**Video 2.2: PHP Class & Objects – Typed Properties – Constructors & Destructors**

Let's talk about classes and objects. A class, as mentioned in the previous lesson, is like a blueprint and an object is something that you build from that blueprint, or in other words, an instance of that class. Object is a PHP data type that we haven't covered yet. In the first section of the course, we covered four scalar types, we covered arrays, callables, resources, and null data types.

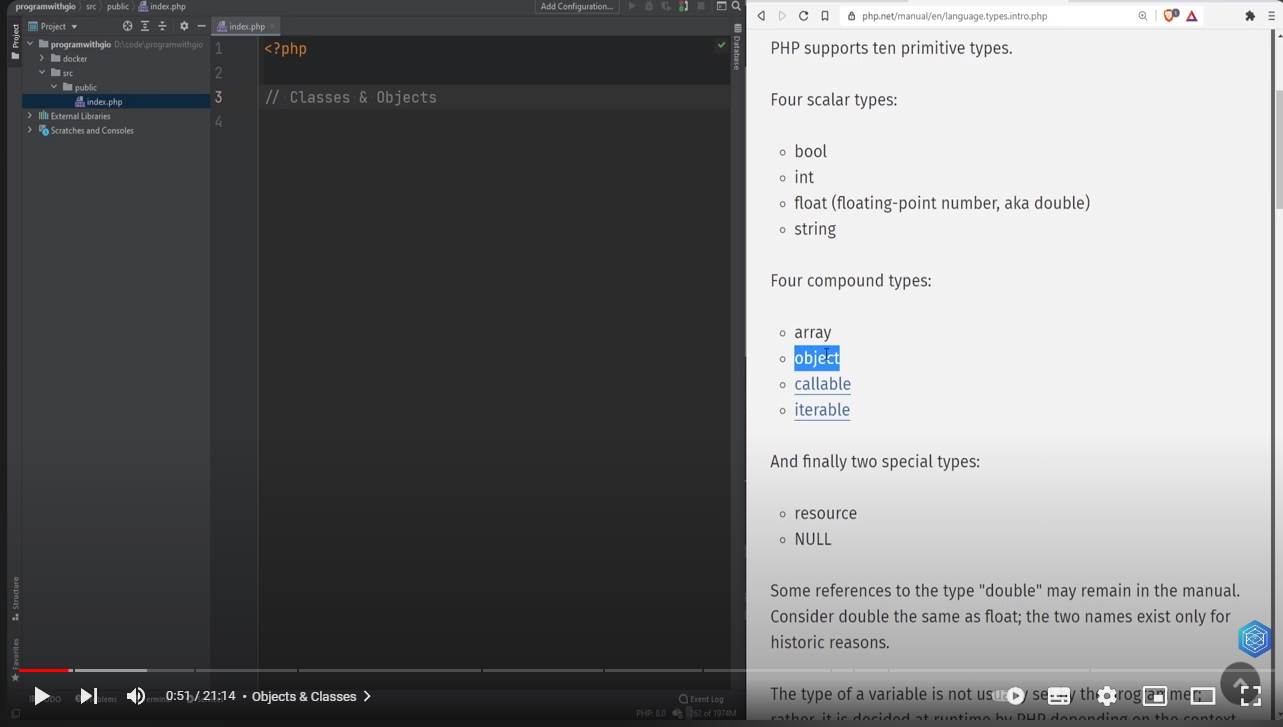
* Hãy nói về các lớp và đối tượng. Một lớp, như đã đề cập trong bài học trước, giống như một bản thiết kế và một đối tượng là thứ mà bạn xây dựng từ bản thiết kế đó, hay nói cách khác, một thể hiện của lớp đó. Đối tượng là kiểu dữ liệu PHP mà chúng ta chưa đề cập đến. Trong phần đầu tiên của khóa học, chúng tôi đã đề cập đến bốn loại vô hướng, chúng tôi đã đề cập đến các mảng, dữ liệu có thể gọi được, tài nguyên và loại dữ liệu null.

That leaves us with two data types that we haven't covered yet, and that is iterables and objects. Iterables will be covered later in the course, and today we'll talk about object data type. PHP has a generic class implementation called std class that you could use to create generic objects, or you could cast your variables that are of another data types into objects.

* Điều đó để lại cho chúng ta hai loại dữ liệu mà chúng ta chưa đề cập đến, đó là dữ liệu lặp và đối tượng. Các tính năng có thể lặp lại sẽ được đề cập ở phần sau của khóa học và hôm nay chúng ta sẽ nói về kiểu dữ liệu đối tượng. PHP có một triển khai lớp chung được gọi là lớp std mà bạn có thể sử dụng để tạo các đối tượng chung hoặc bạn có thể chuyển các biến thuộc loại dữ liệu khác thành đối tượng.

And we'll talk about that towards the end of this lesson, so make sure to watch it till the end. In addition to being able to create objects using generic STD class, you can also create your own custom classes. Let's create a class called Transaction because it applies to the project that we'll be working on.

* Và chúng ta sẽ nói về điều đó ở cuối bài học này, vì vậy hãy nhớ xem đến cuối. Ngoài việc có thể tạo các đối tượng bằng lớp STD chung, bạn cũng có thể tạo các lớp tùy chỉnh của riêng mình. Hãy tạo một lớp có tên là Giao dịch vì nó áp dụng cho dự án mà chúng ta đang thực hiện.



The definition must begin with a class keyword followed by the class name, which should begin either with a letter or an underscore, followed by letters, numbers, or underscores. So in this case, that's just transaction. And then we put curly braces to enclose the class definition. Note here that you could follow different standards and conventions when it comes to naming your variables or classes or overall your code structure.

* Định nghĩa phải bắt đầu bằng từ khóa lớp, theo sau là tên lớp, bắt đầu bằng chữ cái hoặc dấu gạch dưới, theo sau là chữ cái, số hoặc dấu gạch dưới. Vì vậy, trong trường hợp này, đó chỉ là giao dịch. Và sau đó chúng ta đặt dấu ngoặc nhọn để kèm theo định nghĩa lớp. Lưu ý ở đây rằng bạn có thể tuân theo các tiêu chuẩn và quy ước khác nhau khi đặt tên cho các biến hoặc lớp hoặc tổng thể cấu trúc mã của bạn.

You don't have to name your classes after your file, but that is the recommended and widely used. convention. You could also have more than one class definitions in a file, which would work perfectly fine. But again, it is not the recommended way. The recommended way is to have a single class per file and nothing else in that file.

* Bạn không cần phải đặt tên các lớp theo tên tệp của mình, nhưng đó là cách được khuyến nghị và sử dụng rộng rãi. quy ước. Bạn cũng có thể có nhiều định nghĩa lớp trong một tệp, điều này sẽ hoạt động hoàn toàn tốt. Nhưng một lần nữa, đó không phải là cách được khuyến khích. Cách được đề xuất là có một lớp duy nhất cho mỗi tệp và không có gì khác trong tệp đó.

We'll cover the PSR standards and conventions in this course, so don't worry about it now. So now that we have a class, let's Create an object. As I mentioned before, you could also create the object within the same class right after the closing curly brace, but I'm not going to do that because it is not the recommended way.

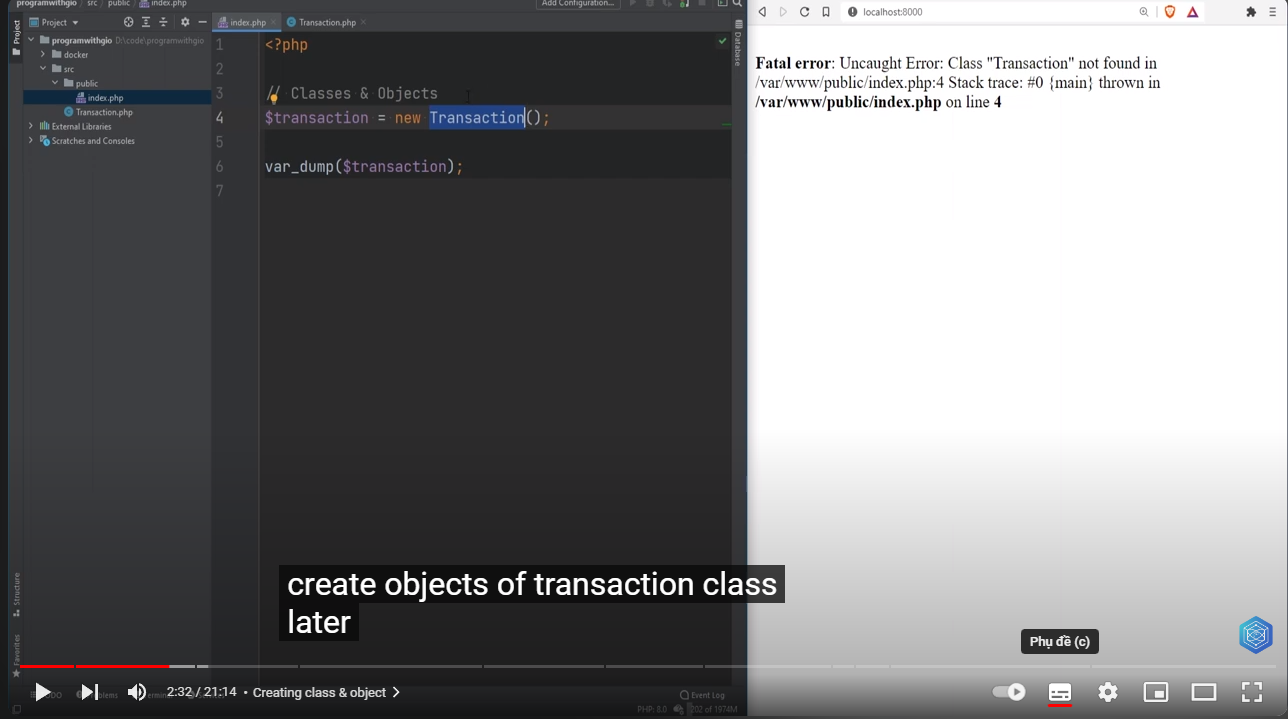
* Chúng tôi sẽ đề cập đến các tiêu chuẩn và quy ước của PSR trong khóa học này, vì vậy đừng lo lắng về điều đó ngay bây giờ. Bây giờ chúng ta đã có một lớp, hãy tạo một đối tượng. Như tôi đã đề cập trước đó, bạn cũng có thể tạo đối tượng trong cùng một lớp ngay sau dấu ngoặc nhọn đóng, nhưng tôi sẽ không làm điều đó vì đó không phải là cách được khuyến nghị.

So I'm going to create the object within the index. php in public directory. So I'm going to call my variable. Transaction. And to create an object, we need to use the new keyword followed by the class name, which is transaction followed by parentheses. And parentheses is not always required depending on your constructor, but it's always a good idea to stay consistent and just use parentheses regardless.

* Vì vậy, tôi sẽ tạo đối tượng trong chỉ mục. php trong thư mục công cộng. Vì vậy, tôi sẽ gọi biến của mình. Giao dịch. Và để tạo một đối tượng, chúng ta cần sử dụng từ khóa new theo sau là tên lớp, đó là giao dịch theo sau là dấu ngoặc đơn. Và không phải lúc nào cũng cần có dấu ngoặc đơn tùy thuộc vào hàm tạo của bạn, nhưng bạn nên luôn nhất quán và chỉ sử dụng dấu ngoặc đơn bất kể điều gì.

So let's var dump transaction. and let's open it in the browser and we get fatal error. That's because index. php does not know where the transaction class exists. It hasn't loaded that file. We need to explicitly include or require this file for index. php to be able to create objects of transaction class.

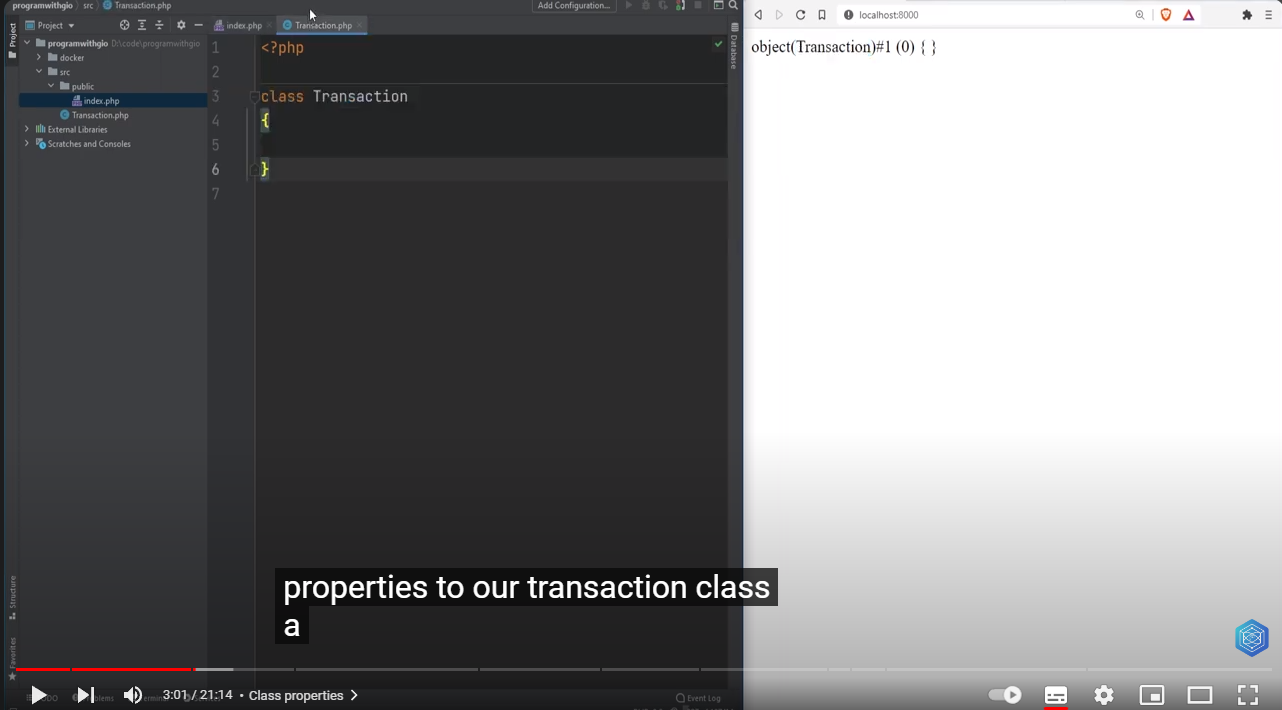
* Vì vậy, hãy thực hiện giao dịch var dump. và hãy mở nó trong trình duyệt và chúng tôi gặp lỗi nghiêm trọng. Đó là bởi vì chỉ mục. php không biết lớp giao dịch tồn tại ở đâu. Nó chưa tải tập tin đó. Chúng tôi cần bao gồm hoặc yêu cầu tệp này một cách rõ ràng cho chỉ mục. php để có thể tạo các đối tượng của lớp giao dịch.



Later in the course we're going to cover something called namespaces and autoloading. where you will not have to do that. But for now we need to require this. So I'm going to add require once and the transaction class is in the parent directory. So dot dot transaction dot PHP. And if I refresh the page, now it prints an empty object.

* Ở phần sau của khóa học, chúng ta sẽ đề cập đến một thứ gọi là không gian tên và tự động tải. nơi bạn sẽ không phải làm điều đó. Nhưng bây giờ chúng ta cần yêu cầu điều này. Vì vậy, tôi sẽ thêm yêu cầu một lần và lớp giao dịch nằm trong thư mục mẹ. Vì vậy, chấm giao dịch chấm chấm PHP. Và nếu tôi làm mới trang, bây giờ nó sẽ in một đối tượng trống.

We see that the data type is object and it's instance of a transaction class. A class can have variables which are called properties and functions, which are called methods. So let's add a couple of properties to our transaction class. A transaction at minimum should have an amount and description properties, right?

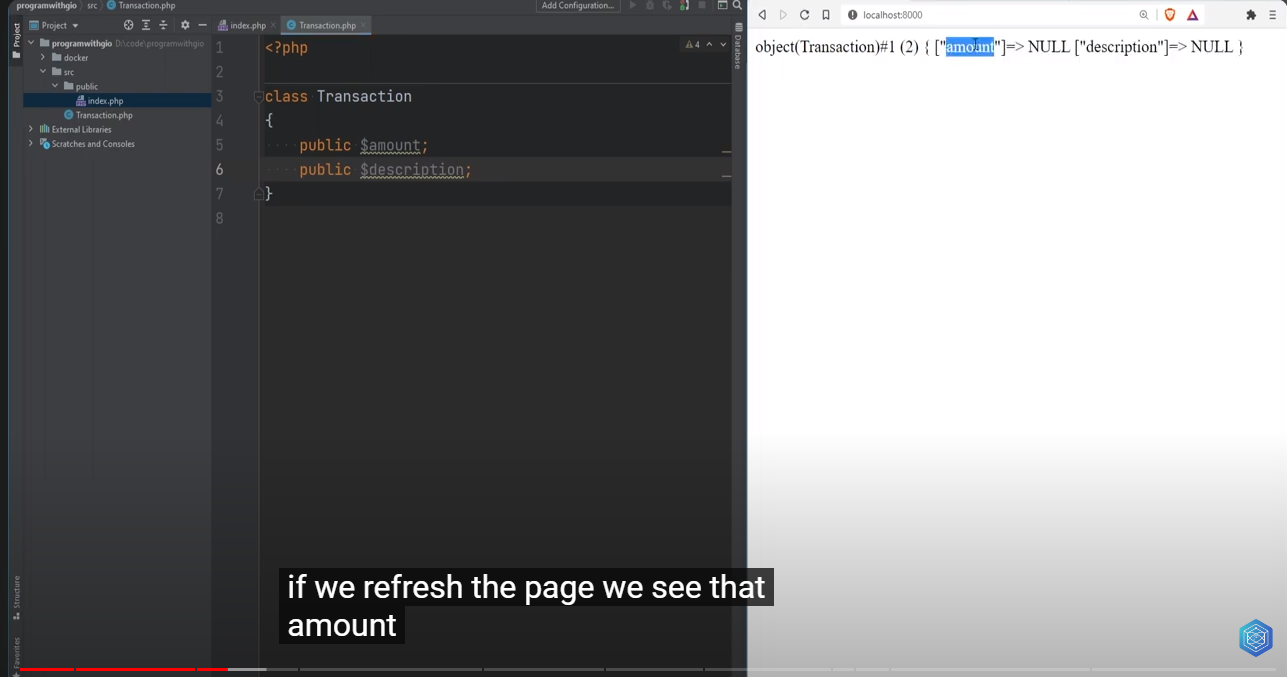
* Chúng ta thấy rằng kiểu dữ liệu là đối tượng và là thể hiện của lớp giao dịch. Một lớp có thể có các biến được gọi là thuộc tính và hàm, được gọi là phương thức. Vì vậy, hãy thêm một vài thuộc tính vào lớp giao dịch của chúng ta. Một giao dịch tối thiểu phải có số tiền và thuộc tính mô tả, phải không?
* 

So let's add those two properties. We need to define the visibility of the property to either public, private, or protected. And these are also called access modifiers. And they should be added to both properties and methods. So we start with the access modifier. In this case, let's do public. Then the variable name.

* Vì vậy, hãy thêm hai thuộc tính đó. Chúng ta cần xác định khả năng hiển thị của thuộc tính ở chế độ công khai, riêng tư hoặc được bảo vệ. Và đây còn được gọi là công cụ sửa đổi truy cập. Và chúng nên được thêm vào cả thuộc tính và phương thức. Vì vậy, chúng tôi bắt đầu với công cụ sửa đổi truy cập. Trong trường hợp này, hãy làm công khai. Sau đó là tên biến.

So that is amount and semicolon. And we do the same thing for description. And now we have two properties that are publicly accessible. If we refresh the page, we see that amount and description are available there, and they're set to null. The values of these properties will be set to null if you don't assign them the value.

* Vậy đó là số tiền và dấu chấm phẩy. Và chúng tôi làm điều tương tự cho phần mô tả. Và bây giờ chúng tôi có hai thuộc tính có thể truy cập công khai. Nếu làm mới trang, chúng tôi sẽ thấy số tiền và mô tả có sẵn ở đó và chúng được đặt thành rỗng. Giá trị của các thuộc tính này sẽ được đặt thành null nếu bạn không gán giá trị cho chúng.



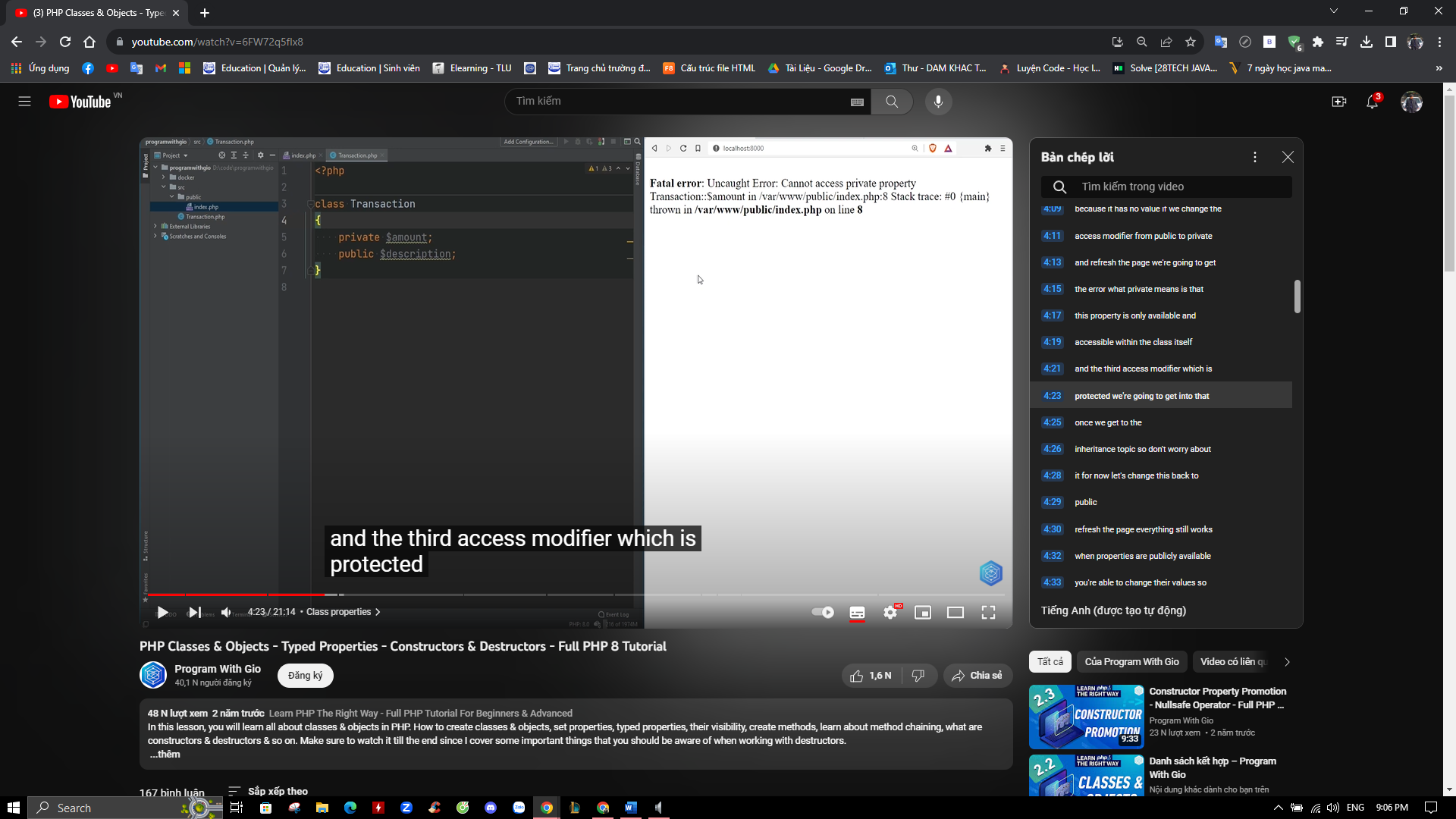
So the null will be the default value for them. Now, there is one exception to this when using type hinting, and we'll get to that in a minute. As mentioned before, public means that it's accessible to everyone interacting with the object, even outside of the class. So if we wanted to var dump the value of amount, all we have to do is use the object operator, which is dash greater than sign, and as you can see, my IDE autocompletes showing me all the publicly available methods and properties.

