Visitor Design Pattern Tutorial

Posted by Derek Banas on Nov 2, 2012 in Java Video Tutorial | 36 comments

Welcome to my Visitor Design Pattern Tutorial! This is the last part of my design pattern video tutorial.

The Visitor design pattern allows you to add methods to classes of different types without much altering to those classes. You can make completely different methods depending on the class used with this pattern.

With both the video below and the code that follows you should be able to start using this pattern in your code easily.

If you like videos like this, it helps if you tell Google with a click [googleplusone]

Share this video if you know someone who may like it

Code from the Video

VISITOR.JAVA

```java
// The visitor pattern is used when you have to perform
// the same action on many objects of different types

interface Visitor {

    // Created to automatically use the right
    // code based on the Object sent
    // Method Overloading

    public double visit(Liquor liquorItem);

    public double visit(Tobacco tobaccoItem);

    public double visit(Necessity necessityItem);

}
```

TAXVISITOR.JAVA

```java
import java.text.DecimalFormat;
```

```java

// Concrete Visitor Class

class TaxVisitor implements Visitor {

    // This formats the item prices to 2 decimal places

    DecimalFormat df = new DecimalFormat("#.##");

    // This is created so that each item is sent to the
    // right version of visit() which is required by the
    // Visitor interface and defined below

    public TaxVisitor() {
    }

    // Calculates total price based on this being taxed
    // as a liquor item

    public double visit(Liquor liquorItem) {
        System.out.println("Liquor Item: Price with Tax");
        return Double.parseDouble(df.format((liquorItem.getPrice() * .18)
+ liquorItem.getPrice()));
    }

    // Calculates total price based on this being taxed
    // as a tobacco item

    public double visit(Tobacco tobaccoItem) {
        System.out.println("Tobacco Item: Price with Tax");
        return Double.parseDouble(df.format((tobaccoItem.getPrice() * .32)
+ tobaccoItem.getPrice()));
    }

    // Calculates total price based on this being taxed
    // as a necessity item

    public double visit(Necessity necessityItem) {
        System.out.println("Necessity Item: Price with Tax");
        return Double.parseDouble(df.format(necessityItem.getPrice()));
    }

}
```

VISITABLE.JAVA

```java
interface Visitable {

    // Allows the Visitor to pass the object so
    // the right operations occur on the right
    // type of object.

    // accept() is passed the same visitor object
    // but then the method visit() is called using
    // the visitor object. The right version of visit()
    // is called because of method overloading

    public double accept(Visitor visitor);

}
```

LIQUOR.JAVA

```java
class Liquor implements Visitable {

    private double price;

    Liquor(double item) {
        price = item;
    }

    public double accept(Visitor visitor) {
        return visitor.visit(this);
    }

    public double getPrice() {
        return price;
    }

}
```

NECESSITY.JAVA

```java
class Necessity implements Visitable {

    private double price;

    Necessity(double item) {
        price = item;
    }

    public double accept(Visitor visitor) {
        return visitor.visit(this);
    }

    public double getPrice() {
        return price;
    }

}
```

TOBACCO.JAVA

```java
class Tobacco implements Visitable {

    private double price;

    Tobacco(double item) {
        price = item;
    }

    public double accept(Visitor visitor) {
        return visitor.visit(this);
    }

    public double getPrice() {
        return price;
    }

}
```

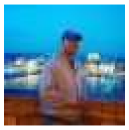
TAXHOLIDAYVISITOR.JAVA

```java
import java.text.DecimalFormat;

// Concrete Visitor Class

class TaxHolidayVisitor implements Visitor {

    // This formats the item prices to 2 decimal places

    DecimalFormat df = new DecimalFormat("#.##");

    // This is created so that each item is sent to the
    // right version of visit() which is required by the
    // Visitor interface and defined below

    public TaxHolidayVisitor() {
    }

    // Calculates total price based on this being taxed
    // as a liquor item

    public double visit(Liquor liquorItem) {
        System.out.println("Liquor Item: Price with Tax");
        return Double.parseDouble(df.format((liquorItem.getPrice() * .10)
+ liquorItem.getPrice()));
    }

    // Calculates total price based on this being taxed
    // as a tobacco item

    public double visit(Tobacco tobaccoItem) {
        System.out.println("Tobacco Item: Price with Tax");
        return Double.parseDouble(df.format((tobaccoItem.getPrice() * .30)
+ tobaccoItem.getPrice()));
    }

    // Calculates total price based on this being taxed
    // as a necessity item

    public double visit(Necessity necessityItem) {
        System.out.println("Necessity Item: Price with Tax");
        return Double.parseDouble(df.format(necessityItem.getPrice()));
    }

}
```

