

[JAVA TUTORIAL](#)[#INDEX POSTS](#)[#INTERVIEW QUESTIONS](#)[RESOURCES](#)[HIRE ME](#)[DOWNLOAD ANDROID APP](#)[CONTRIBUTE](#)**Subscribe to Download Java Design Patterns eBook****DOWNLOAD NOW**[HOME](#) » [JAVA](#) » [DESIGN PATTERNS](#) » [COMPOSITE DESIGN PATTERN IN JAVA](#)

# Composite Design Pattern in Java

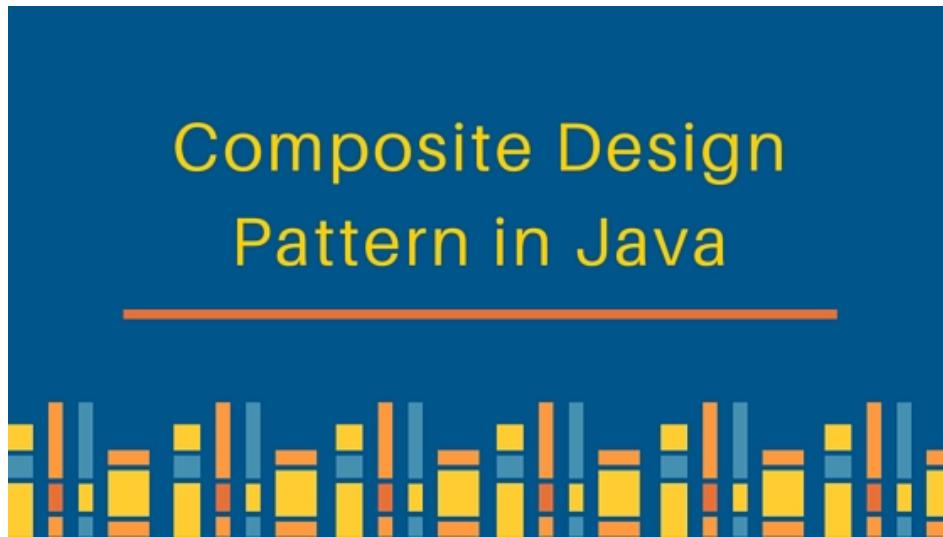
APRIL 4, 2018 BY [PANKAJ](#) — [10 COMMENTS](#)

Composite pattern is one of the Structural [design pattern](#). Composite design pattern is used when we have to represent a part-whole hierarchy.

## Table of Contents [\[hide\]](#)

- [1 Composite Design Pattern](#)
  - [1.1 Composite Pattern Base Component](#)
  - [1.2 Composite Design Pattern Leaf Objects](#)
  - [1.3 Composite object](#)
  - [1.4 Composite Design Pattern Client Program](#)
  - [1.5 Composite Pattern Important Points](#)

## Composite Design Pattern



When we need to create a structure in a way that the objects in the structure has to be treated the same way, we can apply composite design pattern.

Lets understand it with a real life example – A diagram is a structure that consists of Objects such as Circle, Lines, Triangle etc. When we fill the drawing with color (say Red), the same color also gets applied to the Objects in the drawing. Here drawing is made up of different parts and they all have same operations.

Composite Pattern consists of following objects.

1. **Base Component** – Base component is the interface for all objects in the composition, client program uses base component to work with the objects in the composition. It can be an interface or an **abstract class** with some methods common to all the objects.
2. **Leaf** – Defines the behaviour for the elements in the composition. It is the building block for the composition and implements base component. It doesn't have references to other Components.
3. **Composite** – It consists of leaf elements and implements the operations in base component.

Here I am applying composite design pattern for the drawing scenario.

## Composite Pattern Base Component

Composite pattern base component defines the common methods for leaf and composites. We can create a class Shape with a method `draw(String fillColor)` to draw the shape with given color.

Shape.java

```
package com.journaldev.design.composite;  
  
public interface Shape {
```

```
        public void draw(String fillColor);  
    }  
}
```

## Composite Design Pattern Leaf Objects

Composite design pattern leaf implements base component and these are the building block for the composite. We can create multiple leaf objects such as Triangle, Circle etc.

Triangle.java

```
package com.journaldev.design.composite;  
  
public class Triangle implements Shape {  
  
    @Override  
    public void draw(String fillColor) {  
        System.out.println("Drawing Triangle with color "+fillColor);  
    }  
  
}
```

Circle.java

```
package com.journaldev.design.composite;  
  
public class Circle implements Shape {  
  
    @Override  
    public void draw(String fillColor) {  
        System.out.println("Drawing Circle with color "+fillColor);  
    }  
  
}
```

## Composite object

A composite object contains group of leaf objects and we should provide some helper methods to add or delete leafs from the group. We can also provide a method to remove all the elements from the group.

