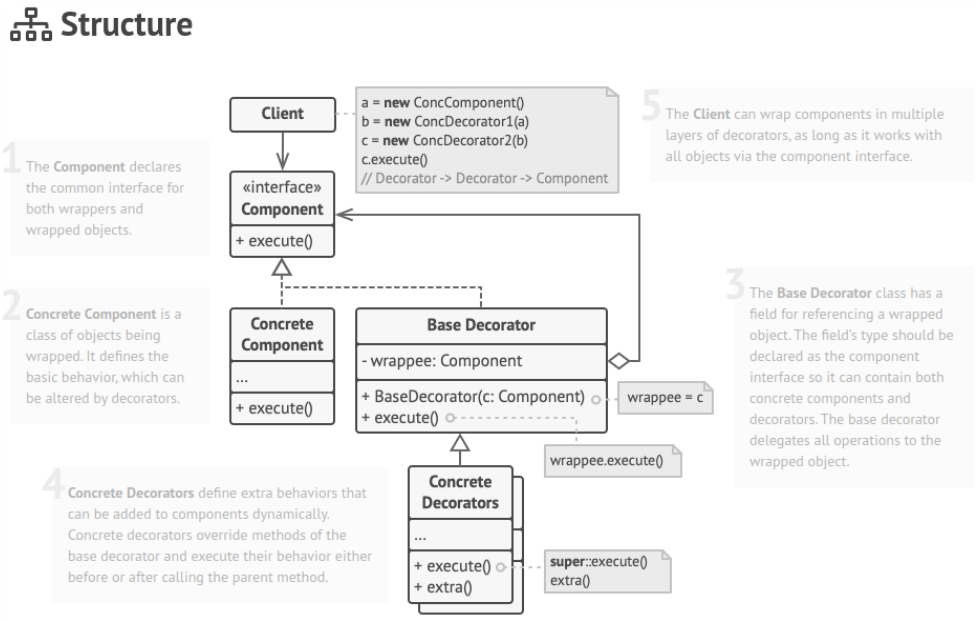
Lista 9

Dodawanie zachowania do klas: programowanie z wykorzystaniem cech/domieszek/wielodziedziczenia i wzorca projektowego dekorator

Przez termin *domieszka* (ang. *mixin*) najczęściej się rozumie klasę lub klasopodobny konstrukt, który ma za zadanie dostarczyć metod (funkcjonalności) do użycia poprzez "wstrzyknięcie" je do klasy bez konieczności stanowienia klasy nadrzędnej. Często przyjmuje się, że domieszki działają podobnie do związku generalizacja-specjalizacja, jednak nie niosą wraz z sobą semantyki "IS-A".

Implementacja i sposób wykorzystania domieszek zależy od sposobu, w jaki język programowania wspiera paradygmat obiektowy. Zazwyczaj, do klas może zostać dołączonych wiele domieszek. Domieszki mogą rozwiązać problem pozyskiwania zachowania z wielu klas bez potrzeby wprowadzania wielodziedziczenia, które w wielu językach programowania jest albo zabronione, albo posiada konsekwencje (np. problem diamentu).

W przypadku, gdy zdefiniowanie pełnego zachowania obiektu w czasie kompilacji byłoby niemożliwe, trudne lub nieefektywne, dobrym rozwiązaniem jest wykorzystanie wzorca projektowego *Dekorator*. Wykorzystanie tego wzorca projektowego pozwala zdefiniować komponent, który będzie posiadał odwołanie do wewnętrznego, bazowego obiektu (poprzez kompozycję), a zarazem dostarczał taki sam interfejs dla klienta, jak bazowy obiekt (poprzez podtypowanie). Co więcej, "dekoracja" samego obiektu może być wykonana dynamicznie, w czasie działania programu.



Źródło: https://refactoring.guru/design-patterns/decorator

Zadanie 1

Jeśli jeszcze nie było okazji, zapoznaj się z mechanizmem zarządzania wersjami kodu git. W szczególności,

- jak klonować istniejące repozytoria,
- jak inicjalizować repozytorium i konfigurować repozytoria zdalne,
- jak tworzyć wrzutki (ang. *commits*),
- jak rozgałęziać repozytoria.

Jeśli jeszcze nie zostało to zrobione, przy użyciu polecenia `git init` utwórz repozytorium w ramach swojego projektu z listy 8. Zadania 2 i 3 należy wykonać i zapisać w ramach oddzielnych gałęzi różnych od gałęzi master/main.

Zadanie 2

Wykorzystując mechanizm domieszek lub mimikując go w wybranym języku programowania, zmodyfikuj program z listy 8 w taki sposób aby dodać do niego opisane poniżej funkcjonalności.

- Samodzielnie zaprojektuj i zaimplementuj mechanizm tworzenia *singletonów*, tj. takich typów elementów, które mogą na scenie pojawić się tylko raz. Niech rolę

singletonu pełni pewien wybrany typ elementów na scenie, np. trójkąt. Niech utworzenie kolejnego *singletonu* spowoduje usunięcie już istniejącego na scenie i nadpisanie go nowym.

Zadanie 3

Wykorzystując wzorzec projektowy *dekorator*, zmodyfikuj program z listy 8 w taki sposób aby dodać do niego opisane poniżej funkcjonalności.

- Samodzielnie zaprojektuj i zaimplementuj mechanizm wybierania obiektów na scenie. Zaznaczenie obiektu powinno spowodować, iż mechanizm renderowania obiektów wyświetli na scenie prostokąt o współrzędnych określonych przez metodę `getBoundingBox`. Sposób implementacji samego zaznaczania jest dowolny, np. poprzez najechanie na element myszką, wybór z listy, etc.