Strategy Design Pattern Tutorial

Posted by Derek Banas on Aug 24, 2012 in Java Video Tutorial | 113 comments

Here is my Strategy design patterns tutorial. You use this pattern if you need to dynamically change an algorithm used by an object at run time. Don't worry, just watch the video and you'll get it.

The pattern also allows you to eliminate code duplication. It separates behavior from super and subclasses. It is a super design pattern and is often the first one taught.

All of the code follows the video to help you learn.

If you liked this video, tell Google so more people can see it [googleplusone]

Sharing is nice

Code & Comments from the Video

ANIMAL.JAVA

```
01 public class Animal {
02
03     private String name;
04     private double height;
05     private int weight;
06     private String favFood;
07     private double speed;
08     private String sound;
09
10     // Instead of using an interface in a traditional way
11     // we use an instance variable that is a subclass
12     // of the Flys interface.
13
14     // Animal doesn't care what flyingType does, it just
15     // knows the behavior is available to its subclasses
16
17     // This is known as Composition : Instead of inheriting
18     // an ability through inheritance the class is composed
19     // with Objects with the right ability
20
21     // Composition allows you to change the capabilities of
22     // objects at run time!
23
24     public Flys flyingType;
25
26     public void setName(String newName){ name = newName; }
```

```
27      public String getName(){ return name; }
28
29      public void setHeight(double newHeight){ height = newHeight; }
30      public double getHeight(){ return height; }
31
32      public void setWeight(int newWeight){
33          if (newWeight > 0){
34              weight = newWeight;
35          } else {
36              System.out.println("Weight must be bigger than 0");
37          }
38      }
39      public double getWeight(){ return weight; }
40
41      public void setFavFood(String newFavFood){ favFood = newFavFood; }
42      public String getFavFood(){ return favFood; }
43
44      public void setSpeed(double newSpeed){ speed = newSpeed; }
45      public double getSpeed(){ return speed; }
46
47      public void setSound(String newSound){ sound = newSound; }
48      public String getSound(){ return sound; }
49
50      /* BAD
51      * You don't want to add methods to the super class.
52      * You need to separate what is different between subclasses
53      * and the super class
54      public void fly(){
55
56          System.out.println("I'm flying");
57
58      }
59      */
60
61      // Animal pushes off the responsibility for flying to flyingType
62
63      public String tryToFly(){
64
65          return flyingType.fly();
66
67      }
68
69      // If you want to be able to change the flyingType dynamically
70      // add the following method
71
72      public void setFlyingAbility(Flys newFlyType){
73
74          flyingType = newFlyType;
75
76      }
77
78  }
```

DOG.JAVA

```
01 public class Dog extends Animal{
02
03     public void digHole(){
04
05         System.out.println("Dug a hole");
06
07     }
08
09     public Dog(){
10
11         super();
12
13         setSound("Bark");
14
15         // We set the Flys interface polymorphically
16         // This sets the behavior as a non-flying Animal
17
18         flyingType = new CantFly();
19
20     }
21
22     /* BAD
23     * You could override the fly method, but we are breaking
24     * the rule that we need to abstract what is different to
25     * the subclasses
26     *
27     public void fly(){
28
29         System.out.println("I can't fly");
30
31     }
32     */
33
34 }
```

BIRD.JAVA

```
01 public class Bird extends Animal{
02
03     // The constructor initializes all objects
04
05     public Bird(){
06
07         super();
08
09         setSound("Tweet");
10
11         // We set the Flys interface polymorphically
12         // This sets the behavior as a non-flying Animal
13
14         flyingType = new ItFlys();
15
16     }
17
```

```
}
```