* Vì vậy, null sẽ là giá trị mặc định cho chúng. Bây giờ, có một ngoại lệ đối với vấn đề này khi sử dụng gợi ý kiểu và chúng ta sẽ giải quyết vấn đề đó sau một phút. Như đã đề cập trước đó, public có nghĩa là mọi người tương tác với đối tượng đều có thể truy cập được, ngay cả bên ngoài lớp. Vì vậy, nếu chúng ta muốn var dump giá trị của money, tất cả những gì chúng ta phải làm là sử dụng toán tử đối tượng, dấu gạch ngang lớn hơn dấu và như bạn có thể thấy, IDE tự động hoàn thành hiển thị cho tôi tất cả các phương thức và thuộc tính có sẵn công khai.

In this case, we only have two properties and no methods, so we're going to use amount, and we var dump that, and we get null, because it has no value. If we change the access modifier from public to private, and refresh the page, we're going to get the error. What private means is that this property is only available and accessible within the class itself.

* Trong trường hợp này, chúng ta chỉ có hai thuộc tính và không có phương thức, vì vậy chúng ta sẽ sử dụng số lượng và chúng ta kết xuất giá trị đó và nhận được giá trị rỗng vì nó không có giá trị. Nếu chúng tôi thay đổi công cụ sửa đổi quyền truy cập từ công khai thành riêng tư và làm mới trang, chúng tôi sẽ gặp lỗi. Ý nghĩa riêng tư là thuộc tính này chỉ khả dụng và có thể truy cập được trong chính lớp đó.
* 

And the third access modifier, which is protected, we're going to get into that once we get to the inheritance topic. So don't worry about it for now. Let's change this back to public, refresh the page, everything still works. When properties are publicly available, you're able to change their values. So right here, we could say transaction.

* Và công cụ sửa đổi quyền truy cập thứ ba, được bảo vệ, chúng ta sẽ đi sâu vào vấn đề đó khi chúng ta chuyển sang chủ đề kế thừa. Vì vậy, đừng lo lắng về nó bây giờ. Hãy chuyển nó về chế độ công khai, làm mới trang, mọi thứ vẫn hoạt động. Khi thuộc tính được hiển thị công khai, bạn có thể thay đổi giá trị của chúng. Vì vậy, ngay tại đây, chúng ta có thể nói giao dịch.
* 

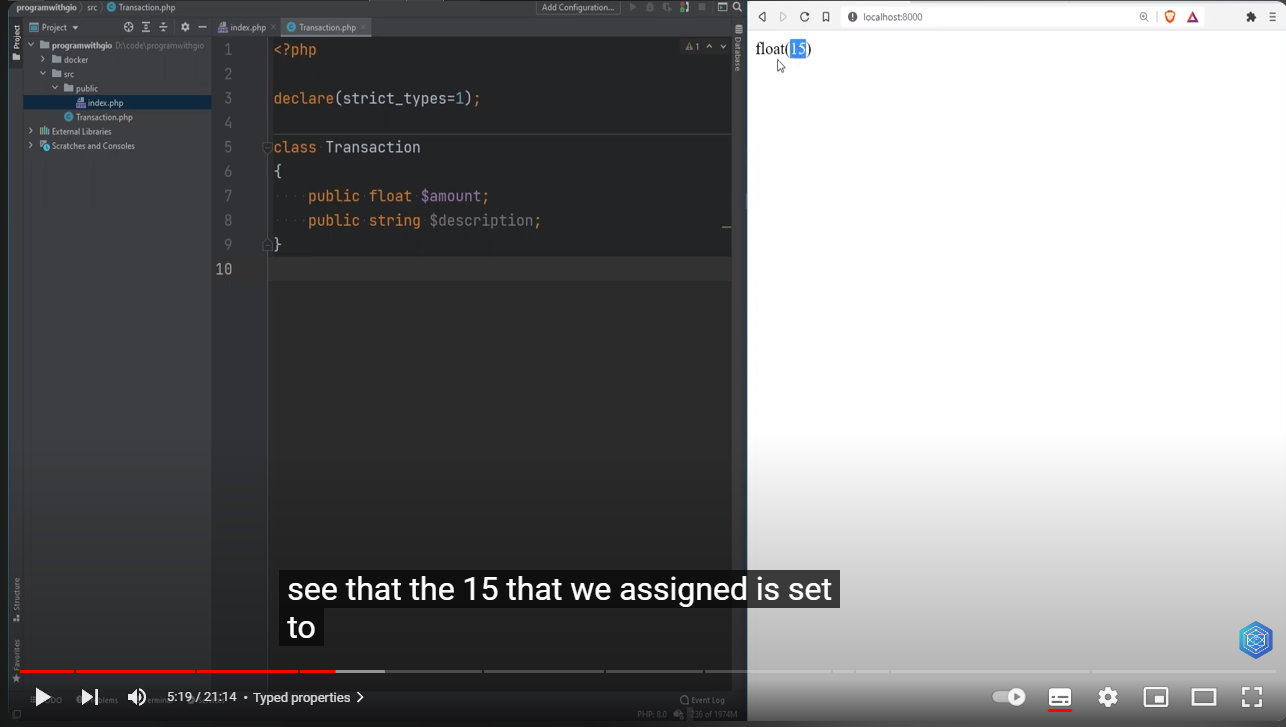
Amount equals to 15. And if we refresh the page, now when we access it, it's set to 15. PHP 7. 4 introduced typed properties, which means that you could set the type of the property in the definition. So right here, as you can see, my ID is underlining these two properties, letting me know that I should type hint them.

* Số tiền bằng 15. Và nếu chúng ta làm mới trang, bây giờ khi chúng ta truy cập trang, nó được đặt thành 15. PHP 7. 4 đã giới thiệu các thuộc tính được nhập, có nghĩa là bạn có thể đặt loại thuộc tính trong định nghĩa. Vì vậy, ngay tại đây, như bạn có thể thấy, ID của tôi đang gạch chân hai thuộc tính này, cho tôi biết rằng tôi nên nhập gợi ý chúng.

And type hinting is recommended, but not required. It is up to you if you want to use it or not. I personally prefer type hinting and strict types. So you will see me type hinting everything and using strict types in most of my code. So I'm going to type in this float, and I'm going to type in this string, and I'm going to declare strictTypes equals one.

* Và gợi ý loại được khuyến khích, nhưng không bắt buộc. Tùy thuộc vào bạn nếu bạn muốn sử dụng nó hay không. Cá nhân tôi thích kiểu gợi ý và kiểu nghiêm ngặt hơn. Vì vậy, bạn sẽ thấy tôi gõ gợi ý mọi thứ và sử dụng các loại nghiêm ngặt trong hầu hết mã của tôi. Vì vậy tôi sẽ gõ vào float này, và tôi sẽ gõ vào chuỗi này, và tôi sẽ khai báo strictTypes bằng một.

Now, if we refresh the page, everything will still work, and we see that the 15 that we assigned is set to float. Now, what do you think would happen if we remove this assignment from here, and try to var dump the property that hasn't been assigned a value yet? Before, when we had no type definition, we were getting null.

* Bây giờ, nếu chúng ta làm mới trang, mọi thứ sẽ vẫn hoạt động và chúng ta thấy rằng số 15 mà chúng ta đã chỉ định được đặt ở trạng thái nổi. Bây giờ, bạn nghĩ điều gì sẽ xảy ra nếu chúng ta xóa nhiệm vụ này khỏi đây và cố gắng loại bỏ thuộc tính chưa được gán giá trị? Trước đây, khi chúng ta không có định nghĩa kiểu, chúng ta nhận được giá trị rỗng.
* 

However, with the type definition, if we refresh the page, you're not going to get null. You're going to get a fatal error stating that the property should not be accessed before initialization. Now, you might say that without type definition, the default value of property should be null, right? Because when we didn't define the type, we were getting null as the default value.

* Tuy nhiên, với định nghĩa kiểu, nếu chúng ta làm mới trang, bạn sẽ không nhận được giá trị rỗng. Bạn sẽ gặp phải một lỗi nghiêm trọng cho biết rằng thuộc tính không được truy cập trước khi khởi tạo. Bây giờ, bạn có thể nói rằng nếu không có định nghĩa kiểu, giá trị mặc định của thuộc tính sẽ là null, phải không? Bởi vì khi chúng tôi không xác định loại, chúng tôi sẽ nhận giá trị null làm giá trị mặc định.

So can't you simply make this nullable type by adding the question mark? So something like this. So if the amount gets assigned the default value null, then you could say that this property is floating data type, but it could also be null. Then do you think this would work? The answer is no. If I refresh the page, we'll still get the fatal error.

* Vì vậy, bạn không thể đơn giản tạo loại có thể rỗng này bằng cách thêm dấu chấm hỏi sao? Vì vậy, một cái gì đó như thế này. Vì vậy, nếu số tiền được gán giá trị mặc định là null thì bạn có thể nói rằng thuộc tính này là kiểu dữ liệu nổi nhưng cũng có thể là null. Vậy bạn có nghĩ việc này sẽ hiệu quả không? Câu trả lời là không. Nếu tôi làm mới trang, chúng tôi vẫn gặp lỗi nghiêm trọng.

That's because PHP is not able to determine whether a property that has a type was actually set or you simply forgot to set the value. This is why the new state was introduced called uninitialized. When type is defined and value is not assigned to a property, it is in uninitialized. state. And let me show you that.

* Đó là bởi vì PHP không thể xác định xem một thuộc tính có loại thực sự đã được đặt hay đơn giản là bạn quên đặt giá trị. Đây là lý do tại sao trạng thái mới được giới thiệu có tên là chưa được khởi tạo. Khi loại được xác định và giá trị không được gán cho một thuộc tính, nó sẽ ở trạng thái chưa được khởi tạo. tình trạng. Và hãy để tôi cho bạn thấy điều đó.

I'm gonna remove this question mark from here and let's var dump the class without accessing the property. If we refresh the page, we see that amount is set to uninitialized float data type and description is set to uninitialized string data type. If we remove the type from the definition here, we refresh, description is set to null, uninitialized.

* Tôi sẽ xóa dấu chấm hỏi này khỏi đây và hãy kết xuất lớp mà không cần truy cập vào thuộc tính. Nếu làm mới trang, chúng ta sẽ thấy số tiền đó được đặt thành kiểu dữ liệu float chưa được khởi tạo và mô tả được đặt thành kiểu dữ liệu chuỗi chưa được khởi tạo. Nếu chúng tôi xóa loại khỏi định nghĩa ở đây, chúng tôi sẽ làm mới, mô tả được đặt thành null, chưa được khởi tạo.

So to solve the issue of accessing properties before initialization is to actually assign some kind of value, either directly in the definition or in something called constructor or a setter that we're going to talk about in a minute. As I just mentioned, you're able to set the default value direct. So you could say equals to 15 for example.

* Vì vậy, để giải quyết vấn đề truy cập các thuộc tính trước khi khởi tạo là thực sự gán một loại giá trị nào đó, trực tiếp trong định nghĩa hoặc trong một thứ gọi là hàm tạo hoặc hàm setter mà chúng ta sẽ nói đến sau đây. Như tôi vừa đề cập, bạn có thể đặt trực tiếp giá trị mặc định. Vì vậy, bạn có thể nói bằng 15 chẳng hạn.

And now if we try to access Amount, this will work. Something to note here is that you're only able to assign constant values as the default values. You cannot have complex expressions here. Like function calls and so on. If you wanted to have the nullable property then you would add the question mark in front of the type definition and you would set the default value to null.

* Và bây giờ nếu chúng ta cố gắng truy cập Số tiền, thao tác này sẽ hoạt động. Điều cần lưu ý ở đây là bạn chỉ có thể gán các giá trị không đổi làm giá trị mặc định. Bạn không thể có các biểu thức phức tạp ở đây. Giống như các cuộc gọi hàm, v.v. Nếu bạn muốn có thuộc tính null thì bạn sẽ thêm dấu hỏi vào trước định nghĩa kiểu và bạn sẽ đặt giá trị mặc định thành null.

In that case it would work. But this kind of assignment doesn't really make sense. for this class, because we're kind of hard coding the amount to value 15. When we're creating multiple transaction objects, they're not all going to have the amount 15, right? They will have different amounts. So this is not the proper way, though there are many use cases where you would want to set the default in the definition.

* Trong trường hợp đó nó sẽ hoạt động. Nhưng loại nhiệm vụ này thực sự không có ý nghĩa. đối với lớp này, bởi vì chúng ta mã hóa số tiền thành giá trị 15. Khi chúng ta tạo nhiều đối tượng giao dịch, không phải tất cả chúng đều có số tiền 15, phải không? Họ sẽ có số tiền khác nhau. Vì vậy, đây không phải là cách thích hợp, mặc dù có nhiều trường hợp sử dụng mà bạn muốn đặt mặc định trong định nghĩa.

But in this case, we'll use something called constructor to initialize our properties. The constructor method is a special function, also known as a magic method, that will be called whenever a new instance So the start with a double underscore, and if I type double underscore, my IDE will autocomplete all the available magic methods that PHP offers, and the one that we need is the construct.

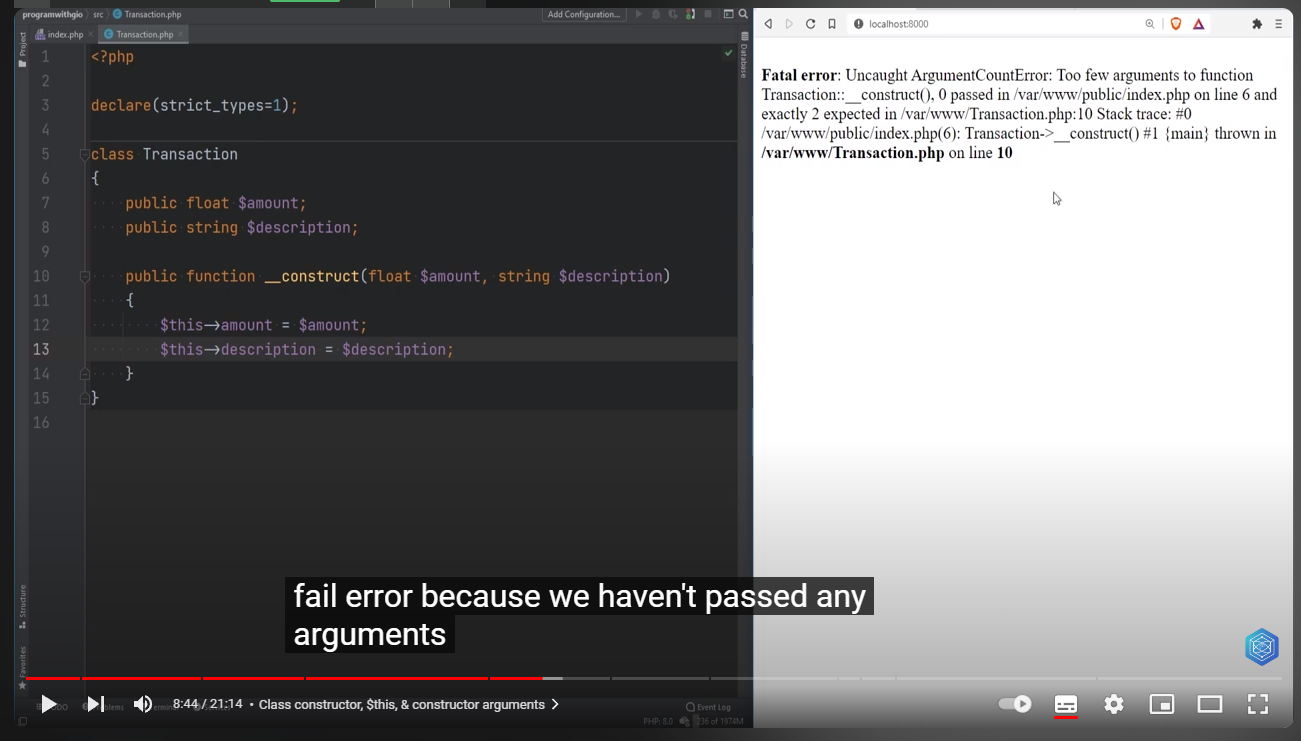
* Nhưng trong trường hợp này, chúng ta sẽ sử dụng một thứ gọi là hàm tạo để khởi tạo các thuộc tính của mình. Phương thức xây dựng là một hàm đặc biệt, còn được gọi là phương thức ma thuật, sẽ được gọi bất cứ khi nào có một phiên bản mới. Vì vậy, hãy bắt đầu bằng dấu gạch dưới kép và nếu tôi gõ dấu gạch dưới kép, IDE của tôi sẽ tự động hoàn thành tất cả các phương thức ma thuật có sẵn mà PHP cung cấp , và thứ chúng ta cần là cấu trúc.

It's just like a regular method, the difference is that this will get called every time a new object is created. You can accept arguments here, so we could accept... float amount and we could accept string description. And now we could assign these values to the properties. To access the properties of the object or the class within the class itself, we need to use a variable called this.

* Nó giống như một phương thức thông thường, điểm khác biệt là phương thức này sẽ được gọi mỗi khi một đối tượng mới được tạo. Bạn có thể chấp nhận các đối số ở đây, vì vậy chúng tôi có thể chấp nhận... số tiền thực và chúng tôi có thể chấp nhận mô tả chuỗi. Và bây giờ chúng ta có thể gán những giá trị này cho các thuộc tính. Để truy cập các thuộc tính của đối tượng hoặc lớp trong chính lớp đó, chúng ta cần sử dụng một biến có tên là this.

And this variable refers to the calling object, or in other words, it refers to the instance from which the method was called. So we can say this amount equals to amount and this description equals to description. Now if we refresh the page we're going to get a fatal error because we haven't passed any arguments to our constructor.

* Và biến này đề cập đến đối tượng đang gọi, hay nói cách khác, nó đề cập đến phiên bản mà phương thức được gọi. Vì vậy, chúng ta có thể nói số tiền này bằng số lượng và mô tả này bằng mô tả. Bây giờ, nếu chúng ta làm mới trang, chúng ta sẽ gặp một lỗi nghiêm trọng vì chúng ta chưa chuyển bất kỳ đối số nào cho hàm tạo của mình.



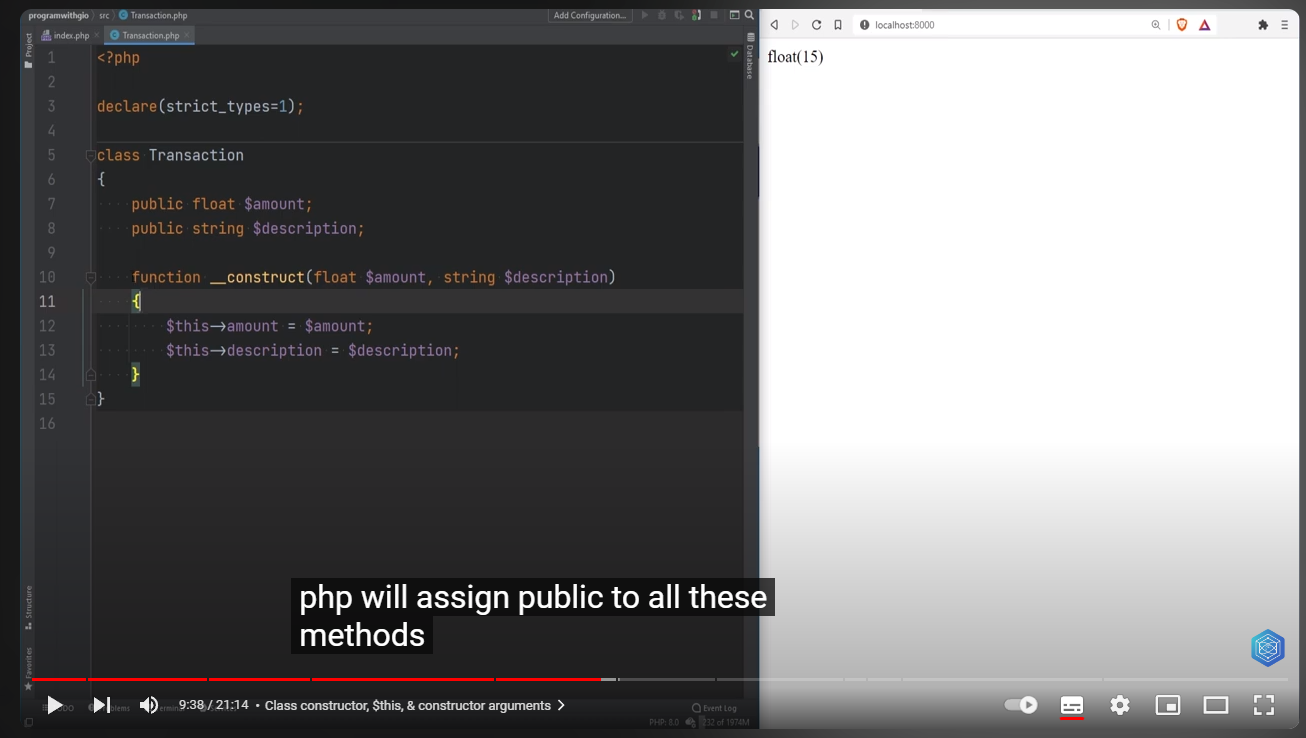
Remember these parentheses that I mentioned before? This is where you would pass the arguments. So we can say 15 and transaction 1 as the description and if we refresh we get float 15. Let's enable the strict types here as well. And refresh the page, everything still works. And as you remember from my previous lesson about the data types and float data type, you are able to pass integer when floating data type is expected, even when strict mode is enabled.

