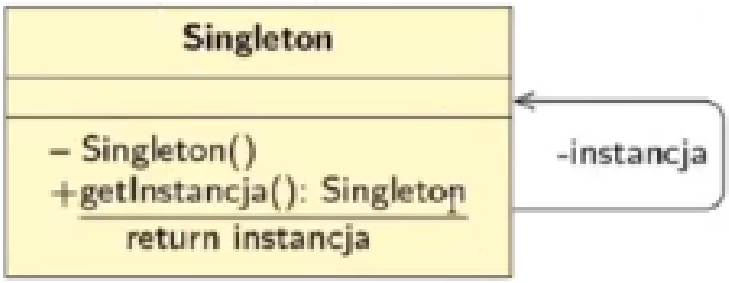
# Contents


# 1 wzorce kreacyjne

## 1.1 singleton




Rysunek 9: Schemat wzorca *Singleton*

- zagwaratowac ze jest jeden obiekt tego typu (np. konfiguracja/stan globalny)

### 1.1.1 implementacja

```
class singleton {

private static singleton; //nasz obiekt
```

```
public static singleton getSingleton() //statyczna publiczna funkcja do otrzymywania te
{
if(instancja==null)
instancja = new Singleton();

return singleton;
}
};
```

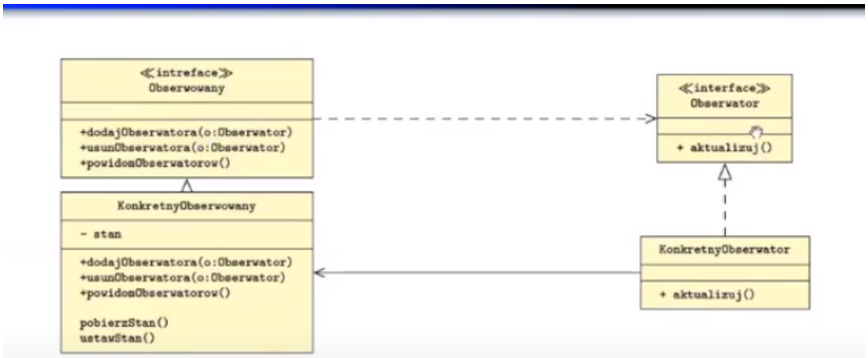
### 1.2 iterator

- hermetyzacja iteracji

```
Iterator iterator = menuCostam.utworzIterator();
while (iterator.hasNext())
{
 pozycjaMenu pozycja = iterator.next();
}
```

## 2 wzorce behawioralne

### 2.1 Obserwator




**Rysunek 3:** Diagram klas *wzorca Obserwator* (pull)

- okresla zaleznosc jeden do wiele miedzy obiektami
- gdy jeden obiekt zmienia stan wszystkie obiekty od niego zalezne sa o tym automatycznie powiadamiane i uaktualniane (np. w kalkulatorze

mamy 3 klasy wypisywania ktore maja w sobie string do wypisywania, kiedy wprowadzamy nowe dzialanie wszyskie sa updatowane)

- wydaje mi sie ze realizowany w grach -> bo trzeba updatowac stan obiektow a one musza znac stan innych

### 2.1.1 kontekst

zmiana stanu jednego obiektu wymaga zmiany innych i nie wiadomo, ile obiektow trzeba zmienic

### 2.1.2 problem

obiekt powinien byc w stanie powiadamiac inne obiekty, nie przyjmujac zadnych zalozen co do tego, co te obiekty reprezentuja - wynikiem sa luzniejsze powiazania miedzy obiektami

### 2.1.3 implementacja

`https://refactoring.guru/design-patterns/observer` zagwarantowanie ze przed rozeslaniem powiadomienia stan obserwowanergo jest wewnetrznie spojny

model push (obserwowany wysyla wszystkie informacje same) model pull (obserwowany wysyla POWIADOMIENIE a kazdy inny pyta sie to czego potrzebuje z jakiejs zmiany)

## 2.2 Stan

- umozliwia obiektowi zmiane zachowania, gdy zmienia sie jego stan wewnetrzny (np. ktos zmienia typ konta bankowego)