FLYS.JAVA

```
// The interface is implemented by many other
// subclasses that allow for many types of flying
// without effecting Animal, or Flys.

// Classes that implement new Flys interface
// subclasses can allow other classes to use
// that code eliminating code duplication

// I'm decoupling : encapsulating the concept that varies

public interface Flys {

   String fly();

}

// Class used if the Animal can fly

class ItFlys implements Flys{

    public String fly() {

        return "Flying High";

    }

}

//Class used if the Animal can't fly

class CantFly implements Flys{

    public String fly() {

        return "I can't fly";

    }

}
```

ANIMALPLAY.JAVA

```
public class AnimalPlay{

    public static void main(String[] args){

        Animal sparky = new Dog();
        Animal tweety = new Bird();

        System.out.println("Dog: " + sparky.tryToFly());
```

```
09
10          System.out.println("Bird: " + tweety.tryToFly());
11
12          // This allows dynamic changes for flyingType
13
14          sparky.setFlyingAbility(new ItFlys());
15
16          System.out.println("Dog: " + sparky.tryToFly());
17
18      }
19
20  }
```

113 Responses to “Strategy Design Pattern Tutorial”

1. *Saleh* says:
[September 28, 2012 at 9:52 am]

It is the first time I encounter this design pattern which uses interface in a nice way.

Reply

   -  *admin* says:
   [September 28, 2012 at 1:15 pm]

   I’m glad to hear that. I’ll cover many more. If you are into thinking about building great software, you’ll love the tutorials I make over the next few months 🙂

   Reply

2. *Thuy Nguyen* says:
[November 12, 2012 at 4:25 pm]

You’re awesome! Thank you so much for your tutorials.

Reply

   -  *admin* says:
   [November 12, 2012 at 4:59 pm]

   You’re very welcome 🙂 Thank you for the kind words

   Reply

3. *Trini Allan* says:
[December 5, 2012 at 6:04 pm]

Your tutorials are pretty awesome, the best stuff I've seen on youtube hands down…, anyway thanks alot for taking the time to post. I am dutifully reviewing the code as you recommended and I came across something I wasn't sure of…

In the comments line above you mention the following:

// without effecting Animal, or Flys.

Not sure if it is my novice thinking or not, but wanted to know if "effecting" or "affecting" was the word intended here.

Thanks for clarifying,
Allan.

Reply

-  *admin* says:
  December 5, 2012 at 6:13 pm

  Thank you very much 🙂 I do my best to make the best videos I can. In regards to your question, honestly I don't know the difference between the two words. I'm really good at a few things, but grammar has never been my strong point. Sorry about that

  Reply

-  *se7en* says:
  January 23, 2014 at 5:55 pm

  Alan, it's clearly "effecting" meaning – "to have consequence"
  Derek was right the first time, maybe you should look up "pedantic" 🙂

  Reply

  -  *Minae* says:
    March 17, 2014 at 8:31 am

    Actually, "affecting" would be right in this case.

    "Effecting" means to bring about, or to cause/create . Affecting means to influence, or to cause/create a change in . You are not bringing Animal about. You are causing a change in Animal. If you wanted to use "effecting", you could, but you would have to say it like this: "without effecting a change in Animal".

    Reply

4.  *Jayant* says:
   January 29, 2013 at 12:18 pm

   Thank you very much for the tutorials they are just awesome !!! Learnt them pretty fast.

Reply

-  *admin* says:
  January 29, 2013 at 1:55 pm

  You're very welcome 🙂

  Reply

5.  *Jinil* says:
   February 11, 2013 at 10:01 pm

   Nice tutorials. I think, the best video tutorials I have ever seen in YouTube. Thank you so much boss. 🙂
   Keep up the good work.

   Reply

   -  *Derek Banas* says:
     February 12, 2013 at 11:27 am

     Thank you very much 🙂 most people don't know about my videos so I'm happy you found and liked them

     Reply

6.  *Venkatesh* says:
   February 25, 2013 at 6:59 pm

   Thank you for the awesome videos on design pattern. well done!!!
   You have done amazing videos and very simple to understand with the code. Thanks a lot 🙂

   Reply

   -  *Derek Banas* says:
     February 26, 2013 at 11:05 am

     You're very welcome 🙂 I'm glad you liked them

     Reply

7.  *DoubleYou* says:
   February 26, 2013 at 7:32 am

   First of all; great tutorials!!