* Bạn có nhớ những dấu ngoặc đơn mà tôi đã đề cập trước đây không? Đây là nơi bạn sẽ vượt qua các đối số. Vì vậy, chúng ta có thể nói 15 và giao dịch 1 làm mô tả và nếu làm mới, chúng ta sẽ nhận được float 15. Hãy kích hoạt các loại nghiêm ngặt ở đây. Và làm mới trang, mọi thứ vẫn hoạt động. Và như bạn nhớ từ bài học trước của tôi về các kiểu dữ liệu và kiểu dữ liệu float, bạn có thể truyền số nguyên khi kiểu dữ liệu float được mong đợi, ngay cả khi chế độ nghiêm ngặt được bật.

However, if I change this to string, now it's going to result in a fatal error. Let's create a method that will add the tax amount to the transaction amount. As mentioned before, methods should also have the access modifiers similar to the properties. As you can see, the constructor method has the access modifier public by default, And this is actually not required, but it is highly recommended to always have the access modifiers on methods.

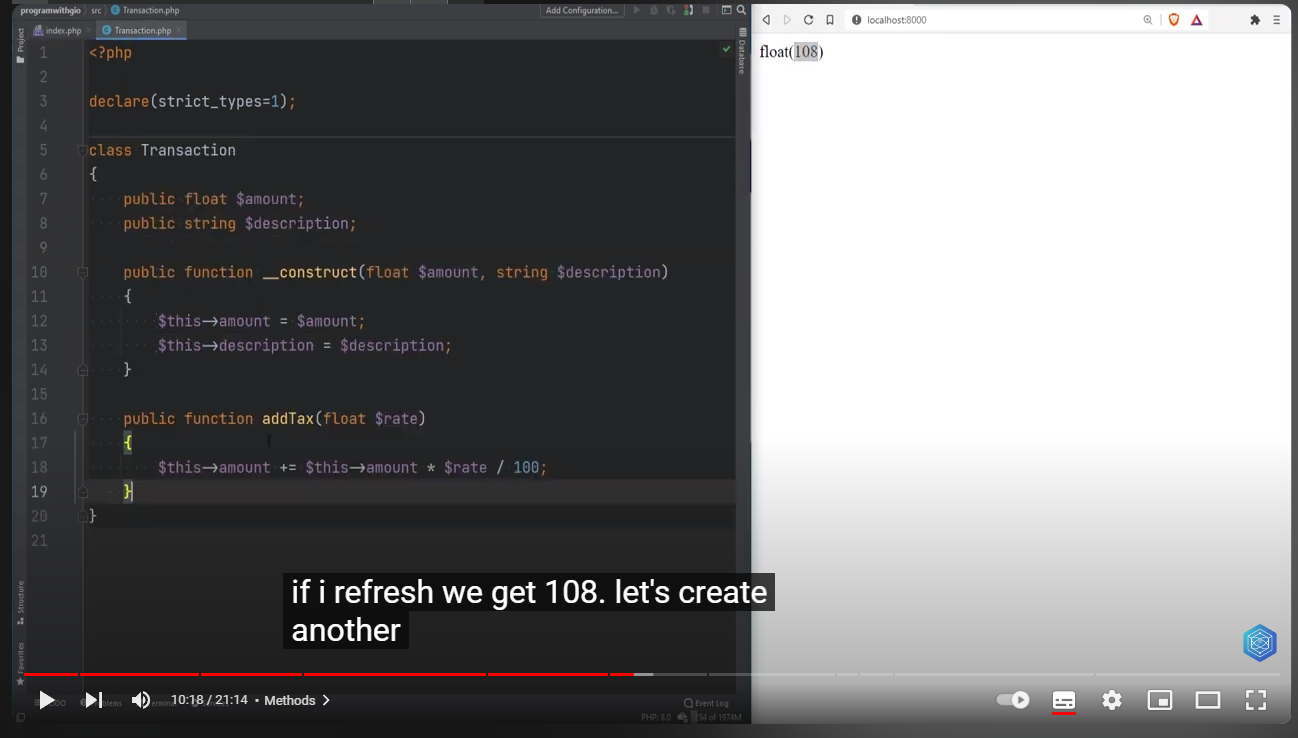
* Tuy nhiên, nếu tôi thay đổi chuỗi này thành chuỗi thì bây giờ nó sẽ gây ra lỗi nghiêm trọng. Hãy tạo một phương thức sẽ thêm số tiền thuế vào số tiền giao dịch. Như đã đề cập trước đó, các phương thức cũng phải có các công cụ sửa đổi truy cập tương tự như các thuộc tính. Như bạn có thể thấy, phương thức xây dựng có công cụ sửa đổi truy cập theo mặc định. Và điều này thực sự không bắt buộc, nhưng chúng tôi khuyên bạn nên luôn có các công cụ sửa đổi truy cập trên các phương thức.

If I remove this, by default, PHP will assign public to all these methods that don't have the access modifier. So if I refresh the page here, it will still work. Though it is highly recommended to always set the access modifier to your methods, to stay consistent, and also it's better for readability. So we'll bring the public back here, and let's add a method called addTax, which will accept the rate as an argument.

* Nếu tôi loại bỏ điều này, theo mặc định, PHP sẽ gán công khai cho tất cả các phương thức không có công cụ sửa đổi truy cập này. Vì vậy nếu tôi làm mới trang ở đây, nó vẫn hoạt động. Mặc dù bạn nên luôn đặt công cụ sửa đổi quyền truy cập cho các phương thức của mình để duy trì tính nhất quán và cũng giúp dễ đọc hơn. Vì vậy, chúng ta sẽ đưa công chúng trở lại đây và hãy thêm một phương thức có tên là addTax, phương thức này sẽ chấp nhận tỷ giá làm đối số.
* 

The way we calculate tax amount is by multiplying the transaction amount by the rate and then dividing by 100 and adding that amount to the original transaction amount. So let's go here and let's add tax of say 8 percent and let's change this to 100 and the final amount should be 108. If I refresh, we get 108.

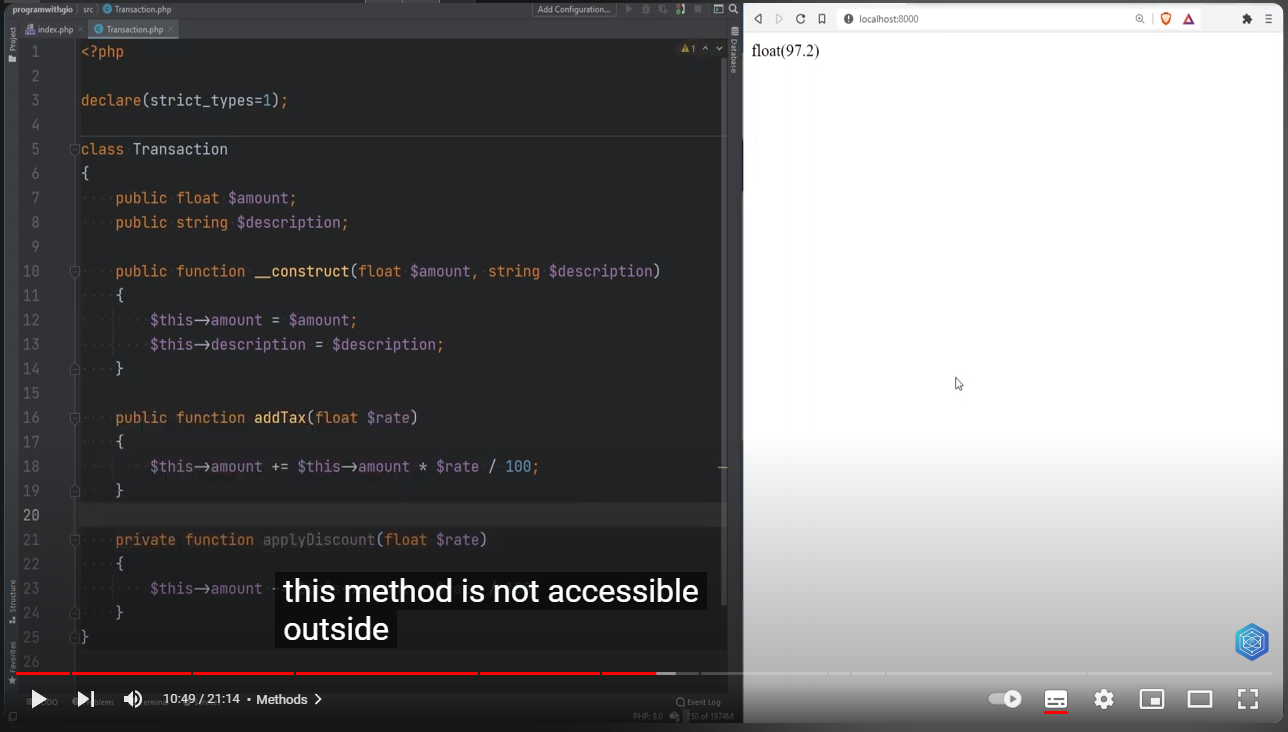
* Cách chúng tôi tính số tiền thuế là nhân số tiền giao dịch với tỷ lệ rồi chia cho 100 và cộng số tiền đó vào số tiền giao dịch ban đầu. Vì vậy, hãy đến đây và cộng thuế, chẳng hạn là 8 phần trăm và thay đổi giá trị này thành 100 và số tiền cuối cùng sẽ là 108. Nếu tôi làm mới, chúng ta sẽ nhận được 108.



Let's create another method here to apply discount to the transaction amount. So let's do public function, applyDiscountFlowRate. And in this case, we're going to subtract from the amount, the discount amount, and we'll calculate it the same way. So amount times rate divided by 100. And let's apply discount of 10%.

* Hãy tạo một phương thức khác ở đây để áp dụng chiết khấu cho số tiền giao dịch. Vì vậy, hãy thực hiện chức năng công khai, applyDiscountFlowRate. Và trong trường hợp này, chúng ta sẽ trừ đi số tiền, số tiền chiết khấu và chúng ta sẽ tính toán theo cách tương tự. Vì vậy, số tiền nhân tỷ lệ chia cho 100. Và hãy áp dụng chiết khấu 10%.

And we refresh page, that's 97. 2, which is correct. So as you can see, this works as expected. If I were to change this to private, this will no longer work, because this method is not accessible outside of the current class. If I refresh, we get that error. So let's change this back to public, and it works.

* Và chúng tôi làm mới trang, đó là 97,2, đúng. Vì vậy, như bạn có thể thấy, điều này hoạt động như mong đợi. Nếu tôi thay đổi nó thành riêng tư, nó sẽ không còn hoạt động nữa vì phương thức này không thể truy cập được bên ngoài lớp hiện tại. Nếu tôi làm mới, chúng tôi sẽ gặp lỗi đó. Vì vậy, hãy thay đổi điều này trở lại thành công khai và nó sẽ hoạt động.
* 

Now let's also change the visibility of amount and description to private, Because it doesn't make sense to have amount publicly available. That can cause some unnecessary bugs where the amount is explicitly changed outside of the class. We should not let anything other than the transaction class itself change the amount property.

* Bây giờ, chúng ta cũng hãy thay đổi mức độ hiển thị của số tiền và mô tả thành riêng tư, Bởi vì việc công khai số tiền sẽ không có ý nghĩa gì. Điều đó có thể gây ra một số lỗi không cần thiết trong đó số tiền được thay đổi rõ ràng bên ngoài lớp học. Chúng ta không nên để bất cứ thứ gì ngoài chính lớp giao dịch thay đổi thuộc tính money.

But instead to access the amount property we could simply create a getter function called getAmount and that returns a float and let's return amount. Now in here we can simply do getAmount and everything will still work. And we'll talk more about. The encapsulation and getters and setters in a separate video.

* Nhưng thay vì truy cập vào thuộc tính money, chúng ta có thể chỉ cần tạo một hàm getter tên là getAmount và hàm đó trả về một số float và hãy trả về số tiền. Bây giờ ở đây chúng ta chỉ cần thực hiện getAmount và mọi thứ vẫn hoạt động. Và chúng ta sẽ nói nhiều hơn về Việc đóng gói và getters và setters trong một video riêng biệt.

So one thing to note here is that I'm calling add tax and apply discount on the same object, right? So we're using this variable multiple times. Let's say we had another six methods that we had to call before getting amount. So in addition to adding tax, applying discount, maybe you want to do some calculation you might.

* Vậy một điều cần lưu ý ở đây là mình đang gọi cộng thuế và áp dụng giảm giá cho cùng một đối tượng phải không? Vì vậy, chúng tôi đang sử dụng biến này nhiều lần. Giả sử chúng ta có sáu phương thức khác mà chúng ta phải gọi trước khi nhận được số tiền. Vì vậy, ngoài việc thêm thuế, áp dụng giảm giá, có thể bạn muốn thực hiện một số phép tính.

To maybe add some additional amount or something like that. Maybe you want to call another six methods here instead of duplicating these lines and calling those functions. That way you could actually chain methods by returning the current object from those methods. So in this case, add tag. apply discount, they don't return anything currently.

* Có thể thêm một số tiền bổ sung hoặc một cái gì đó tương tự. Có thể bạn muốn gọi sáu phương thức khác ở đây thay vì sao chép các dòng này và gọi các hàm đó. Bằng cách đó, bạn thực sự có thể xâu chuỗi các phương thức bằng cách trả về đối tượng hiện tại từ các phương thức đó. Vì vậy, trong trường hợp này, hãy thêm thẻ. áp dụng giảm giá, hiện tại họ không trả lại bất cứ thứ gì.

And we could simply just return this variable, which refers to the calling object, which is the instance of the transaction class, and we could specify transaction as the return type. There is another way to specify this by using the self keyword, but we'll get into that later in the course once we get to the inheritance and stuff.

* Và chúng ta có thể chỉ cần trả về biến này, tham chiếu đến đối tượng đang gọi, là thể hiện của lớp giao dịch và chúng ta có thể chỉ định giao dịch làm kiểu trả về. Có một cách khác để chỉ định điều này bằng cách sử dụng từ khóa self, nhưng chúng ta sẽ đề cập đến vấn đề đó sau trong khóa học khi chúng ta đề cập đến tính kế thừa và nội dung.

So for now, I'm going to use the class name and same goes for the applied discount. We can return the transaction instance and return this. And if we go back here, everything still works. But now we are able to do something like this where methods are chained. If we refresh, it still works. And we could even do something like this.

* Vì vậy, bây giờ, tôi sẽ sử dụng tên lớp và áp dụng giảm giá tương tự. Chúng tôi có thể trả lại phiên bản giao dịch và trả lại phiên bản này. Và nếu chúng ta quay lại đây, mọi thứ vẫn hoạt động. Nhưng bây giờ chúng ta có thể làm điều gì đó giống như thế này khi các phương thức được xâu chuỗi. Nếu chúng tôi làm mới, nó vẫn hoạt động. Và chúng ta thậm chí có thể làm điều gì đó như thế này.

Put this in parentheses, remove this, and this will still work. And for better readability, we can put this on a new line and put this on another line. and put the other six methods that we would have on separate lines and this would still work. Even better we can set this to amount if we don't care to have the reference to that object and simply add get amount here and delete it from here and change this to amount and it will still work.

* Đặt cái này vào ngoặc đơn, xóa cái này đi và cái này vẫn hoạt động. Và để dễ đọc hơn, chúng ta có thể đặt dòng này trên một dòng mới và đặt dòng này trên một dòng khác. và đặt sáu phương thức khác mà chúng ta có trên các dòng riêng biệt và phương pháp này vẫn hoạt động. Thậm chí tốt hơn là chúng ta có thể đặt số tiền này thành số tiền nếu chúng ta không quan tâm đến việc tham chiếu đến đối tượng đó và chỉ cần thêm số tiền nhận được ở đây và xóa nó khỏi đây và thay đổi số tiền này thành số tiền và nó vẫn hoạt động.

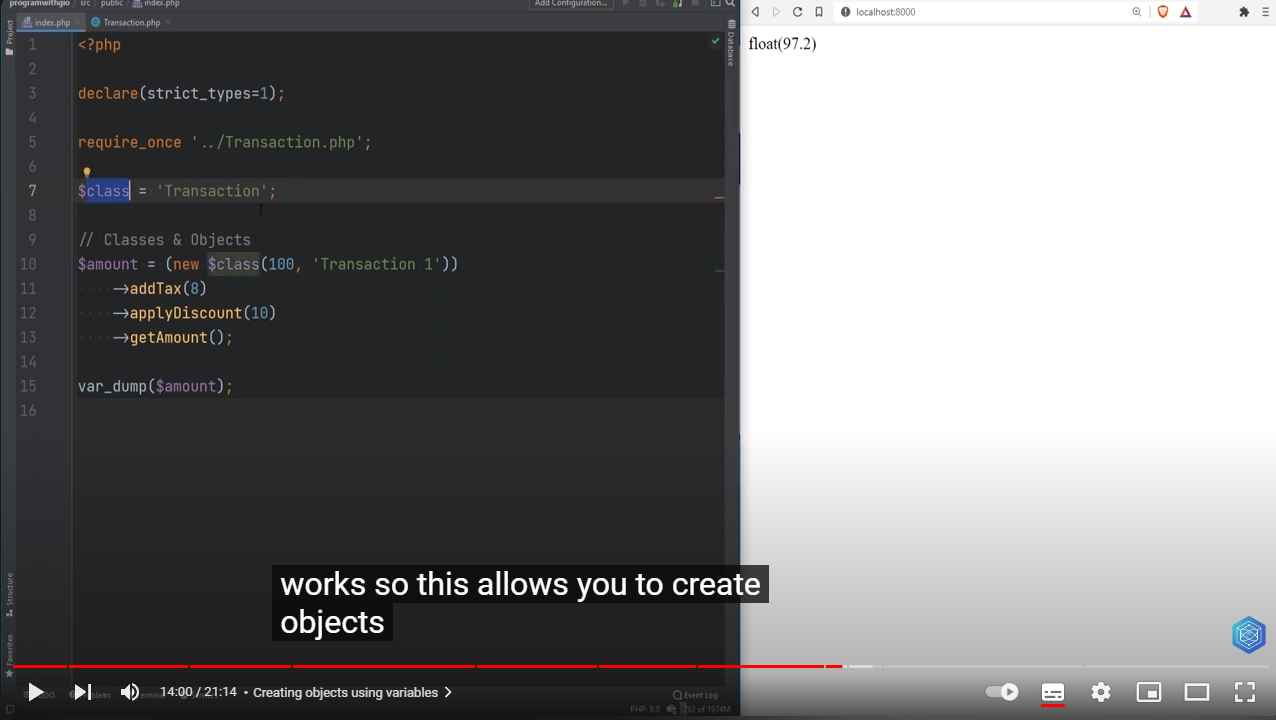
As you can see chaining methods creates this more readable way of building your objects. And we'll talk more about this later in the course, but note that chaining methods will not always make sense. Chaining methods would not make sense. If we had to return a tax here, something like instead of add tax, we had to get tax, then chaining methods would not make sense because we need to return some type of value, not the current instance.

* Như bạn có thể thấy, các phương thức xâu chuỗi tạo ra cách xây dựng đối tượng dễ đọc hơn này. Và chúng ta sẽ nói nhiều hơn về vấn đề này ở phần sau của khóa học, nhưng lưu ý rằng các phương pháp xâu chuỗi không phải lúc nào cũng có ý nghĩa. Phương pháp xâu chuỗi sẽ không có ý nghĩa. Nếu chúng ta phải trả lại thuế ở đây, chẳng hạn như thay vì thêm thuế, chúng ta phải lấy thuế, thì các phương pháp xâu chuỗi sẽ không có ý nghĩa vì chúng ta cần trả về một số loại giá trị, không phải phiên bản hiện tại.

Also don't try to force yourself creating chained method structures in all of your classes. It only makes sense for certain classes where, as I mentioned before, you want to kind of build up the object before getting the final value. Another way you could create objects is by using variables instead of the class name.

* Ngoài ra, đừng cố ép bản thân tạo các cấu trúc phương thức chuỗi trong tất cả các lớp của bạn. Nó chỉ có ý nghĩa đối với một số lớp nhất định, như tôi đã đề cập trước đây, bạn muốn xây dựng đối tượng trước khi nhận được giá trị cuối cùng. Một cách khác bạn có thể tạo đối tượng là sử dụng biến thay vì tên lớp.

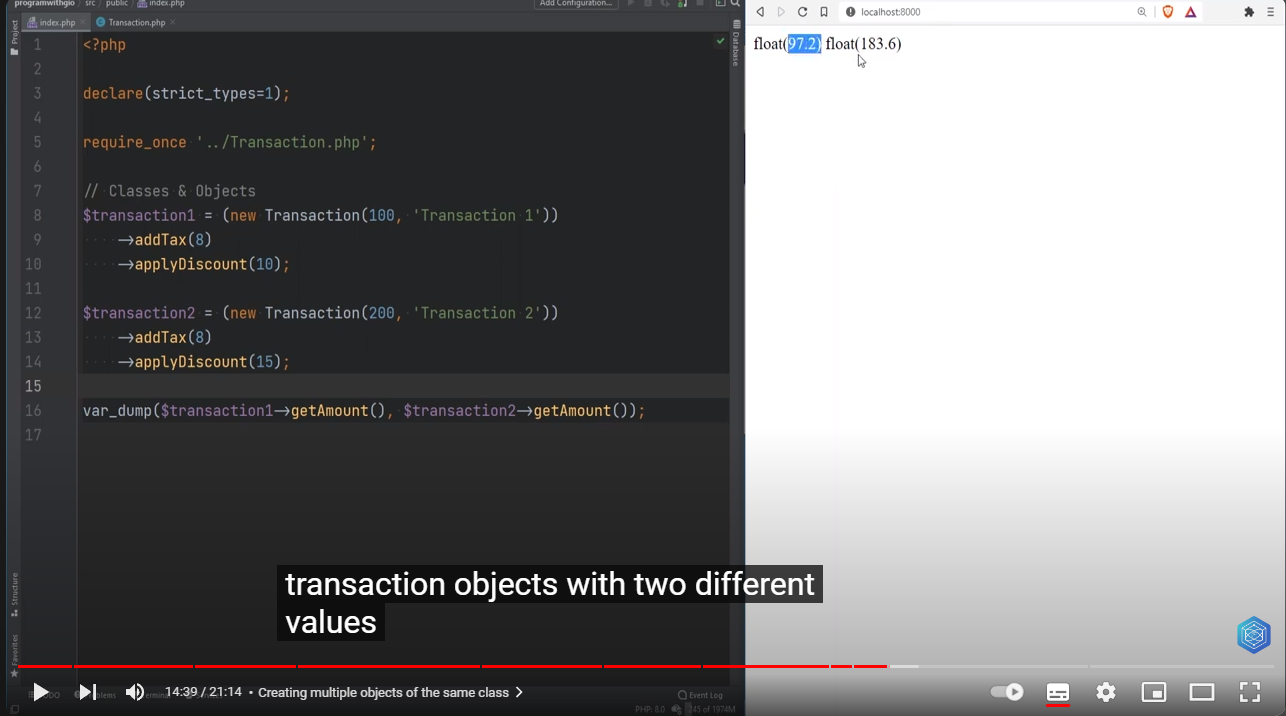
So in here, we could simply call a variable like this, and we could assign the variable class, and you could call this variable whatever you want, and set this to transaction like that, and if you refresh, everything still works, so this allows you to create objects. where the class name is stored in a variable.