VISITORTEST.JAVA

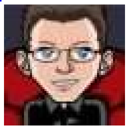
```
public class VisitorTest {
    public static void main(String[] args) {

        TaxVisitor taxCalc = new TaxVisitor();
        TaxHolidayVisitor taxHolidayCalc = new TaxHolidayVisitor();

        Necessity milk = new Necessity(3.47);
        Liquor vodka = new Liquor(11.99);
        Tobacco cigars = new Tobacco(19.99);

        System.out.println(milk.accept(taxCalc) + "\n");
        System.out.println(vodka.accept(taxCalc) + "\n");
        System.out.println(cigars.accept(taxCalc) + "\n");

        System.out.println("TAX HOLIDAY PRICES\n");

        System.out.println(milk.accept(taxHolidayCalc) + "\n");
        System.out.println(vodka.accept(taxHolidayCalc) + "\n");
        System.out.println(cigars.accept(taxHolidayCalc) + "\n");

    }
}
```

36 Responses to “Visitor Design Pattern Tutorial”

1.  *Joy* says:
   November 15, 2012 at 12:53 pm

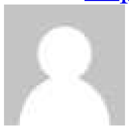
   Good job!! code and video together is a nice approach, thanks.

   Reply

   -  *admin* says:
     November 16, 2012 at 11:43 am

     Thank you 🙂 I’m glad you enjoyed it

     Reply

2.  *Paul* says:

November 19, 2012 at 1:09 am

Thank you so much for the video! You cleared up A LOT for me.

I'm still stuck on one thing though…I still don't understand what VISITABLE.JAVA does. I read the comments in the source code, but I still don't get it 🙁

Reply

- *admin* says:
  November 19, 2012 at 7:24 am

  You're very welcome. Think of visitable as a way to sneak into the classes that implement it. It would be like if you knew someone hid a key to their house under their door mat. You can go in anytime you want. visitable places a method accept in the class. Then you'll be able to call visitor.visit(this) when ever you like, which passes the class object that implements visitable with (this) to the visit method. Because I have 3 visit methods the right one is executed depending on whether a Liquor, Tobacco, or Necessity object is passed. Does that help. It is all about placing a method call in an object and then executing the right method using method overloading

  Reply

3. *MReddy* says:
   March 9, 2013 at 5:26 am

   Dear Derek,
   Thanks for the effort that you are putting to create all these videos and articles which are very informative. Can you also consider creating articles/videos on Core J2EE Design Patterns which also would be very helpful for many, across the world?

   Reply

   - *Derek Banas* says:
     March 9, 2013 at 2:11 pm

     Thank you 🙂 Yes I will cover J2EE and the patterns specific to it and all of the frameworks as well. I'll do my best to make videos as quickly as possible

     Reply

4. *Frank* says:
   March 22, 2013 at 4:15 am

   Thanks For all the videos..Very Informative..Removes any confusion..
   Thanks Again..
   Frank

   Reply

- *Derek Banas* says:
  March 22, 2013 at 9:15 am

  You’re very welcome. Thank you for stopping by my little website 🙂

  Reply

5. *boris* says:
March 29, 2013 at 1:49 am

cant wait to look over all of these videos..im excited!