### 2.2.1 kontekst

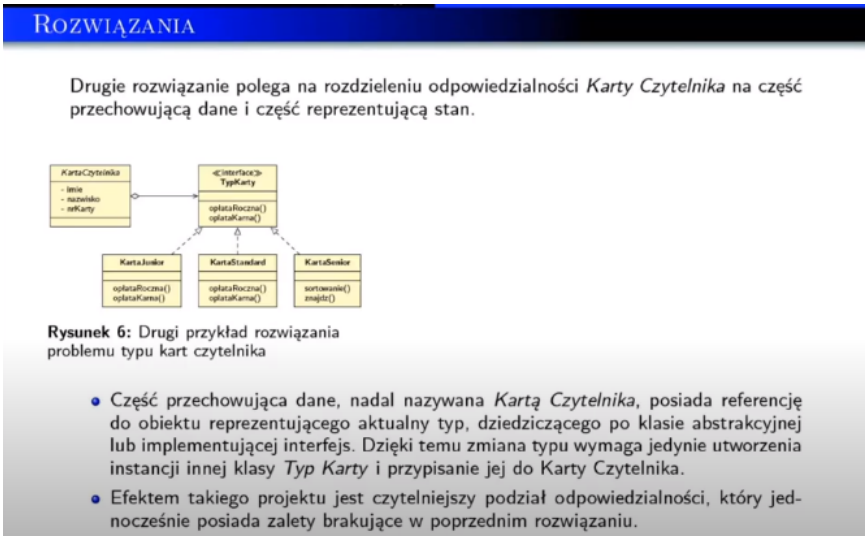
- zachowanie obiektu zalezy od jego stanu, a obiekt ten musi zmieniac swoje zachowanie w czasie wykonywania programu w zaleznosci od stanu

- operacje zawieraja duze, wieloczesciowe instrukcje warunkowe ktore zaleza od stanu obiektu - wzorzec State przenosi kazde rozgalezienie do specjalnej klasy z inna implementacja np. pobierz podatek

### 2.2.2 problem

chemy umozliwic obiektowi zmiane zachowania w momencie zmiany wewnetrzengo stanu obiektu hermetyzujac stan w postaci klasy

### 2.2.3 implementacja

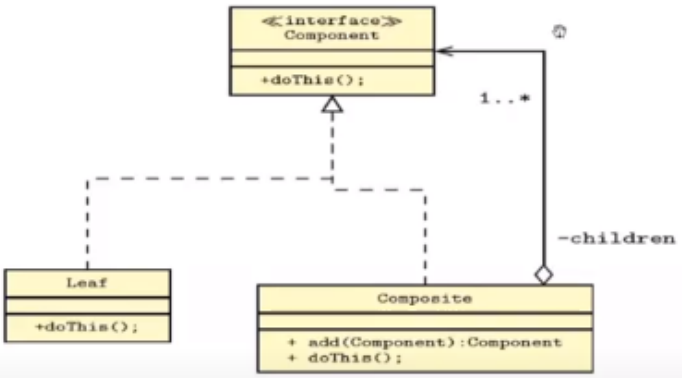



## 2.3 strategia

-

# 3 wzorce strukturalne

## 3.1 kompozyt




**Rysunek 2:** Schemat *wzorca Kompozyt*

TLDR: Drzewko w ktorym lisc zawiera siebie + liste dzieci

- zadaniem jest laczenie obiektow w struktura tak, ze reprezentuja hierarchie czesci-calosci, unifikujac dostep do kolekcji jak i pojedynczego obiektu.
- + umozliwa to klientom jednolite traktowanie pojedynczych obiektow i rowniez ich kompozycji

### 3.1.1 kontekst

chcemy przedstawic hierarchie obiektow czesc-calosc Hierarchia obiektow ma wspolna klase bazowa (abstrakcyjną)

### 3.1.2 problem

chcemy, aby klienci mogli ignorowac roznice miedzy zlozeniami obiektow a pojedynczymi obiektami - klienci beda wtedy jednakowo traktowac wszyskie obiekty wystepujace w strukturze