

Projektovanje softvera

Sastav

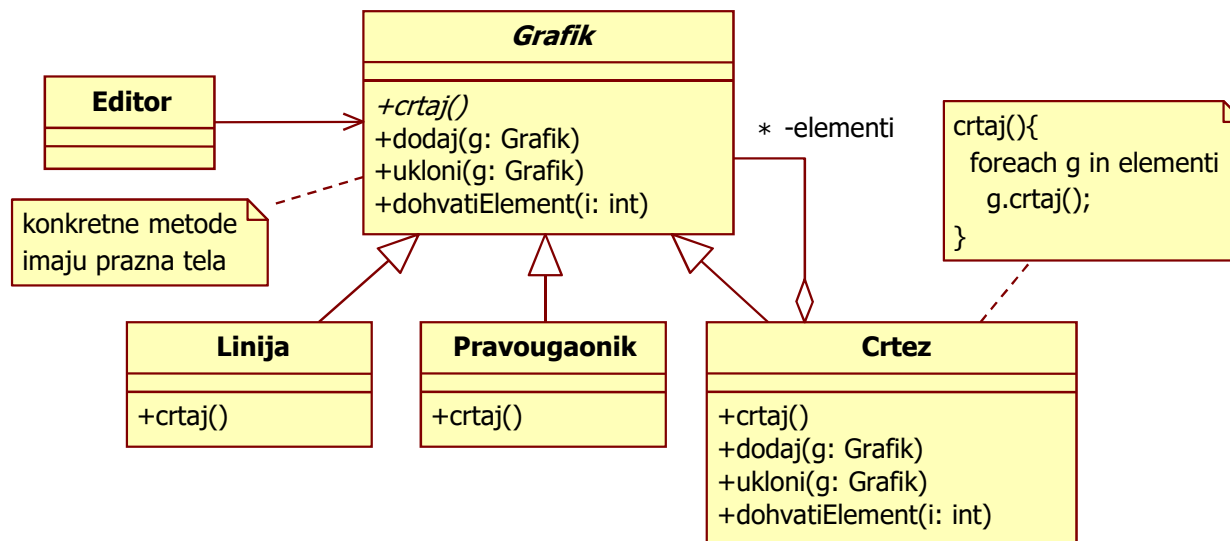


Sastav (1)

- Ime i klasifikacija:
 - Sastav (Sklop, Kompizicija, engl. *Composite*) – objektni uzorak strukture
- Namena:
 - komponuje objekte u strukturu stabla (hijerarhija celina-deo)
 - omogućava klijentima da uniformno tretiraju
 - individualne objekte
 - njihove kompozicije
- Motivacija:
 - grafičke aplikacije kao što su editori šema:
 - omogućavaju crtanje kompleksnih šema sastavljenih od jednostavnih grafičkih elemenata (listova)
 - elementi se mogu grupisati da se formiraju složeni elementi (sastavi)
 - složeni element je potrebno tretirati i kao vrstu grafičkog elementa
 - ključno: apstraktna klasa koja predstavlja i proste i složene elemente

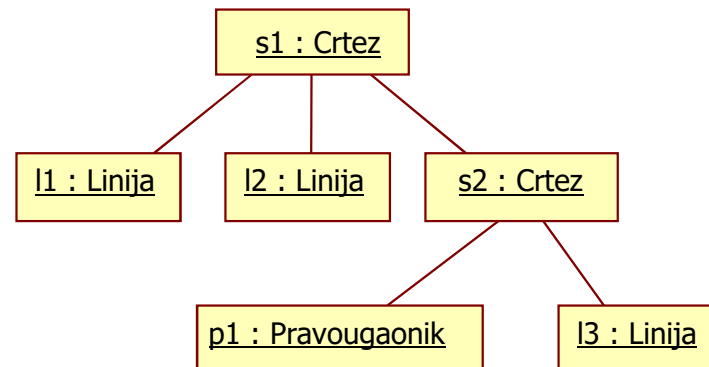
Sastav (2)

