```
}

// Driver class
public class Company
{
    public static void main (String[] args)
    {
        Developer dev1 = new Developer(100, "Lokesh Sharma", "Pro Developer");
        Developer dev2 = new Developer(101, "Vinay Sharma", "Developer");
        CompanyDirectory engDirectory = new CompanyDirectory();
        engDirectory.addEmployee(dev1);
        engDirectory.addEmployee(dev2);

        Manager man1 = new Manager(200, "Kushagra Garg", "SEO Manager");
        Manager man2 = new Manager(201, "Vikram Sharma ", "Kushagra's Manager");

        CompanyDirectory accDirectory = new CompanyDirectory();
        accDirectory.addEmployee(man1);
        accDirectory.addEmployee(man2);

        CompanyDirectory directory = new CompanyDirectory();
        directory.addEmployee(engDirectory);
        directory.addEmployee(accDirectory);
        directory.showEmployeeDetails();
    }
}
```

Run on IDE

Output :

```
100 Lokesh Sharma Pro Developer
101 Vinay Sharma Developer
200 Kushagra Garg SEO Manager
201 Vikram Sharma  Kushagra's Manager
```

When to use Composite Design Pattern?

Composite Pattern should be used when clients need to ignore the difference between compositions of objects and individual objects. If programmers find that they are using multiple objects in the same way, and often have nearly identical code to handle each of them, then composite is a good choice, it is less complex in this situation to treat primitives and composites as homogeneous.

1. Less number of objects reduces the memory usage, and it manages to keep us away from errors related to memory like java.lang.OutOfMemoryError.
2. Although creating an object in Java is really fast, we can still reduce the execution time of our program by sharing objects.

When not to use Composite Design Pattern?

1. Composite Design Pattern makes it harder to restrict the type of components of a composite. So it should not be used when you don't want to represent a full or partial hierarchy of objects.
2. Composite Design Pattern can make the design overly general. It makes harder to rest the components of a composite. Sometimes you want a composite to have only certain comp its.

With Composite, you can't rely on the type system to enforce those constraints for you. Instead you'll have to use run-time checks.

This article is contributed by **Saket Kumar**. If you like GeeksforGeeks and would like to contribute, you can also write an article using contribute.geeksforgeeks.org or mail your article to contribute@geeksforgeeks.org. See your article appearing on the GeeksforGeeks main page and help other Geeks.

Please write comments if you find anything incorrect, or you want to share more information about the topic discussed above.

GATE CS Corner   Company Wise Coding Practice

Design Pattern

Recommended Posts:

Prototype Design Pattern
Proxy Design Pattern
Decorator Pattern | Set 1 (Background)
Business Delegate Pattern
Curiously recurring template pattern (CRTP)
Data Access Object Pattern
Front Controller Design Pattern
MVC Design Pattern
Intercepting Filter Pattern
Design the Data Structures(classes and objects)for a generic deck of cards

(Login to Rate and Mark)

0 Average Difficulty : **0/5.0**
No votes yet.

Add to TODO List

Mark as DONE

Writing code in comment? Please use ide.geeksforgeeks.org, generate link and share the link here.

Load Comments

Share this post!