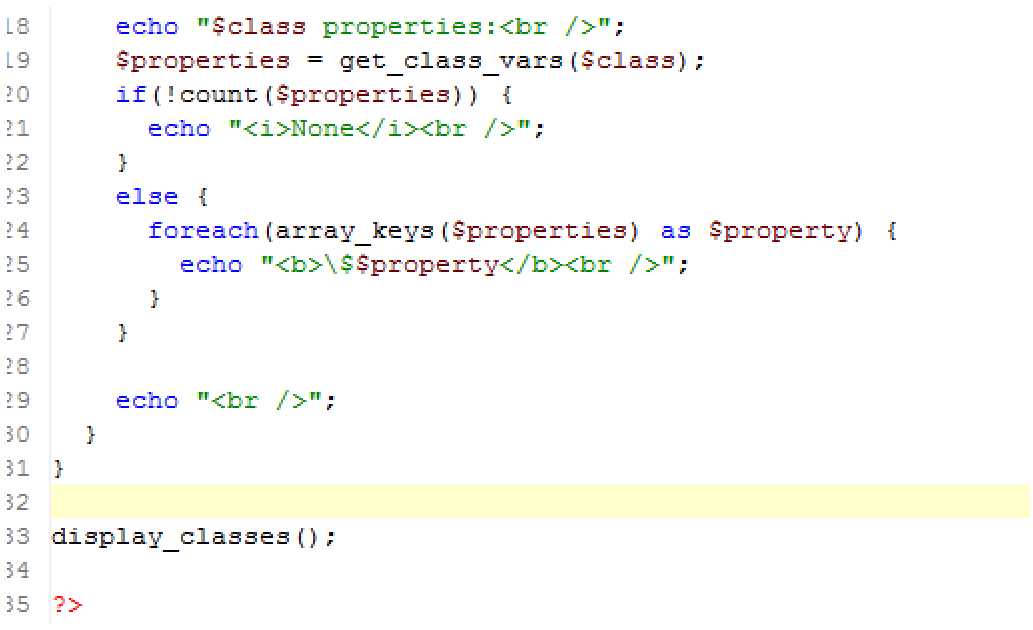
Step 8. Add autoloading class scripts at the beginning of Test_Shape.php so that when runnning Test_Shape.php; it gives the following result:

```
Area: 35
Sides: 4

Area: 10
Sides: 3

Area: 28.274333882308

Color: blue
```

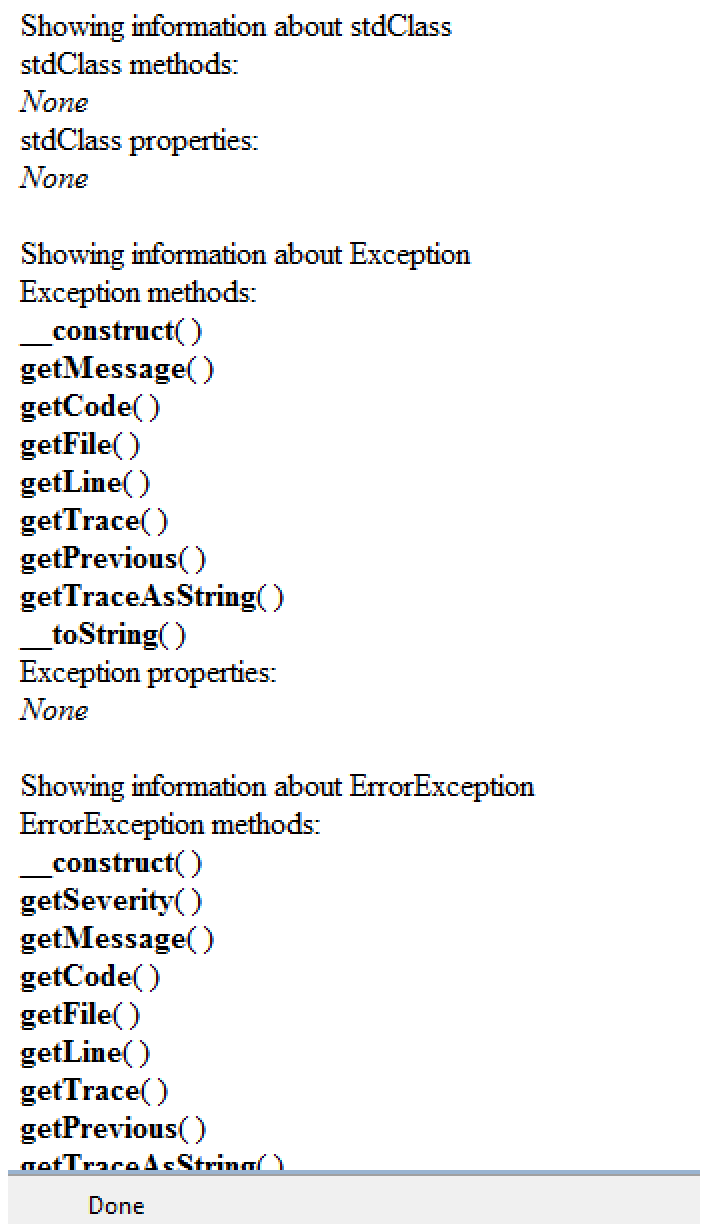
Step 9. Try to use namespace for each class.

