Projektovanje softvera

Komanda

Komanda (1)

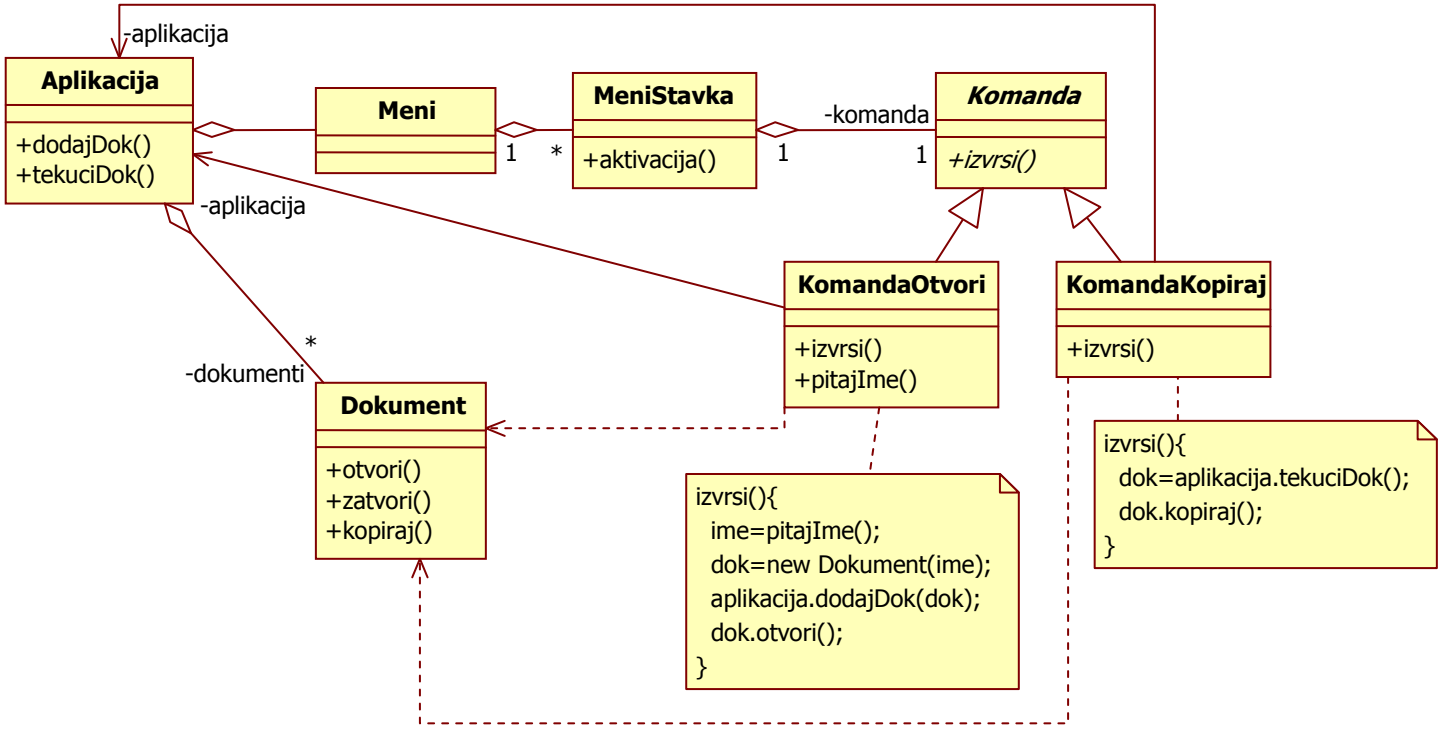
- Ime i klasifikacija:
  - Komanda (engl. *Command*) – objektni uzorak ponašanja
- Namena:
  - kapsulira zahtev u jedan objekat, omogućavajući:
    - da se klijenti parametrizuju različitim zahtevima,
    - da se zahtevi isporučuju kroz red čekanja,
    - da se pravi dnevnik *(log)* zahteva i
    - da se efekti izvršenog zahteva poništе (*undo*)
- Drugo ime:
  - Akcija, Transakcija (engl. *Action*, *Transaction*)

Komanda (2)

