

3.11: Objects and Classes

We can group related functions together into an item called a class. Classes are collections of functions (when they are in classes, they are called methods) and variables that are related or collectively represent our knowledge or actions that can be performed on a concept. We can create a class by writing:

```
1. class math{  
2. }
```

This gives us an empty class called Math. Let us include our adding function:

```
1. class math{  
2. function add($num1, $num2){  
3. return ($num1 + $num2);  
4. }  
5. }
```

and add a couple more functions:

```
1. class math{  
2. function add($num1, $num2){  
3. return ($num1 + $num2);  
4. }  
5. function subtract($num1, $num2){  
6. return ($num1—$num2);  
7. }  
8. function divide($num1, $num2){  
9. if($num2!=0){return ($num1 / $num2);}  
10. else return "error";  
11. }  
12. }
```

By creating the class, we have declared that add, subtract, and divide all belong to Math. To use our new class, we create a copy of it, called an instance, in our script. In the divide function you can see we check to make sure we will not divide by zero, which would cause an error. This is an example of how we can use classes and functions to expand functionality and protect our program. We can reference the variable that is our object to use the functions and data members (variables that are inside a class) inside it:

```
1. $ourMath = new math();  
2. echo $ourMath->add(5,9);
```

The arrow tells us to use something inside our class, in this case the add method. We can use these methods over several lines:

```
1. $temp = $ourMath->add(5,9);  
2. $temp = $ourMath->divide($temp/2);
```

or nest them:

```
1. $temp = $ourMath->divide($ourMath->add(5,9),2);
```

Classes are commonly used to create libraries of related actions, like the Math example we made, or to create collections of methods and data members that pertain to a concept in our system. For example, if this was an online course system we might have objects to represent concepts like students and courses.

Methods in the student class would revolve around the student, like calculating their GPA or generating a list of courses the student is enrolled in. The courses class might have a similar method that calculates the overall average of all students in a class, but these could only be run from an object of the appropriate type and permission. An object is an instance (or copy) of the class that represents a particular item, like a particular student or course.

While most functions in PHP can be used by any script, we can control what pieces of our class structures can be accessed by programmers when an instance of the class is used. We do this with the keywords public, private, and protected that precede our

methods or data members in our class. This gives us a means to protect our information and/or control what can be done when the object is being used.

Public items can be accessed by “anyone” (or anything, such as other classes), meaning we can reference them using `->` just like in our example. Protected items can be accessed by other methods in the class, or by methods in parent or child classes that use the class in question—we will cover class inheritance here though. Finally, private items can only be accessed by the class itself. This means that a method or data member marked private in our Math example could only be used by other methods in the class, not by us when we write code to use an object based on the class. To see this in action, we will look at public and private in action by adjusting our math class. To learn more about inheritance and protected methods, you can refer to this chapter’s [“Learn more”](#) section.

```
1. class math{
2. public function add($num1, $num2){
3. return ($num1 + $num2);
4. }
5. private function subtract($num1, $num2){
6. return ($num1—$num2);
7. }
8. public function divide($num1, $num2){
9. return ($num1 / $num2);
10. }
11. }
```

Since our add and divide methods are set to public, our previous example still works—we are able to add and divide from our code without any issues. If you try to use subtract though (`$ourMath->subtract(5,2);`) you will end up with an error:

```
1. Fatal error: Call to private method math::subtract()
```

The only way for the subtract method to be used is by being called from within another method in the Math class. If we had a more advanced method like calculating a longer formula, that method could call subtract as part of its execution.

Learn more

Keywords, search terms: Functions, classes, objects, inheritance

More examples: <http://php.net/manual/en/language.oop5.visibility.php>

Constructors and inheritance: <http://net.tutsplus.com/tutorials/php/object-oriented-php-for-beginners/>

PHP Objects Patterns and Practice 3rd Edition: <http://it-ebooks.info/book/411/>

3.11: Objects and Classes is shared under a [not declared](#) license and was authored, remixed, and/or curated by LibreTexts.

- **3.11: Objects and Classes** by [Michael Mendez](#) has no license indicated.