5.7. ClassIntrospection.php

```php
<?php
function display_classes ( ) {
  $classes = get_declared_classes( );
  foreach($classes as $class) {
    echo "Showing information about $class<br />";

    echo "$class methods:<br />";
    $methods = get_class_methods($class);
    if(!count($methods)) {
      echo "<i>None</i><br />";
    }
    else {
      foreach($methods as $method) {
        echo "<b>$method</b>( )<br />";
      }
    }

```

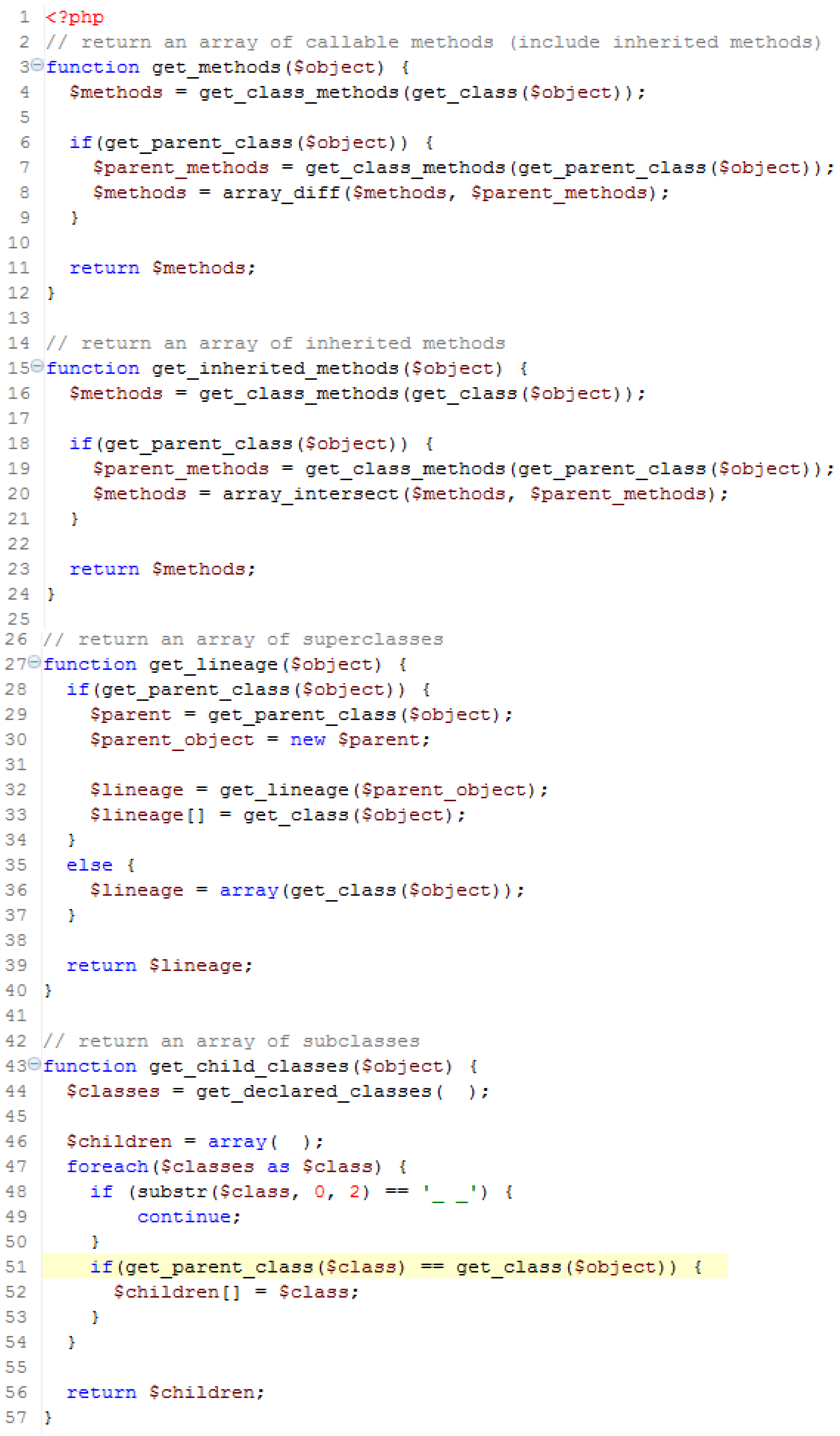


```php
    echo "$class properties:<br />";
    $properties = get_class_vars($class);
    if(!count($properties)) {
      echo "<i>None</i><br />";
    }
    else {
      foreach(array_keys($properties) as $property) {
        echo "<b>\$$property</b><br />";
      }
    }

    echo "<br />";
  }
}

display_classes();

?>
```

