Visitor Design Pattern - return type

I used the Visitor design pattern to solve one of the issues within our system. As a reference how to implement it I used DoFactory site and This YouTube video.

In DoFactory example, the Visitor uses a method with return type of "void" and in YouTube video, author uses "double".

Why I ask:
After presenting the solution to company CTO, he accepted to call it Visitor, but he claims like if Visitor is not "void" as stated in GoF, than it like abuse the real Visitor pattern.

Question:
Is it required for Visitor pattern to return "void"?
I mean in order to be "real Visitor pattern" as DoFactory (GoF) describes it, or it can be any return type and can still be called "real Visitor pattern"?

design-patterns architecture visitor-pattern

asked Apr 12 '15 at 18:36

Alex Dn

3,043 5 30 60

1 It would be great to know the reasons of the *other* author about why he chose to return `double` instead of nothing (`void`); then analyze if those reasons are valid or not in such context / scenario. Most of the time, talking about ***pure*** this or that pattern is kind of counterproductive. Just use what works for you the way it works best; don't worry too much about what to call it. If you call either of those *visitor pattern*, most people would still agree to and understand what you are talking about: that's the main goal, in the first place. Let's see... define "real". – Leandro Apr 12 '15 at 19:07

1 I wrote an answer on Programmers.SE that touches on this point. In short, the GoF design patterns book literally includes example code that returns arbitrary objects. This has largely gone unnoticed because that code is in Smalltalk, using dynamic typing. The C++ example code has to use void because semantic restrictions of that language, not because a void return type is a central feature of the Visitor Pattern. In Java, we can use generics for non-void return types in a type-safe manner. – amon Mar 28 '16 at 17:06

1 Answer

Design Patterns are meant to be used as a guide to show how to solve common computer science problems. You are welcome to deviate from the "real" implementation any way you wish.

As for your example youtube video, the author of that shows a how you can use the visitor pattern to compute taxes for different types of items. Each `visit` method returned a double for the amount of money including taxes each item would cost. Different Visitor implementations were then made to show how you could have different ways to compute taxes (normal vs tax holiday etc) without changing your code.

This example was a "toy" problem and was meant to teach how the visitor pattern worked in an easy to understand way- and it does a good job.

While I say you are welcome to deviate from the GoF implementation, it might not be a good idea to mimic the code from this video. There were a few things in the video that would be a bad idea to use in a real program. For example using `double` for money. I think the return double (for money) was just a way to quickly show how the visitor worked and you probably shouldn't use it.

If you wanted to modify the code in the video to return void instead. The easiest solution would be to have a private field in the `TaxVisitor` that accumulates the total value and increment it inside each visit method. Then have a getter to get the final total.

The author also explicitly invokes each food item in his visitor example which doesn't show off the power of the Visitor Pattern. I would have had a container object of the grocery items that would be visitable and its accept method would visit each item in the receipt.

```
GroceryList groceries = new GroceryList();

groceries.add(new Milk(...));
groceries.add(new Liquor(...));
   ...


 TaxVisitor visitor = new TaxVisitor();

 visitor.accept(groceries);


 Money tax = visitor.getTax();

 Money preTaxTotal = groceries.getTotalPreTax();
```

```
 Money total = preTaxTotal.plus(tax);

//or compute tax during tax holiday
TaxVisitor holidayVisitor = new TaxHolidayVisitor();
  holidayVisitor.accept(groceries);


 Money holidayTax = holidayVisitor.getTax();

  Money holidayTotal = preTaxTotal.plus(holidayTax);
```

answered Apr 13 '15 at 1:21

dkatzel

24.2k 2 38 53

Very helpful explanation. I'm aware about "double" for money problem, it was in my question just for example. Thank you! – Alex Dn Apr 13 '15 at 6:37