Class Abstraction

PHP has abstract classes and methods. Classes defined as abstract cannot be instantiated, and any class that contains at least one abstract method must also be abstract. Methods defined as abstract simply declare the method's signature; they cannot define the implementation.

When inheriting from an abstract class, all methods marked abstract in the parent's class declaration must be defined by the child class, and follow the usual [inheritance](https://www.php.net/manual/en/language.oop5.inheritance.php) and [signature compatibility](https://www.php.net/manual/en/language.oop5.basic.php#language.oop.lsp) rules.

Example #1 Abstract class example

<?php  
abstract class AbstractClass  
{  
    // Force Extending class to define this method  
    abstract protected function getValue();  
    abstract protected function prefixValue($prefix);  
  
    // Common method  
    public function printOut() {  
        print $this->getValue() . "\n";  
    }  
}  
  
class ConcreteClass1 extends AbstractClass  
{  
    protected function getValue() {  
        return "ConcreteClass1";  
    }  
  
    public function prefixValue($prefix) {  
        return "{$prefix}ConcreteClass1";  
    }  
}  
  
class ConcreteClass2 extends AbstractClass  
{  
    public function getValue() {  
        return "ConcreteClass2";  
    }  
  
    public function prefixValue($prefix) {  
        return "{$prefix}ConcreteClass2";  
    }  
}  
  
$class1 = new ConcreteClass1;  
$class1->printOut();  
echo $class1->prefixValue('FOO\_') ."\n";  
  
$class2 = new ConcreteClass2;  
$class2->printOut();

echo $class2->prefixValue('FOO\_') ."\n";  
?>

The above example will output:

ConcreteClass1

FOO\_ConcreteClass1

ConcreteClass2

FOO\_ConcreteClass2

Example #2 Abstract class example

<?php  
abstract class AbstractClass  
{  
    // Our abstract method only needs to define the required arguments  
    abstract protected function prefixName($name);  
  
}  
  
class ConcreteClass extends AbstractClass  
{  
  
    // Our child class may define optional arguments not in the parent's signature  
    public function prefixName($name, $separator = ".") {  
        if ($name == "Pacman") {  
            $prefix = "Mr";  
        } elseif ($name == "Pacwoman") {  
            $prefix = "Mrs";  
        } else {  
            $prefix = "";  
        }  
        return "{$prefix}{$separator} {$name}";  
    }  
}  
  
$class = new ConcreteClass;  
echo $class->prefixName("Pacman"), "\n";  
echo $class->prefixName("Pacwoman"), "\n";  
?>

The above example will output:

Mr. Pacman

Mrs. Pacwoman