Projektovanje softvera

Dekorater

Dekorater (1)

- Ime i klasifikacija:
  - Dekorater (engl. *Decorator*) – objektni uzorak strukture
- Namena:
  - dinamički dodaje odgovornosti (mogućnosti) nekom objektu
  - fleksibilna alternativa izvođenju za proširivanje funkcionalnosti
- Druga imena:
  - Dopuna, Omotač (engl. *Wrapper*)

Dekorater (2)

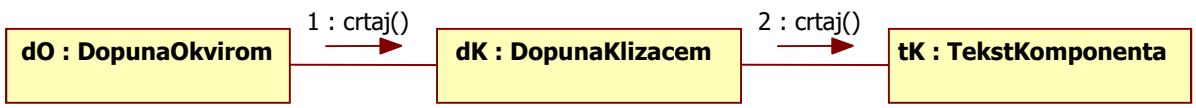
- Motivacija:
  - alati za GUI podržavaju dodavanje “ukrasa” na razne komponente
    - klizači, okviri sa raznim efektima,…
  - nefleksibilan način za dodavanje ukrasa – nasleđivanje sa proširivanjem
    - nasleđivanje je nefleksibilno, jer se ukrasi definišu statički
    - ukrasi se ne mogu dinamički kombinovati na osnovnoj komponenti
    - opasnost od eksplozije broja izvedenih klasa kada se kombinuju ukrasi
  - fleksibilniji pristup je da se komponenta obuhvati drugom komponentom
    - komponenta koja obuhvata osnovnu dodaje ukras (okvir, klizače)
    - ukrasi se dodaju pojedinom objektu, a ne celoj klasi
    - ukrasi se mogu menjati dinamički
    - generalno – proširenja mogu biti specifična stanja ili ponašanja

Dekorater (3)

- Motivacija (nastavak):
  - komponenta koja obuhvata osnovnu se naziva
    - dekoraterom, omotačem ili dopunom
  - dekorater prosleđuje zahteve klijenta obuhvaćenoj komponenti i može da obavi dodatne akcije
    - kao što je crtanje okvira ili dodavanje klizača za pomeranje
  - dekorater nasleđuje interfejs komponente koju ukrašava
  - prisustvo dekoratera je transparentno za klijente komponente
    - transparentnost omogućava neograničen broj dodataka uvedenih posebnim dekoraterima

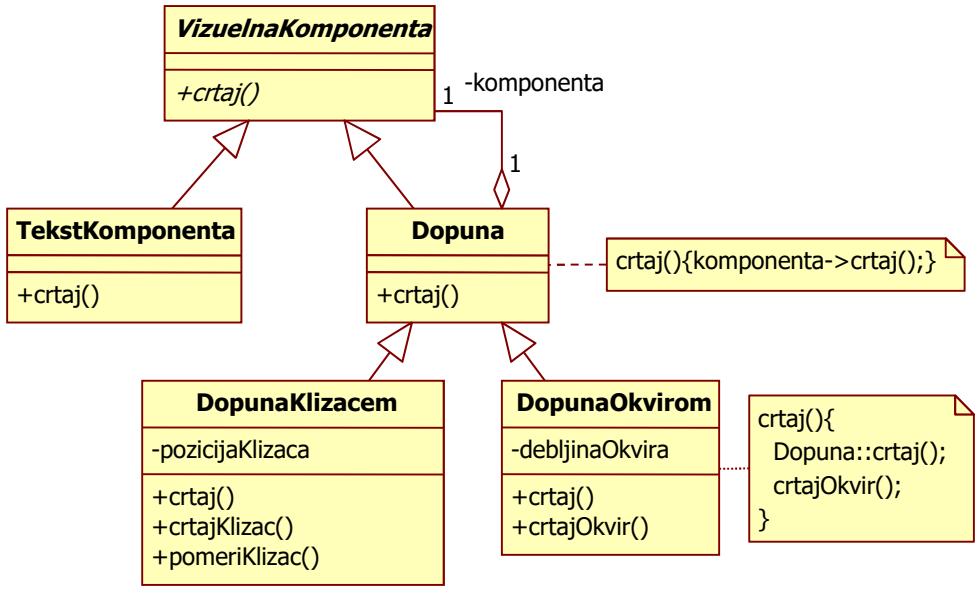
Dekorater (4)

- Motivacija (nastavak):
  - primer:
    - `TekstKomponenta` prikazuje tekst u prozoru i nema klizače za H/V pomeranje sadržaja, ni okvir
    - ako su potrebni klizači
      - dodaje se objekat klase `DopunaKlizacem` koji ih dodaje
    - ako se želi i okvir
      - dodaje se i objekat klase `DopunaOkvirom` koji crta okvir
    - sledeći dijagram prikazuje objektnu kompoziciju i interakciju:



Dekorater (5)