Watched most of the design patterns. Had just one question about the explanation at this pattern. You're talking about Composition (Animal.java) but i had the feeling it was Aggregation? I find it hard to tell the difference. (I know about there definitions)

Reply

-  *Derek Banas* says:
  February 26, 2013 at 11:10 am

  Thank you 🙂 yes I misspoke and you are correct. Most people just refer to everything as composition, but you understand the difference.

  Reply

  -  *DoubleYou* says:
    February 26, 2013 at 12:00 pm

    Thank you for the quick reply!!

    it was probably…. you….that had told me the differences in some tutorial ;). i'll continue with the refactoring tutorials and i am looking forward to the upcoming Android tuts! Your videos are easy to follow and nicely backed by the code on the website. Amazing job!

    Reply

    -  *Derek Banas* says:
      February 26, 2013 at 5:00 pm

      Thank you very much 🙂 I do my best to present everything in an understandable format. I glad you like the videos

      Reply

8.  *Hoang Edward* says:
   March 19, 2013 at 8:52 pm

   Firstly, thanks for your great video and enclosed source code, but I also have a question for you:
   Why don't you encapsulate the flyingType field in the Animal class with protected modifier so that the sub-classes can directly access to this field but the clients can't, they must use the setFlyingAbility method instead.

   Reply

   -  *Derek Banas* says:
     March 20, 2013 at 7:09 am

You could definitely do that. There are many ways to create each design pattern. They are but a guide for writing flexible code

Reply

9.  *Joe Contreras* says:
March 30, 2013 at 4:59 pm

Great Video.

Question:

I want to design an object for exporting a file to XLS; CSV; Word in ASP.NET. They have the same properties but have different values. The initial reaction is to use a Switch statement which breaks the Open-Close principle. However I don't want to create an object for each export type during implementation. Instead I want to take the value from the button click event for the export process and using that value create a reference that evaluates the value being passed and determine which object to retrieve like an XLS export object; CSV export etc.

Feedback on design would be great

Reply

-  *Derek Banas* says:
April 2, 2013 at 6:22 am

I'm sorry, but I haven't used ASP for many years and I don't think I could help you. Sorry

Reply

10.  *DanMan* says:
April 15, 2013 at 9:53 am

I'm confused.. how does the Animal class know about Flys.. maybe I need to watch the video again.

Reply

-  *Derek Banas* says:
April 16, 2013 at 4:10 pm

Flys flyingType is stored in every Animal object as a field. In the Bird class we then give it the ability to fly flyingType = new ItFlys().

I cover the strategy pattern again here Code Refactoring Strategy

I hope that helps

Reply

11.  *Vivek* says:
[April 24, 2013 at 9:28 pm]

Great stuff ! Thanks

[Reply]

    -  *Derek Banas* says:
    [April 26, 2013 at 6:47 am]

    Thank you very much 🙂

    [Reply]

12.  *Radhika* says:
[May 19, 2013 at 8:22 am]

Thank you so much for your time and effort! 🙂 This really helps!

[Reply]

    -  *Derek Banas* says:
    [May 19, 2013 at 1:18 pm]

    You're very welcome 🙂 I'm glad I was able to help

    [Reply]

13.  *Phoenix* says:
[May 29, 2013 at 6:55 pm]

Hello Derek,

Very good explanation of Strategy Design Pattern. I have one question regarding this. Can strategy class have access to members of animal class? If yes how should animal class access should be given to strategy class?