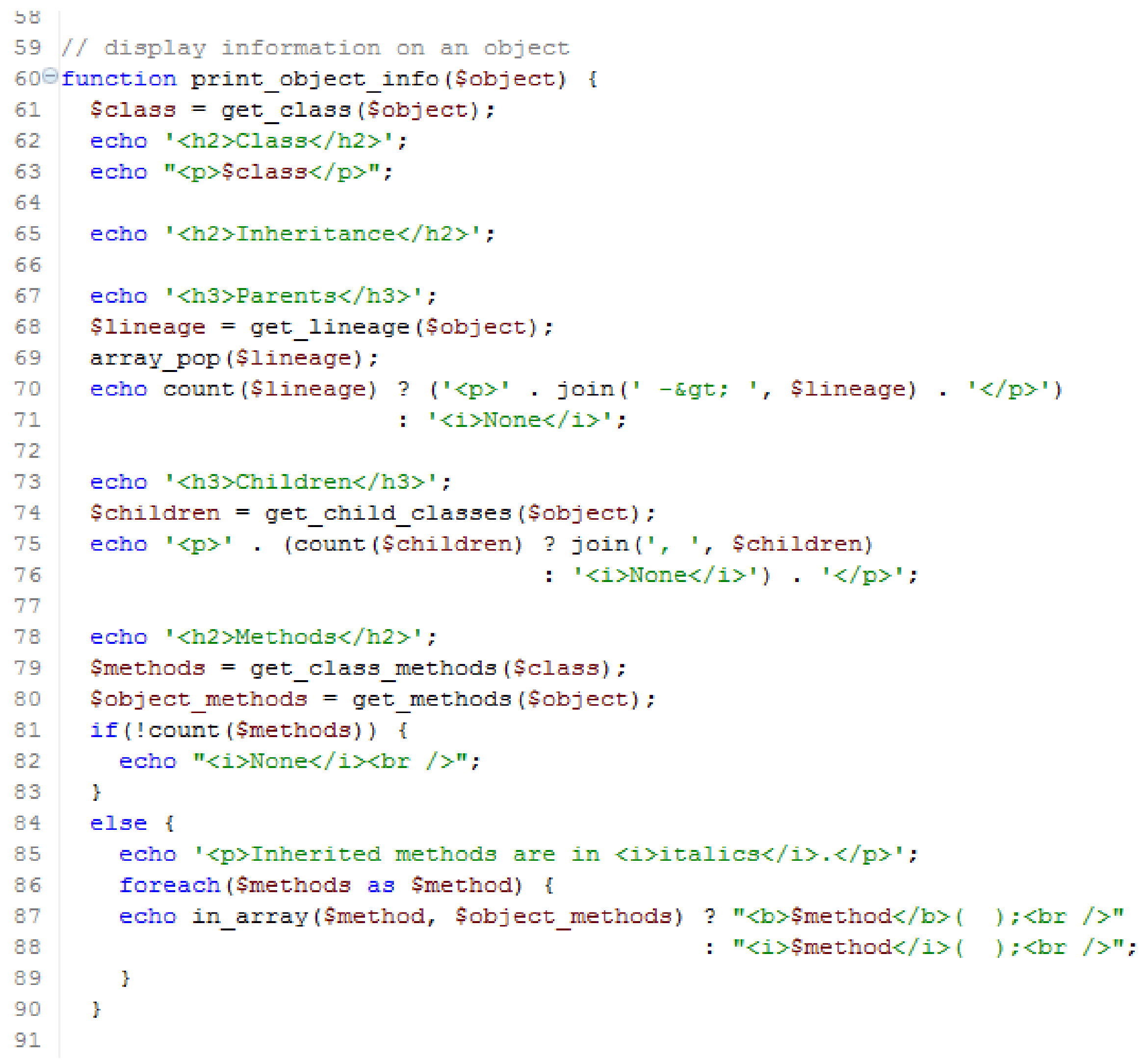


```
Showing information about stdClass
stdClass methods:
None
stdClass properties:
None

Showing information about Exception
Exception methods:
__construct( )
getMessage( )
getCode( )
getFile( )
getLine( )
getTrace( )
getPrevious( )
getTraceAsString( )
__toString( )
Exception properties:
None

Showing information about ErrorException
ErrorException methods:
__construct( )
getSeverity( )
getMessage( )
getCode( )
getFile( )
getLine( )
getTrace( )
getPrevious( )
getTraceAsString( )
```