Drawing.java

```

package com.journaldev.design.composite;

import java.util.ArrayList;
import java.util.List;

public class Drawing implements Shape{

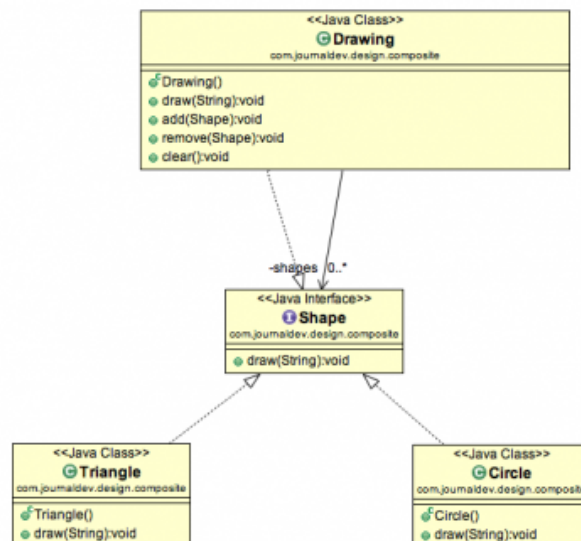
    //collection of Shapes
    private List<Shape> shapes = new ArrayList<Shape>();

    @Override
    public void draw(String fillColor) {
        for(Shape sh : shapes)
        {
            sh.draw(fillColor);
        }
    }

    //adding shape to drawing
    public void add(Shape s){
        this.shapes.add(s);
    }
}

```

Notice that composite also implements component and behaves similar to leaf except that it can contain group of leaf elements.



Our composite pattern implementation is ready and we can test it with a client program.

## Composite Design Pattern Client Program

TestCompositePattern.java

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

import com.journaldev.design.composite.Circle;
import com.journaldev.design.composite.Drawing;
import com.journaldev.design.composite.Shape;
import com.journaldev.design.composite.Triangle;

public class TestCompositePattern {

    public static void main(String[] args) {
        Shape tri = new Triangle();
        Shape tri1 = new Triangle();
        Shape cir = new Circle();

        Drawing drawing = new Drawing();
        drawing.add(tri1);
        drawing.add(tri1);
        drawing.add(cir);

        drawing.draw("Red");

        drawing.clear();
    }
}
```

Output of the above composite pattern client program is:

