

Projektovanje softvera

Adapter



Adapter (1)

- Ime i klasifikacija:
 - *Adapter* – klasni i objektni uzorak strukture (različite realizacije)
- Namena:
 - konvertuje interfejs klase u drugi interfejs koji klijenti očekuju
 - omogućava da rade zajedno klase koje inače to ne bi mogle, zbog različitog zahtevanog i realizovanog interfejsa
- Drugo ime:
 - Omotač (engl. *Wrapper*) – prikladno za objektni Adapter
 - dvoznačno, koristi se i za *Dekorater*

Adapter (2)

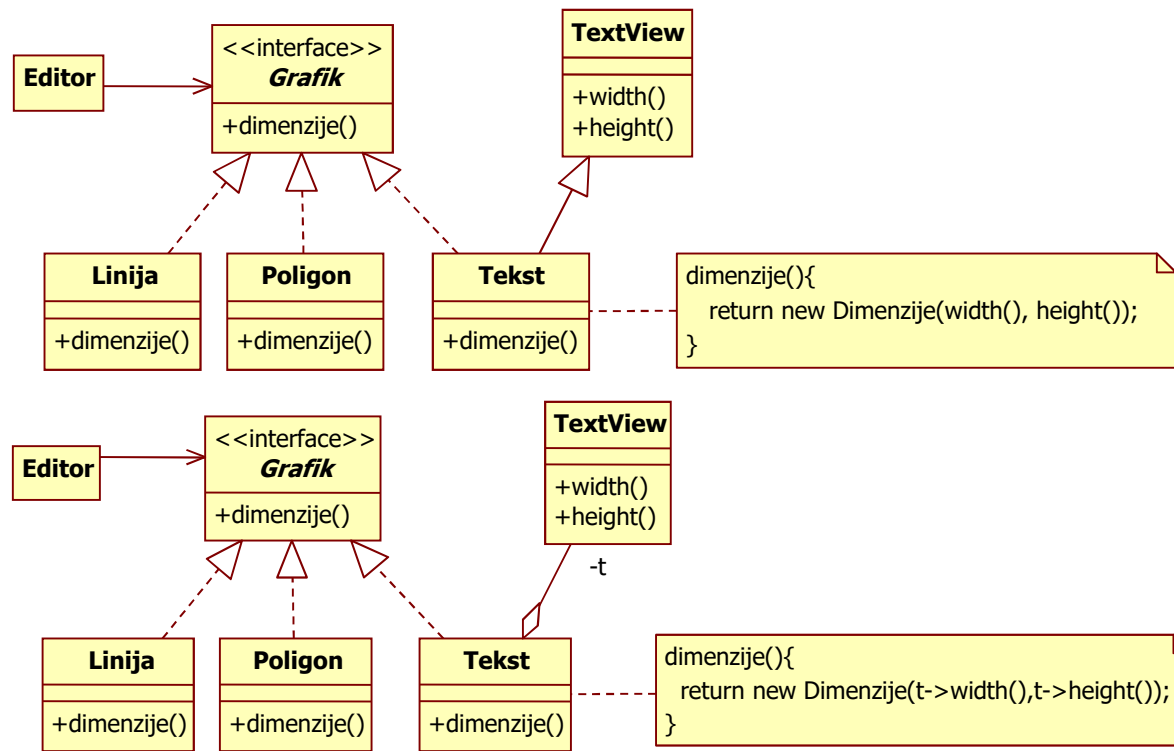
- Motivacija:
 - ponekad neka korisna klasa iz biblioteke nije upotrebljiva
 - razlog: njen interfejs ne odgovara domenski specifičnom interfejsu kakav očekuje aplikacija
 - razmatra se primer grafičkog editora
 - editor omogućava crtanje i aranžiranje grafičkih elemenata
 - grafički elementi su linije, poligoni, tekst
 - oni se nalaze na crtežima
 - ključna apstrakcija grafičkog editora je grafički objekat
 - grafički objekat ima editabilan oblik i može sam da se nacrtava
 - interfejs za grafičke objekte je definisan apstraktnom klasom (`Grafik`)
 - editor definiše potklase klase `Grafik` za svaki tip grafičkog objekta: `Linija`, `Poligon`, `Tekst`, ...

Adapter (3)

- Motivacija (nastavak):
 - klasa `Tekst` je značajno složenija za implementaciju od drugih klasa
 - postojeća klasna biblioteka nudi klasu `TextView`
 - za prikaz i editovanje teksta
 - problem:
 - klasna biblioteka nije imala u vidu klasu `Grafik` kada je definisan `TextView`
 - ne mogu da se koriste naslednici `Grafik` i `TextView` polimorfno (različiti interfejsi)
 - rešenje:
 - definisati `Tekst` na način da adaptira interfejs `TextView` prema interfejsu `Grafik`
 - implementirati `Tekst` pomoću `TextView`
 - ovo može da se uraditi na jedan od dva načina:
 - implementacijom interfejsa `Grafik` i nasleđivanjem `TextView` ili
 - agregacijom `TextView` u `Tekst`
 - ova dva pristupa odgovaraju klasnoj i objektnoj verziji uzorka *Adapter*

Adapter (4)

- Motivacija (nastavak):



Adapter

22.11.2023.