[Reply]

    -  *Derek Banas* says:
    [May 31, 2013 at 11:37 am]

    Thank you 🙂 If by doing that you are increasing coupling, then that should be avoided. i hope that helps

Reply

-  *Phoenix* says:
  June 1, 2013 at 3:16 pm

  Darek,

  In my application I need access members of Animal class in strategy class? I am planning to use interface which gives only few members access to strategy. Is it good for avoiding coupling or need to something else?

  Reply

  -  *Derek Banas* says:
    June 4, 2013 at 6:26 am

    I may not be understanding the question. The strategy pattern is used to separate behavior from the super and subclasses. So, it would defeat the point if they communicated directly with each other.

    Reply

14.  *Muhammad* says:
    June 3, 2013 at 5:21 am

    Thanks for this perfect tutorial, and for your effort.

    Reply

    -  *Derek Banas* says:
      June 4, 2013 at 5:45 am

      You're very welcome 🙂 Thank you for visiting!

      Reply

15.  *Anuj* says:
    June 7, 2013 at 4:52 am

    amazing explanation…. 🙂 Thanx 🙂

    Reply

    -  *Derek Banas* says:
      June 10, 2013 at 7:37 am

Thank you 🙂

Reply

16.  *Prasanna* says:
June 16, 2013 at 5:48 am

Very good explanation. I really liked the way you presented the patterns. Keep up the good work Sir.

Reply

-  *Derek Banas* says:
June 18, 2013 at 6:27 pm

Thank you very much 🙂

Reply

17.  *Srinivasan* says:
June 28, 2013 at 2:00 am

Great Tutorial Brian. Thanks a lot!

Reply

18.  *Ariel* says:
June 28, 2013 at 5:08 pm

Congratulations. This is the firt video I watched, and now I will carry on watching all the rest. Great staff!

Reply

-  *Derek Banas* says:
June 30, 2013 at 8:57 am

Thank you very much 🙂 I did my best to make the design patterns easy to understand and fun to learn about

Reply

19.  *Chow* says:
July 17, 2013 at 6:03 pm

Great stuff, sir. Awesome voice too!

Reply

-  *Derek Banas* says:
  July 19, 2013 at 1:11 pm

  Thank you 🙂

  Reply

20.  *Rey* says:
August 7, 2013 at 5:32 am

Great tutorial. Thanks for sharing your wisdom.

Reply

-  *Derek Banas* says:
  August 8, 2013 at 10:55 am

  Thank you for the compliment 🙂 You're very welcome

  Reply

21.  *Rajesh H* says:
August 7, 2013 at 10:21 pm

Hi Derek,

These days I am working n C# and I had used Java around 7 years back in my college days, but you explained things with such a beauty that it helped me a lot.

Hats off to you… Excellent Job
I am a fan of yours… Cheers

Reply

-  *Derek Banas* says:
  August 8, 2013 at 10:52 am

  Thank you very much 🙂 I try to do the best I can.

  Reply

22.  *varanak* says:
August 9, 2013 at 11:55 am

Awesome work !

Reply

-  *Derek Banas* says:
  August 10, 2013 at 1:08 pm

  Thank you very much 🙂

  Reply

23.  *da Hacedor* says:
August 24, 2013 at 12:04 pm

I've just started with design patterns. To begin with, I was told by my friends to start with some book but after going through first four videos on your youtube channel, I have cancelled the order for that book. I find this sufficient enough to start implementing the patterns.
Thanks for all the stuff. And, please accept my congratulations for the same.

Reply

-  *Derek Banas* says:
  August 28, 2013 at 1:41 pm

  I'm very happy that you are enjoying the design patterns videos. I also cover Refactoring, which is normally covered after design patterns.

  You're very welcome – Derek

  Reply

24.  *Byron Sanchez* says:
September 20, 2013 at 12:24 pm

I'm working through memorizing every design pattern I can and you've made it that much easier. I really appreciate the work on the videos. Thank you!

Reply

-  *Derek Banas* says:
  September 22, 2013 at 3:42 pm

  I'm very happy I have been able to help 🙂 You're very welcome.