```
Drawing Triangle with color Red
Drawing Triangle with color Red
Drawing Circle with color Red
Clearing all the shapes from drawing
Drawing Triangle with color Green
Drawing Circle with color Green
```

## Composite Pattern Important Points

- Composite pattern should be applied only when the group of objects should behave as the single object.
- Composite design pattern can be used to create a tree like structure.

`java.awt.Container#add(Component)` is a great example of Composite pattern in java and used a lot in Swing.

Earlier structural design pattern articles:

- [Bridge Pattern in Java](#)
- [Adapter Design Pattern in Java](#)

## « PREVIOUS

[Bridge Design Pattern in Java](#)

## NEXT »

[Decorator Design Pattern in Java Example](#)

### 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

**ravi says**

JUNE 18, 2017 AT 10:11 PM

This is not composite pattern. check this for right explanation

<http://www.dofactory.com/net/composite-design-pattern>.

in composite pattern there have to be nodes and leaves to form a tree. Just aggregating objects into another object is not a composite pattern.

[Reply](#)

**avmohan says**

AUGUST 11, 2017 AT 8:37 AM

This is a composite pattern. The main idea in composite pattern is that the client shouldn't need to know whether it's an atomic object or a composite.

Using the terminology of the composite pattern (from

[https://en.wikipedia.org/wiki/Composite\\_pattern](https://en.wikipedia.org/wiki/Composite_pattern)) itself, here, the drawing is a Composite and triangle, & circle are Leaves and Shape is the Component.

[Reply](#)

**Anil Chowdhury says**

DECEMBER 19, 2017 AT 7:36 PM

Hi avmohan,

I agree with Ravi, this is not composite pattern. What pankaj has shown in the above example is aggregation of objects.

A composite pattern should treat each object hierarchically either as leaf or parent. The wiki link itself says "The composite pattern describes a group of objects that is treated the same way as a single instance of the same type of object. The intent of a composite is to "compose" objects into tree structures"

Without parent-child relationship composite pattern cannot exist.

A good example is Filesystem..whether each node could be parent (sub-folder) or leaf (i.e. files) please refer to this link : <http://www.oodeign.com/composite-pattern.html>

[Reply](#)

**Manoj Singh says**

APRIL 4, 2017 AT 11:37 PM

Nice article pankaj, you are the champ.

[Reply](#)

**Nabil Amrouche says**

DECEMBER 9, 2014 AT 7:31 AM

Thanks a lot !

Your tutorials were so useful for me !!! and i'm sure, i'm not the only one !

Design Patterns especially, That got me a good mark in my exams :)))

Thanks; another time !!!

Good Continuation !

[Reply](#)**perminder singh says**

SEPTEMBER 28, 2014 AT 5:50 AM

Really nice post. help me a lot to get the concept.

[Reply](#)**Zhacantal says**

MARCH 13, 2014 AT 12:27 AM

It really help me a lot..

Can you give an example of composite pattern that has a GUI

Thank you!!!

[Reply](#)**Amit says**

JULY 14, 2013 AT 10:03 AM

I like the example you have provided.

[Reply](#)**Yogesh Kharode says**

JULY 6, 2013 AT 4:52 AM

Hi Pankaj,

Thanks for sharing best core java tutorial for freshers and experience.

Its really helpful for java developer.

kindly request to you please share Spring MVC+Hibernate application as per industry stranded and useful for better performance.

Thanks and Regards

Yogesh Kharode



[Reply](#)**Pankaj says**

JULY 6, 2013 AT 9:34 AM

Thanks Yogesh for the kind words, please do check out the design pattern articles. I am planning on starting on Spring tutorial in coming month.

[Reply](#)

## Leave a Reply

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

Comment

Name \*

Email \*

☐

Save my name, email, and website in this browser for the next time I comment.

POST COMMENT

---

[DOWNLOAD ANDROID APP](#)

---



---

## DESIGN PATTERNS TUTORIAL

---

### Java Design Patterns

#### Creational Design Patterns

- > [Singleton](#)
- > [Factory](#)
- > [Abstract Factory](#)
- > [Builder](#)
- > [Prototype](#)

#### Structural Design Patterns

- > [Adapter](#)
- > [Composite](#)
- > [Proxy](#)
- > [Flyweight](#)
- > [Facade](#)
- > [Bridge](#)
- > [Decorator](#)

#### Behavioral Design Patterns

- > [Template Method](#)
- > [Mediator](#)
- > [Chain of Responsibility](#)
- > [Observer](#)
- > [Strategy](#)
- > [Command](#)
- > [State](#)
- > [Visitor](#)
- > [Interpreter](#)
- > [Iterator](#)
- > [Memento](#)

#### Miscellaneous Design Patterns

- > [Dependency Injection](#)
- > [Thread Safety in Java Singleton](#)

---

## RECOMMENDED TUTORIALS

---

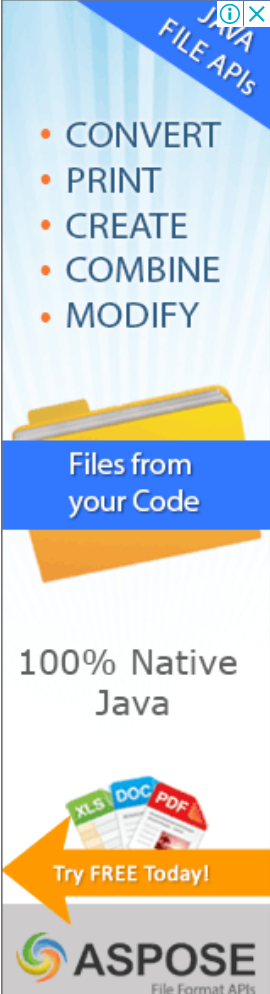
### Java Tutorials

- > [Java IO](#)

- > [Java Regular Expressions](#)
- > [Multithreading in Java](#)
- > [Java Logging](#)
- > [Java Annotations](#)
- > [Java XML](#)
- > [Collections in Java](#)
- > [Java Generics](#)
- > [Exception Handling in Java](#)
- > [Java Reflection](#)
- > [Java Design Patterns](#)
- > [JDBC Tutorial](#)

## Java EE Tutorials

- > [Servlet JSP Tutorial](#)
- > [Struts2 Tutorial](#)
- > [Spring Tutorial](#)
- > [Hibernate Tutorial](#)
- > [Primefaces Tutorial](#)
- > [Apache Axis 2](#)
- > [JAX-RS](#)
- > [Memcached Tutorial](#)



**JAVA FILE APIs**

- CONVERT
- PRINT
- CREATE
- COMBINE
- MODIFY

Files from your Code

100% Native Java

Try FREE Today!

**ASPOSE**  
File Format APIs

The advertisement features a blue header with a white 'i' icon and a close button. Below the header, a list of five actions (Convert, Print, Create, Combine, Modify) is shown with orange bullet points. A yellow folder icon is positioned above a blue banner that reads 'Files from your Code'. Below this, the text '100% Native Java' is displayed. At the bottom, there is a graphic of three document icons labeled 'XLS', 'DOC', and 'PDF' next to an orange arrow pointing left with the text 'Try FREE Today!'. The ASPOSE logo, consisting of a colorful swirl and the text 'ASPOSE File Format APIs', is at the very bottom.