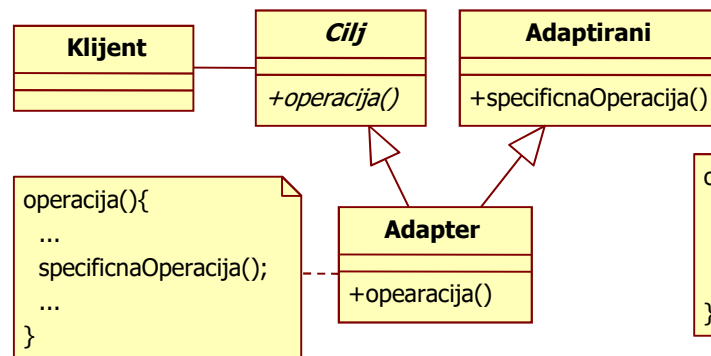
Adapter (5)

- **Primenljivost:**
 - generalno, uzorak treba koristiti kada
 - postoji potreba da se koristi neka raspoloživa klasa koja nema interfejs kakav odgovara
 - postoji potreba da se projektuje reupotreblijiva klasa koja saraduje sa nepovezanim i nepredviđenim klasama, t.j. klasama čiji interfejsi nisu neophodno kompatibilni
 - objektni adapter treba koristiti kada
 - treba koristiti nekoliko postojećih klasa, ali je nepraktično adaptirati njihove interfejse višestrukim izvođenjem iz svake od tih klasa

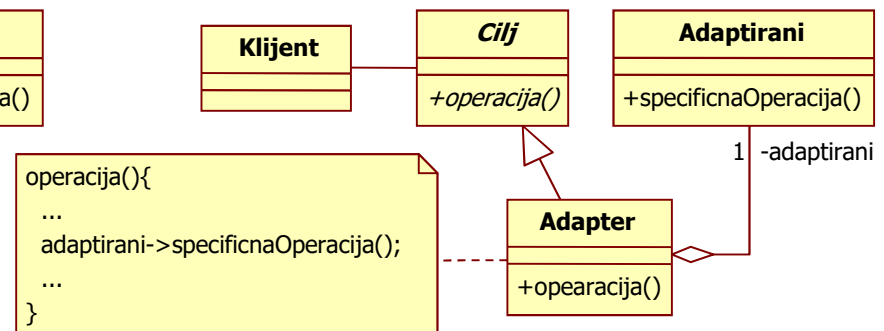
Adapter (6)

- Struktura:

- klasni adapter:



- objektni adapter:



Adapter (7)

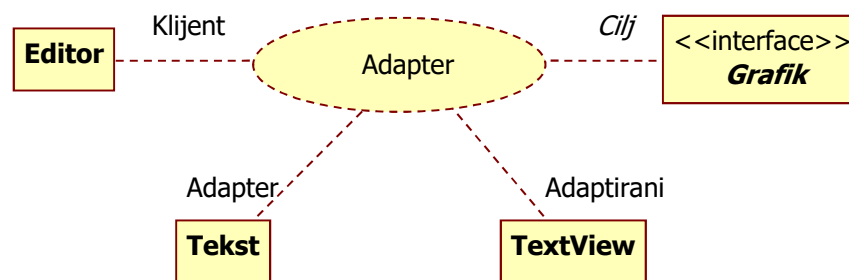
- Učesnici:
 - *Cilj* (interfejs *Grafik*)
 - definiše domenski-specifičan interfejs koji koristi klijent
 - Klijent (klasa *Editor*)
 - saraduje sa objektima koji poštuju interfejs *Cilj*
 - Adaptirani (klasa *TextView*)
 - definiše postojeću implementaciju sa drugačijim interfejsom koja zahteva adaptiranje
 - Adapter (klasa *Tekst*)
 - adaptira interfejs *Adaptirani* na interfejs *Cilj*
- Saradnja:
 - klijenti pozivaju operacije objekta adaptera, a adapter poziva operacije adaptiranog podobjekta (nasleđivanjem ili agragacijom) da izvrše zahtev

Adapter (8)

- **Posledice:**
 - klasni *Adapter* ima sledeće karakteristike:
 - ne odgovara kada se želi istovremeno adaptiranje više potklasa neke klase
 - dopušta adapteru kao potklasi da redefiniše metode adaptirane natklase
 - uvodi samo jedan objekat i nema potrebe za dodatnom indirekcijom da se stigne do adaptiranog objekta
 - objektni *Adapter* ima sledeće karakteristike:
 - dopušta jednom adapteru da radi sa hijerarhijom adaptiranih klasa (adapter može da doda funkcionalnost za sve adaptirane klase odjednom)
 - moguća dinamička promena adaptiranog objekta iz hijerarhije klasa
 - identitet ciljnog objekta i adaptiranog objekta se razlikuje
 - dvosmerni *Adapter* – zadržava i interfejs `Adaptirani`
 - objekat može da se koristiti i kao `Cilj` i kao `Adaptirani`
 - postiže se transparentnost za različite klijente

Adapter (9)

- UML notacija:



- Implementacija:

- C++: klasni uzorak može da se realizuje
 - javnim izvođenjem iz `Cilj` i privatnim iz `Adaptirani`
 - ukoliko se ne želi dvosmerni adapter

- Povezani uzorci:

- i *Dekorater* i objektni *Adapter* prave omotače nekog objekta
- *Dekorater* dopunjuje drugi objekat ne menjajući mu interfejs, a *Adapter* upravo menja interfejs adaptiranog objekta
 - posledica: *Dekorater* podržava rekurzivne kompozicije, a *Adapter* ne
- *Most* ima sličnu strukturu objektom adapteru
 - namena je drugačija