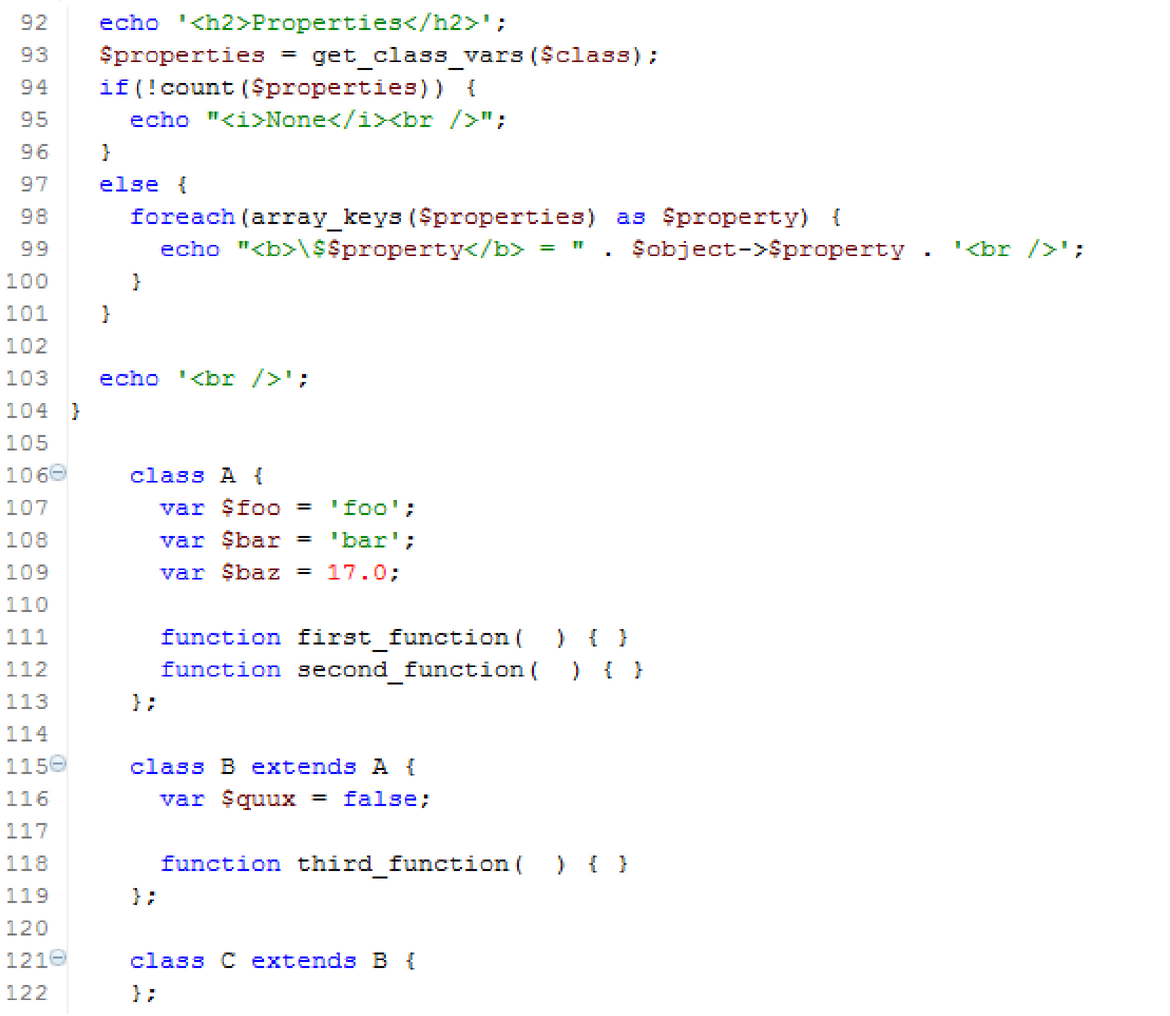
5.8. ObjectInstrospection.php



```php
<?php
// return an array of callable methods (include inherited methods)
function get_methods($object) {
  $methods = get_class_methods(get_class($object));

  if(get_parent_class($object)) {
    $parent_methods = get_class_methods(get_parent_class($object));
    $methods = array_diff($methods, $parent_methods);
  }

  return $methods;
}

// return an array of inherited methods
function get_inherited_methods($object) {
  $methods = get_class_methods(get_class($object));

  if(get_parent_class($object)) {
    $parent_methods = get_class_methods(get_parent_class($object));
    $methods = array_intersect($methods, $parent_methods);
  }

  return $methods;
}

// return an array of superclasses
function get_lineage($object) {
  if(get_parent_class($object)) {
    $parent = get_parent_class($object);
    $parent_object = new $parent;

    $lineage = get_lineage($parent_object);
    $lineage[] = get_class($object);
  }
  else {
    $lineage = array(get_class($object));
  }

  return $lineage;
}

// return an array of subclasses
function get_child_classes($object) {
  $classes = get_declared_classes(  );

  $children = array(  );
  foreach($classes as $class) {
    if (substr($class, 0, 2) == '__') {
        continue;
    }
    if(get_parent_class($class) == get_class($object)) {
      $children[] = $class;
    }
  }

  return $children;
}
```



```php

// display information on an object
function print_object_info($object) {
  $class = get_class($object);
  echo '<h2>Class</h2>';
  echo "<p>$class</p>";

  echo '<h2>Inheritance</h2>';

  echo '<h3>Parents</h3>';
  $lineage = get_lineage($object);
  array_pop($lineage);
  echo count($lineage) ? ('<p>' . join(' -&gt; ', $lineage) . '</p>')
                       : '<i>None</i>';

  echo '<h3>Children</h3>';
  $children = get_child_classes($object);
  echo '<p>' . (count($children) ? join(', ', $children)
                                 : '<i>None</i>') . '</p>';

  echo '<h2>Methods</h2>';
  $methods = get_class_methods($class);
  $object_methods = get_methods($object);
  if(!count($methods)) {
    echo "<i>None</i><br />";
  }
  else {
    echo '<p>Inherited methods are in <i>italics</i>.</p>';
    foreach($methods as $method) {
    echo in_array($method, $object_methods) ? "<b>$method</b>(  );<br />"
                                            : "<i>$method</i>(  );<br />";
    }
  }