  Reply

25.  *Stas* says:
October 5, 2013 at 2:19 am

Great video!
I've started reading some book about Design Patterns, but it is lack of simple examples as you show in your tutorials!

would it make any difference, if I would make the Fly interface as an abstract class & ItsFly,CantFly as derived classes of Fly?

Reply

26.  *Badhri* says:
October 19, 2013 at 12:23 pm

Awesome!… I referred multiple examples but I was not able to understand the real benefit of this pattern. But your video gave me a very clear and thorough understanding. Great work and thanks for all your effort on this!…

Reply

-  *Derek Banas* says:
October 22, 2013 at 12:52 pm

Thank you 🙂 I'm very happy that I was able to clear it up

Reply

27.  *Sriram* says:
December 6, 2013 at 10:27 pm

Well, this is so much better than reading those theoretical books which do not give much insight into the code. The best tutorial for sure! Thanks 🙂

Reply

-  *Derek Banas* says:
December 7, 2013 at 1:40 pm

Thank you 🙂 I'm glad you enjoyed it.

Reply

28.  *Marc-André* says:
December 12, 2013 at 10:48 pm

Waw! Another great video!

I got a final exam tommorow at 9 and you did a tutorial about every pattern we have covered during the semester !

Definetly the most effective study method i've used in many years!

Thanks again to use those simple examples, every teacher should be like you!

Reply

-  *Derek Banas* says:
  December 13, 2013 at 8:21 am

  I'm very happy that I was able to help. Good luck on your exams 🙂

  Reply

29.  *Tirtha Chakraborty* says:
December 28, 2013 at 10:54 am

Your explanation and so easy to remember technique is something that can't be thanked enough. Thanks a lottt..

Reply

-  *Derek Banas* says:
  December 29, 2013 at 4:23 pm

  Thank you for the nice compliment 🙂 You're very welcome.

  Reply

30.  *shiva* says:
January 1, 2014 at 3:14 am

hey Thanks for bringing up this tutorial,it is very easy to understand, I liked it now i can learn design pattern very fast.

Reply

-  *Derek Banas* says:
  January 1, 2014 at 8:53 am

  You're very welcome 🙂 I'm glad I was able to clear up this topic for so many people

  Reply

31.  *weldu Hashenge* says:
February 1, 2014 at 1:47 pm

Sir,

I was wondering how the Animal.java know about the Flys interface???I could n't understand how they are related….
Thank you for clarifying

Reply

-  *Derek Banas* says:
  February 3, 2014 at 6:57 pm

  Take a look at this line

  // Composition allows you to change the capabilities of objects at run time
  public Flys flyingType;

  The Flys object is stored in every Animal. It can then be changed if needed without disturbing the code

  Reply

32.  *Sarunas* says:
February 18, 2014 at 12:38 am

I have only one word for you: YourTutorialsAreGreat!! Keep it up. I'm going to watch all of them..

Reply

-  *Derek Banas* says:
  February 18, 2014 at 6:01 pm

  Thank you very much 🙂 I'm happy that you like them.

  Reply

33.  *Anonymous* says:
March 1, 2014 at 11:10 am

Thanks for the videos. Great job Derek!
Code is very helpful in understanding the design patterns. It will be very helpful if we can download the slides as well. They have some very important information such as when to use particular design pattern. If they are available for download, can you please share the link?
Thanks

Reply

34.  *Sam* says:

March 1, 2014 at 11:12 am

Thanks for the videos. Great job Derek!
Code is very helpful in understanding the design patterns. It will be very helpful if we can download the slides as well. They have some very important information such as when to use particular design pattern. If they are available for download, can you please share the link?
Thanks

Reply

-  *Derek Banas* says:
  March 4, 2014 at 6:57 pm

  You're very welcome 🙂 I have all of them here. They aren't neat because I never expected to give them away, but every slide is here.