* Vì vậy, ở đây, chúng ta có thể chỉ cần gọi một biến như thế này và gán lớp biến và bạn có thể gọi biến này là bất cứ điều gì bạn muốn và đặt biến này thành giao dịch như thế và nếu bạn làm mới, mọi thứ vẫn hoạt động, vậy nên điều này cho phép bạn tạo các đối tượng. nơi tên lớp được lưu trữ trong một biến.
* 

So let's change that back. And as I mentioned in the beginning of the lesson, the class is the blueprint and an object is the instance of that class, right? And you could create multiple objects of the same class. And we could do the same thing here. We can create as many transactions as we want with different values.

* Vậy hãy thay đổi lại điều đó. Và như tôi đã đề cập ở đầu bài, lớp là bản thiết kế và một đối tượng là thể hiện của lớp đó, phải không? Và bạn có thể tạo nhiều đối tượng của cùng một lớp. Và chúng ta có thể làm điều tương tự ở đây. Chúng ta có thể tạo bao nhiêu giao dịch tùy thích với các giá trị khác nhau.

Let's call this transaction 2 for the amount of 200. And we're going to apply 8 percent tax. And maybe 15 percent discount, and we don't need to get amount here. And let's print transaction one amount and transaction two amount. Let's refresh the page. And as you can see, we have two separate transaction objects with two different values.

* Hãy gọi giao dịch này là 2 với số tiền là 200. Và chúng ta sẽ áp dụng mức thuế 8%. Và có thể giảm giá 15 phần trăm, và chúng ta không cần lấy số tiền ở đây. Và hãy in giao dịch một số tiền và giao dịch hai số tiền. Hãy làm mới trang. Và như bạn có thể thấy, chúng ta có hai đối tượng giao dịch riêng biệt với hai giá trị khác nhau.
* 

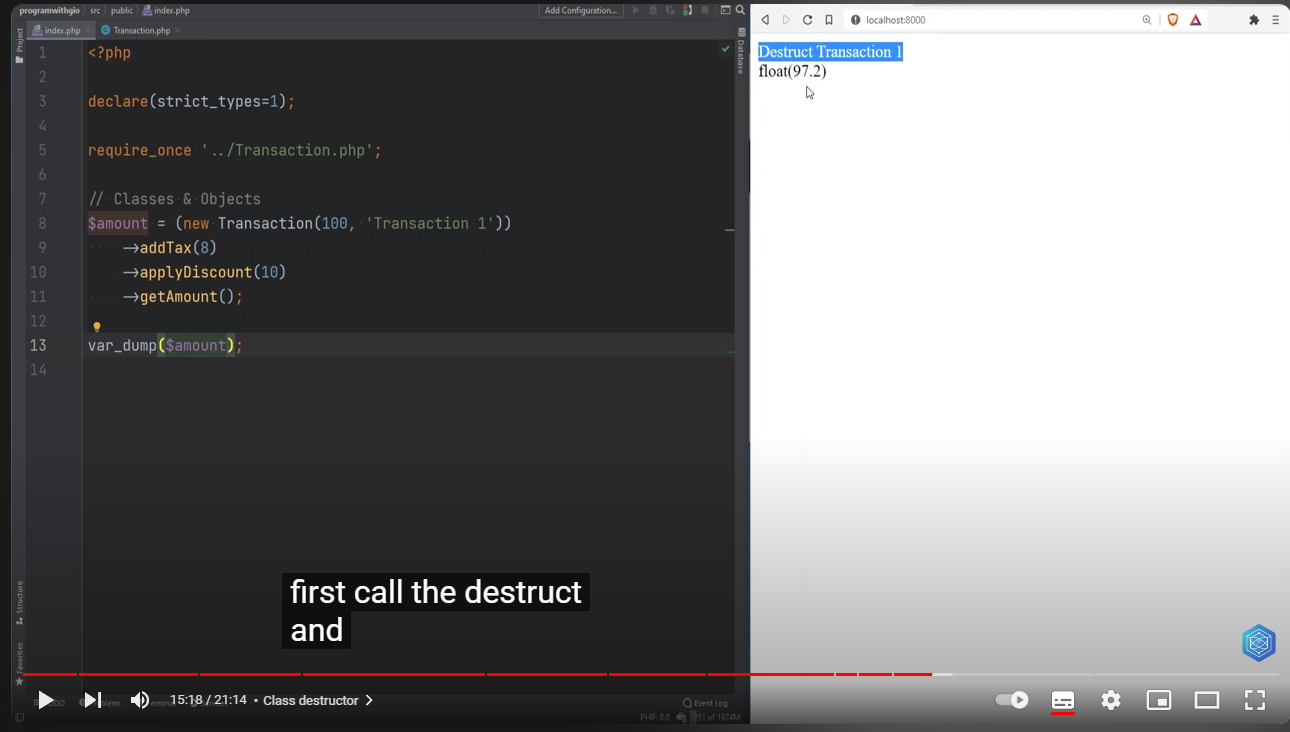
All right, let's move on to destructors. Destructor, like constructor, is also a magic method, and it needs to be defined with two underscores in the beginning. So let's type two underscores and as mentioned before we see all the available magic methods here. The destruct is the one that we need and the destruct method is called whenever there is no more reference available to the object or when the object is destroyed.

* Được rồi, hãy chuyển sang phần hủy diệt. Hàm hủy, giống như hàm tạo, cũng là một phương thức kỳ diệu và nó cần được xác định bằng hai dấu gạch dưới ở đầu. Vì vậy, hãy gõ hai dấu gạch dưới và như đã đề cập trước khi chúng ta thấy tất cả các phương pháp ma thuật có sẵn ở đây. Hàm hủy là thứ chúng ta cần và phương thức hủy được gọi bất cứ khi nào không còn tham chiếu nào cho đối tượng hoặc khi đối tượng bị hủy.

So let's print destruct and the description of the transaction with the breakline and let's keep a single transaction here and let's do get amount so that I can show you two cases. So let's do amount and let's print. amount we refresh and as you can see it will first call the destruct and then print the amount.

* Vì vậy, hãy in lệnh hủy và mô tả giao dịch có đường ngắt và hãy giữ một giao dịch duy nhất ở đây và lấy số tiền để tôi có thể chỉ cho bạn hai trường hợp. Vì vậy, hãy tính số lượng và in. số tiền chúng tôi làm mới và như bạn có thể thấy, trước tiên nó sẽ gọi hàm hủy và sau đó in số tiền.

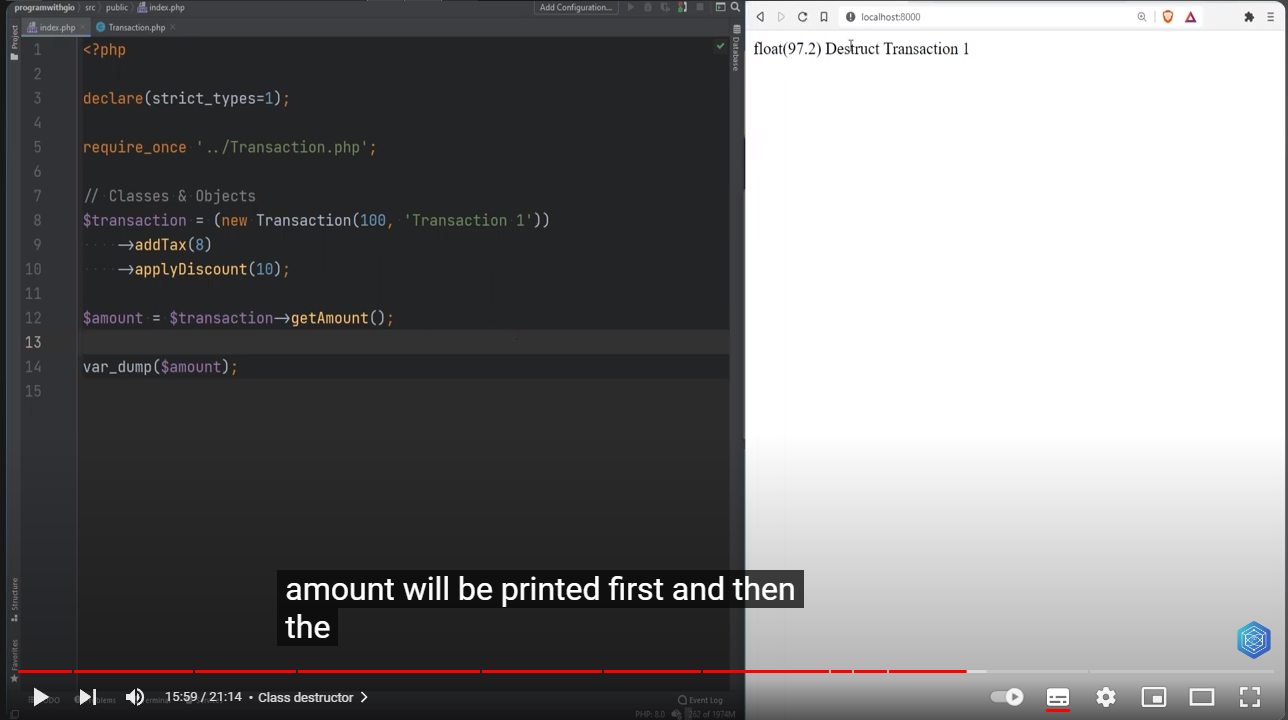
What this means is that there is actually no reference available to this transaction aside from here and therefore it's calling the destruct and it's printing destruct transaction one and then we're printing the amount. If we change this to a transaction object and remove the getAmount. Now we still have the reference available to it.

* Điều này có nghĩa là thực sự không có tài liệu tham khảo nào có sẵn cho giao dịch này ngoài đây và do đó, nó gọi lệnh hủy và nó đang in một giao dịch hủy và sau đó chúng tôi in số tiền. Nếu chúng ta thay đổi đối tượng này thành đối tượng giao dịch và xóa getAmount. Bây giờ chúng tôi vẫn có sẵn tài liệu tham khảo cho nó.
* 

So if we try to do getAmount here, it's going to print the amount first and then call the distract method. Now, what if we did something like this? Amount equals transaction getAmount, and we printed amount. What do you think will happen? Do you think the destruct will be called first or the amount will be printed first?

* Vì vậy, nếu chúng ta cố gắng thực hiện getAmount ở đây, nó sẽ in số tiền trước rồi gọi phương thức phân tâm. Bây giờ, nếu chúng ta làm điều gì đó như thế này thì sao? Số tiền bằng với giao dịch getAmount và chúng tôi đã in số tiền. Bạn nghĩ điều gì sẽ xảy ra? Bạn có nghĩ rằng lệnh hủy sẽ được gọi trước hay số tiền sẽ được in trước?

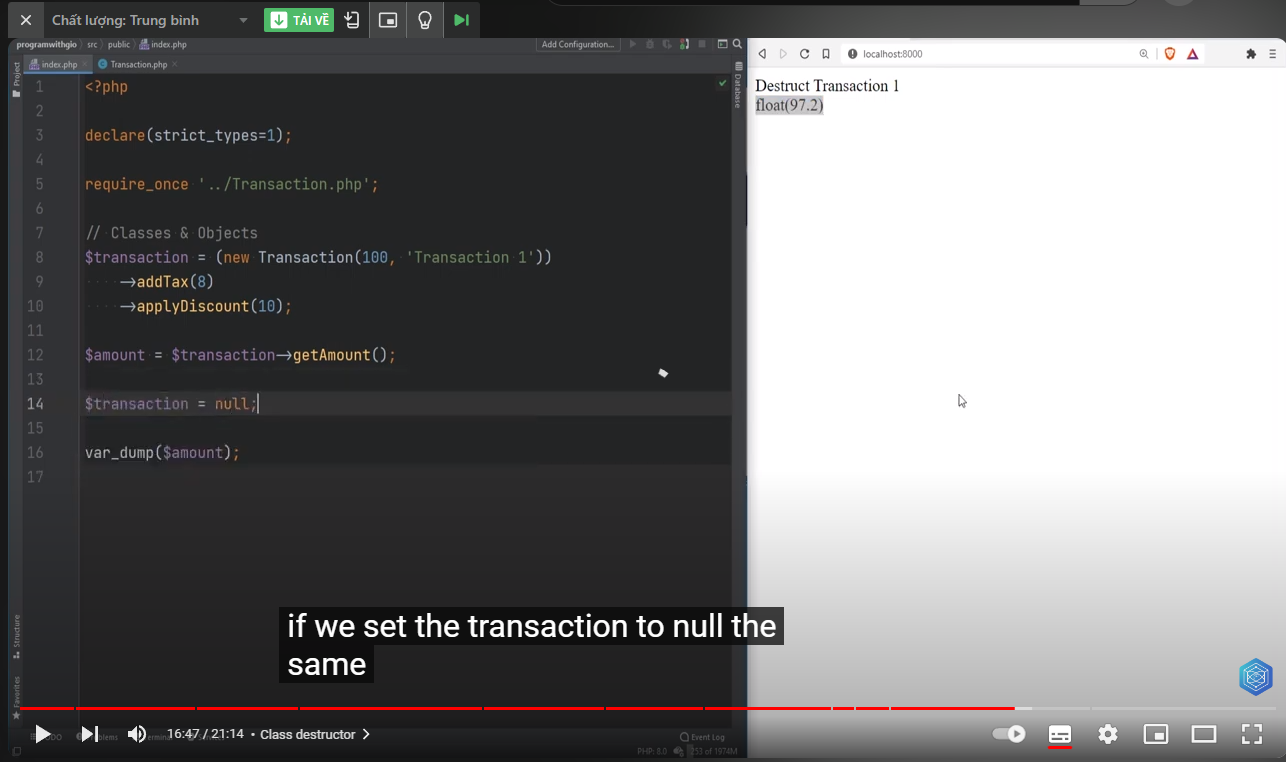
The answer is that amount will be printed first and then the destructor will call. The reason for that is because we still have the reference of that object available. Because even though we are assigning the value here, this reference still is available until the end of the script. When the script ends, that's when the destructor is called.

* Câu trả lời là số tiền đó sẽ được in trước và sau đó hàm hủy sẽ gọi. Lý do là vì chúng ta vẫn có sẵn tham chiếu của đối tượng đó. Bởi vì mặc dù chúng ta đang gán giá trị ở đây nhưng tham chiếu này vẫn có sẵn cho đến cuối tập lệnh. Khi tập lệnh kết thúc, đó là lúc hàm hủy được gọi.
* 

And that is why it first prints the amount and then it calls the destruct. While in previously when we called getAmount here, there was actually no reference available anywhere to that transaction object. That's why the destruct was called first and then the amount was printed. You're able to destroy your object by calling onset function or setting the value of that object to null.

* Và đó là lý do tại sao đầu tiên nó in số tiền và sau đó gọi hàm hủy. Mặc dù trước đây khi chúng tôi gọi getAmount ở đây, thực tế không có tài liệu tham khảo nào có sẵn ở bất kỳ đâu cho đối tượng giao dịch đó. Đó là lý do tại sao lệnh hủy được gọi đầu tiên và sau đó số lượng được in. Bạn có thể hủy đối tượng của mình bằng cách gọi hàm khởi động hoặc đặt giá trị của đối tượng đó thành null.

So for example, if we set onset transaction here, this will call the destructor. So destructor will be called first and then the amount. If we set the transaction to null, the same thing will happen. It will call the destructor first and then the amount. Destructors can be used to perform some sort of cleanup or release locks or close open resources or database connections and so on.

* Vì vậy, ví dụ: nếu chúng ta đặt giao dịch khởi động ở đây, thì hàm này sẽ gọi hàm hủy. Vì vậy, hàm hủy sẽ được gọi đầu tiên và sau đó là số tiền. Nếu chúng ta đặt giao dịch thành null, điều tương tự sẽ xảy ra. Nó sẽ gọi hàm hủy trước và sau đó là số tiền. Hàm hủy có thể được sử dụng để thực hiện một số loại dọn dẹp hoặc giải phóng khóa hoặc đóng các tài nguyên mở hoặc kết nối cơ sở dữ liệu, v.v.
* 

I personally don't use destructors a lot, only when I really need to and have the actual use case for it. It is recommended that if you have a resource or a database connection open, to close it as soon as you're done working with them and as soon as you no longer need them. Just in case you have a long run in script, because if you have a long run in script, and you don't close the connection as soon as you're done using it, it's going to keep that connection open until the script finishes running.

* Cá nhân tôi không sử dụng hàm hủy nhiều, chỉ khi tôi thực sự cần và có trường hợp sử dụng thực tế cho nó. Chúng tôi khuyên rằng nếu bạn có một tài nguyên hoặc kết nối cơ sở dữ liệu đang mở, hãy đóng nó ngay khi bạn làm việc xong với chúng và ngay khi bạn không còn cần đến chúng nữa. Đề phòng trường hợp bạn có tập lệnh dài, vì nếu bạn có tập lệnh dài và bạn không đóng kết nối ngay sau khi sử dụng xong, nó sẽ giữ kết nối đó mở cho đến khi tập lệnh kết thúc đang chạy.

So unless you have a good use case for destructors, I would try to stay away from them and close your connections as soon as you're done using them. A couple of things to know about destructors is that if you use exit statement to terminate your script, it will still call the destructor. So if I refresh the page here, it's going to print destruct transaction 1, which means the destructor ran, but it's not going to Print amount because we exited here.

* Vì vậy, trừ khi bạn có trường hợp sử dụng tốt cho hàm hủy, tôi sẽ cố gắng tránh xa chúng và đóng kết nối của bạn ngay sau khi bạn sử dụng xong chúng. Một số điều cần biết về hàm hủy là nếu bạn sử dụng câu lệnh thoát để chấm dứt tập lệnh của mình, nó vẫn sẽ gọi hàm hủy. Vì vậy, nếu tôi làm mới trang ở đây, nó sẽ in giao dịch hủy 1, nghĩa là hàm hủy đã chạy, nhưng nó sẽ không in số tiền vì chúng ta đã thoát ở đây.

Another thing to know about disruptors is that if you put the exit statement inside the destruct method, whenever this specific destruct method runs, no other destruct methods will run on other classes, which can cause some issues if you're performing some actions and relying on those methods to run that are within the disruptors of other classes.

* Một điều khác cần biết về trình phá vỡ là nếu bạn đặt câu lệnh thoát bên trong phương thức hủy, bất cứ khi nào phương thức hủy cụ thể này chạy, sẽ không có phương thức hủy nào khác chạy trên các lớp khác, điều này có thể gây ra một số vấn đề nếu bạn đang thực hiện một số hành động và dựa vào về các phương thức chạy đó nằm trong phạm vi gây rối của các lớp khác.

Before we finish this lesson, I want to talk a little bit about PHP's std class that can be used to create generic objects and also some functions return instances of std class. So for example, let's say that we have a JSON string, something like this, and then we JSON decode this into an array, so we use a built in function called JSON decode.

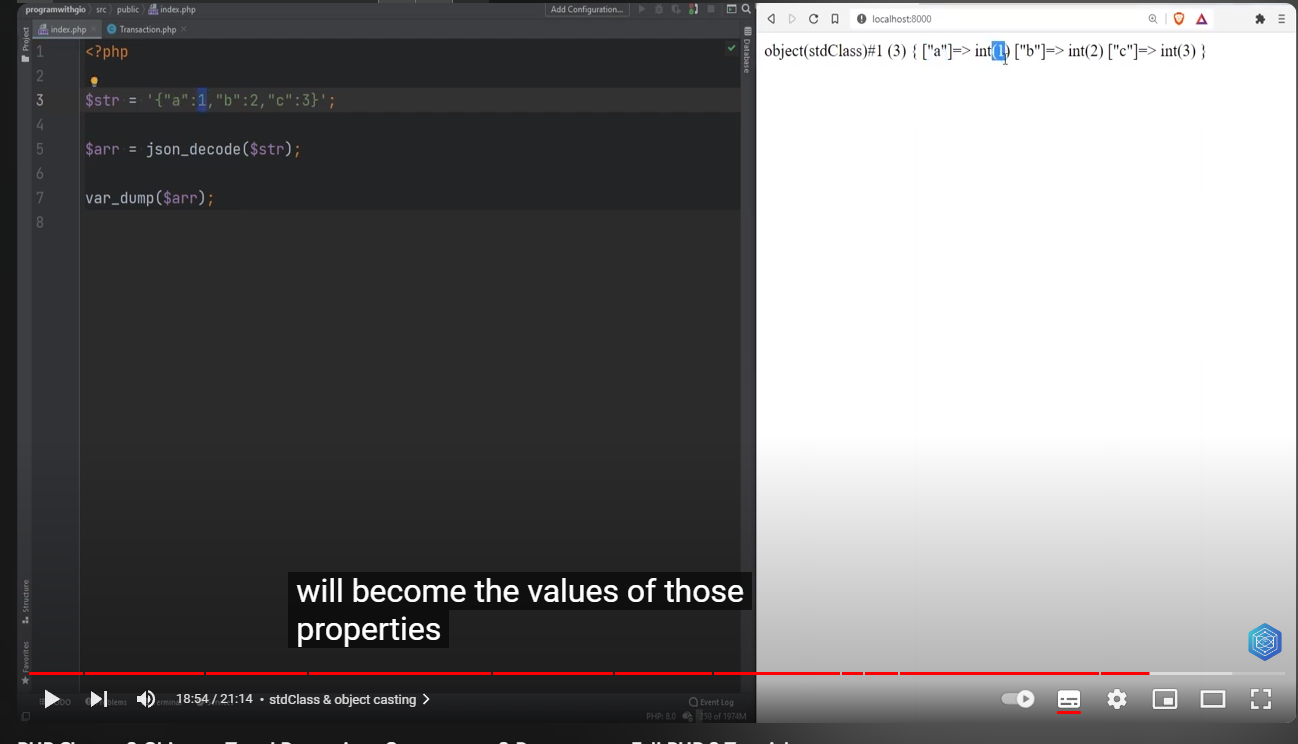
* Trước khi chúng ta kết thúc bài học này, tôi muốn nói một chút về lớp std của PHP có thể được sử dụng để tạo các đối tượng chung và một số hàm trả về các thể hiện của lớp std. Vì vậy, ví dụ: giả sử chúng ta có một chuỗi JSON, đại loại như thế này, sau đó chúng ta giải mã JSON thành một mảng, vì vậy chúng ta sử dụng một hàm dựng sẵn có tên là giải mã JSON.

