Wzorzec **Kompozyt** (ang. Composite) jest wzorcem strukturalnym, który pozwala na tworzenie struktur drzewiastych z obiektów podobnych do siebie. Każdy obiekt w tej strukturze może być pojedynczym elementem lub składać się z innych obiektów (komponentów). Dzięki temu możliwe jest traktowanie pojedynczych elementów i całych składników w sposób jednolity.

Wzorzec Kompozyt jest często stosowany w programowaniu interfejsów graficznych użytkownika (GUI), gdzie możliwe jest tworzenie skomplikowanych struktur złożonych z pojedynczych elementów lub grup elementów. Może być również stosowany w aplikacjach biznesowych do tworzenia struktur hierarchicznych, takich jak drzewa organizacyjne lub struktury produktów.

W .NET Core wzorzec Kompozyt może być stosowany do tworzenia struktur drzewiastych z obiektów opartych na interfejsie lub klasie bazowej. Możliwe jest również wykorzystanie wbudowanych kontroli kompozytowych, takich jak Panel lub StackPanel, które pozwalają na dodawanie innych kontroli jako ich elementów.

Przykład:

Załóżmy, że chcemy stworzyć aplikację do tworzenia struktur drzewiastych z węzłami (ang. nodes). Możemy utworzyć interfejs INode, który zdefiniuje metody do dodawania i usuwania węzłów oraz wyświetlania ich nazwy. Następnie możemy stworzyć klasę Leaf, która będzie reprezentować pojedynczy węzeł i implementuje interfejs INode oraz klasę Composite, która będzie reprezentować grupę węzłów i również implementuje interfejs INode. Dzięki temu będziemy mogli traktować pojedyncze węzły i grupy węzłów w sposób jednolity i tworzyć skomplikowane struktury drzewiaste.

**Observer** (wzorzec projektowy obserwatora) to wzorzec projektowy, który pozwala na zarejestrowanie obiektu jako subskrybenta w celu otrzymywania powiadomień o zmianach stanu innego obiektu. Wzorzec ten jest często stosowany w aplikacjach, w których istnieje wiele obiektów zależnych od innych obiektów i chcemy, aby te obiekty były powiadamiane o zmianach stanu tych obiektów, na których są zależne.

W .NET Core można zastosować wzorzec obserwatora za pomocą interfejsu IObserver<T> i klasy Observable<T>. Interfejs IObserver<T> definiuje trzy metody: OnCompleted, OnError i OnNext. Metoda OnNext jest wywoływana, gdy obiekt nadawcy (tzw. "obserwowany") ma nową informację do przekazania, metoda OnError jest wywoływana, gdy wystąpi błąd podczas przekazywania informacji, a metoda OnCompleted jest wywoływana, gdy obiekt nadawcy zakończy przekazywanie informacji. Klasa Observable<T> implementuje interfejs IObservable<T>, który pozwala na rejestrowanie obiektów jako subskrybentów i przekazywanie im powiadomień o zmianach stanu.

**Strategia** (wzorzec projektowy strategii) to wzorzec projektowy, który pozwala na zdefiniowanie rodziny algorytmów, umożliwiając klientom wybór odpowiedniego algorytmu i umożliwiając łatwe zmiany algorytmów w czasie działania programu. Wzorzec ten pozwala na oddzielenie algorytmów od obiektów, które z nich korzystają, co umożliwia łatwe rozszerzenie lub zmianę algorytmów bez wpływu na obiekty, które z nich korzystają.

W .NET Core można zastosować wzorzec strategii poprzez utworzenie interfejsu lub klasy abstrakcyjnej, która definiuje metodę lub metody implementowane przez różne algorytmy, oraz utworzenie konkretnych klas implementujących te metody dla każdego algorytmu. Następnie klient może wybrać odpowiedni algorytm poprzez utworzenie obiektu odpowiedniej klasy i użycie go zgodnie z potrzebami.

**Metoda** **Wytwórcza** (wzorzec projektowy metody wytwórczej) to wzorzec projektowy, który pozwala na oddzielenie procesu tworzenia obiektów od ich reprezentacji, co umożliwia klientom używanie tych obiektów bez konieczności wiedzy o sposobie ich tworzenia. Wzorzec ten jest szczególnie przydatny, gdy proces tworzenia obiektów jest skomplikowany lub gdy istnieje potrzeba tworzenia obiektów z różnymi typami reprezentacji.

W .NET Core można zastosować wzorzec metody wytwórczej poprzez utworzenie interfejsu lub klasy abstrakcyjnej, która definiuje metodę lub metody służące do tworzenia obiektów, oraz utworzenie konkretnych klas implementujących te metody dla każdego typu obiektu. Klient może następnie używać obiektów za pomocą interfejsu lub klasy abstrakcyjnej bez konieczności wiedzy o sposobie ich tworzenia.

**Dekorator** (wzorzec projektowy dekoratora) to wzorzec projektowy, który pozwala na dynamiczne dodawanie nowych funkcjonalności do obiektu bez modyfikowania jego kodu źródłowego. Wzorzec ten jest szczególnie przydatny, gdy chcemy dodać nowe funkcjonalności do istniejącego obiektu, ale nie chcemy modyfikować jego kodu źródłowego lub gdy chcemy umożliwić użytkownikom tworzenie obiektów o różnych kombinacjach funkcjonalności.

W .NET Core można zastosować wzorzec dekoratora poprzez utworzenie interfejsu lub klasy abstrakcyjnej, która definiuje metodę lub metody dla obiektu dekorowanego, oraz utworzenie konkretnej klasy implementującej ten interfejs lub dziedziczącej po tej klasie abstrakcyjnej. Następnie można utworzyć klasę dekoratora, która będzie dziedziczyć po tej samej klasie abstrakcyjnej co obiekt dekorowany i będzie zawierać pole zawierające obiekt dekorowany. Klasa dekoratora będzie również implementować metody z interfejsu lub klasy abstrakcyjnej, ale będzie je rozszerzać o nowe funkcjonalności.