  Reply

35.  *Pradeep* says:
April 19, 2014 at 9:15 am

I have started design pattern series, Thanks Derek.
Now I am clear on Strategy pattern, I have just one question.
What is Association and Aggregation, and what is difference between these with Composition.

Thanks
Pradeep

Reply

-  *Derek Banas* says:
  April 19, 2014 at 5:30 pm

  Hi Pradeep,

  You're very welcome 🙂 This diagram explains everything you asked about with examples as well. I hope it helps.

  Reply

  -  *Pradeep* says:
    April 20, 2014 at 9:30 pm

    Thanks Derek,The diagram helps lot, Now clear on Association, Aggregation and Composition 🙂 .

    Thanks
    Pradeep

Reply

-  *Derek Banas* says:
  April 21, 2014 at 7:53 am

  Great I'm glad it helped 🙂

  Reply

36.  *Asi* says:
April 20, 2014 at 4:45 am

Hi Derek,
thank you very much for your amazing videos… I'm learning so much thank to you.. i just can't tell you how glad I am for came across your videos!

♥

Reply

-  *Derek Banas* says:
  April 20, 2014 at 8:38 am

  Thank you 🙂 I'm very happy that you enjoy them. Many more are coming.

  Reply

37.  *Martin Gwarada* says:
April 20, 2014 at 1:49 pm

Derek, thank you buddy. You are the best!!!

Reply

-  *Derek Banas* says:
  April 20, 2014 at 4:56 pm

  Thank you 🙂

  Reply

38.  *Don Cooper* says:
April 24, 2014 at 7:55 pm

Very good series! You are truly demystifying an important topic. Above, in the comments here to the Strategy Pattern, you say, "The strategy pattern is used to separate behavior from the super and

subclasses.” With that one sentence, I finally clearly understood the purpose of it. Often, with design patterns, teachers focus get lost in explaining the mechanics, and the reason-to-use gets lost in the wash of information. Can you make a list of these kinds succinct formulations, and post them on your site? ie, one sentence per each design pattern. I think for certain patterns, like Strategy, the “why” is not obvious whereas the mechanics are simple. I would truly appreciate it if you focused like a laser on the why’s, so that we know when to use a given pattern. And, as you do in your video, please state the why’s in different ways (perhaps two or more sentences instead of one) so that the clarification may come from different angles.

Thanks again for your excellent efforts!

Reply

-  *Derek Banas* says:
  April 25, 2014 at 5:33 pm

  Thank you 🙂 It is always great to hear that I was able to clear everything up for people. I’ll see what I can do about that list. These tutorials became popular many months after I originally posted them, so I stopped short of making a video like you requested. I’ll see what i can do now.

  Reply

  -  *Don Cooper* says:
    April 26, 2014 at 2:02 pm

    Derek,

    Glad to hear! I will look forward to that list very much.

    I was thinking of a more complete sentence to describe the Strategy pattern, “The strategy pattern is used to separate behavior from the super and subclasses, when you want to invoke this dissimilar behavior with the same function in the client (making use of polymorphism).” Also, I was reading http://oodesign.com on the Strategy pattern page (http://oodesign.co/strategy-pattern.html), and it says there that the Strategy pattern and the Bridge pattern have the same UML diagram, “but differ in their intent”. Again, another example of the teacher being clear on the easy stuff (mechanics/structure) but using jargon, almost maniacally trying to keep secret the main point: why. Why would you use one or the other? This is where the esoteric side of design patterns starts to lose us regular coders who are not academics, but simply want to write better code. The jargon starts to get too thick, and the reasons for using a particular pattern are obscured. We want credible examples and, more importantly, scenarios indicating when a given pattern is appropriate.

    Thanks for listening to my mini rant.

    Reply

    -  *Derek Banas* says:
      April 27, 2014 at 7:09 am

      I’ll try to explain them in simple terms.