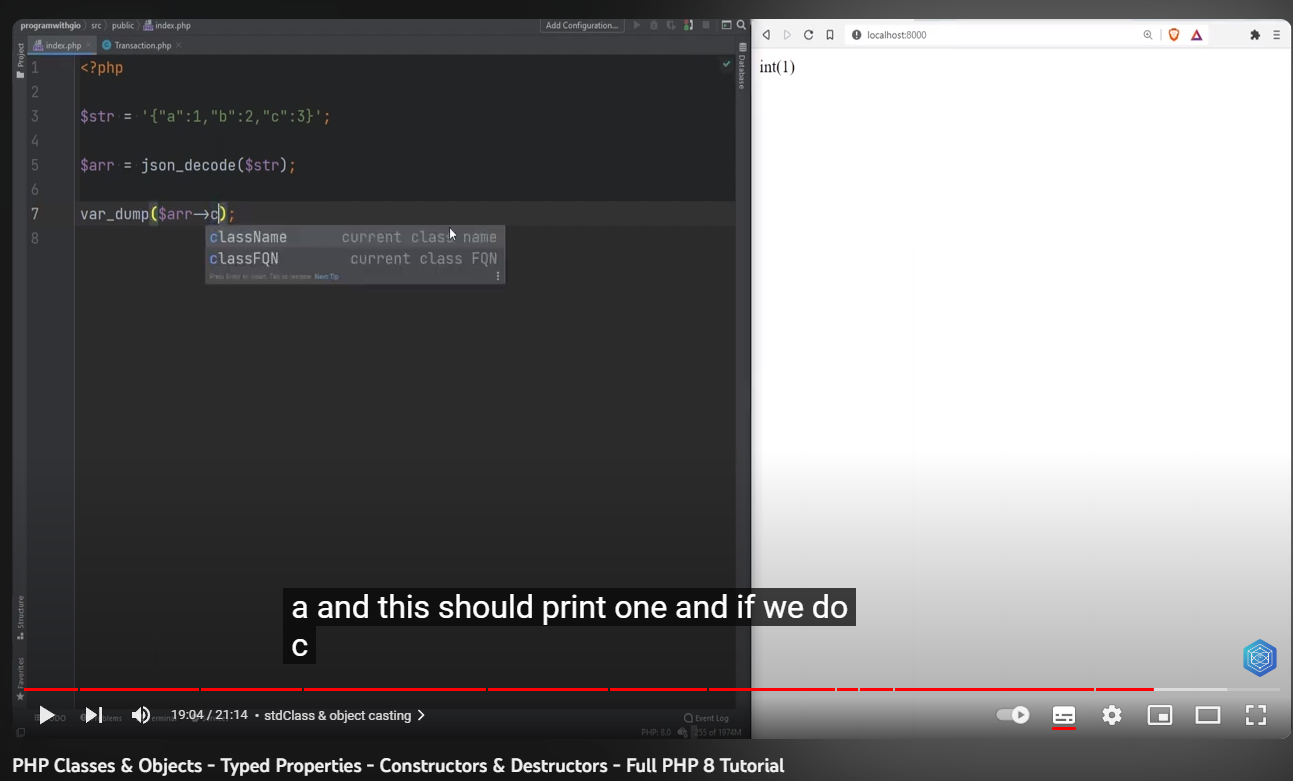
We pass the string and we pass the second parameter as true to return it as an associative array. If we then var dump array, we refresh the page, we get the array. However, if you don't pass the second parameter here and just use JSON decode, we refresh, we get an object which is an instance of std class.

* Chúng ta truyền chuỗi và truyền tham số thứ hai là true để trả về chuỗi đó dưới dạng một mảng kết hợp. Sau đó, nếu chúng ta sử dụng mảng kết xuất var, chúng ta làm mới trang, chúng ta sẽ nhận được mảng. Tuy nhiên, nếu bạn không chuyển tham số thứ hai ở đây và chỉ sử dụng giải mã JSON, chúng tôi sẽ làm mới, chúng tôi sẽ nhận được một đối tượng là một thể hiện của lớp std.

And the keys of the array or the keys of the JSON will become the properties of the class and the values will become the values of those properties. So now you're able to access. those properties using the object operator. So we would use object operator and then A, and this should print one. And if we do C, this prints three and so on.

* Và các khóa của mảng hoặc các khóa của JSON sẽ trở thành thuộc tính của lớp và các giá trị sẽ trở thành giá trị của các thuộc tính đó. Vì vậy, bây giờ bạn có thể truy cập. các thuộc tính đó bằng cách sử dụng toán tử đối tượng. Vì vậy, chúng ta sẽ sử dụng toán tử đối tượng rồi đến A, và cái này sẽ in ra một cái. Và nếu chúng ta làm C, nó sẽ in ra ba, v.v.





You can create custom object by using the STD class itself. So you could do new STD class and then set the properties the way you would set public properties on your own custom class. So you would do object A equals one object. b equals two, and then we carry var dump object, and refresh, we see that it has a and b.

* Bạn có thể tạo đối tượng tùy chỉnh bằng cách sử dụng chính lớp STD. Vì vậy, bạn có thể thực hiện lớp STD mới và sau đó đặt thuộc tính theo cách bạn đặt thuộc tính công khai trên lớp tùy chỉnh của riêng bạn. Vì vậy, bạn sẽ làm đối tượng A bằng một đối tượng. b bằng hai, sau đó chúng ta mang đối tượng var dump và làm mới, chúng ta thấy rằng nó có a và b.

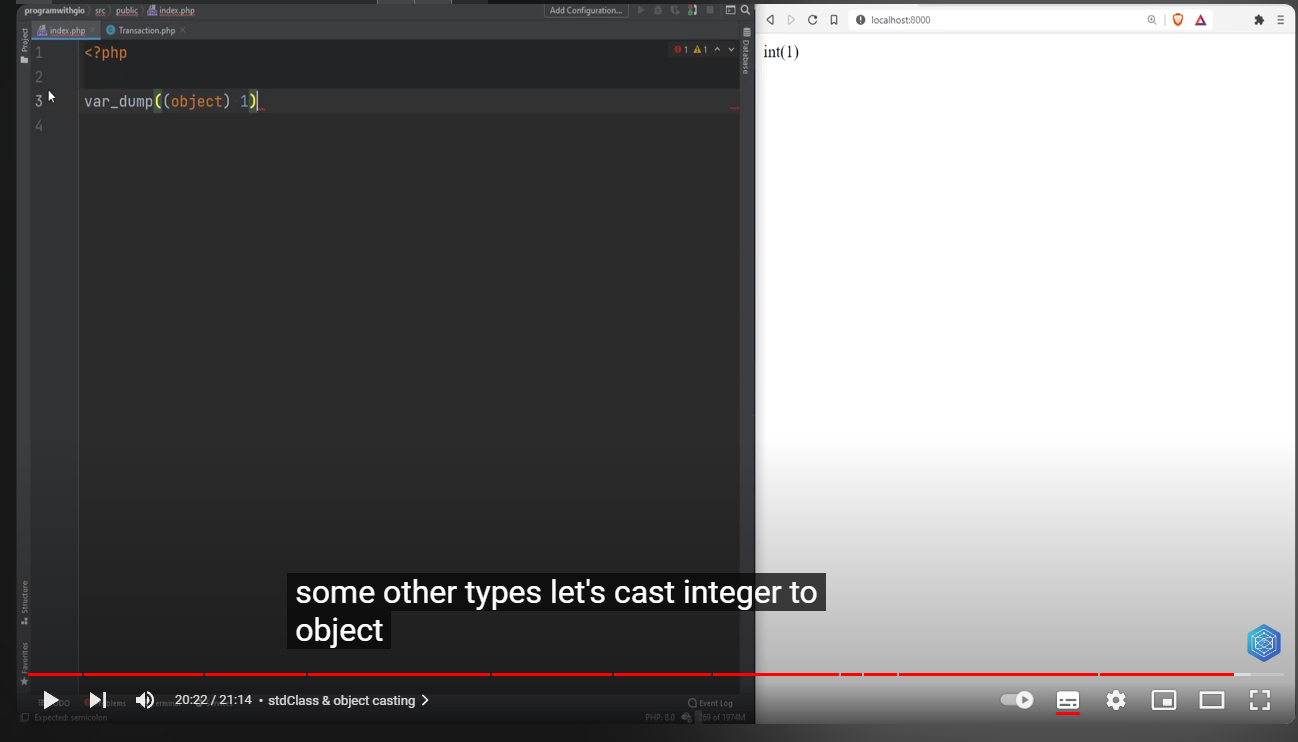
Now let's see how the casting works. So let's say we have an array that just contains one, two, and three as elements, and this is numerically indexed array. So this is at the zeroth index, this is at one, and this is at two. We can cast things to object by using the regular casting and just specifying the object as the type.

* Bây giờ hãy xem quá trình casting diễn ra như thế nào. Vì vậy, giả sử chúng ta có một mảng chỉ chứa một, hai và ba phần tử và đây là mảng được lập chỉ mục bằng số. Vậy đây là chỉ số thứ 0, đây là số 1, và đây là số 2. Chúng ta có thể chuyển các thứ thành đối tượng bằng cách sử dụng tính năng truyền thông thường và chỉ xác định đối tượng làm loại.

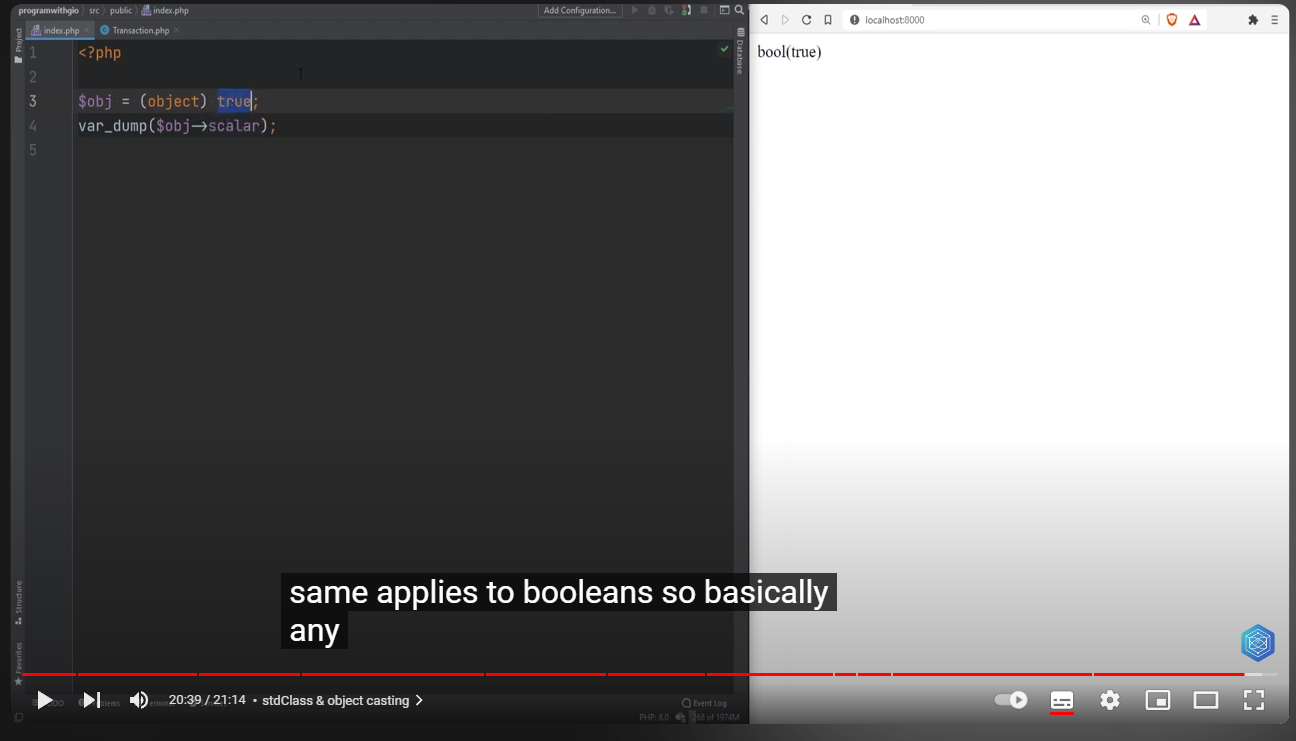
So let's do var dump object array, And this is going to convert array into object. And if we refresh, we get the keys as the properties again, which is 0, 1 and 2, and we're able to access them. So let's assign this to a variable object. And then here we can do object and we would not do one like this. It would not work.

* Vì vậy, hãy thực hiện var dump mảng đối tượng. Và điều này sẽ chuyển đổi mảng thành đối tượng. Và nếu chúng ta làm mới, chúng ta sẽ lấy lại các khóa làm thuộc tính, đó là 0, 1 và 2 và chúng ta có thể truy cập chúng. Vì vậy, hãy gán cái này cho một đối tượng biến. Và ở đây chúng ta có thể làm đối tượng và chúng ta sẽ không làm như thế này. Nó sẽ không hoạt động.

You have to surround this with brackets. And now it will work. If we access the index one, that is the second element. So it should be two and it is two. And if we access the first element, it will be one. Now let's cast some other types. Let's cast integer to object. If we refresh, we're going to get an object with the scalar as the property and the value, whatever the value we're casting.

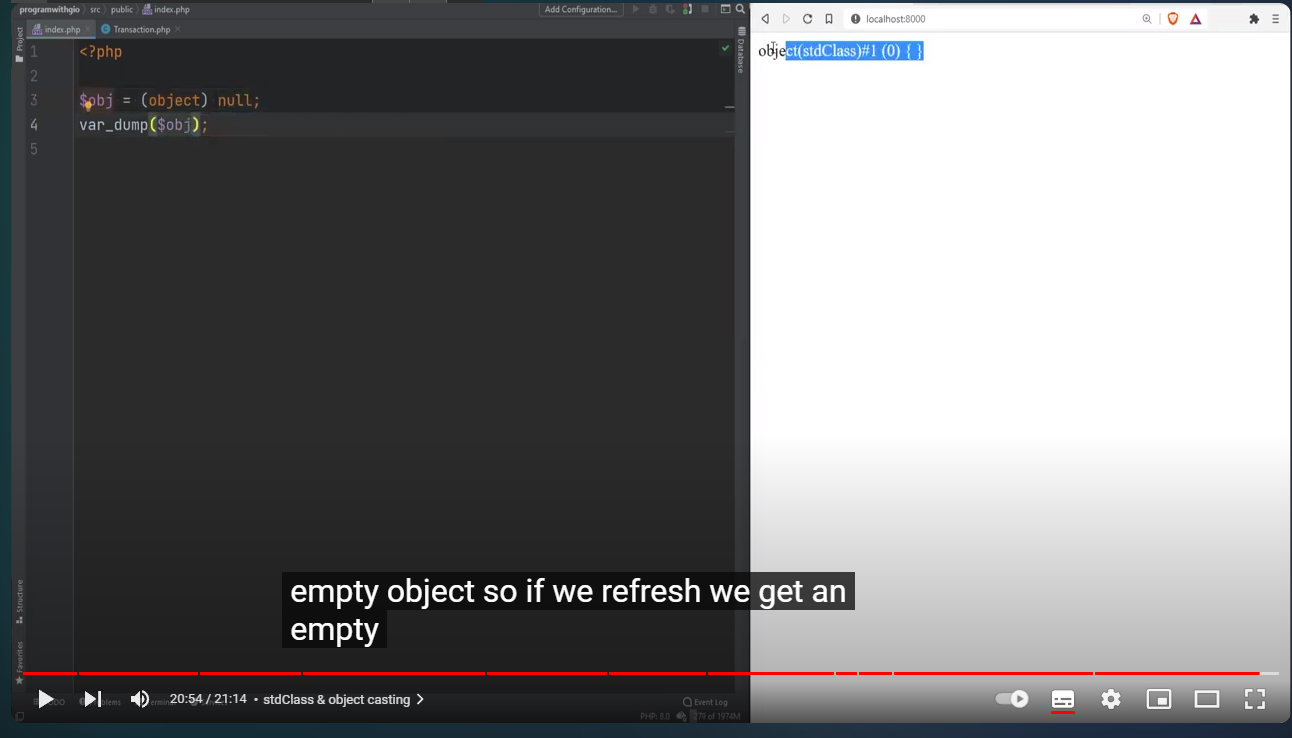
* Bạn phải bao quanh cái này bằng dấu ngoặc. Và bây giờ nó sẽ hoạt động. Nếu chúng ta truy cập vào chỉ mục một thì đó là phần tử thứ hai. Vì vậy, nó phải là hai và nó là hai. Và nếu chúng ta truy cập phần tử đầu tiên, nó sẽ là một phần tử. Bây giờ hãy đúc một số loại khác. Hãy truyền số nguyên cho đối tượng. Nếu làm mới, chúng ta sẽ nhận được một đối tượng có thuộc tính và giá trị là vô hướng, bất kể giá trị mà chúng ta đang truyền.
* 

And if we were to access that, and if we set this to object, we can access that by using scalar. property and this will work. Same applies to booleans. So basically any scalar value will be converted to object this way and you will have the property scalar with that value. That includes integer, floats, strings and booleans.

* Và nếu chúng ta truy cập vào đó và nếu chúng ta đặt cái này thành đối tượng, chúng ta có thể truy cập nó bằng cách sử dụng vô hướng. tài sản và điều này sẽ hoạt động. Áp dụng tương tự cho booleans. Vì vậy, về cơ bản, bất kỳ giá trị vô hướng nào cũng sẽ được chuyển đổi thành đối tượng theo cách này và bạn sẽ có thuộc tính vô hướng với giá trị đó. Điều đó bao gồm số nguyên, số float, chuỗi và booleans.
* 

The null value is cast to object as an empty object. So if we refresh we get an empty object without any properties. So this is it for this video. Thank you so much for watching. Please give this video a thumbs up, share and subscribe if you enjoyed it and I'll see you on the next video where we're going to talk about some new PHP 8 features that are related to the things that we talked today such as constructor property promotion and null safe operator.

* Giá trị null được truyền tới đối tượng dưới dạng đối tượng trống. Vì vậy, nếu chúng ta làm mới, chúng ta sẽ nhận được một đối tượng trống mà không có bất kỳ thuộc tính nào. Vì vậy, đây là nó cho video này. Cảm ơn bạn rất nhiều vì đã xem. Vui lòng thích video này, chia sẻ và đăng ký nếu bạn thích nó và tôi sẽ gặp bạn ở video tiếp theo nơi chúng ta sẽ nói về một số tính năng PHP 8 mới có liên quan đến những điều chúng ta đã nói hôm nay, chẳng hạn như khuyến mãi thuộc tính của nhà xây dựng và nhà điều hành an toàn vô giá trị.



**Video 2.3: Contrucstor Prooerty Promotion – Nullsafe Operator**

Constructor property promotion is a feature that was added in PHP 80 .

Quảng cáo thuộc tính của nhà xây dựng là một tính năng đã được thêm vào PHP 80.

It allows you to get rid of the boilerplate code where you define properties and then assign them to the arguments that are passed in your constructor .

Nó cho phép bạn loại bỏ mã soạn sẵn trong đó bạn xác định các thuộc tính và sau đó gán chúng cho các đối số được truyền trong hàm tạo của bạn.

Basically it lets you combine class properties, constructor and assignment into one shorter syntax .

Về cơ bản, nó cho phép bạn kết hợp các thuộc tính lớp, hàm tạo và phép gán thành một cú pháp ngắn hơn.

Let's take this example from the last lesson where we have the transaction class which has two properties amount and description, and they get assigned to the arguments that are passed in the constructor .

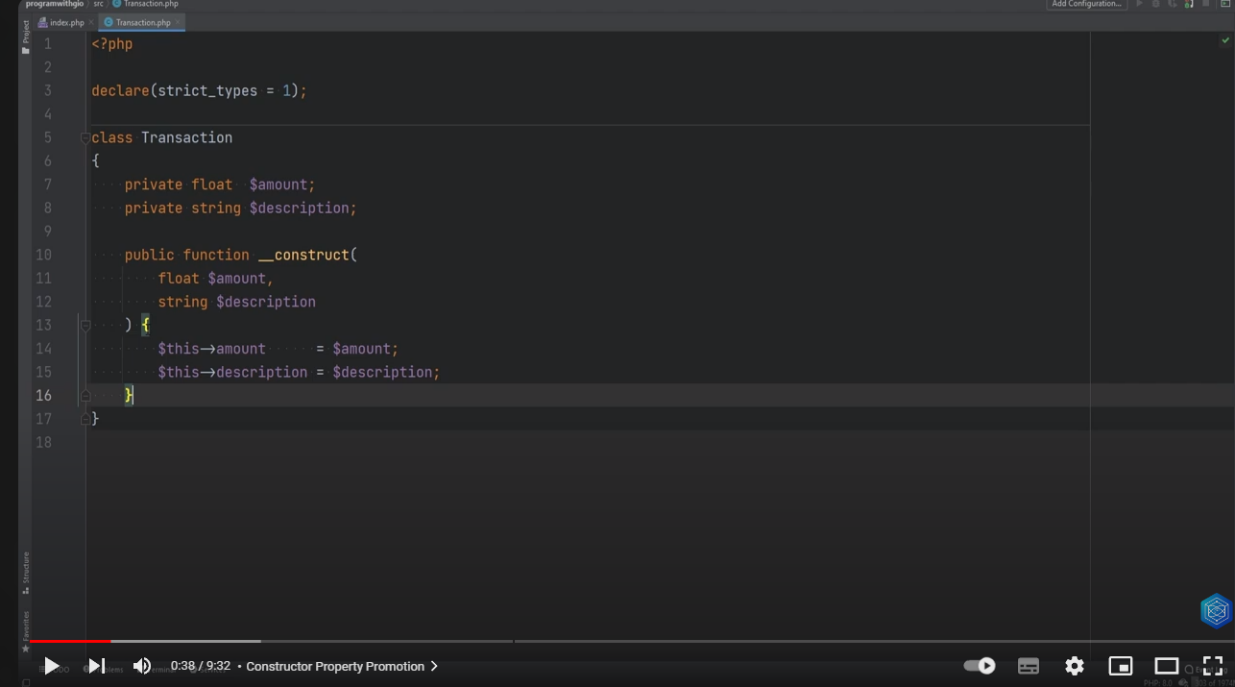
Hãy lấy ví dụ này từ bài học trước, trong đó chúng ta có lớp giao dịch có hai thuộc tính là money và description và chúng được gán cho các đối số được truyền trong hàm tạo.

