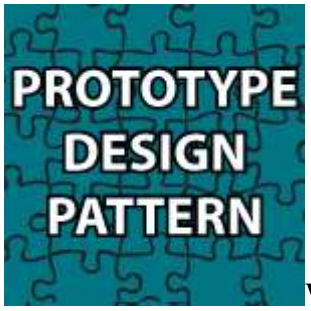
Prototype Design Pattern Tutorial

Posted by Derek Banas on Sep 15, 2012 in Java Video Tutorial | 20 comments



Welcome to my Prototype Design Pattern Tutorial. The Prototype design pattern is used for creating new objects (instances) by cloning (copying) other objects.

It allows for the adding of any subclass instance of a known super class at run time. It is used when there are numerous potential classes that you want to only use if needed at runtime. The major benefit of using the Prototype pattern is that it reduces the need for creating potentially unneeded subclasses.

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

If you like videos like this, please tell Google [googleplusone]

Sharing is super great

Code from the Video

ANIMAL.JAVA

```
01 // By making this class cloneable you are telling Java
02 // that it is ok to copy instances of this class
03 // These instance copies have different results when
04 // System.identityHashCode(System.identityHashCode(bike))
05 // is called
06
07 public interface Animal extends Cloneable {
08
09     public Animal makeCopy();
10
11 }
```

SHEEP.JAVA

```
01 public class Sheep implements Animal {
02
03     public Sheep(){
04
05         System.out.println("Sheep is Made");
06
07     }
08
09     public Animal makeCopy() {
10
11         System.out.println("Sheep is Being Made");
12
```

```
        Sheep sheepObject = null;

        try {

            // Calls the Animal super classes clone()
            // Then casts the results to Sheep

            sheepObject = (Sheep) super.clone();

        }

        // If Animal didn't extend Cloneable this error
        // is thrown

        catch (CloneNotSupportedException e) {

            System.out.println("The Sheep was Turned to Mush");

            e.printStackTrace();

         }

        return sheepObject;
    }

    public String toString(){

        return "Dolly is my Hero, Baaaaa";

    }

}
```

CLONEFACTORY.JAVA

```
public class CloneFactory {

    // Receives any Animal, or Animal subclass and
    // makes a copy of it and stores it in its own
    // location in memory

    // CloneFactory has no idea what these objects are
    // except that they are subclasses of Animal

    public Animal getClone(Animal animalSample) {

        // Because of Polymorphism the Sheeps makeCopy()
        // is called here instead of Animals

        return animalSample.makeCopy();

    }

}
```

TESTCLONING.JAVA

```java
public class TestCloning {

    public static void main(String[] args){

        // Handles routing makeCopy method calls to the
        // right subclasses of Animal

        CloneFactory animalMaker = new CloneFactory();

        // Creates a new Sheep instance

        Sheep sally = new Sheep();

        // Creates a clone of Sally and stores it in its own
        // memory location

        Sheep clonedSheep = (Sheep) animalMaker.getClone(sally);

        // These are exact copies of each other

        System.out.println(sally);

        System.out.println(clonedSheep);

        System.out.println("Sally HashCode: " +
System.identityHashCode(System.identityHashCode(sally)));

        System.out.println("Clone HashCode: " +
System.identityHashCode(System.identityHashCode(clonedSheep)));
    }

}
```

20 Responses to “Prototype Design Pattern Tutorial”

1.  *Shriram* says:
   November 27, 2012 at 10:08 am

   I just came across your design pattern tutorials.You have explained the concepts in a simple manner and its amazing..!!!

   Looking forward to view and learn all your videos….

   Great JOB..

   Reply

   -  *admin* says:

November 27, 2012 at 4:57 pm

Thank you very much 🙂 I did my best to make them easy to understand. I'll dive into using them and recognizing when they can help in my next tutorial on refactoring. Thank you for taking the time to tell me you liked them

Reply

2.  *Arun* says:
February 10, 2013 at 11:52 pm

I really appreciate the way you have explained the java design pattern. It clears all my doubts.

Thank you again for your awesome description..:)

Reply

   -  *Derek Banas* says:
February 11, 2013 at 4:36 pm

You are very welcome and thank you for taking the time to tell me the tutorial helped 🙂

Reply

3.  *neha* says:
February 27, 2013 at 12:52 pm

Very good, clean and clear tutorial. Best part is the availability of the code to be practiced by the learner. If possible, if you include class diagram then it would be perfect.

