

ChatGPT

Explore GPTs

Today

Composite Pattern Library System

Library Management Decorator

Library Management Facade

Adapter Pattern Example

Builder Pattern Library System

Abstract Factory Pattern Example

Check Python Init Method

Enhanced Abstract Factory Pattern

Movie Ticket Factory Pattern

Design Patterns Refactoring

composite pattern with python code, you can write code from below domain,

Library Management System,
Parking Lot,
Online Shopping System - E-Commerce,
Stack Overflow,
Movie Ticket Booking System,
Design an ATM,
Airline Management System,
Hotel Management System,
Restaurant Management system,
Design Chess,
Design an Online Stock Brokerage System,
Design a Car Rental System,
Design LinkedIn,
Design Cricinfo,
Design Facebook Social Network,
Design payment gateway

Get citation

Sure! Let's use the Composite pattern to design a simple **Library Management System**. The Composite pattern is useful when you want to treat individual objects and compositions of objects uniformly.

Composite Pattern Overview

In the Composite pattern:

- **Component:** An abstract class or interface that declares the interface for all concrete objects and compositions.
- **Leaf:** Represents the individual objects that do not have any children.
- **Composite:** Represents a group of objects and implements the `Component` interface, allowing them to have children.

Example: Library Management System

Let's design a basic library system where we have books, magazines, and sections. The Composite pattern will help us treat both individual items and sections uniformly.

1. Define the Component

```
python Copy code

from abc import ABC, abstractmethod

class LibraryComponent(ABC):
    @abstractmethod
    def get_name(self):
        pass

    @abstractmethod
    def get_description(self):
        pass

    @abstractmethod
    def add(self, component):
        pass

    @abstractmethod
    def remove(self, component):
        pass

    @abstractmethod
    def display(self, depth):
        pass
```

2. Define the Leaf

```
python Copy code

class Book(LibraryComponent):
```

```

def __init__(self, name, description):
    self.name = name
    self.description = description

def get_name(self):
    return self.name

def get_description(self):
    return self.description

def add(self, component):
    raise NotImplementedError("Cannot add to a leaf component")

def remove(self, component):
    raise NotImplementedError("Cannot remove from a leaf component")

def display(self, depth):
    print(" " * depth + f"Book: {self.get_name()} - {self.get_description()}")

```

3. Define the Composite

```

python Copy code

class Section(LibraryComponent):
    def __init__(self, name):
        self.name = name
        self.components = []

    def get_name(self):
        return self.name

    def get_description(self):
        return f"Section: {self.name}"

    def add(self, component):
        self.components.append(component)

    def remove(self, component):
        self.components.remove(component)

    def display(self, depth):
        print(" " * depth + f"Section: {self.get_name()}")
        for component in self.components:
            component.display(depth + 1)

```

4. Use the Composite Pattern

```

python Copy code

# Create leaf components
book1 = Book("Book One", "Description of Book One")
book2 = Book("Book Two", "Description of Book Two")

# Create composite components
section1 = Section("Fiction")
section1.add(book1)
section1.add(book2)

# Create another composite
library = Section("Library")
library.add(section1)

# Display the library structure
library.display(0)

```

Output

```

yaml Copy code

Section: Library
  Section: Fiction
    Book: Book One - Description of Book One
    Book: Book Two - Description of Book Two

```

In this example:

- `LibraryComponent` is the abstract class that defines the interface.
- `Book` is a leaf node representing individual books.
- `Section` is a composite that can contain other `LibraryComponent` objects, either `Book` or other `Section`.

This setup allows you to add, remove, and display both individual books and sections of the library uniformly.

< 3/3 > 🔊 📄 ↺ 🗨️ ↻

Design Patterns Recommendation

★ Upgrade plan
Get GPT-4, DALL-E, and more



Message ChatGPT



ChatGPT can make mistakes. Check important info.

?