Right here, this is perfectly fine, but now there is a shorthand version of this .

Ở đây, điều này hoàn toàn ổn, nhưng bây giờ có một phiên bản tốc ký của tệp .

So basically instead of this now you're able to add the visibility or access modifier directly in your constructor .

Vì vậy, về cơ bản, thay vì điều này, giờ đây bạn có thể thêm trực tiếp công cụ sửa đổi mức độ hiển thị hoặc quyền truy cập vào hàm tạo của mình.



So in here we can set this to private, public or protected .

Vì vậy, ở đây chúng ta có thể đặt cái này thành riêng tư, công khai hoặc được bảo vệ.

So let's set this to private as well and ignore the underlining .

Vì vậy, hãy đặt nó ở chế độ riêng tư và bỏ qua phần gạch chân .

Now I'll explain what that is in a second .

Bây giờ tôi sẽ giải thích đó là gì trong giây lát.

So when PHP sees that and detects that we have added the access modifier in the constructor, it will interpret it as both the property and constructor argument and it will assign the value that is passed to the constructor to that property behind the scenes .

Vì vậy, khi PHP thấy điều đó và phát hiện ra rằng chúng ta đã thêm công cụ sửa đổi truy cập vào hàm tạo, nó sẽ hiểu nó là cả đối số thuộc tính và hàm tạo, đồng thời nó sẽ gán giá trị được truyền cho hàm tạo cho thuộc tính đó đằng sau hậu trường.

So basically it still does the property definition and property assignment behind the scenes for you,

so you don't have to do it .

Vì vậy, về cơ bản, nó vẫn thực hiện định nghĩa thuộc tính và gán thuộc tính ở hậu trường cho bạn,

vì vậy bạn không cần phải làm điều đó.

So you can delete this and also get rid of this .

Vì vậy, bạn có thể xóa cái này và cũng có thể thoát khỏi cái này.

And this will work in PHP eight .

Và điều này sẽ hoạt động trong PHP 8.

So as you can see, constructor property promotion feature offers a shorthand version and much cleaner syntax and less boilerplate .

Vì vậy, như bạn có thể thấy, tính năng quảng bá thuộc tính của hàm tạo cung cấp một phiên bản tốc ký và cú pháp rõ ràng hơn nhiều cũng như ít bản soạn sẵn hơn.

While behind the scenes PHP still does the property definition and assignment for you though there are a few things that you need to be aware of and let's get into that right now .

Mặc dù đằng sau hậu trường, PHP vẫn thực hiện định nghĩa và gán thuộc tính cho bạn, mặc dù có một số điều bạn cần lưu ý và hãy đi sâu vào vấn đề đó ngay bây giờ.

So first thing, you can type Hint, any type except Callable .

Vì vậy, điều đầu tiên, bạn có thể gõ Gợi ý, bất kỳ loại nào ngoại trừ Callable .

That's because you cannot type hint properties with Callable type .

Đó là do bạn không thể gõ thuộc tính gợi ý bằng kiểu Callable.

So if we revert it back to this here and remove this, we were not able to type hint Callable .

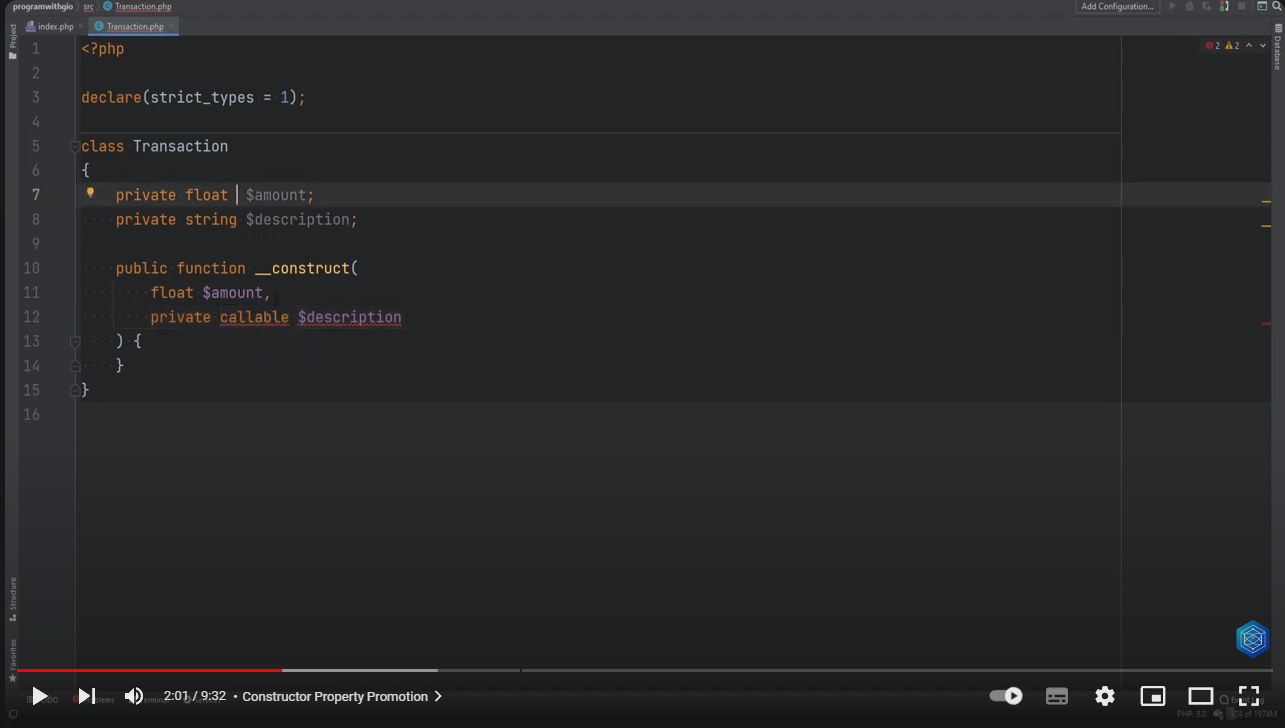
Vì vậy, nếu chúng tôi hoàn nguyên nó về cái này ở đây và xóa cái này, chúng tôi không thể nhập gợi ý Callable .

PHP does not allow that .

PHP không cho phép điều đó.

However, in the arguments you're able to type hint Callable but if you're promoting these properties and you set the access modifier, then this becomes illegal because you're not allowed to type hint properties with Callable .

Tuy nhiên, trong các đối số, bạn có thể nhập gợi ý Có thể gọi được nhưng nếu bạn đang quảng cáo các thuộc tính này và bạn đặt công cụ sửa đổi quyền truy cập thì điều này sẽ trở thành bất hợp pháp vì bạn không được phép nhập thuộc tính gợi ý bằng Callable .

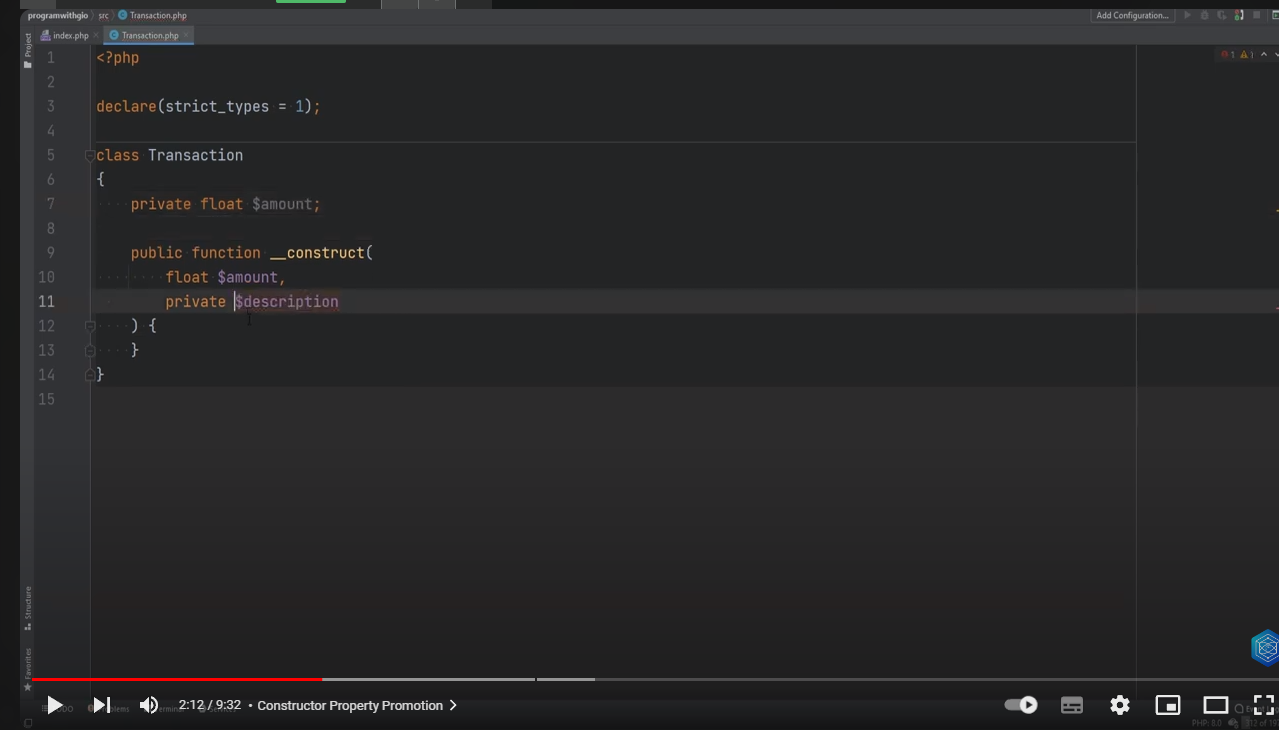


Also you're able to combine promoted properties with class properties so you don't have to promote all the properties .

Ngoài ra, bạn có thể kết hợp các thuộc tính được thăng cấp với thuộc tính lớp để không phải quảng bá tất cả các thuộc tính.

If you only wanted to promote description for example and not the amount, you could delete this and remove this from here and this is perfectly valid syntax .

Nếu bạn chỉ muốn quảng cáo mô tả chẳng hạn chứ không phải số tiền, bạn có thể xóa phần này và xóa phần này khỏi đây và đây là cú pháp hoàn toàn hợp lệ.



So now you're promoting description and here you can set amount or do whatever you need to do with the amount property .

Vì vậy, bây giờ bạn đang quảng cáo mô tả và tại đây bạn có thể đặt số tiền hoặc làm bất cứ điều gì bạn cần làm với thuộc tính money .

Another thing you need to be aware of is that you cannot have duplicates, meaning that you cannot have a property and promote it property at the same time with the same name .

Một điều nữa bạn cần lưu ý là bạn không thể có tài sản trùng lặp, nghĩa là bạn không thể có một tài sản và quảng bá tài sản đó cùng một lúc với cùng một tên.

So right here we have private description, we cannot have private description here . \

Vì thế ở đây chúng ta có mô tả riêng, không thể có mô tả riêng ở đây được.

This is not going to work .

Điều này sẽ không hiệu quả

You don't have to type hint arguments, you can promote untyped properties as well .

Bạn không cần phải nhập các đối số gợi ý, bạn cũng có thể quảng cáo các thuộc tính chưa được gõ.

So in here, as you can see, there is no type and this will still work though you can add type if you want to or remove the type . \

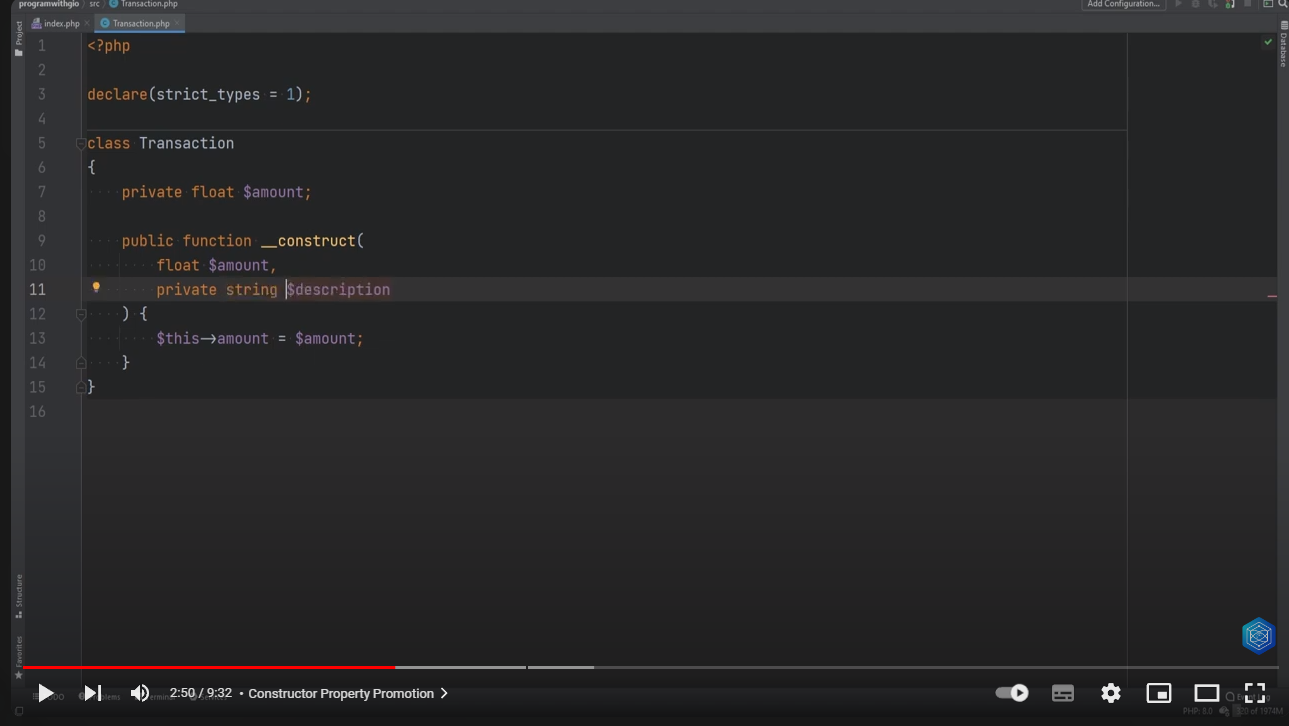
Vì vậy, ở đây, như bạn có thể thấy, không có loại nào và nó vẫn hoạt động mặc dù bạn có thể thêm loại nếu muốn hoặc xóa loại .

That is entirely up to you .

Tất cả đều phụ thuộc vào bạn

Next thing you should know is that you could assign default values to Promoted properties but has to follow the same rule .

Điều tiếp theo bạn nên biết là bạn có thể chỉ định giá trị mặc định cho Thuộc tính được quảng cáo nhưng phải tuân theo quy tắc tương tự.



You can only have the simple expressions or constant values as default values .

Bạn chỉ có thể có các biểu thức đơn giản hoặc các giá trị không đổi làm giá trị mặc định.

You cannot use function calls or complex expressions .

Bạn không thể sử dụng lệnh gọi hàm hoặc biểu thức phức tạp.

So as you remember you were able to set the default value here .

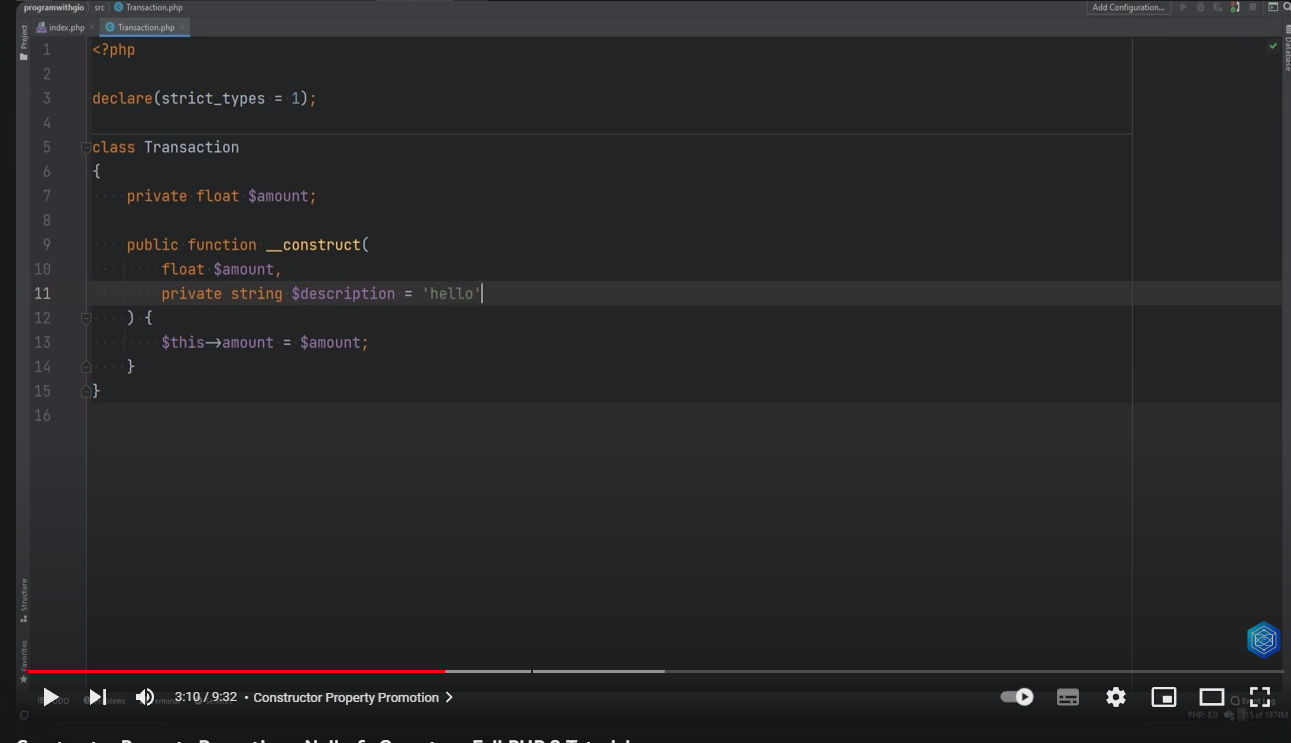
Vì vậy, như bạn nhớ, bạn có thể đặt giá trị mặc định ở đây.

You can do the same thing here .

Bạn có thể làm điều tương tự ở đây.

So you can set the default value here and it will work .

Vì vậy, bạn có thể đặt giá trị mặc định ở đây và nó sẽ hoạt động.

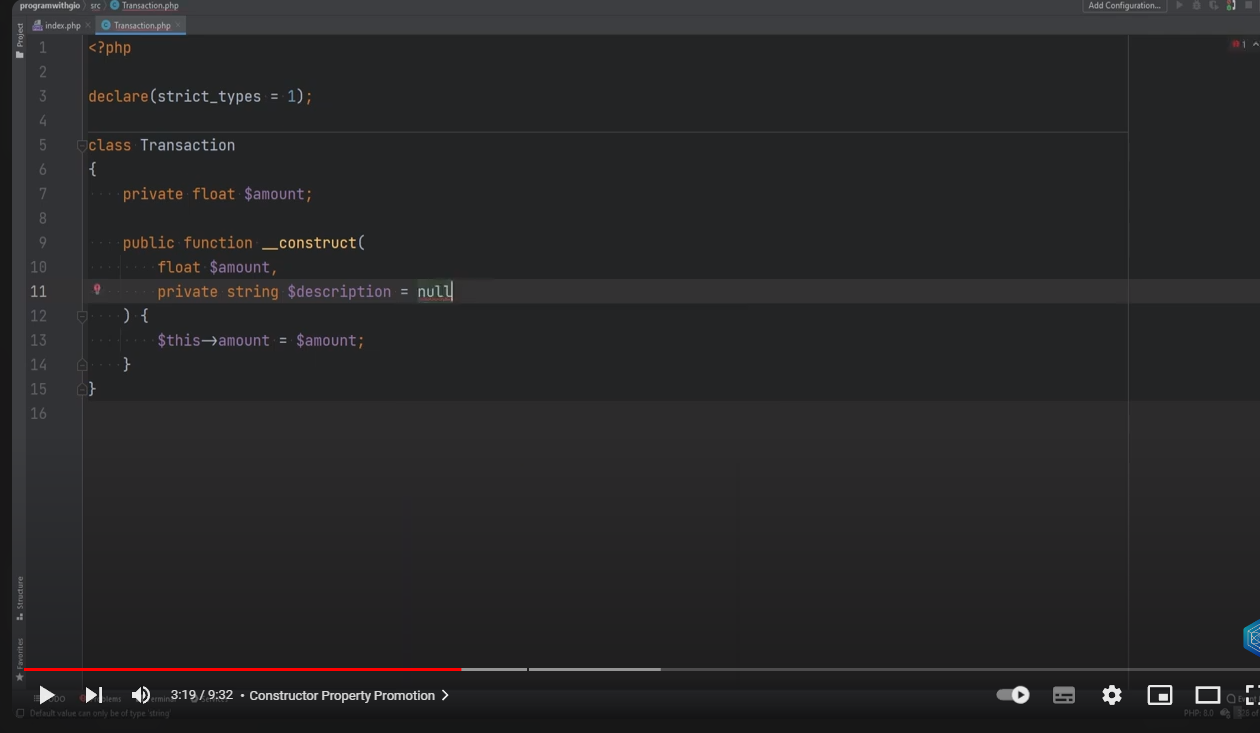


Though one thing you should know about the default values is that if you're assigning a default value of null you will need to make that property nullable type .

Mặc dù có một điều bạn nên biết về các giá trị mặc định là nếu bạn đang gán một giá trị mặc định là null thì bạn sẽ cần phải đặt thuộc tính đó thành loại nullable.

So for example, if you set this to null this will not work because we're expecting string type and we cannot set that to null .

Vì vậy, ví dụ: nếu bạn đặt giá trị này thành null thì điều này sẽ không hoạt động vì chúng tôi đang mong đợi loại chuỗi và chúng tôi không thể đặt giá trị đó thành null .



So in order for this to work you need to make this nullable type .