The Bridge Pattern allows you to create 2 types of abstract objects that can interact with each other in numerous extendable ways. In my bridge design pattern example I created a way for an infinite variety of remote controls to interact with an infinite variety of different devices.

The strategy pattern allows you to change what an object can do while the program is running. In my strategy design pattern example I showed how you can add an object that represents whether an Animal can fly or not to the Animal class. This attribute was then added to every class that extended from Animal. Then at run time I was able to decide if an Animal could fly or not.

I hope that helps 🙂

Reply

39.  *Andrea lee* says:
April 28, 2014 at 11:19 am

I am a little bit confused on the use of strategy and factory both created at runtime. Kindly please elaborate the difference.
Thank you in advance and I love your videos.

Reply

-  *Derek Banas* says:
April 28, 2014 at 4:12 pm

In my strategy design pattern tutorial I demonstrated how you can at run time give any Animal subclass the ability to fly.

1. I added the Flys object to Animal which all subclasses then receive with : public Flys flyingType;

2. ItFlys and CantFly implement the Flys interface

3. At runtime I can change the version of Flys : sparky.setFlyingAbility(new ItFlys());

4. Now sparky can fly

With the Factory pattern tutorial I showed how the EnemyShipFactory pops out a different type of ship based off of user input :

```
EnemyShip newShip = null;
13
14 if (newShipType.equals("U")){
15
16 return new UFOEnemyShip();
17
18 } else
19
20 if (newShipType.equals("R")){
21
22 return new RocketEnemyShip();
```

```
23
24 } else
25
26 if (newShipType.equals("B")){
27
28 return new BigUFOEnemyShip();
29
30 } else return null;
```

– See more at: http://www.newthinktank.com/2012/09/factory-design-pattern-tutorial/#sthash.aTZJT3zG.dpuf

I hope that helps 🙂

Reply

-  *Andrea lee* says:
  April 28, 2014 at 7:12 pm

  Thank you for your explanation. I have another question if you don't mind. Why some examples use factory in calculator program and others use strategy?

  Reply

  -  *Derek Banas* says:
    April 29, 2014 at 4:08 pm

    Most patterns have the same goal which is to add flexibility or to streamline the code. They can very often be used interchangeably.

    Reply

    -  *Andrea lee* says:
      April 30, 2014 at 2:20 am

      Thank you very much. For almost 12 hours I have been going back and forth to understanding strategy n factory. They are almost closely the same even their uml.
      Your video are good and I also like the code refactoring.

      When are you going to start j2ee, spring, hibernate…..I guess many are eager for those videos. I hope soon and hopefully it will happen this year

      Reply

      -  *Derek Banas* says:
        April 30, 2014 at 11:13 am

        You're very welcome 🙂 Before Java Enterprise I want to make an Android tutorial that teaches everything in a format in which anyone will

be able to make any Android app they can imagine. Sorry it is taking so long, but I'm very serious about teaching Android right now.

Reply

40.  *Sandeep* says:
June 1, 2014 at 2:36 pm

You are like the best Tutor I've ever seen. The quality of the lessons and the way you teach is amazing. Thanks a ton

Reply

-  *Derek Banas* says:
June 2, 2014 at 7:12 am

Thank you very much 🙂

Reply

41.  *Anon* says:
June 7, 2014 at 3:24 am

Hi, our professor taught us this pattern but I still don't get on how do I access the dog digHole() method if I used the class Animal other than forcing it.

((Dog) Sparky).digHole(); which is not appropriate 🙂

Reply

-  *Derek Banas* says:
June 8, 2014 at 4:27 pm

I'm confused. That code you have there does work. You wouldn't have to cast it though if you had the original method in the super class though.