- Motivacija (nastavak):
 - klasni dijagram



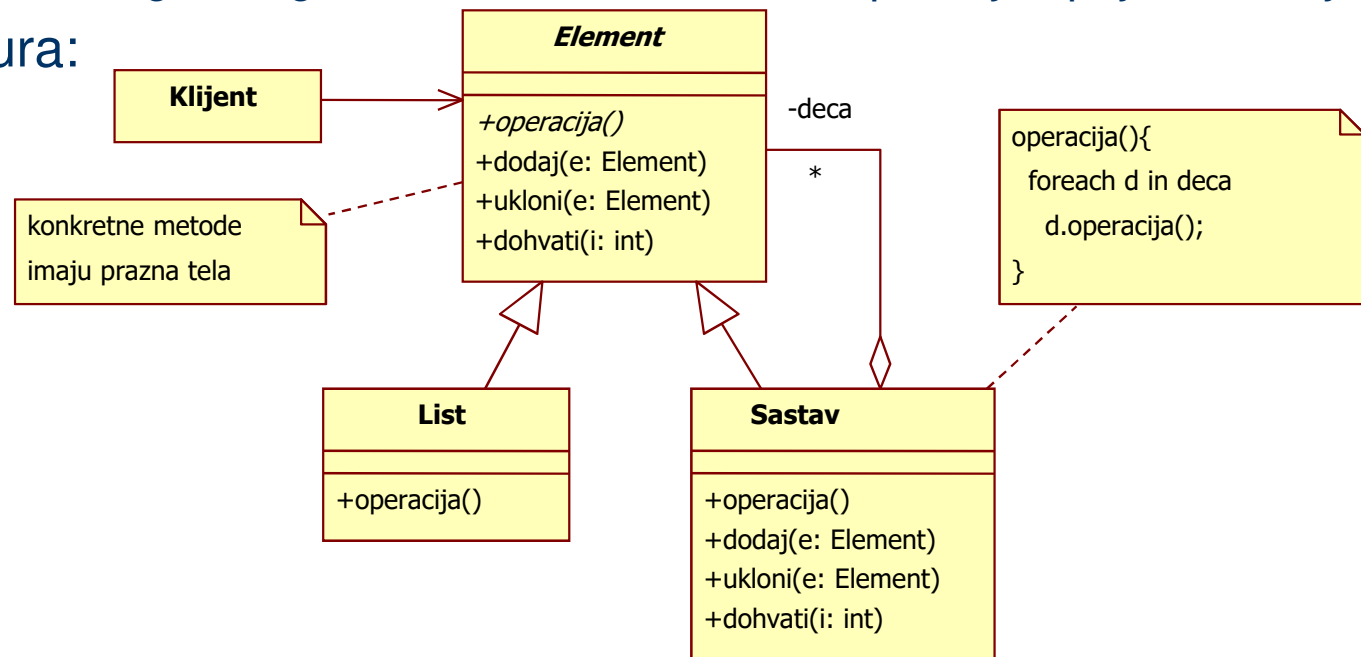
Sastav (3)

- Motivacija (nastavak):
 - objektni dijagram – struktura rekurzivno komponovanih grafičkih objekata:



Sastav (4)

- Primenljivost: uzorak treba koristiti kada se želi da
 - postoje hijerarhije objekata celina-deo takve da su celina i deo iste vrste
 - klijenti mogu da ignorišu razlike između kompozicija i pojedinih objekata
- Struktura:

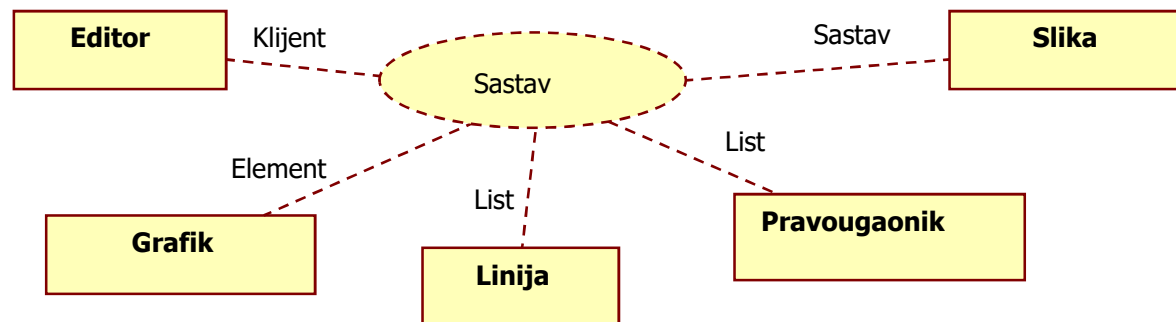


Sastav (5)

- Učesnici:
 - `Element` (klasa `Grafik`)
 - deklarira zajednički interfejs za sve objekte u sastavu
 - implementira podrazumevano ponašanje zajedničko za sve klase
 - deklarira interfejs za pristupanje i upravljanje decom
 - implementira prazne metode za pristup i upravljanje decom (zbog listova)
 - opcionalno deklarira i implementira interfejs za pristup roditelju
 - `List` (klase `Linija`, `Pravougaonik`)
 - reprezentuje individualne objekte – listove u stablu
 - definiše ponašanje za jednostavne objekte
 - `Sastav` (klasa `Crtez`)
 - definiše ponašanje za objekte koji imaju decu
 - sadrži komponente decu
 - implementira operacije za pristup i upravljanje decom
 - `Klijent` (klasa `Editor`)
 - manipuliše objektima u kompoziciji kroz interfejs klase `Element`

Sastav (6)

- Saradnja:
 - klijenti koriste interfejs apstraktne klase `Element` da interaguju sa objektima složene strukture
 - ako je primalac zahteva `List`, zahtev se neposredno izvršava
 - ako je primalac zahteva `Sastav`, obično se zahtev prosleđuje deci
- UML notacija:
 - na primeru grafičkog editora



Sastav (7)

- Posledice:
 - uzorak čini jedostavnim klijente
 - oni tretiraju na jedinstven način sve objekte u hijerarhiji
 - uzorak čini jednostavnim dodavanje nove vrste elemenata
 - nije jednostavno ograničiti vrste elemenata koje neki sastavi sadrže
- Povezani uzorci:
 - često se veza element-roditelj koristi za uzorak *Lanac odgovornosti*
 - ako element ne može da odgovori na zahtev – prosleđuje ga roditeljskom objektu
 - *Dekorater (Dopuna)* ima sličnu strukturu klasa i često se koristi sa *Sastavom*
 - kada se koriste zajedno obično imaju zajedničku natklasu
 - *Muva* dozvoljava strukture sa nedeljenim sastavima i deljenim listovima
 - *Iterator* se koristi često za obilazak strukture *Sastava*
 - *Posetilac* se koristi sa *Sastavom* da lokalizuje operacije i ponašanje koje bi inače bilo distribuirano između klasa `Sastav` i `List`