Vì vậy, để nó hoạt động, bạn cần tạo loại nullable này.

So let's talk about accessing the promoted properties .

Vì vậy, hãy nói về việc truy cập các thuộc tính được thăng cấp.

Let me promote this property as well .

Hãy để tôi quảng bá tài sản này là tốt.

The way you could access these properties is by either using this variable .

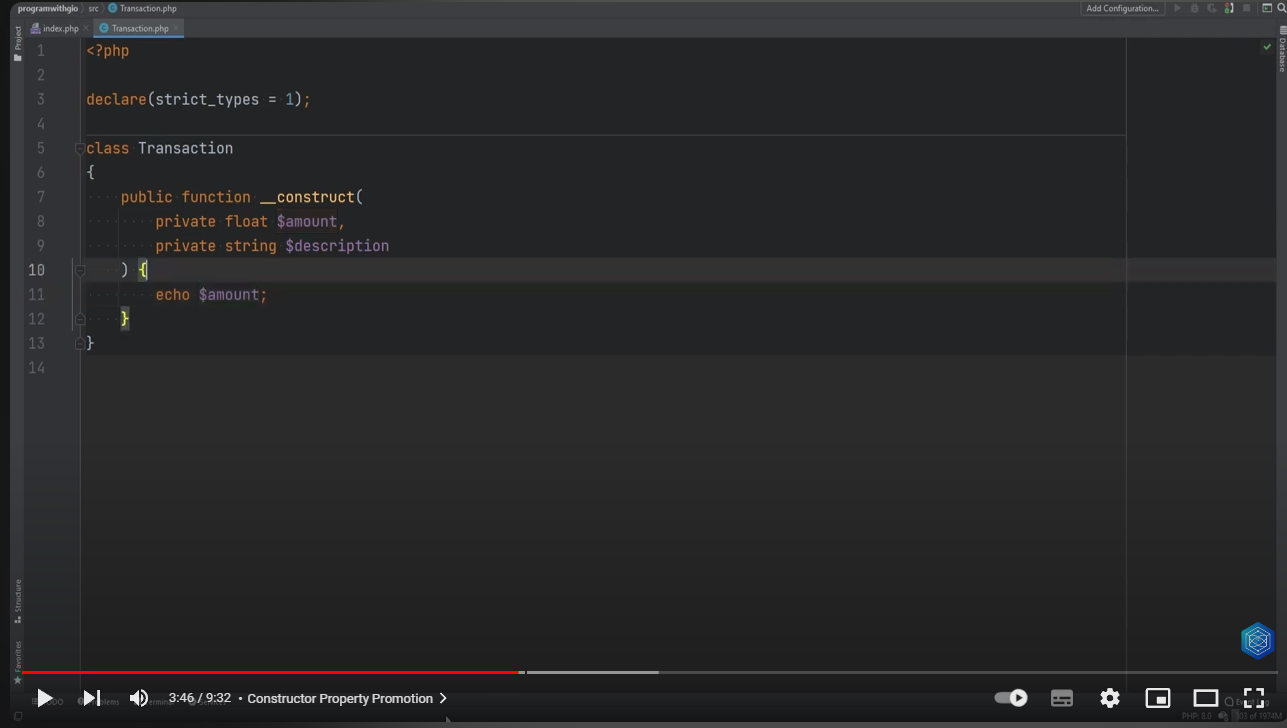
Cách bạn có thể truy cập các thuộc tính này là sử dụng biến này.

So something like echo this amount and this will of course work .

Vì vậy, một cái gì đó như lặp lại số lượng này và tất nhiên điều này sẽ hoạt động.

So let's test this out, we refresh, we get five .

Vì vậy, hãy kiểm tra điều này, chúng tôi làm mới, chúng tôi nhận được năm .



Another way you could access it is by simply just accessing the variable without this and it will still work .

Một cách khác mà bạn có thể truy cập nó là chỉ cần truy cập vào biến mà không có cái này và nó vẫn hoạt động.

So if I go back to the browser and refresh we still get five .

Vì vậy, nếu tôi quay lại trình duyệt và làm mới, chúng tôi vẫn nhận được năm tệp .

Let's talk about the null Safe operator .

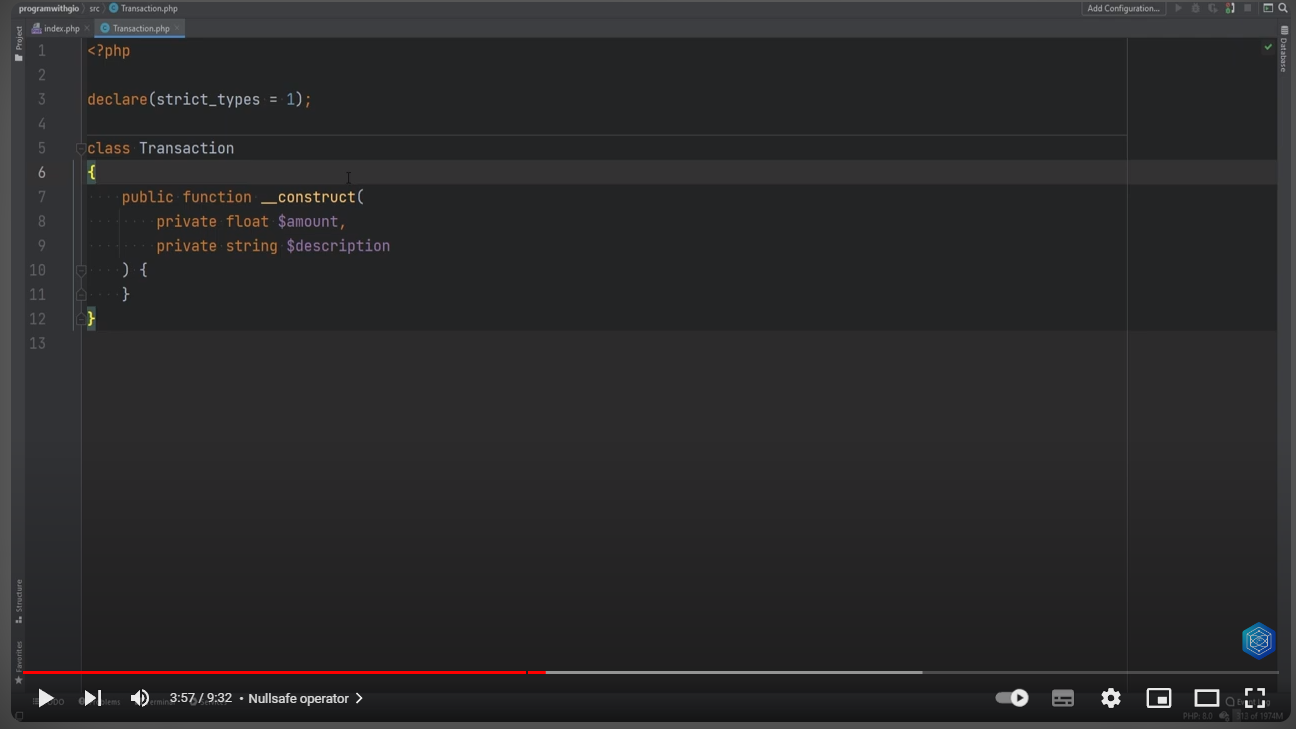
Hãy nói về toán tử an toàn null.

The null Safe operator allows you to chain properties and method calls even if one of them returns null .

Toán tử an toàn null cho phép bạn xâu chuỗi các thuộc tính và lệnh gọi phương thức ngay cả khi một trong số chúng trả về null.

So for example, let's say that we want to access the Payment Profile ID which is nested two to three levels deep within the transaction object . So you would have something like Transaction Customer, Payment Profile ID and let's say that we want to echo that out .

Vì vậy, ví dụ: giả sử chúng ta muốn truy cập ID hồ sơ thanh toán được lồng sâu từ hai đến ba cấp trong đối tượng giao dịch . Vì vậy, bạn sẽ có thông tin nào đó như Khách hàng giao dịch, ID hồ sơ thanh toán và giả sử rằng chúng tôi muốn thể hiện điều đó .



Of course we don't have these properties yet, but I have already created the Payment Profile and Customer classes and they're just simple classes .

Tất nhiên là chúng ta chưa có những thuộc tính này, nhưng tôi đã tạo các lớp Hồ sơ thanh toán và Khách hàng và chúng chỉ là các lớp đơn giản.

Customer Class has a nullable Payment Profile and Payment Profile is just a simple class that assigns a random number to the ID .

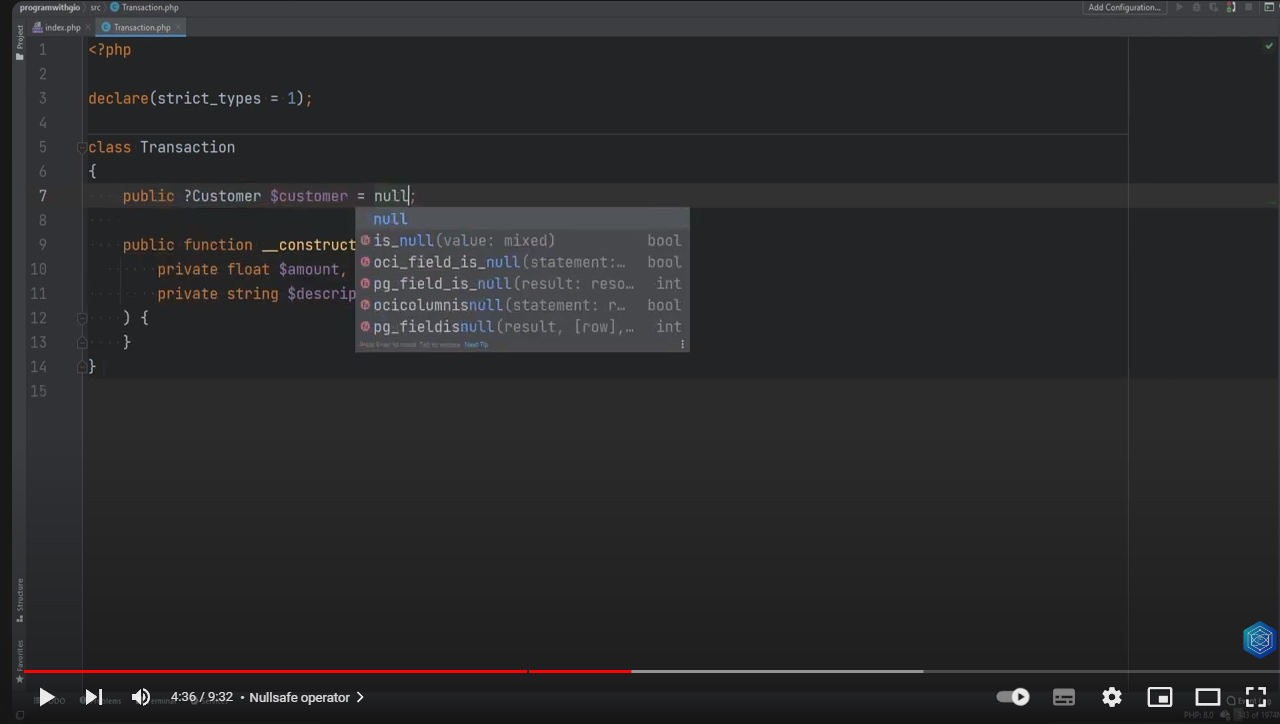
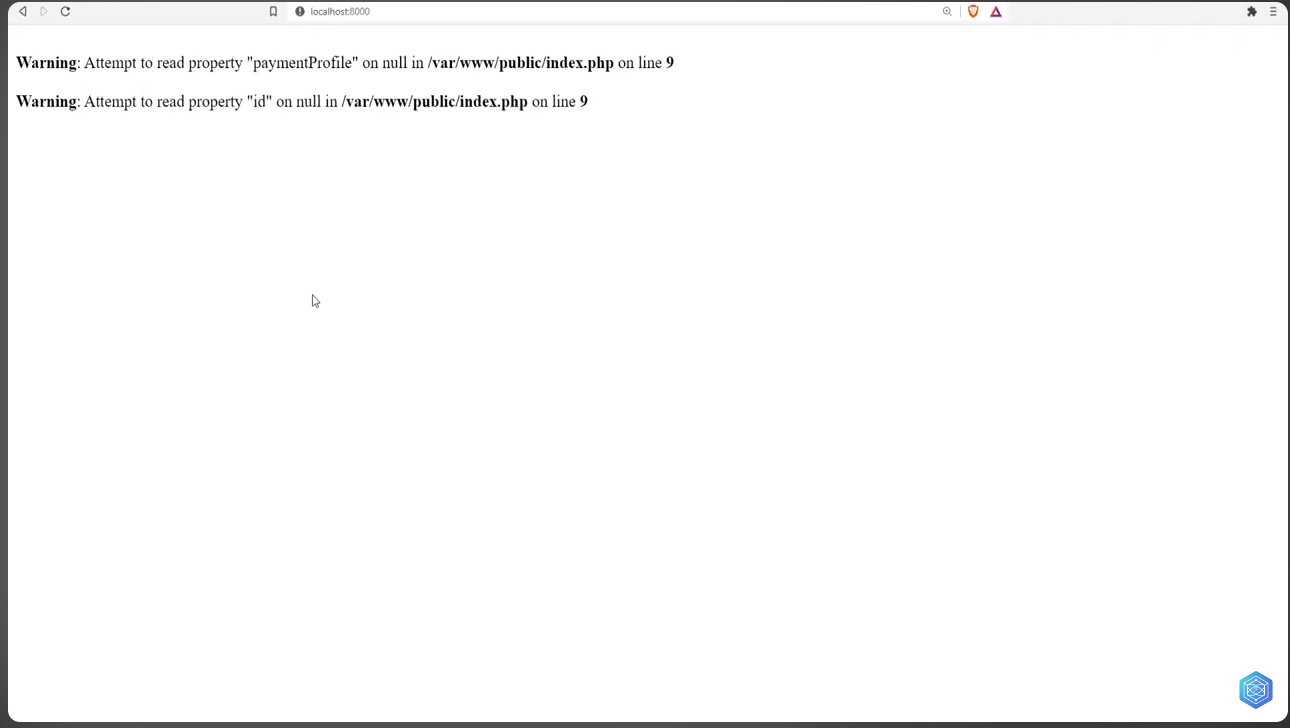
Lớp khách hàng có Hồ sơ thanh toán không thể rỗng và Hồ sơ thanh toán chỉ là một lớp đơn giản gán một số ngẫu nhiên cho ID .

Just to demonstrate this feature I'm going to close those out and I just need to add the Customer property to the transaction so we can do Public Customer and this can also be nullable for our case .

Để chứng minh tính năng này, tôi sẽ đóng chúng lại và tôi chỉ cần thêm thuộc tính Khách hàng vào giao dịch để chúng tôi có thể thực hiện Khách hàng công khai và điều này cũng có thể vô hiệu đối với trường hợp của chúng tôi.

So let's go back here now and of course we're going to get an error because Customer is null and therefore we cannot access the Payment Profile or ID .

Vì vậy, hãy quay lại đây ngay bây giờ và tất nhiên chúng ta sẽ gặp lỗi vì Customer là null và do đó chúng ta không thể truy cập Hồ sơ thanh toán hoặc ID.

So if we go back and refresh we get attempt to read Property Payment Profile on null and then attempt to read Property ID on null .

Vì vậy, nếu chúng tôi quay lại và làm mới, chúng tôi sẽ cố gắng đọc Hồ sơ thanh toán tài sản trên null và sau đó thử đọc ID tài sản trên null .

You could use a null Safe operator here which prefixes the object operator with a question mark .

Bạn có thể sử dụng toán tử An toàn null ở đây để đặt trước toán tử đối tượng bằng dấu chấm hỏi .

So this is the object operator which is dash and a greater than sign and you can prefix that with a question mark and now this becomes a null safe operator .

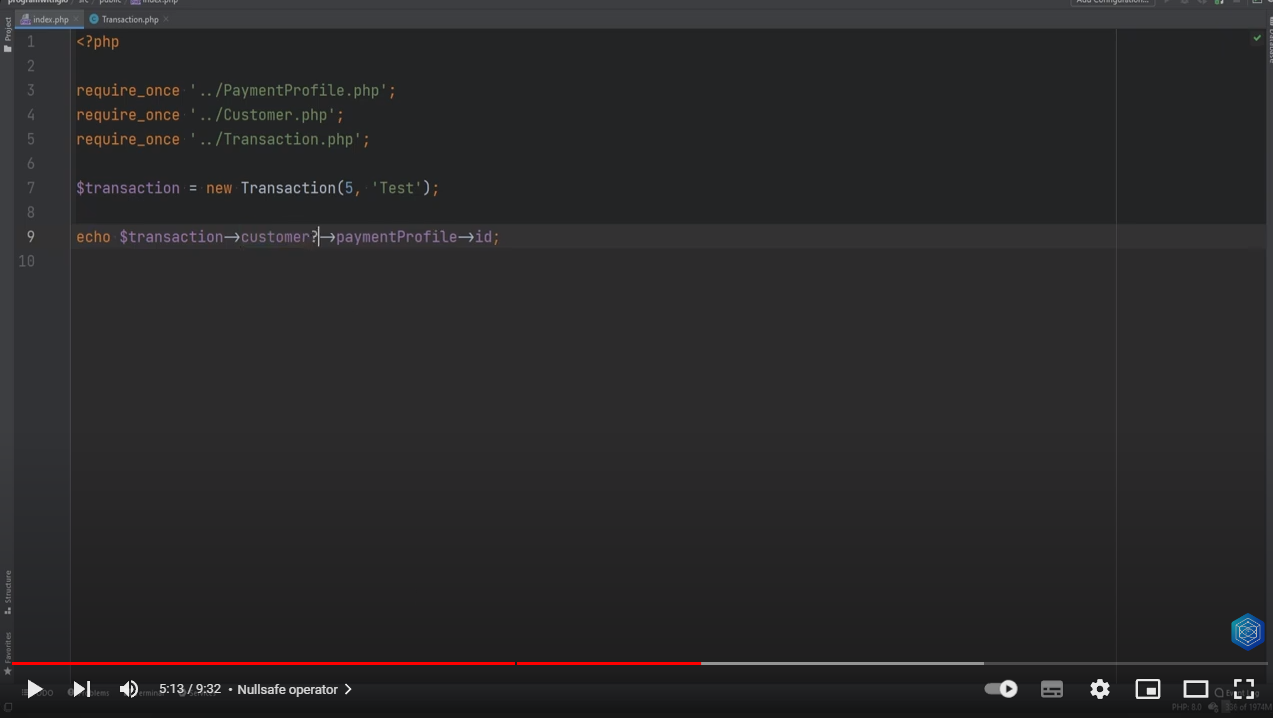
Vì vậy, đây là toán tử đối tượng là dấu gạch ngang và dấu lớn hơn và bạn có thể thêm tiền tố vào dấu hỏi đó và bây giờ điều này trở thành toán tử an toàn rỗng.

Here we don't need to make this null safe because transaction is always available .

Ở đây chúng ta không cần phải đảm bảo giá trị rỗng này là an toàn vì giao dịch luôn có sẵn.

We need to make null safe customer because customer might be null .

Chúng ta cần đảm bảo khách hàng là null vì khách hàng có thể là null .



And now if we go ahead and refresh the page we no longer get the error .

Và bây giờ nếu chúng ta tiếp tục và làm mới trang thì chúng ta sẽ không còn gặp lỗi nữa.

Though this is not perfect yet because what if we do transaction customer equals new customer .

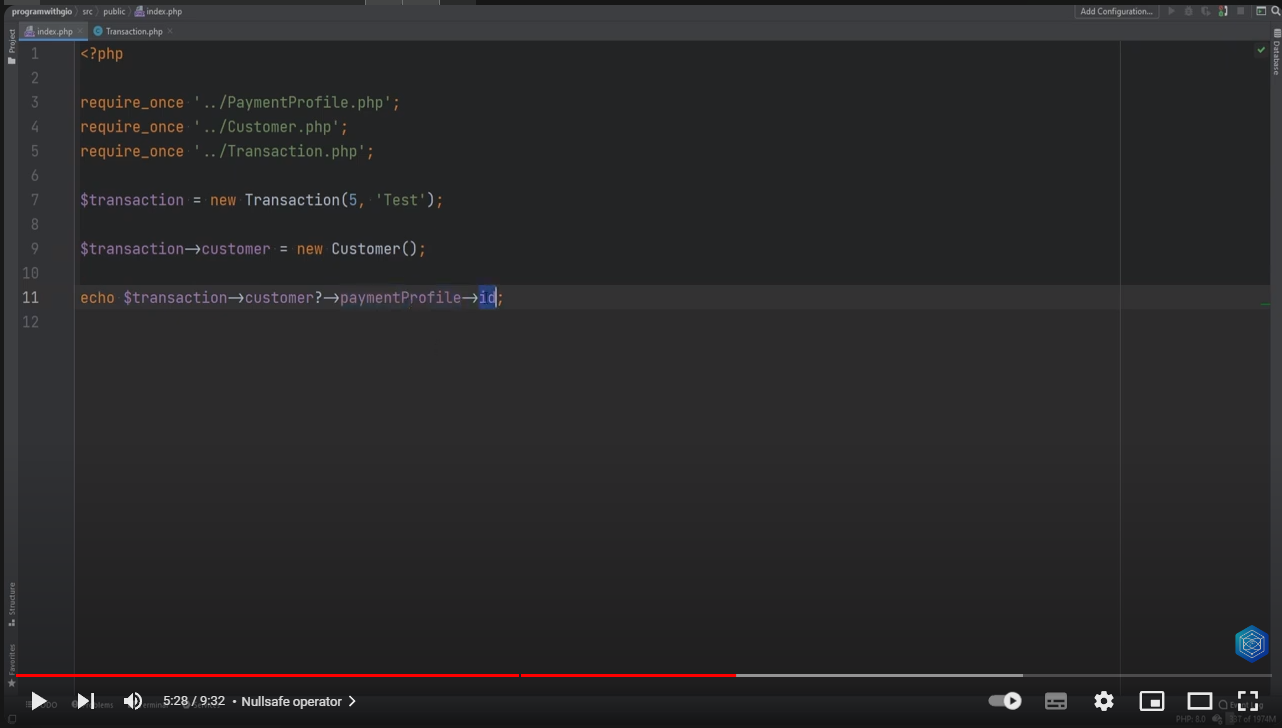
Mặc dù điều này vẫn chưa hoàn hảo vì điều gì sẽ xảy ra nếu chúng ta thực hiện giao dịch với khách hàng tương đương với khách hàng mới.

Now customer is set but the payment profile is null and we're trying to access ID on the payment profile we'll still get an error .