```



```php
  echo '<h2>Properties</h2>';
  $properties = get_class_vars($class);
  if(!count($properties)) {
    echo "<i>None</i><br />";
  }
  else {
    foreach(array_keys($properties) as $property) {
      echo "<b>\$$property</b> = " . $object->$property . '<br />';
    }
  }

  echo '<br />';
}

    class A {
      var $foo = 'foo';
      var $bar = 'bar';
      var $baz = 17.0;

      function first_function(  ) { }
      function second_function(  ) { }
    };

    class B extends A {
      var $quux = false;

      function third_function(  ) { }
    };

    class C extends B {
    };
    $a = new A;
    $a->foo = 'sylvie';
    $a->bar = 23;

    $b = new B;
    $b->foo = 'bruno';
    $b->quux = true;

    $c = new C;

    print_object_info($a);
    print_object_info($b);
    print_object_info($c);
?>
```

Result:

Class

A

Inheritance

Parents

None

Children

B

Methods

Inherited methods are in *italics*.

first_function( );
second_function( );

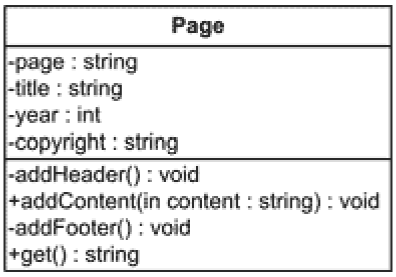
Properties

$foo = sylvie
$bar = 23
$baz = 17

Class

5.9. Exercise

Represented as a UML diagram, the Page class is shown in the following figure:



Write classes and web pages to generate several different pages and save/display them in a web browser.