Method naming in visitor design pattern

Ask Question

Here's the context/sample code of visitor design pattern.

```
public interface Visitable{
    public void accept(Visitor v);
}

public class Book implements Visitable{
    public void accept(Visitor v){
        v.visit(this);
    }
    public void read() {}
    /**
    book stuff
    **/
}
public class Movie implements Visitable{
    public void accept(Visitor v){
        v.visit(this);
    }
    public void watch() {}
    /**
    movie stuff
    **/
}

public interface Visitor{
    public void visit(Book b);
    public void visit(Movie m);
}

public class Person implements Visitor{
    public void visit(Book b){
        b.read();
    }
    public void visit(Movie m){
        m.watch();
    }
}
```

My instructor says that it's no a good idea to overload `visit` method and I should give a distinct name to each visit method that looks like the following. I'm not convinced by this idea. Can someone explain what's the downside of overloading `visit` method?

```
public interface Visitor{
    public void visitBook(Book b);
    public void visitMovie(Movie m);
}

public class Person implements Visitor{
    public void visitBook(Book b){
        b.read();
    }
    public void visitMovie(Movie m){
        m.watch();
    }
}
```

design-patterns design

asked Dec 9 '13 at 3:02

xiamx
1,716 4 17 28

1 I want to know the answer to this too because I agree with you. I've never seen an implementation that does this. Aside from reading awkwardly, I find it strange to have method names that reflect abstractions you're visiting. The parameters have that covered. – Vidya Dec 9 '13 at 3:25

@Vidya then give a up vote :D – xiamx Dec 9 '13 at 15:26

Done. Let's see what happens. – Vidya Dec 9 '13 at 15:32

3 Answers

John Vlissides's book *Pattern Hatching* (one of the GOF authors and his book is sometimes considered a supplement to *Design Patterns*) explains the advantages both both ways to implement Visitor.

The reason he says using the different variable names is this:

> A more substantial advantage comes when there's a resonalbe default behavior, and subclasses tend to override just a few of the operations. When we overload, subclasses must override *all* of the functions; otherwise your friendly C++ compiler will probably complain that your selective overrides hide one or more of the base class operations. We get around this probelm when we give Visitor operations difference names. Subclasses can then redefine a subset of the operations with impunity. - Pattern Hatching p.36

answered Dec 9 '13 at 17:02

dkatzel
24.2k 2 38 53

Thanks, that sounds like a good argument. Is this a `feature` that is unique to C++ compiler or is it the general behaviour in OO languages (C#, java)? – xiamx Dec 9 '13 at 17:13

I believe that is C++ specific behavior. For java, according to the JLS §9.4.1.: Methods are overridden on a signature-by-signature basis. `If, for example, an interface declares two public methods with the same name (§9.4.2), and a subinterface overrides one of them, the subinterface still inherits the other method.` – dkatzel Dec 9 '13 at 19:08

As per my understanding the purpose of the Visitor Design pattern was to solve the problem of single Dispatch problem. Visitor Pattern provides **Double Dispatch approach**. Dynamic dispatch is only based on the type of the calling object that is only possible using overriding not Overloading. Using Overloading will let the function depends upon the argument passed during calling which we don't want in visitor.

The whole concept of using overriding over overloading is to achieve Double Dispatch mechanism in visitor.

answered Dec 9 '13 at 10:39

Anupam Alok
174 12

1 "Dynamic dispatch is only based on the type of the calling object that is only possible using overriding not Overloading"

< not true – xiamx Dec 9 '13 at 15:24

As per my understanding Dynamic dispatch mechanism is what we call runtime polymorphism which is resolved at runtime whereas overloading is a compile time binding which is not possible for Dynamic dispatch if you can provide me some example to prove the statement wrong that would be a great help for me and to improve the concept. – Anupam Alok Nov 21 '14 at 6:32

The Gang of Four book, aka the Patterns Bible, does as your instructor advises.

> The Visitor class would be declared like this in C++:
>
> ```
> class Visitor {
> public:
>     virtual void VisitElementA(ElementA*);
>     virtual void VisitElementB(ElementB*);
>
>     // and so on for other concrete elements
> protected:
>     Visitor();
> };
> ```

Case closed.

See my point on @vadya's answer above: overloading implies that the methods do the same thing, but support that thing on different types, which is not the case with Visitor.

edited Dec 9 '13 at 16:18

answered Dec 9 '13 at 4:58

Rob

6,853 6 26 48

What's the rational behind this? I can find examples in textbooks where one uses overloading too, but that does not answer my question. – xiamx Dec 9 '13 at 14:58

I would say if you are arguing that The Gang of Four is wrong you need to provide the rationale. I generally try to avoid mentioning a type in a method where the type is a param, but I think in this case it does make sense, since the whole idea of the visitor is all the methods that do anything are in one place. – Rob Dec 9 '13 at 15:29

I adore the Go4 and the idea of design patterns as much as the next guy, but to blindly and religiously accept something without question is a bad idea in anything. Besides, the OP isn't saying the Go4 is "wrong." He is asking why his idea can't work also. This is more nuanced than a black-and-white right-or-wrong issue. Let's not forget the great lesson from Obi-Wan. – Vidya Dec 9 '13 at 15:39

lesson from Obi-Wan. – Vidya Dec 9 '13 at 15:39

Omg. Yeah Star Wars is not my ethics text of choice but thanks for that. And I think you are wrong, op is saying he thinks his instructor and Go4 are wrong. – Rob Dec 9 '13 at 15:42

2 Except that the OP never actually uses the word "wrong." He says instead he is "not convinced by this idea." I'm not convinced by the idea of fat-free ranch dressing, but I wouldn't call it "wrong" either. You seem to be inferring more than you should. – Vidya Dec 9 '13 at 15:58