Hiện tại, khách hàng đã được đặt nhưng hồ sơ thanh toán không có giá trị và chúng tôi đang cố gắng truy cập ID trên hồ sơ thanh toán nhưng vẫn gặp lỗi.

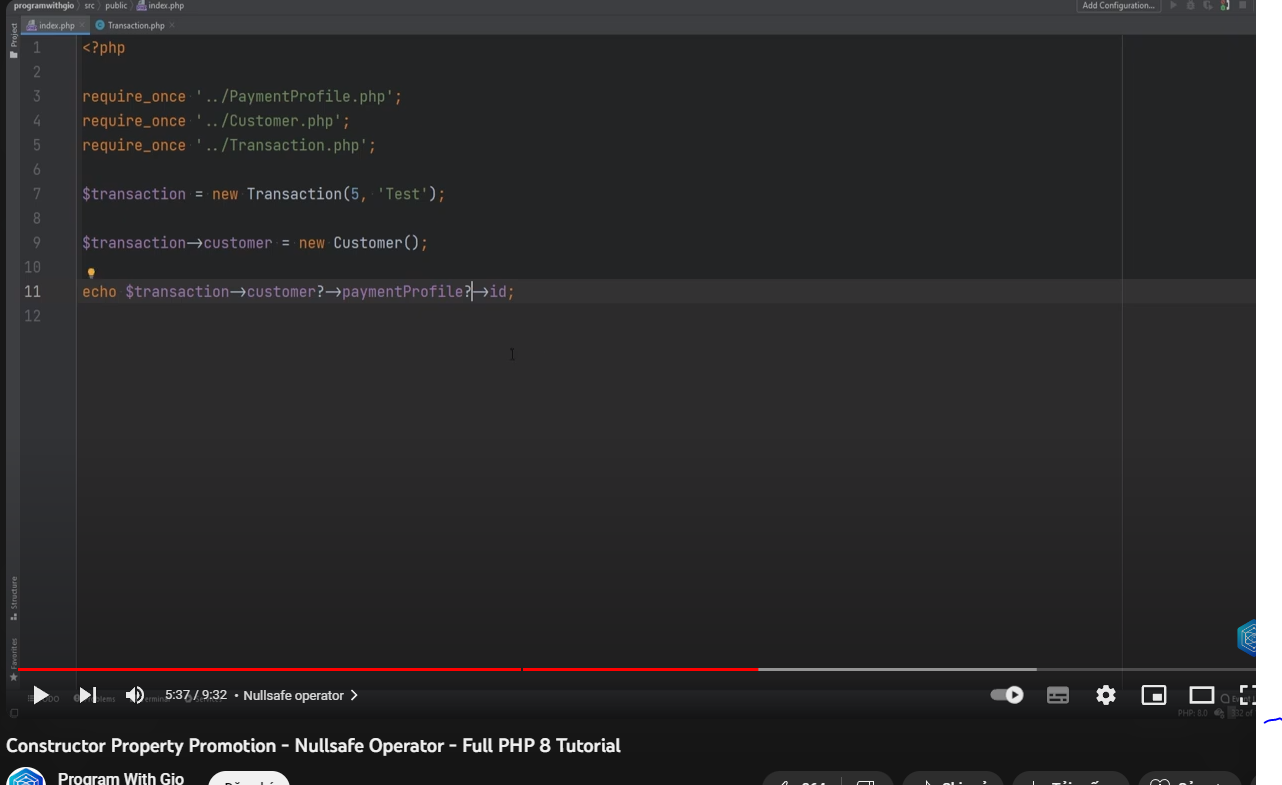
So if we go back here and we refresh we'll get an error and you could change the null safe operator so you could make this null safe as well .

Vì vậy, nếu chúng ta quay lại đây và làm mới, chúng ta sẽ gặp lỗi và bạn có thể thay đổi toán tử an toàn null để bạn cũng có thể làm cho giá trị rỗng này trở nên an toàn.



And if we go back here and refresh, the error is gone .

Và nếu chúng ta quay lại đây và làm mới thì lỗi sẽ không còn nữa.



Now you might be asking can't you simply use the null coalescing operator instead? The answer is yes .

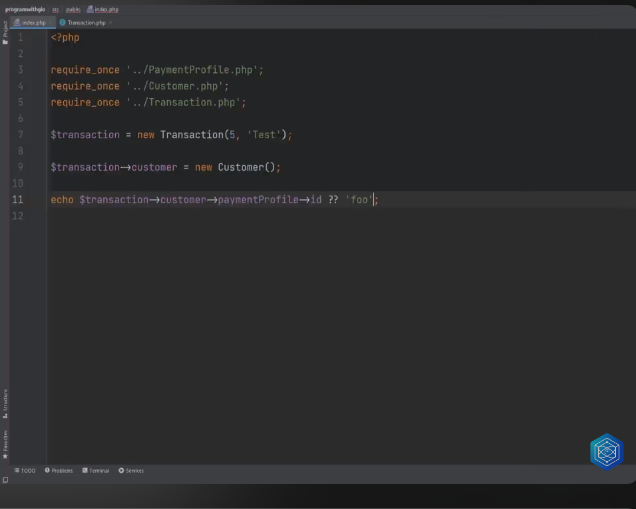
Bây giờ bạn có thể hỏi rằng thay vào đó, bạn có thể sử dụng toán tử hợp nhất null không? Câu trả lời là có

In this case you could use no coalescing operator but there is a difference and we'll talk about that in a second .

Trong trường hợp này, bạn có thể không sử dụng toán tử kết hợp nào nhưng có một sự khác biệt và chúng ta sẽ nói về điều đó sau.

So if we revert this and use no coalescing operator, something like this and we go to the browser and refresh, this is going to work .

Vì vậy, nếu chúng ta hoàn nguyên điều này và không sử dụng toán tử kết hợp, đại loại như thế này và chúng ta truy cập trình duyệt và làm mới, điều này sẽ hoạt động.

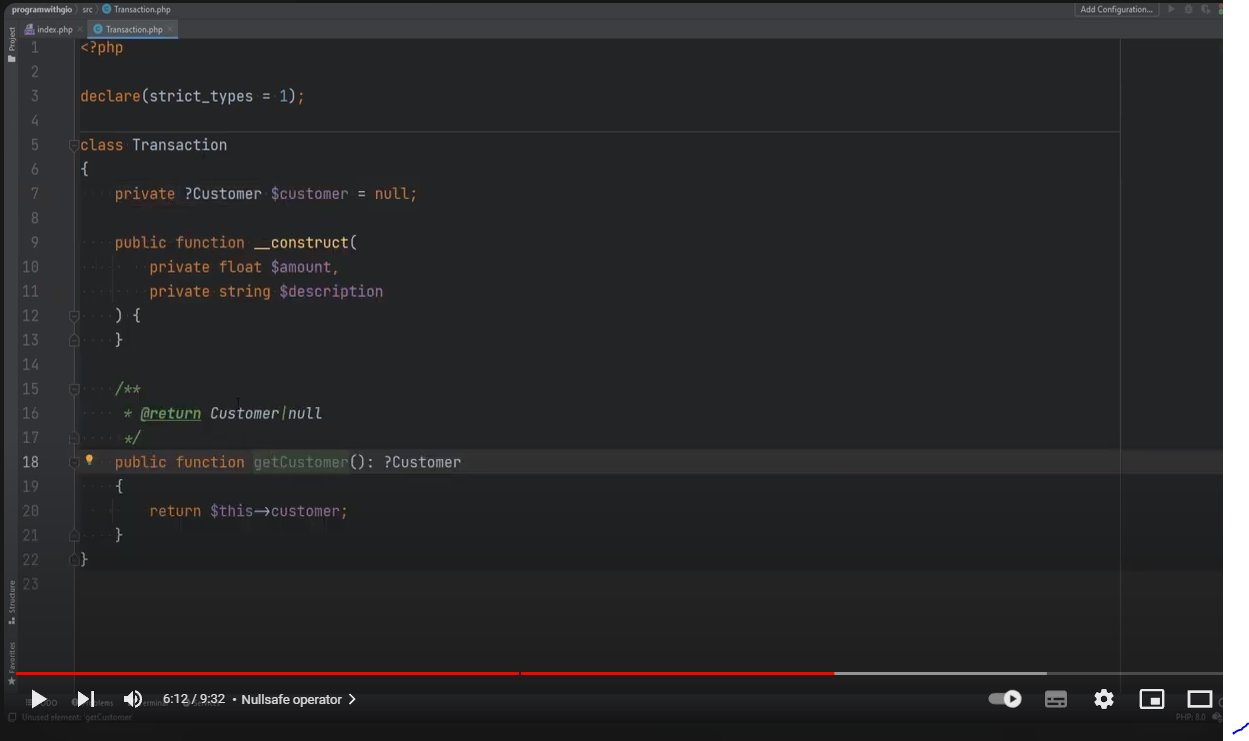


No errors though what if we change these to methods instead of the properties? So right now these are public properties, right? So let's change them to private .

Không có lỗi nhưng nếu chúng ta thay đổi chúng thành phương thức thay vì thuộc tính thì sao? Vậy hiện tại đây là tài sản công phải không? Vì vậy, hãy thay đổi chúng thành riêng tư.

So I'm going to change them to private and I'm going to make a Getter and same thing for the customer change payment profile to private and create Getter .

Vì vậy, tôi sẽ thay đổi chúng thành riêng tư và tôi sẽ tạo Getter và điều tương tự cho khách hàng thay đổi hồ sơ thanh toán thành riêng tư và tạo Getter.

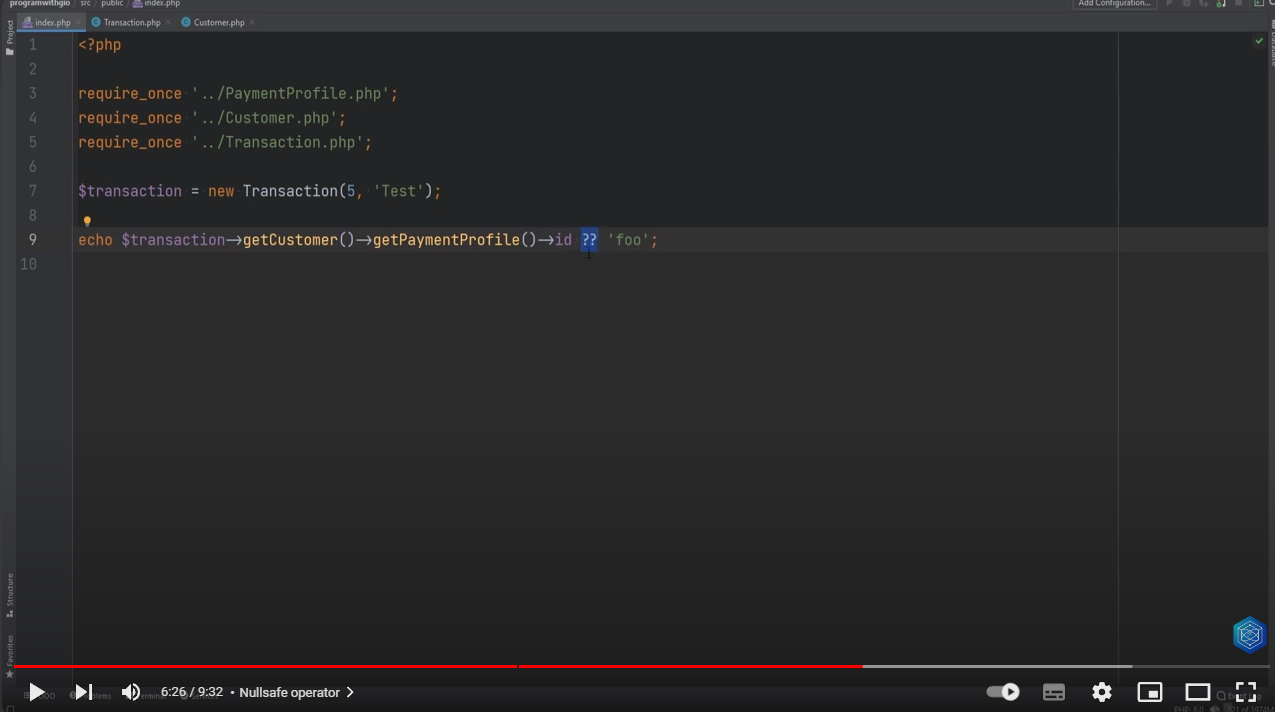


So here we can remove this and change this to Get customer and Get payment profile .

Vì vậy, ở đây chúng ta có thể xóa phần này và thay đổi phần này thành Nhận khách hàng và Nhận hồ sơ thanh toán .

And as you know this will not work because null coalescing operator will not work with the method calls .

Và như bạn biết điều này sẽ không hoạt động vì toán tử hợp nhất null sẽ không hoạt động với các lệnh gọi phương thức.



So if I go back to the browser refresh we're going to get an error .

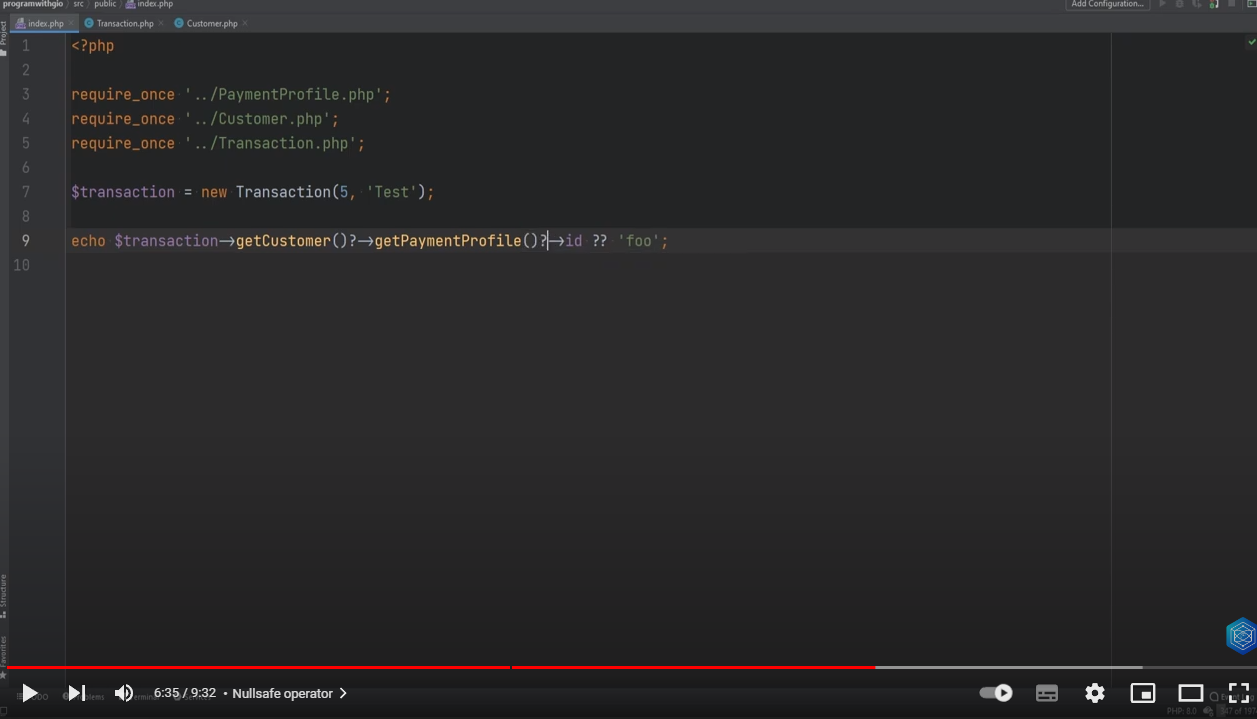
Vì vậy, nếu tôi quay lại làm mới trình duyệt, chúng tôi sẽ gặp lỗi.

So this is where the null safe operator can save you .

Vì vậy, đây là lúc toán tử an toàn null có thể cứu bạn.

So you can do this and you can do this and we go back here and now we get full because we're combining the null safe operator with the null coalescing operator and it works just fine .

Vì vậy, bạn có thể làm điều này và bạn có thể làm điều này và chúng ta quay lại đây và bây giờ chúng ta đã hoàn thành đầy đủ vì chúng ta đang kết hợp toán tử an toàn null với toán tử kết hợp null và nó hoạt động rất tốt.



If we didn't do this, then it would simply just echo null and when null is echoed in the screen, it's just an empty string .

Nếu chúng ta không làm điều này thì nó sẽ chỉ lặp lại null và khi null được lặp lại trên màn hình, nó chỉ là một chuỗi trống.

Without the null safe operator you would be nesting a lot of null check conditionals .

Nếu không có toán tử an toàn null, bạn sẽ lồng rất nhiều điều kiện kiểm tra null.

So how does null safe operator work? Null safe operator basically solves this problem by short circuiting the property or method access and will return null immediately as soon as itis null without executing rest of the expression .

Vậy toán tử an toàn null hoạt động như thế nào? Toán tử an toàn Null về cơ bản giải quyết vấn đề này bằng cách đoản mạch quyền truy cập thuộc tính hoặc phương thức và sẽ trả về null ngay lập tức ngay khi nó là null mà không thực hiện phần còn lại của biểu thức.

So for example, if Get customer returns null, which it does in this case everything on the right side which is this will get discarded and the same goes on this side .

Vì vậy, ví dụ: nếu Nhận khách hàng trả về giá trị rỗng, thì trong trường hợp này, mọi thứ ở phía bên phải sẽ bị loại bỏ và điều tương tự cũng xảy ra ở phía bên này.

So if this did not return no, then we move on to the get payment profile and if this returns no, then everything on the right side which is this will get discarded which can have its pros and cons and we'll talk about that in a second .

Vì vậy, nếu điều này không trả về không thì chúng ta chuyển sang hồ sơ nhận thanh toán và nếu điều này trả về không thì mọi thứ ở phía bên phải sẽ bị loại bỏ, điều này có thể có những ưu và nhược điểm và chúng ta sẽ nói về điều đó trong một giây.

So let's talk about a few things to know and be aware of when working with null safe operators .

Vì vậy, hãy nói về một số điều cần biết và cần lưu ý khi làm việc với các toán tử an toàn null.

First of all, it's read only meaning that you cannot write or assign values to it .

Trước hết, nó chỉ đọc có nghĩa là bạn không thể viết hoặc gán giá trị cho nó.

So for example, we cannot do something like ID equals ten .

Vì vậy, ví dụ: chúng tôi không thể làm điều gì đó như ID bằng mười .

This is not going to work .

Điều này sẽ không hiệu quả.

As I mentioned before, you could chain null safe operators like this, but you should be aware that it will short circuit as soon as the null value is encountered wherever the null safe operator is used .

Như tôi đã đề cập trước đây, bạn có thể xâu chuỗi các toán tử an toàn null như thế này, nhưng bạn nên lưu ý rằng nó sẽ bị đoản mạch ngay khi gặp giá trị null ở bất kỳ nơi nào sử dụng toán tử an toàn null.

Also, be careful not to pass important function calls that need to be executed even when nulls are present in the chain .

Ngoài ra, hãy cẩn thận để không vượt qua các lệnh gọi hàm quan trọng cần được thực thi ngay cả khi có giá trị rỗng trong chuỗi.

Let's say that instead of the get payment profile here we have something like set payment profile and here we're passing the profile that gets created as a result of some kind of function call .

Giả sử rằng thay vì nhận hồ sơ thanh toán ở đây, chúng tôi có nội dung như đặt hồ sơ thanh toán và ở đây chúng tôi chuyển hồ sơ được tạo do một loại lệnh gọi hàm nào đó.

So let's assume for the sake of this example that profile gets created in some kind of create profile function .

Vì vậy, hãy giả sử vì ví dụ này rằng hồ sơ được tạo bằng một loại chức năng tạo hồ sơ nào đó.

Now the problem with this could be that if get customer returns null then create profile function will never run and your customer's profile will never be created .

Bây giờ, vấn đề với điều này có thể là nếu khách hàng trả về giá trị rỗng thì chức năng tạo hồ sơ sẽ không bao giờ chạy và hồ sơ khách hàng của bạn sẽ không bao giờ được tạo.



And that might be intended behavior, but it might also cause some bugs because if you wanted to run this function regardless of what's happening here, then you would need to take this out and put it in a variable before that and run it and then pass that variable as the argument and that would work .

Và đó có thể là hành vi có chủ đích, nhưng nó cũng có thể gây ra một số lỗi vì nếu bạn muốn chạy hàm này bất kể chuyện gì đang xảy ra ở đây, thì bạn cần phải lấy cái này ra và đặt nó vào một biến trước đó rồi chạy nó rồi chuyển biến đó làm đối số và điều đó sẽ hoạt động.

But on the other hand, if this function was very expensive, you might not want to run it unless the customer object is set .

Nhưng mặt khác, nếu chức năng này rất đắt tiền, bạn có thể không muốn chạy nó trừ khi đối tượng khách hàng được đặt.

So in some cases you might not want to run certain functions .

Vì vậy trong một số trường hợp bạn có thể không muốn chạy một số chức năng nhất định.

So calling it here will save you on resources in cases where get customer returns .

Vì vậy, gọi nó ở đây sẽ giúp bạn tiết kiệm tài nguyên trong trường hợp khách hàng trả lại.

No, in that case you're not running that expensive function .

Không, trong trường hợp đó bạn không chạy chức năng đắt tiền đó.

So as I mentioned before, it can go both ways .

Vì vậy, như tôi đã đề cập trước đó, nó có thể đi theo cả hai hướng.

It mainly depends on your application design or your use case .

Nó chủ yếu phụ thuộc vào thiết kế ứng dụng hoặc trường hợp sử dụng của bạn.

So just be aware of what short circuiting is and how it works .

Vì vậy, chỉ cần lưu ý ngắn mạch là gì và nó hoạt động như thế nào.

If you want to know more about either null safe operator or the Constructor property promotion, I'm going to leave the links to the RFCs in the description and you can read more about these if you want to .

Nếu bạn muốn biết thêm về nhà điều hành an toàn null hoặc quảng cáo thuộc tính của Trình xây dựng, tôi sẽ để lại các liên kết đến RFC trong phần mô tả và bạn có thể đọc thêm về những điều này nếu muốn.

So this is it for this video .

Vì vậy, đây là nó cho video này.

I hope you enjoyed it .

Tôi hy vọng bạn thích nó .

Thank you so much for watching .

Cảm ơn bạn rất nhiều vì đã xem .

Please give this video a thumbs up, share and subscribe and I'll see you next week .

Vui lòng thích video này, chia sẻ và đăng ký và tôi sẽ gặp bạn vào tuần tới.

Where we're going to cover PSR standards, auto loading and composer .

Nơi chúng tôi sẽ đề cập đến các tiêu chuẩn PSR, tính năng tự động tải và trình soạn thảo .