Reply

- *Derek Banas* says:
  March 30, 2013 at 9:57 am

  Thank you 🙂 I hope you like them

  Reply

6. *Raj* says:
April 10, 2013 at 1:55 am

In the VisitorTest class you have called the appropriate methods by milk.accept(taxCalc) but that can be also called like taxCalc.visit(milk/vodka/cigars) or taxHolidayCalc .visit(milk/vodka/cigars) then what is the advantage here?

```
public class VisitorTest {
public static void main(String[] args) {

TaxVisitor taxCalc = new TaxVisitor();
TaxHolidayVisitor taxHolidayCalc = new TaxHolidayVisitor();

Necessity milk = new Necessity(3.47);
Liquor vodka = new Liquor(11.99);
Tobacco cigars = new Tobacco(19.99);

System.out.println(milk.accept(taxCalc) + “\n”);
System.out.println(vodka.accept(taxCalc) + “\n”);
System.out.println(cigars.accept(taxCalc) + “\n”);

System.out.println(“TAX HOLIDAY PRICES\n”);

System.out.println(milk.accept(taxHolidayCalc) + “\n”);
System.out.println(vodka.accept(taxHolidayCalc) + “\n”);
System.out.println(cigars.accept(taxHolidayCalc) + “\n”);
}
}
```

Reply

7. *Roopa* says:
May 7, 2013 at 4:19 am

   Fantastic video on design patters. Made me understand the patterns in detail . Great job !

   Reply

   - *Derek Banas* says:
     May 8, 2013 at 5:23 pm

     Thank you very much 🙂

     Reply

8. *Zeeshan Jamal* says:
July 10, 2013 at 3:40 am

   Are you also gonna make tutorials on Frameworks Like strut or spring and hibernate/JPI for developing enterprise level applications???

   Reply

   - *Derek Banas* says:
     July 12, 2013 at 6:23 am

     Yes I will cover this topics after I finish teaching how to make Android apps and games.

     Reply

9. *Ariel* says:
August 16, 2013 at 6:57 am

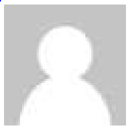
   Hi Derek, thanks so much for all these videos! greatly helping me 🙂 is there a package with all source codes from the design pattern videos?

   Reply

   - *Derek Banas* says:
     August 22, 2013 at 11:44 am

Hi, You're very welcome 🙂 Sorry, but I can't sort all of them out into one package. I do however have all the videos and links to the code on one page here Design Patterns Video Tutorial. I hope that helps

Reply

- *Ariel* says:
  September 2, 2013 at 10:14 am

  Yes it did, I actually packaged all except one.

  If you ever want to add subtitles to your videos, I can provide Hebrew.

  Thanks again.

  Reply

  - *Derek Banas* says:
    September 4, 2013 at 7:21 am

    Thank you for the subtitle offer. I'd do that if I could find an easy way to do it. I have been writing transcripts for all of my recent tutorials. I'm not sure if they help or not though?

    Reply

10. *Vishwanath* says:
September 15, 2013 at 2:23 am

Hi Derek,

I've been learning so much from these design patterns videos and thank you so much for it ! The content is just great !

As for this visitor pattern example, the visit() and accept() methods have a 'double' return value. But doesn't that restrict the kind of future-operations that could be added ?

A few other examples that I went through have the visit() and accept() methods as void-methods and store the results in instance variables within the visitor. This second approach looks to be more flexible.

So, If I were to add a DescriptionVisitor that could visit Liquor,Tobacco and Necessity classes, I wouldn't have to add a method like:
String take(Visitor)
in each of those Visitable classes.

Your thoughts ?

Cheers,
Vishwa

Reply

- [Derek Banas](#) says:
  [September 16, 2013 at 4:53 pm](#)

  Hi Vishwa,

  That is a very good point and I agree with you. The only reason why I locked them down and removed flexibility was for tutorial reasons. I have to constantly think about 2 things when I'm making these videos. First I need them to be accurate. Secondly I need them to be as easy to understand as possible.

  In my mind making something easier to understand normally forces me to make code that is less flexible. That is a shortcoming that I'm working on. That is why I constantly try to improve and I very much welcome great insights like you have provided here.

  Thank you for helping me make the tutorials better
  Derek

  [Reply](#)

11. *Praveen Rampally* says:
    [September 24, 2013 at 12:33 am](#)

    Hi Derek,
    I went through couple of videos of design patterns and OO design.And as you said in one of the videos i absolutely agree with you saying nobody does this kind of videos on you tube.

    i have a request though if you can do TDD in similar lines of OO design, that would do world of good. As i thought you would be the best person to do such stuff once i saw your OO design videos. please take a use case / user story and if you can cover end – end like how you design and related thought process before you start with initial test. as i browsed a lot and see no body as ever provided such videos on TDD. every body say a problem and show the tests. but no body tell what is the design / thought process lead them to write such tests in front. hope i didn't confused you.

    and thanks a ton for all you work.

