9. opisać wzorce projektowe i napisać jakie mają zastosowanie w .NET Core:

Wzorce projektowe to rozwiązania, które zostały sprawdzone w rozwiązywaniu różnych problemów związanych z projektowaniem. W .NET Core można je stosować tak samo jak w innych językach programowania. .NET Core dostarcza również gotowych implementacji wielu wzorców projektowych w postaci klas bibliotecznych. Są one często używane do tworzenia aplikacji, które są elastyczne i łatwe do rozszerzenia. Ich stosowanie może pomóc poprawić czytelność i jakość kodu oraz ułatwić utrzymanie aplikacji w przyszłości.

**Kompozyt**

Kompozyt jest wzorcem projektowym, który pozwala na traktowanie pojedynczych obiektów i ich zbiorów w sposób jednakowy. Umożliwia on budowanie złożonych struktur z prostych obiektów, takich jak gałęzie i liście w drzewie. W ten sposób klient może ignorować różnicę między pojedynczymi obiektami a ich zbiorami i traktować je wszystkie jako jedną jednostkę.

**Obserwator**

Strategia jest wzorcem projektowym, który pozwala na budowanie obiektów, które będą reagować na zmiany w stanie innych obiektów. Może być stosowany do tworzenia systemów, w których jedne obiekty informują inne o swoich zmianach stanu, np. w systemie powiadomień o zmianach na stronie internetowej lub w aplikacji mobilnej.

**Strategia**

Strategia jest wzorcem projektowym, który pozwala na definiowanie różnych algorytmów, które mogą być wykorzystywane przez obiekt w różnych sytuacjach. Wzorzec ten polega na oddzieleniu algorytmu od obiektu, który go wykorzystuje, co umożliwia łatwe zmiany algorytmu bez konieczności modyfikowania kodu obiektu. Strategia pozwala na przeprowadzanie zmian w aplikacji bez konieczności modyfikowania kodu istniejących klas, co zwiększa jej elastyczność i pozwala na łatwe dostosowywanie do zmieniających się potrzeb.

**Metoda wytwórcza**

Metoda wytwórcza jest wzorcem projektowym, który polega na tworzeniu obiektów za pomocą specjalnej metody, zwanej fabryką. Ta metoda jest odpowiedzialna za tworzenie obiektów, zamiast tego, aby tworzyć je bezpośrednio w kodzie klienta. Dzięki temu, klient nie musi znać szczegółów implementacji obiektu, a jedynie jego interfejs. W przypadku potrzeby zmiany implementacji obiektu, nie trzeba modyfikować kodu klienta, tylko fabrykę.

Wzorzec ten pozwala na abstrahowanie procesu tworzenia obiektów, co pozwala na łatwe dostosowywanie aplikacji do zmieniających się potrzeb.

**Dekorator**

Dekrator jest wzorcem projektowym, który pozwala na dynamiczne dodawanie nowych funkcjonalności do obiektów bez zmiany ich kodu źródłowego. Jest to uzyskiwane poprzez tworzenie klas dekoratorów, które przechowują referencję do obiektu, który mają dekorować oraz implementują tą samą interfejs co obiekt dekorowany.

Dekorator pozwala na tworzenie rozszerzającej się funkcjonalności poprzez dodawanie kolejnych dekoratorów, a także na łatwe usuwanie funkcjonalności poprzez usunięcie odpowiedniego dekoratora.