Reply

42.  *Karthick* says:
June 8, 2014 at 1:53 am

Great video and awesome teaching!! Thanks a lot for taking the effort and making things very interesting 😀

Reply

-  *Derek Banas* says:
  June 8, 2014 at 4:21 pm

  Thank you very much 🙂

  Reply

43.  *Sunil* says:
  July 6, 2014 at 7:36 am

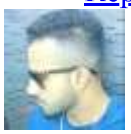
  Awesome tutorial, Clearly described. Thanks a lot for your videos.

  Reply

  -  *Derek Banas* says:
    July 6, 2014 at 7:45 am

    Thank you 🙂 You're very welcome

    Reply

44.  *zal* says:
  July 12, 2014 at 10:11 pm

  Hey Derek! I have been watching most of your OOP videos. well, I am not proud of myself for not having the courtesy to thank you until now.
  you r an amazing teacher. I will soon convert all the code you used in Design Pattern videos into Php and put on github. I will let u know when I finish 🙂

  Reply

  -  *Derek Banas* says:
    July 17, 2014 at 8:30 am

    Thank you for taking the time to tell me that you have found them useful. I greatly appreciate that. Yes definitely post your link.

    Reply

45.  *Raf* says:
  July 22, 2014 at 12:58 pm

  Hello Derek,

  Thank you for these videos they are very informative and helpful.

Can you please explain to me why you have the flyingType variable is public? If it is public why do you need the setter method?

Reply

-  *Derek Banas* says:
  July 25, 2014 at 6:41 pm

  You're very welcome 🙂 It didn't need to be public.

  Reply

46.  *Pratik Prasad* says:
July 22, 2014 at 9:08 pm

Really really nice tutorials, I love watching your tutorials on Design Patterns, I have planned that I'll watch all of your tutorials on youtube.
Thanks a lot for all the tutorials 🙂

Can you please tell me the screencasting software you used to create these tutorials

Reply

-  *Derek Banas* says:
  July 25, 2014 at 6:39 pm

  Thank you 🙂 I use Camtasia 2 to record.

  Reply

47.  *Mazen* says:
October 1, 2014 at 1:44 am

would be really nice if you redo the videos in C#. I understand that most concept are the same but still better for and for a whole lot of newbies. thank you so much for videos, really great stuff although you are talking little bit fast 🙂

Reply

-  *Derek Banas* says:
  October 3, 2014 at 9:45 am

  I'm working on a C# tutorial. I'll see what I can do patterns wise as well.

  Reply

48.  *Nick* says:
[October 1, 2014 at 9:25 am]

As a current Design Patterns student going through Head First Design Patterns, your videos are a helpful reinforcement. Thanks!

[Reply]

-  *Derek Banas* says:
[October 3, 2014 at 9:41 am]

Thank you 🙂 I'm very glad that I could help

[Reply]

49.  *Denis* says:
[October 21, 2014 at 9:15 am]

Thanks! This videos are many times more interesting for me, that the booring books and lectures))) In addition, it helps to learn English (it's not my native language).

[Reply]

-  *Derek Banas* says:
[October 29, 2014 at 10:46 am]

You're very welcome 🙂

[Reply]

50.  *Mostafa* says:
[January 16, 2015 at 6:24 pm]

Banas for president,

Thank you for your help

[Reply]

-  *Derek Banas* says:
[January 24, 2015 at 6:16 am]

That's funny 🙂 You're very welcome.

[Reply]

51.  *David N.* says:
[January 20, 2015 at 1:19 pm](#)

What if you just had an interface Flys, and have Bird implement it. Can you elaborate on why this option is duplicate code, and not good. Thanks.

[Reply](#)

-  *Derek Banas* says:
[January 24, 2015 at 6:00 am](#)

You could definitely do that instead. I just wanted to demonstrate the pattern in a simple way.

[Reply](#)

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
- May 2011
- April 2011
- March 2011
- February 2011
- January 2011
- December 2010
- November 2010
- October 2010
- September 2010
- August 2010
- July 2010
- June 2010
- May 2010
- April 2010
- March 2010
- February 2010
- January 2010
- December 2009