when a class’s methods return the $this keyword, they can be chained together to create much more streaming code.

For instance, in our Car class, let’s say that we want to measure how much fuel we have in our car’s tank. The amount of fuel in the tank is dependent upon the number of miles we have driven in our car, as well as the amount of fuel that we put in the tank.

In order to achieve our goal, we are going to put a public property $tank to our class that represents the number of gallons of fuel that we have in the car’s tank.

class Car {

public $tank;

}

We must also add two methods to our Car class:

1. The fill() method adds gallons of fuel to our car’s tank.

2. The ride() method calculates how much fuel we consume when we ride a certain distance,

and then subtracts it from the tank. In our example, we assume that the car consumes 1 gallon

of fuel every 50 miles.

class Car {

public $tank;

// Add gallons of fuel to the tank when we fill it

public function fill($float)

{

$this-> tank += $float;

return $this;

}

// Subtract gallons of fuel from the tank as we ride the car

public function ride($float)

{

$miles = $float;

$gallons = $miles/50;

$this-> tank -= $gallons;

return $this;

}

}

Now, we can create an object from the Car class with the name of $bmw and find out the number of gallons of fuel left in our car’s tank after we have filled the tank with 10 gallons of fuel and driven

40 miles.

// Create an object from the Car class

$bmw = new Car();

// Add 10 gallons of fuel, then ride 40 miles

// and get the number of gallons in the tank

$tank = $bmw -> fill(10) -> ride(40) -> tank;

// Print the results to the screen

echo "The number of gallons left in the tank: " . $tank . " gal.";

Result:

The number of gallons left in the tank: 9.2 gal.