**C# Decorator**

Паттерн проектування Декоратор динамічно додає додаткові обов'язки до об'єкта. Цей паттерн є гнучкою альтернативою підкласам для розширення функціональності.

До класів та об'єктів, що беруть участь у цьому паттерні, відносяться наступні:

* **Component (LibraryItem) -** визначає інтерфейс для об'єктів, до яких можна динамічно додавати обов'язки.
* **ConcreteComponent (Book, Video) -** визначає об'єкт, до якого можна додати додаткові обов'язки.
* **Decorator (Decorator) -** зберігає посилання на об'єкт ***Component*** і визначає інтерфейс, який відповідає інтерфейсу Component.
* **ConcreteDecorator (Borrowable) -** додає обов'язки до компонента.

**Structural code in C#**

using System;

using System.Collections.Generic;

namespace Decorator.RealWorld

{

public class Program

{

public static void Main(string[] args)

{

// Create book

Book book = new Book("Worley", "Inside ASP.NET", 10);

book.Display();

// Create video

Video video = new Video("Spielberg", "Jaws", 23, 92);

video.Display();

// Make video borrowable, then borrow and display

Console.WriteLine("\nMaking video borrowable:");

Borrowable borrowvideo = new Borrowable(video);

borrowvideo.BorrowItem("Customer #1");

borrowvideo.BorrowItem("Customer #2");

borrowvideo.Display();

// Wait for user

Console.ReadKey();

}

}

public abstract class LibraryItem

{

private int numCopies;

public int NumCopies

{

get { return numCopies; }

set { numCopies = value; }

}

public abstract void Display();

}

public class Book : LibraryItem

{

private string author;

private string title;

// Constructor

public Book(string author, string title, int numCopies)

{

this.author = author;

this.title = title;

this.NumCopies = numCopies;

}

public override void Display()

{

Console.WriteLine("\nBook ------ ");

Console.WriteLine(" Author: {0}", author);

Console.WriteLine(" Title: {0}", title);

Console.WriteLine(" # Copies: {0}", NumCopies);

}

}

public class Video : LibraryItem

{

private string director;

private string title;

private int playTime;

// Constructor

public Video(string director, string title, int numCopies, int playTime)

{

this.director = director;

this.title = title;

this.NumCopies = numCopies;

this.playTime = playTime;

}

public override void Display()

{

Console.WriteLine("\nVideo ----- ");

Console.WriteLine(" Director: {0}", director);

Console.WriteLine(" Title: {0}", title);

Console.WriteLine(" # Copies: {0}", NumCopies);

Console.WriteLine(" Playtime: {0}\n", playTime);

}

}

public abstract class Decorator : LibraryItem

{

protected LibraryItem libraryItem;

// Constructor

public Decorator(LibraryItem libraryItem)

{

this.libraryItem = libraryItem;

}

public override void Display()

{

libraryItem.Display();

}

}

public class Borrowable : Decorator

{

protected readonly List<string> borrowers = new List<string>();

// Constructor

public Borrowable(LibraryItem libraryItem)

: base(libraryItem)

{

}

public void BorrowItem(string name)

{

borrowers.Add(name);

libraryItem.NumCopies--;

}

public void ReturnItem(string name)

{

borrowers.Remove(name);

libraryItem.NumCopies++;

}

public override void Display()

{

base.Display();

foreach (string borrower in borrowers)

{

Console.WriteLine(" borrower: " + borrower);

}

}

}

}

**Output**

Book ------

Author: Worley

Title: Inside ASP.NET

# Copies: 10

Video -----

Director: Spielberg

Title: Jaws

# Copies: 23

Playtime: 92

Making video borrowable:

Video -----

Director: Spielberg

Title: Jaws

# Copies: 21

Playtime: 92

borrower: Customer #1

borrower: Customer #2