Dekoratory

Zalety

1. **Modularność**: Dekorator Vector3DDecorator umożliwia dynamiczne dodanie trzeciego wymiaru (z) do istniejącego obiektu Vector2D, bez konieczności zmiany jego klasy. To elastyczne podejście pozwala używać 2D i 3D z tym samym obiektem.
2. **Reużywalność**: Funkcjonalność dla abs, cdot i cross jest wbudowana w dekorator, dzięki czemu można jej używać bez ponownego definiowania tych metod dla każdego obiektu 2D.

Wady

1. **Trudniejsza debuggowalność**: Dekorator modyfikuje Vector2D dynamicznie, co może utrudnić śledzenie źródeł błędów, zwłaszcza gdy obiekt działa zarówno jako 2D, jak i 3D.
2. **Kompleksowość**: Zastosowanie dekoratora do konwersji 2D na 3D jest mniej czytelne niż użycie oddzielnej klasy Vector3D, co może sprawić, że kod będzie trudniejszy w zrozumieniu dla nowych programistów.

Adaptery

Zalety

1. **Elastyczność**: Adapter umożliwia dostosowanie interfejsu Vector2D tak, aby działał jak Vector3D, co pozwala na korzystanie z dodatkowego wymiaru bez zmiany oryginalnej klasy.
2. **Spójność interfejsu**: Dzięki adapterowi, Vector2D może być używany wszędzie tam, gdzie wymagany jest Vector3D, co ułatwia integrację różnych klas w kodzie.

Wady

1. **Złożoność kodu**: Dodanie adaptera zwiększa poziom abstrakcji, co może wprowadzać dodatkową złożoność i wymagać więcej wysiłku, by zrozumieć interakcje między klasami.
2. **Wydajność**: Adapter wprowadza dodatkową warstwę wywołań, co może minimalnie wpłynąć na wydajność, zwłaszcza w przypadku dużych operacji lub częstego używania.