Wzorzec projektowy - Iterator

Mikołaj Paździor 272571

# Problem

Jednym z popularniejszych typów danych wśród programistów są kolekcje.

Niezależnie od doboru struktury kolekcji, powinna ona udostępniać dostęp do swoich elementów tak, aby można było z nich korzystać w różnych miejscach programu. W szczególności zależałoby nam na dostępie sekwencyjnym, czyli przejściu po elementach po kolei.

O ile w przypadku struktur takich jak lista, problem wydaje się prosty – tak w przypadku np. drzewa problem ten może być skomplikowany.

Klient w tym przypadku może nie być zainteresowany w jaki sposób przechodzimy po elementach kolekcji, oczekuje tylko wyniku.

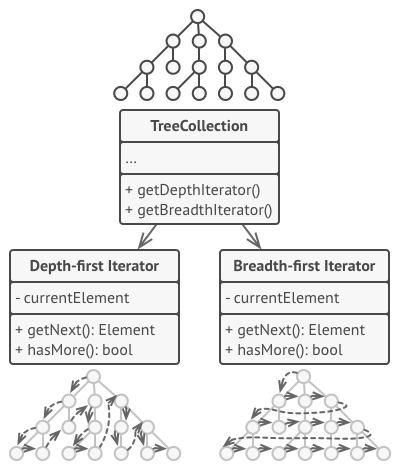
# Rozwiązanie

Główną ideą wzorca Iterator jest ekstrakcja zadań związanych z przechodzeniem przez elementy kolekcji do osobnego obiektu zwanego iteratorem.

Zazwyczaj, iteratory udostępniają jedną główną metodę pobierającą elementy kolekcji. Klient może wywoływać ją raz za razem aż przestanie ona zwracać kolejne obiekty, co oznacza osiągnięcie końca zbioru.

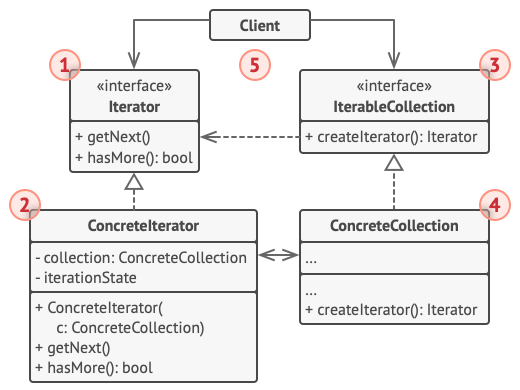
## Wskazówki co do implementacji

1. Zadeklaruj interfejs iteratora. W najprostszym przypadku musi posiadać metodę pobierającą kolejny element kolekcji oraz swój stan (pozycję). Tworzymy interfejs, ponieważ moglibyśmy chcieć posiadać wiele iteratorów dla danej kolekcji, przykład poniżej.



Rysunek Iteratory implementują różne algorytmy sekwencyjnego dostępu do kolejnych elementów. Wiele obiektów iterator może przeskakiwać po elementach jednej kolekcji jednocześnie.

1. Zadeklaruj interfejs kolekcji z metodą pobierającą iterator.
2. Zaimplementuj konkretne klasy iterator dla kolekcji, które umożliwiają przeglądanie struktury. Obiekt iterator musi być powiązany z jedną instancją kolekcji!!! Zazwyczaj takie powiązanie tworzy się w konstruktorze.
3. Przy używaniu wzorca wygląda to następująco:
   1. Klient pobiera nowy obiekt iteratora.
   2. Klient za pomocą iteratora może przejrzeć kolekcję.



# Konsekwencje

## Zalety:

* Zasada pojedynczej odpowiedzialności – przeglądaniem kolekcji zajmuje się jedna klasa.
* Zasada otwarte/zamknięte – można zaimplementować nowe typy kolekcji dla których iterator będzie także działał poprawnie
* Można przeglądać tą samą kolekcję równolegle wieloma iteratorami, ponieważ każdy z nich przechowuje własny stan. Dzięki posiadania stanu można także opóźniać iterację, gdy zajdzie taka potrzeba.

## Wady

* Dla prostych kolekcji iterator jest za dużym skomplikowaniem problemu.
* Używanie iteratora może być mniej efektywne.