**Steps:**

1. **Create a New Java Project:**
   * Create a new Java project named **DecoratorPatternExample**.
2. **Define Component Interface:**
   * Create an interface **Notifier** with a method **send()**.
3. **Implement Concrete Component:**
   * Create a class **EmailNotifier** that implements Notifier.
4. **Implement Decorator Classes:**
   * Create abstract decorator class **NotifierDecorator** that implements **Notifier** and holds a reference to a **Notifier** object.
   * Create concrete decorator classes like **SMSNotifierDecorator**, **SlackNotifierDecorator** that extend **NotifierDecorator**.
5. **Test the Decorator Implementation:**
   * Create a test class to demonstrate sending notifications via multiple channels using decorators.

**Solution:**

using System;

public interface INotifier

{

void Send(string message);

}

public class EmailNotifier : INotifier

{

public void Send(string message)

{

Console.WriteLine($"Email: {message}");

}

}

public abstract class NotifierDecorator : INotifier

{

protected INotifier wrappee;

protected NotifierDecorator(INotifier notifier)

{

wrappee = notifier;

}

public virtual void Send(string message)

{

wrappee.Send(message);

}

}

public class SMSNotifierDecorator : NotifierDecorator

{

public SMSNotifierDecorator(INotifier notifier) : base(notifier) { }

public override void Send(string message)

{

base.Send(message);

Console.WriteLine($"SMS: {message}");

}

}

public class SlackNotifierDecorator : NotifierDecorator

{

public SlackNotifierDecorator(INotifier notifier) : base(notifier) { }

public override void Send(string message)

{

base.Send(message);

Console.WriteLine($"Slack: {message}");

}

}

public class Program

{

public static void Main()

{

INotifier notifier = new EmailNotifier();

notifier = new SMSNotifierDecorator(notifier);

notifier = new SlackNotifierDecorator(notifier);

notifier.Send("System alert: High CPU usage");

}

}

