Dziedziczenie 2D biegunowe (TwoDPolarInheritance)

+ Proste i czytelne: zero dodatkowych obiektów, jedna klasa z rozszerzonymi metodami
- sztywne powiązanie w czasie kompilacji — nie rozszerzysz istniejących obiektów w runtime.

Adapter (Polar2DAdapter)

+ Zgodność interfejsów bez modyfikacji klas źródłowych (przejście z kartezjańskiego na IPolar2D).

+ Działa w runtime: można „podłączyć" nowe metody do gotowych obiektów.

- Wprowadza dodatkową warstwę kodu,

Dziedziczenie 3D (Vector3DInheritance)

+ Jasne i spójne – obiekt 3D ma od razu wszystkie potrzebne metody (np. długość, iloczyn skalarny, wektorowy).

+ Dobra kontrola typów – nie da się przypadkowo pomylić z obiektem 2D.

- Mało elastyczne – obiekt musi być od razu 3D.

Dekorator 3D (Vector3DDecorator)

+ Bardzo elastyczny – można w czasie działania dodać trzeci wymiar do dowolnego wektora 2D.
+ Można łączyć wiele dekoratorów, np. jeden doda wymiar Z, inny doda logowanie operacji.
+ Nie trzeba modyfikować klas bazowych.
+ Zgodność z interfejsem IVector – klient nie musi wiedzieć, że obiekt jest dekorowany.

- Kod staje się trochę bardziej złożony