JAVA TUTORIAL   #INDEX POSTS   #INTERVIEW QUESTIONS   RESOURCES

Instantly Search Tutorials...

Bridge Design Pattern in Java

PANKAJ — 26 COMMENTS

Today we will look into Bridge Design Pattern in java. When we have
interface hierarchies in both interfaces as well as implementations, then
bridge design pattern is used to decouple the interfaces from
implementation and hiding the implementation details from the client
programs.

Bridge Design Pattern

Just like Adapter pattern, bridge design pattern is one of the **Structural
design pattern**.

According to GoF bridge design pattern is:

**Recommended
Tutorials**

✚ **Java Tutorials**

✚ **Java EE Tutorials**

> " Decouple an abstraction from its implementation so that the
> two can vary independently

The implementation of bridge design pattern follows the notion to prefer
Composition over inheritance.

Bridge Design Pattern in Java Example

If we look into bridge design pattern with example, it will be easy to
understand. Lets say we have an interface hierarchy in both interfaces
and implementations like below image.
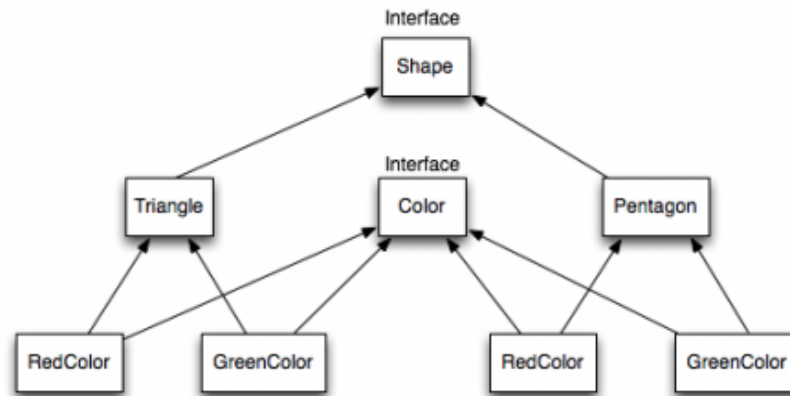
Now we will use bridge design pattern to decouple the interfaces from implementation. UML diagram for the classes and interfaces after applying bridge pattern will look like below image.
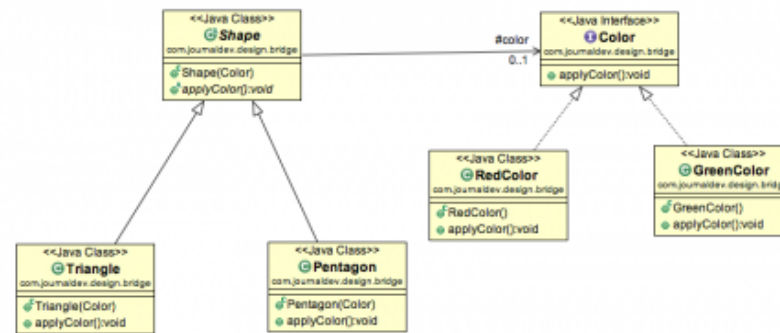
Notice the bridge between `Shape` and `Color` interfaces and use of composition in implementing the bridge pattern.

Here is the java code for Shape and Color interfaces.

```java
Color.java


package com.journaldev.design.bridge;

public interface Color {

        public void applyColor();
}

Shape.java


package com.journaldev.design.bridge;

public abstract class Shape {
        //Composition - implementor
        protected Color color;

        //constructor with implementor as input argument
        public Shape(Color c){
                this.color=c;
        }

        abstract public void applyColor();
}
```

We have Triangle and Pentagon implementation classes as below.

Triangle.java

```java
package com.journaldev.design.bridge;

public class Triangle extends Shape{

    public Triangle(Color c) {
        super(c);
    }

    @Override
    public void applyColor() {
        System.out.print("Triangle filled with color ");
        color.applyColor();
    }

}
```

Pentagon.java

```java
package com.journaldev.design.bridge;

public class Pentagon extends Shape{

    public Pentagon(Color c) {
        super(c);
    }
```

```java
        @Override
        public void applyColor() {
                System.out.print("Pentagon filled with color
");
                color.applyColor();
        }

}
```

Here are the implementation classes for RedColor and GreenColor.

RedColor.java

```java
package com.journaldev.design.bridge;

public class RedColor implements Color{

        public void applyColor(){
                System.out.println("red.");
        }
}
```

GreenColor.java

```java
package com.journaldev.design.bridge;

public class GreenColor implements Color{
```

```java
        public void applyColor(){
                System.out.println("green.");
        }

}
```

Lets test our bridge pattern implementation with a test program.

