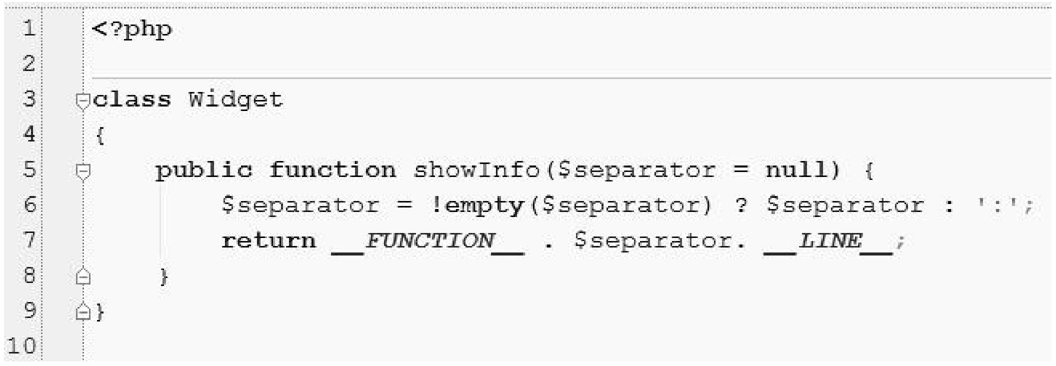
PHP Question 1.1

Consider that SupportBase is declared elsewhere and successfully loaded by the autoloader. Given the described class definitions, what is the output of index.php?

Application/Module/Widget.php

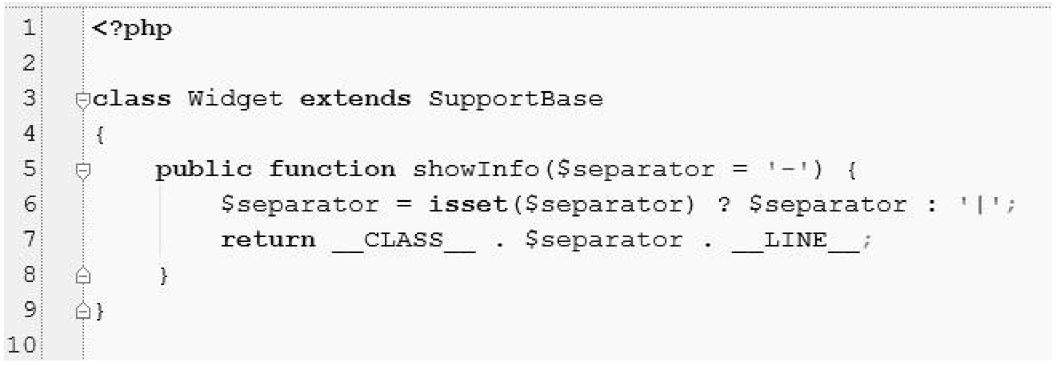


```php
<?php

class Widget
{
    public function showInfo($separator = null) {
        $separator = !empty($separator) ? $separator : ':';
        return __FUNCTION__ . $separator. __LINE__;
    }
}

```

Support/Module/Widget.php

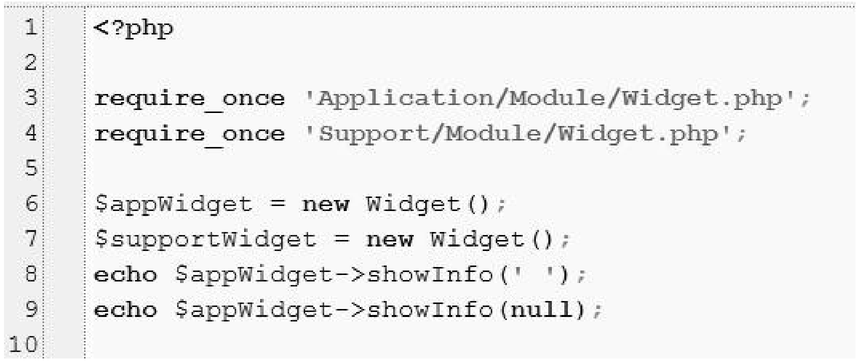


```php
<?php

class Widget extends SupportBase
{
    public function showInfo($separator = '-') {
        $separator = isset($separator) ? $separator : '|';
        return __CLASS__ . $separator . __LINE__;
    }
}

```

index.php



```php
<?php

require_once 'Application/Module/Widget.php';
require_once 'Support/Module/Widget.php';

$appWidget = new Widget();
$supportWidget = new Widget();
echo $appWidget->showInfo(' ');
echo $appWidget->showInfo(null);

```

PHP Answer 1.1

(Provide your answer below. Use as much space as necessary.)

The given code doesn't work the right way because the Widget class names are declared the same way, there will be a conflict in names. Also I would rather use namespaces insted of require_once.

Moreover this kind of naming Widget class will cause future misunderstandings between teammates. So I decided to do the following:

```
class Widget
{
  public function showInfo($separator = null) {
    $separator = !empty($separator) ? $separator : ':';
    return __FUNCTION__ . $separator . __LINE__;
  }
}
require_once './SupportBase.php';
class SupportWidget extends SupportBase
{
  public function showInfo($separator = '-') {
    $separator = isset($separator) ? $separator : '|';
    return __FUNCTION__ . $separator . __LINE__;   }
}
class SupportBase{
  function test(){
    return;
  }
}
require_once './Application/Module/Widget.php';
require_once './Support/Module/SupportWidget.php';

$appWidget = new Widget();
$supportWidget = new SupportWidget();

echo $appWidget->showInfo(' ');
echo $appWidget->showInfo(null);
```

The output would be like:

```
showInfo 7
showInfo:7
```

where the 7 comes from number of the code line it was called from. 'showInfo' is name of the function.