- Motivacija:
  - nekad je potrebno izdati zahtev da se izvrši neka operacija bez znanja o samoj zahtevanoj operaciji i izvršiocu (primaocu zahteva)
  - npr. GUI biblioteke sadrže objekte kao što su dugmad ili meniji
    - ovi objekti izvršavaju zahteve koji su posledica akcije korisnika
    - biblioteka klasa ne može da realizuje adekvatne operacije u ovim objektima
    - samo ciljna aplikacija zna šta je specifična operacija za neko dugme
    - projektant biblioteke ne zna ništa o operaciji ni o primaocu zahteva
  - uzorak *Komanda*:
    - dopušta objektima biblioteke da zahtevaju da nepoznate operacije budu izvršene od nepoznatih objekata aplikacije
    - to postiže smeštajući zahteve za operacijama u posebne objekte
  - objekat sa zahtevom se može zapamtiti ili proslediti drugom objektu

Komanda (3)

- Motivacija (nastavak):



Komanda (4)

- Motivacija (nastavak):
  - ključna apstrakcija uzorka je klasa *`Komanda`*
    - deklariše interfejs za izvršenje operacija
    - u najjednostavnijoj formi, interfejs se sastoji od apstraktne operacije *`izvrsi()`*
  - potklase klase *`Komanda`* specificiraju par (primalac komande, operacija)
    - primalac zna kako da izvrši akcije koje su potrebne za ispunjenje zahteva
  - meniji se lako mogu implementirati koristeći objekte potklasa klase *`Komanda`*
  - uzorak raspreže objekat pozivaoca operacije od onog ko zna kako da je izvrši
    - objekat koji izdaje komandu treba da zna samo kako se ona izdaje
    - on ne treba da zna ništa o tome kako se izvršava i ko je izvršava
  - komande se mogu menjati dinamički
    - u konkretnom primeru menija, ovo omogućava kontekstno zavisne menije
  - jednostavna je implementacija skriptova (složenih komandi - sekvenci operacija)

Komanda (5)

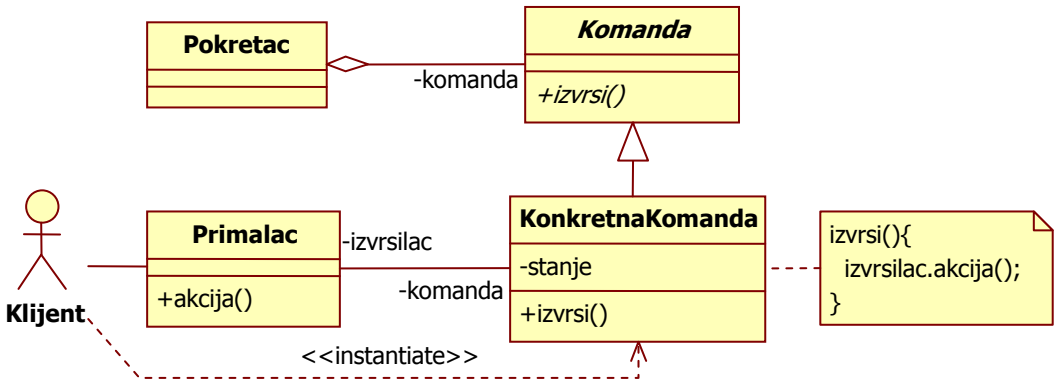
- Primenljivost:
  - kada treba parametrizovati objekte akcijom koju treba da obave
    - zamena za funkciju povratnog poziva (*callback*) u tradicionalnim jezicima
  - kada treba specificirati i stavljati u red čekanja zahteve a kasnije ih izvršavati
    - objekat komande može imati različit životni vek od onog ko izdaje zahtev
    - objekat komande se može prepustiti drugom procesu (promena adresnog prostora), ako se primalac može adresirati univerzalno
  - kada treba podržati *undo*
    - `izvrsi()` operacija može sačuvati u samom objektu stanje za rastauraciju
    - interfejs treba da sadrži i `ponisti()` operaciju koja restaurira stanje
    - neograničen nivo *undo* i *redo* se postiže smeštanjem objekata izvršenih komandi u listu, odnosno prolaskom kroz listu unazad i unapred

Komanda (6)