    [Reply](#)

    - [Derek Banas](#) says:
      [September 24, 2013 at 6:00 pm](#)

      Hi Praveen,

      I did my best to explain test driven design in my Object Oriented Design tutorial. Of course it doesn't really work unless you get a bunch of people in a room in front of a white board and work through the "Real World" process. I have been trying to get people to allow me to video tape that process. As I'm sure you know it mainly involves people throwing out ideas and then we run to our computers and work them out. Then back to the white board over and over again. It is a ton of fun if you are on a good team.

      I'll do my best to get that on video. Thank you for the great request

[Reply]()

- *Anonymous* says:
  [September 24, 2013 at 11:45 pm]()

  Thanks Derek for your quick response. do you have any video / diet plan stuff for indian version as well 🙂

  [Reply]()

  - [*Derek Banas*]() says:
    [September 27, 2013 at 4:36 pm]()

    Sorry, but I'm not very good in regards to Indian cooking. I only know how to make a Green Curry with Eggplant. It is delicious to me, but I'm not sure if it is very accurate.

    [Reply]()

12. *Andre* says:
    [September 27, 2013 at 8:12 am]()

    Hi Derek,

    Great tutorials, thank you very much!

    Is there any reason why the Visitable isn't an abstract class holding the visit method?

    public abstract class Visitable {
    public double accept(Visitor visitor) {
    return visitor.visit(this);
    }
    }

    Thanks

    [Reply]()

    - [*Derek Banas*]() says:
      [September 27, 2013 at 8:39 am]()

      Thank you 🙂 Visitable is a interface because I want to force those classes that implement it into creating the accept method.

      [Reply]()

Trackbacks/Pingbacks

1. [Visitor Pattern Java Tutorial | Video Tutorials - TutsTV]() - [...] Source Code [...]

Leave a Reply

Your email address will not be published.

Comment

Name

Email

Website

Submit Comment

Search

Search

Help Me Make Free Education

Donate Crypto

Social Networks

Facebook

YouTube

Twitter

LinkedIn

Buy me a Cup of Coffee

"Donations help me to keep the site running. One dollar is greatly appreciated." - (Pay Pal Secured)

My Facebook Page

Archives

- March 2022
- February 2022
- January 2022
- June 2021
- May 2021
- April 2021
- March 2021
- February 2021
- January 2021
- December 2020
- November 2020
- October 2020
- September 2020
- August 2020

- July 2020
- June 2020
- May 2020
- April 2020
- March 2020
- February 2020
- January 2020
- December 2019
- November 2019
- October 2019
- August 2019
- July 2019
- June 2019
- May 2019
- April 2019
- March 2019
- February 2019
- January 2019
- December 2018
- October 2018
- September 2018
- August 2018
- July 2018
- June 2018
- May 2018
- April 2018
- March 2018
- February 2018
- January 2018
- December 2017
- November 2017
- October 2017
- September 2017
- August 2017
- July 2017
- June 2017
- May 2017
- April 2017
- March 2017
- February 2017
- January 2017
- December 2016
- November 2016
- October 2016
- September 2016
- August 2016
- July 2016
- June 2016
- May 2016
- April 2016
- March 2016
- February 2016
- January 2016
- December 2015

- November 2015
- October 2015
- September 2015
- August 2015
- July 2015
- June 2015
- May 2015
- April 2015
- March 2015
- February 2015
- January 2015
- December 2014
- November 2014
- October 2014
- September 2014
- August 2014
- July 2014
- June 2014
- May 2014
- April 2014
- March 2014
- February 2014
- January 2014
- December 2013
- November 2013
- October 2013
- September 2013
- August 2013
- July 2013
- June 2013
- May 2013
- April 2013
- March 2013
- February 2013
- January 2013
- December 2012
- November 2012
- October 2012
- September 2012
- August 2012
- July 2012
- June 2012
- May 2012
- April 2012
- March 2012
- February 2012
- January 2012
- December 2011
- November 2011
- October 2011
- September 2011
- August 2011
- July 2011
- June 2011