Reply

   -  *Derek Banas* says:
February 27, 2013 at 5:15 pm

Thank you very much 🙂 I'll see what I can do about the UML diagrams.

Reply

4.  *rotem* says:
March 28, 2013 at 9:01 am

Realy helpful!! thank you..(:

Reply

-  *Derek Banas* says:
  March 30, 2013 at 12:13 pm

  You're welcome 🙂

  Reply

5.  *Helton* says:
   March 30, 2013 at 12:56 pm

   Hi, Derek!

   Your tutorials are awesome! Keep the good work.
   I've a suggestion…
   In the classes Dog and Sheep you can return Dog and Sheep types (respectively) in makeCopy method, because they're subtypes of Animal:

   ```
   //Dog.java file
   @Override
   public Dog makeCopy() {
   (…)
   }

   //Sheep.java file
   @Override
   public Sheep makeCopy() {
   (…)
   }
   ```

   So, you'll avoid cast to this types later, but you won't able to use CloneFactory without do the cast. It'll seems like that:

   ```
   Sheep clonedSheep = sally.makeCopy();
   ```

   []'s
   Helton

   Reply

6.  *Inderjeet* says:
   March 31, 2013 at 12:13 pm

   Hi Derek,
   I never thought in my life that design pattern could be learned in such an easy way. I would really thanks from my bottom of my heart. The way you are explaining the stuff is exceptional. I became your great friend… One small request please upload few core java related stuff such as: collections, Java memory management , Multi Threads , Syncronization. Thanks again and god bless you…

   Reply

-  *Derek Banas* says:
  April 2, 2013 at 6:10 am

  Thank you 🙂 I'm very happy that they have helped. I plan on going back and covering all of the topics you have mentioned and much more. I'm sorry it is taking so long. May God bless you and your family as well

  Reply

7.  *Ryan* says:
   August 29, 2013 at 7:49 am

   I would like to ask an other question , What about making our object's constructors private to prevent direct instantiation ?

   Reply

   -  *Derek Banas* says:
     August 29, 2013 at 7:56 am

     That is very often a good idea and I cover that topic in this tutorial series

     Reply

8.  *Matias* says:
   February 10, 2014 at 12:00 pm

   Not only the explanations are great, but also the quality of the videos. Congrats and thanks for such a hard and great work!

   Reply

   -  *Derek Banas* says:
     February 10, 2014 at 2:32 pm

     Thank you very much 🙂 I try to do my best

     Reply

9.  *Apil Tamang* says:
   May 11, 2014 at 6:14 pm

   I appreciate the videos as they seem to be gentle introductions to design patterns, but a little bit of explanation on how they improves quality would be very nice. For instance, with this video, why go through the trouble of creating a factory, when one could very easily have done this:

Sheep clonedSheep=(Sheep)sally.makeCopy();

I realize that there is some underlying reason as to why the above design is better, and it maybe that the benefits of applying the design patterns do not manifest in small programs , but a gentle reminder of why they ultimately are better would complement these tutorial videos very nicely. Or else, these tutorials would just be a 'hey, so this is this and that is that…', and not 'Hey, but this is why this is so and that is such!'

Reply

10.  *themis* says:
August 8, 2014 at 10:32 pm

thank you ever so much mate! i consider you one of these few rare individuals that really try to make the world a better place, i wish i could shake your hand and buy you a beer 🙂

Reply

-  *Derek Banas* says:
August 15, 2014 at 7:45 am

Thank you for the nice compliments 🙂 I try to do my best

Reply

11.  *Rupesh Kumar Tiwari* says:
January 5, 2015 at 9:56 am

Hi Derek,
I have watched many videos on Design Patterns but the way you have explained is very nice and your examples are so practical and real that it is easy to remember because it related with real world problems.

Please keep up the good work. I would suggest you to pick up such complex concept and keep explaining them in your way…

Reply

-  *Derek Banas* says:
January 5, 2015 at 1:33 pm

Thank you for the nice compliment 🙂 I have an advanced algorithm tutorial that I hope to bring out soon.

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)

Donate

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

About the Author Google+