- Primenljivost:
  - kada treba podržati *recovery*
    - potrebno je snimati promene da bi se one mogle ponovo uraditi
    - u interfejs klase *Komanda* se dodaju operacije za perzistenciju dnevnika promena
    - oporavak se postiže učitavanjem dnevnika sa diska i ponovnim izvršavanjem `izvrsi()`
  - kada treba podržati transakcije
    - transakcije su složene operacije sastavljene od primitivnih
    - transakcije kapsuliraju skup promena podataka

Komanda (7)

- Struktura:

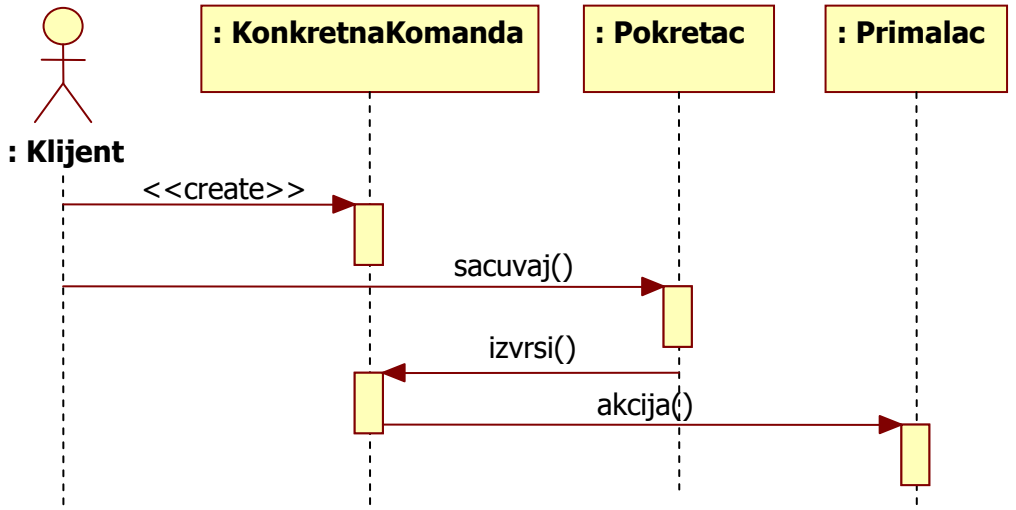


Komanda (8)

- Učesnici:
  - `Komanda` (klasa `Komanda`)
    - deklariše interfejs za izvršenje neke operacije
  - `KonkretnaKomanda` (klase `KomandaOtvori`, `KomandaKopiraj`)
    - definiše vezu između jednog objekta `Primalac` i akcije
  - `Kijent` (klasa `Aplikacija`)
    - kreira objekat `KonkretnaKomanda` i postavlja njen objekat `Primalac`
  - `Pokretac` (klasa `MeniStavka`)
    - traži od komande da izvrši zahtev
  - `Primalac` (klasa `Dokument`)
    - zna kako da izvrši operacije pridružene ispunjavanju zahteva

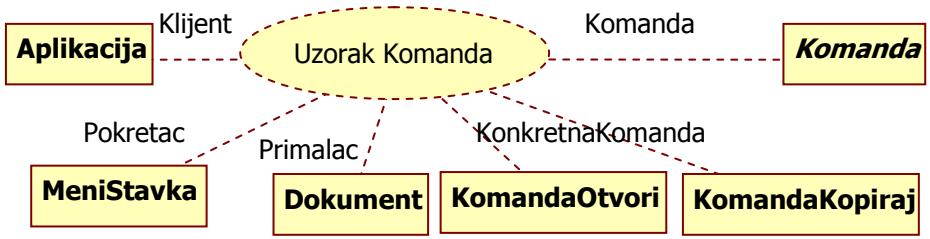
Komanda (9)

- Saradnja:



Komanda (10)

- UML notacija:



- Posledice:
  - raspreže objekat koji pokreće operaciju od onog koji zna kako da je izvrši
  - komande su objekti kao i svi drugi i njima se može manipulisati
  - komande se mogu asemblirati u kompozitne komande (makrokomande)
  - jednostavno je dodavanje novih komandi, ne treba menjati postojeće klase
- Povezani uzorci:
  - *Kompozicija* se koristi za kreiranje makrokomandi (skriptova)
  - *Podsetnik* može da čuva stanje pre izvršenja komande potrebno za *Undo*
  - komanda čija se kopija stavlja u listu istorije se ponaša kao *Prototip*