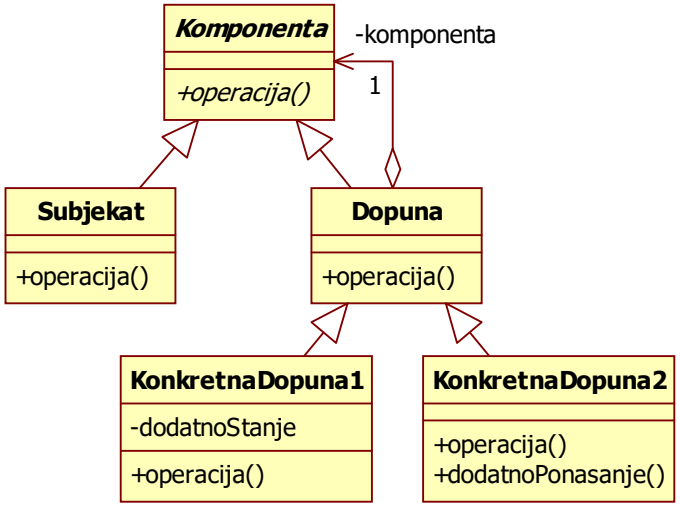
- Motivacija (nastavak):
  - sledeći dijagram prikazuje opisanu klasnu strukturu:



- Primenljivost:
  - kada je potrebno dinamički dodavati odgovornosti objektima na transparentan način
  - kada proširenje izvođenjem nije praktično (kada je moguć veliki broj nezavisnih proširenja, pri čemu se ona mogu kombinovati, što vodi eksploziji broja klasa)

Dekorater (6)

- Struktura:



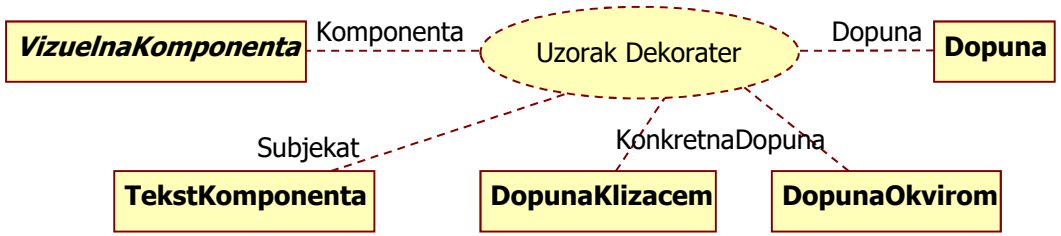
- Učesnici:
  - `Komponenta` (klasa `VizuelnaKomponenta`)
    - definiše interfejs za objekte kojima se mogu dinamički dodavati odgovornosti
  - `Subjekat` (klasa `TekstKomponenta`)
    - definiše objekat kojem se dodaju odgovornosti
  - `Dopuna` (klasa `Dopuna`)
    - održava referencu na objekat klase `Komponenta` i nasleđuje interfejs klase `Komponenta`
  - `KonkretnaDopunaX` (klase `DopunaKlizacem`, `DopunaOkvirom`)
    - dodaje odgovornost objektu tipa `Komponenta`
- Saradnja:
  - objekat `KonkretnaDopunaX`
    - prosleđuje zahteve obuhvaćenom objektu (tipa `Komponenta`)
    - izvršava dodatne operacije pre i/ili posle prosleđivanja zahteva

Dekorater (7)

- Posledice
  - prednosti
    - veća fleksibilnost od statičkog nasleđivanja
      - dinamičko dodavanje odgovornosti
    - izbegavanje eksplozije potklasa koje kombinuju dekoracije
    - izbegavanje bogato parametrizovanih klasa visoko u hijerarhiji
  - mane
    - identitet dekoratera i dekorisanog objekta su različiti
    - objekti se ne razlikuju po klasi već po načinu povezivanja
      - potencijalan problem kod razumevanja i održavanja sistema
    - nasleđeni atribut iz klase `Komponenta` – duplikat u svakoj dopuni
      - dobro je da `Komponenta` nema atribute

Dekorater (8)

- UML notacija:



- Povezani uzorci:
  - *Dekorater* zadržava interfejs, a *Adapter* menja interfejs
  - *Kompozicija* ima sličnu klasnu strukturu
    - *Dekorater* je po objektnoj strukturi degenerisan *Sastav* (*Kompozicija, Sklop)*
      - objekat `Dopuna` sadrži samo jednu komponentu, a objekat `Kompozicija` više
      - *Sastav* formira objektno stablo, a *Dekorater* objektnu listu
    - po nameni je *Dekorater* sasvim različit, jer dodaje odgovornosti i nije namenjen grupisanju
  - *Dekorater* i *Strategija* su alternative za promenu ponašanja objekta
    - *Dekorater* menja spoljašnjost objekta, a *Strategija* unutrašnjost