BridgePatternTest.java

```java
package com.journaldev.design.test;

import com.journaldev.design.bridge.GreenColor;
import com.journaldev.design.bridge.Pentagon;
import com.journaldev.design.bridge.RedColor;
import com.journaldev.design.bridge.Shape;
import com.journaldev.design.bridge.Triangle;

public class BridgePatternTest {

        public static void main(String[] args) {
                Shape tri = new Triangle(new RedColor());
                tri.applyColor();

                Shape pent = new Pentagon(new GreenColor());
                pent.applyColor();
        }

}
```
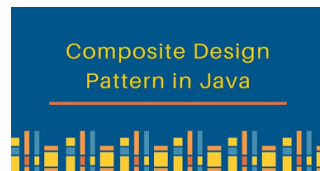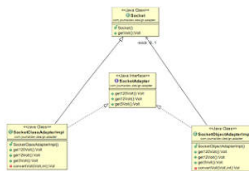
Output of above bridge pattern example program is:

```
Triangle filled with color red.
Pentagon filled with color green.
```

Bridge design pattern can be used when both abstraction and implementation can have different hierarchies independently and we want to hide the implementation from the client application.

Composite Design Pattern in Java

Adapter Design Pattern in Java
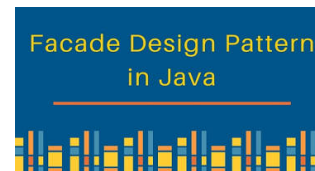
Facade Design Pattern in Java

Mediator Design Pattern in Java

DAO Design Pattern
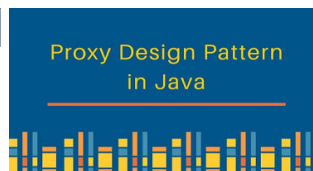
Proxy Design Pattern
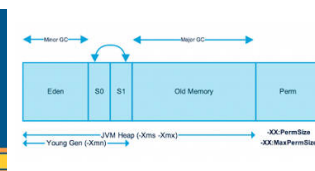
Java (JVM) Memory Model - Memory Management in Java

Java Exam

« PREVIOUS

Adapter Design Pattern in Java

NEXT »

Composite Design Pattern in Java

About Pankaj

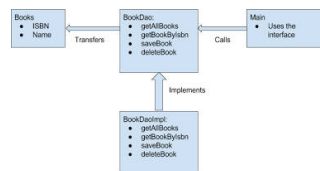If you have come this far, it means that you liked what you are reading. Why not reach little more and connect with me directly on **Google Plus**, **Facebook** or **Twitter**. I would love to hear your thoughts and opinions on my articles directly.

Recently I started creating video tutorials too, so do check out my videos on **Youtube**.

FILED UNDER: DESIGN PATTERNS

Comments

Sumeet Aggarwal says
JANUARY 21, 2019 AT 10:00 AM
Do we have any reference in JAVA JDK/API where Bridge pattern.is used ?
Reply

lukas says

OCTOBER 3, 2018 AT 11:28 AM

code is not displayed properly. go to html view of your browser to see the code nevertheless

KG

Reply

Pankaj says

OCTOBER 3, 2018 AT 10:47 PM

What do you mean by code is not displayed properly. Can you please clear your cookie and cache? The code is properly formatted using syntax highlighter.

Reply

lukas says

OCTOBER 4, 2018 AT 5:09 AM

My browser seemed to have troubles to display the code examples. Now it works again without making any changes, btw: good work!

Reply

Rasa says

MAY 16, 2017 AT 1:41 AM

If I understood this article correct, first schema addresses the state before, and UML diagram addresses the state after applying Bridge pattern. If that's what you meant to show, I guess that schema is not correct. Arrows from RedColor and GreenColor towards Triangle and Pentagon, should be pointing vice-versa.

Reply

Rasa1 says
MAY 16, 2017 AT 2:05 AM

In that case, RedColor and GreenColor would have been abstract classes meant to be extended from Triangle, or Pentagon. Triangle and Pentagon would have to implement Shape interface too.
Or, maybe the names of RedColor and GreenColor are such by mistake and shod have been called RedColorTriangle, RedColorPentagon, GreenColorTriangle and GreenColorPentagon. This make more sense according to the scheme drawing.

Reply

Anupama says
FEBRUARY 23, 2017 AT 1:35 AM

Hi Pankaj,
Thanks for the explanation. I am not fully clear on the example. In the example we could create the color instances and pass it to the

shapes. Then why can not we call the apply color directly on the color objects themselves? Why we are going thru the shape objects? Thanks in advance,

Reply

> **urbandroid says**
>
> AUGUST 5, 2017 AT 4:16 AM
>
> because shape should have that property the color is not a independent from the shape and other way will be bloated with if and elses try to convince yourself.
>
> Reply

Paul says

FEBRUARY 6, 2017 AT 5:40 AM

Great article, I read 2 prior articles which described this pattern, but in no way demonstrated why you would ever have a need to use it. This article does both, and very simply.

Reply

Anurag says

SEPTEMBER 15, 2016 AT 10:23 PM

Please modify UML diagram, you draw association and describing it as composition.

