

Wzorce Projektowe: Dekorator

Tomasz Śmiech i Rafał Kolaska

Krakow, 15.06.2019 r.

Dynamiczne dołączanie dodatkowych funkcji do obiektu.

Chcemy dodać dodatkowe opcje do pojedynczego obiektu, a nie do całej klasy.

Rozwiązanie:

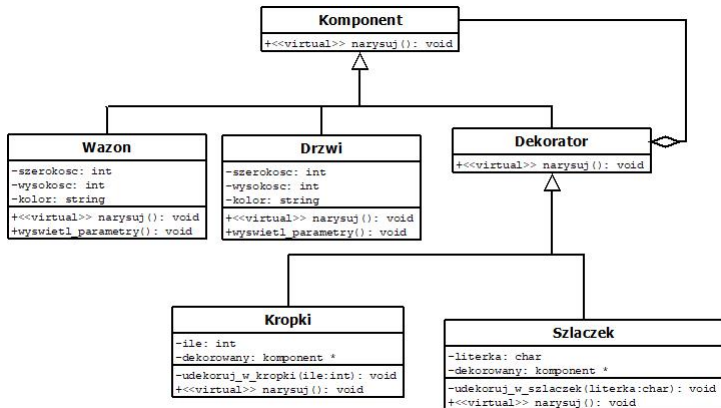
- 1 Dziedziczenie - nieelastyczne, statyczne rozwiązanie
- 2 Dekorator - umieszczenie obiektu w innym obiekcie, który doda nam odpowiednie właściwości

- 1 Większa elastyczność niż statyczne dziedziczenie
- 2 Pozwala uniknąć tworzenia przeładowanych funkcji klas
- 3 Dekorator i komponent nie są identyczne
- 4 Powstanie wielu małych obiektów

Diagram UML Dekoratora

Wzorce
Projektowe:
Dekorator

Tomasz
Śmiech i
Rafał
Kolaska



Jak to działa?

Wzorce
Projektowe:
Dekorator

Tomasz
Śmiech i
Rafał
Kolaska

```
Kropki(Wazon):Dekorator(Wazon) //Wazon dziedziczy po komponet
narysuj {
Dekorator::narysuj ();
Dodatkowa_funkcja() //dekoracja w kropki
}
```

Dekorator wywołuje funkcje narysuj
Wazonu

```
Szlaczek(Kropki):Dekorator(Kropki) //Szlaczek otrzymuje Kropki zawierajoe Wazon
narysuj {
Dekorator::narysuj ();
Dodatkowa_funkcja() //dekoracja w szlaczki
}
```

Dekorator wywołuje funkcje narysuj
Kropek

Wstawiamy do pokoju

```
vector<komponent*> //zbiornik na wstawione elementy
sprawdz() //wyswietla co jest w pokoju
usun() //usowa ostatnio dodany element
```

Funkcja sprawdz() wywołuje funkcje narysuj dla wstawionego elementu
Fk. narysuj wywołuje się począwszy od Wazonu