Reply

12 says

MAY 27, 2016 AT 7:07 AM

Please, change builder to bridge in the first line of the article

Reply

Jay says

MARCH 5, 2016 AT 4:11 AM

Thanks Pankaj for the wonderful article on Design patterns, but you may want to correct the first line on this "Bridge pattern" page:

"When we have interface hierarchies in both interfaces as well as implementations, then builder design "

It is supposed to be bridge pattern and not builder.

Reply

Anil says

JULY 23, 2015 AT 2:20 AM

Cool explanation as always.

Which tool are you using for UML diagram.?

Reply

yousuf says

APRIL 27, 2015 AT 12:39 AM

The opening sentence,

"When we have interface hierarchies in both interfaces as well as implementations"..

Did you mean 'inheritance hierarchies', when you say 'interface hierarchies'..?

Reply

Ashakant says

OCTOBER 3, 2014 AT 2:39 PM

Hi Pankaj ,

Nice , simple explanation , after googling so many Design Pattern , its is one of most simple , that why i have saved as screen shoot . It would be favor for all , if u just shared the code as downloadable .

Reply

KranthiC says

MAY 19, 2014 AT 2:47 AM

Please elaborate on how the decoupling has happened between abstraction and implementation here. Still if we add any abstract method in Shape abstract class, implementation for it is to be provided in Triangle and Pentagon classes. Similarly if any method is declared in interface Color, its implementation needs to be provided in RedColor and GreenColor classes, so how is the decoupling achieved here…

Reply

Himansu Nayak says

JUNE 10, 2014 AT 9:13 PM

Hi Kranthi,

When we say De-Coupling it means "loose coupling" or "no coupling" at all. Pankaj example perfectly shows the loosely coupling of 2 different hierarchy of class and how it is bridged using composition and not interface.

Please don't confused De-Coupling to some T.V. with Remote both communicating via Radio Signal.

Reply

Pravin Katkar says

AUGUST 29, 2014 AT 5:26 AM

This example decouples the shape abstraction from specific apply Color implementation

Reply

Dilip Kumar Pandey says

APRIL 11, 2014 AT 5:41 PM

Thanks Pankaj

Very self explanatory tutorials.

Any chance of that you will upload design-pattern vedioes?

Reply

Pankaj says

APRIL 11, 2014 AT 9:48 PM

I haven't planned for videos in near future, but who knows I might find time to get them too.

Reply

Prasad says

DECEMBER 17, 2013 AT 7:37 AM

Hi Pankaj,

Nice explanation. Can you please clarify below.

If implementation of applyColor() method is different for both Triangle and Pentagon classes, then this design patterns is not suitable right?

Reply

Pankaj says

DECEMBER 17, 2013 AT 8:33 AM

Implementation will obviously be different because both classes have different tasks to do, and this pattern is suitable for this scenario.

Reply

Prasad says

DECEMBER 17, 2013 AT 6:20 PM

Sorry for the confusion i have created here. Let me elaborate my question.

Suppose i write the below code in main() method.

Shape tri = new Triangle(new RedColor());
tri.applyColor();
Shape pent = new Pentagon(new RedColor());
pent.applyColor();

Here i would like to get the below output.
Triangle filled with color light red.
Pentagon filled with color dark red.

How can we achieve this using bridge pattern?

Reply

> **Pankaj says**
> DECEMBER 18, 2013 AT 1:27 AM
> Change the System.out.print() value in Triangle as:
> System.out.print("Triangle filled with color light");
> and Pentagon as:
> System.out.print("Triangle filled with color dark");
>
> Reply
>
> > **Nitsan says**
> > APRIL 23, 2015 AT 4:55 AM
> > I would say add two new color implementations
> > called DarkRedColor and LightRedColor, and keep
> > the Shape implementations unaware of that
> > change.

Basil George says

In this case:

Shape tri = new Triangle(new RedColor(new Dark()));

tri.applyColor();

Shape pent = new Pentagon(new GreenColor(new

Light()));

pent.applyColor();

public interface Density {

}

public class Dark implements Density {

}

public class Light implements Density {

}

public class GreenColor extends Color {

public GreenColor(Density d) {

super(d);

}

@Override

public void applyColor(){

if(density instanceof Dark)

System.out.println("Dark green.");

if(density instanceof Light)

System.out.println("Light green.");

}

}

```java
public class RedColor extends Color {
public RedColor(Density d) {
super(d);
}
@Override
public void applyColor(){
if(density instanceof Dark)
System.out.println("Dark red.");
if(density instanceof Light)
System.out.println("Light red.");
}
}
public abstract class Color {
protected Density density;
//constructor with implementor as input argument
public Color(Density d) {
this.density = d;
}
public abstract void applyColor();
}
```

Reply

Comment Policy: Please submit comments to add value to the post. Comments like "Thank You" and "Awesome Post" will be not published. If you want to post code then wrap them inside <pre> tags. For example **<pre>class Foo { }</pre>**.

If you want to post XML content, then please escape < with < and > with > otherwise they will not be shown properly.

Leave a Reply

Your email address will not be published. Required fields are marked *

Comment

Name *

Email *

POST COMMENT