

Projektovanje softvera

Graditelj



Graditelj (1)

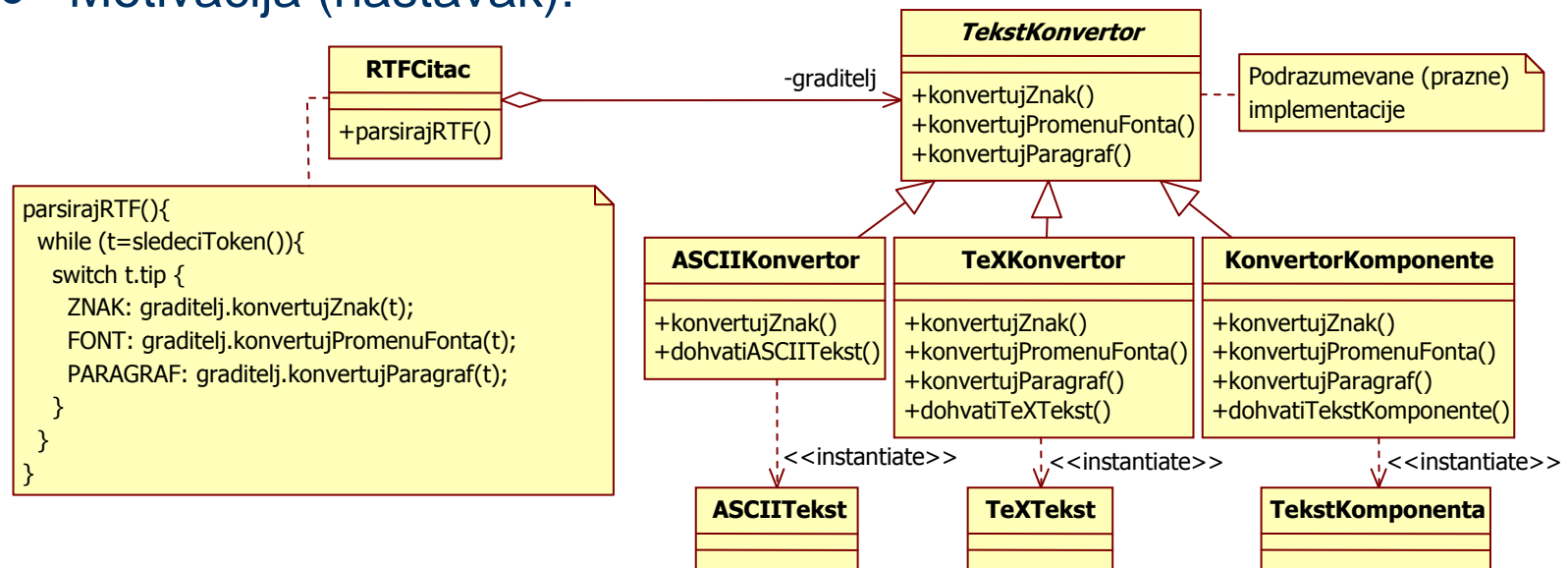
- Ime i klasifikacija:
 - Graditelj (engl. *Builder*) – objektni uzorak kreiranja
- Namena:
 - razdvaja proces konstrukcije kompleksnog objekta od njegove reprezentacije
 - posledica je da isti proces konstrukcije može da kreira različite reprezentacije

Graditelj (2)

- Motivacija:
 - posmatra se aplikacija za čitanje RTF dokumenata (čitač)
 - potrebno je da učitane dokumente može da konvertuje u:
 - čisti ASCII tekst
 - TeX dokument
 - tekstualnu komponentu koja može da se interaktivno edituje
 - ... (broj mogućih konverzija je neograničen)
 - potrebno je da se lako dodaje nova konverzija, bez izmene čitača
 - rešenje je primena uzorka *Graditelj*:
 - konfigurisanje `RTFCitac` klase objektom `TekstKonvertor`
 - `RTFCitac` čita RTF tekst i parsira ga
 - `TekstKonvertor` konvertuje RTF u drugu tekstualnu reprezentaciju
 - kad `RTFCitac` prepozna RTF element
 - on izdaje zahtev `TekstKonvertor` objektu
 - `TekstKonvertor` konvertuje i reprezentuje element u ciljnom formatu
 - potklase `TekstKonvertor` specijalizuju korake u konverziji

Graditelj (3)

- Motivacija (nastavak):



Graditelj (4)

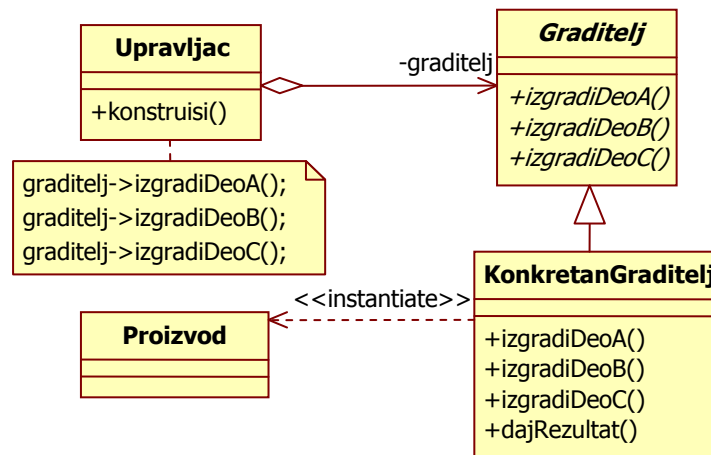
- Motivacija (nastavak):
 - svaka vrsta konvertora, iza apstraktnog interfejsa, ima mehanizam za kreiranje i sastavljanje složenog objekta
 - konvertor je razdvojen od čitača
 - čitač je odgovoran samo za parsiranje ulaznog RTF dokumenta
 - svaka klasa konvertora u uzorku se naziva graditeljem (*builder*)
 - klasa čitača se naziva upravljačem (*director*)
 - u ovom primeru uzorak graditelja je razdvojio
 - algoritam interpretacije tekstualnog formata (parser za RTF dokumente)
 - način kako se konvertovani format stvara i reprezentuje
 - posledica je reupotreba algoritma parsiranja pri kreiranju različitih tekstualnih reprezentacija od RTF dokumenata
 - `RTFCitac` se samo konfigurise objektom neke od potklasa `TekstKonvertor`

Graditelj (5)

- **Primenljivost: uzorak treba koristiti kada**
 - algoritam za kreiranje složenog objekta treba da bude nezavisan
 - od delova koji čine objekat
 - od načina na koji se delovi sklapaju u celinu
 - proces konstrukcije mora da dopusti različite reprezentacije za objekat koji se konstruiše

Graditelj (6)

- **Struktura:**

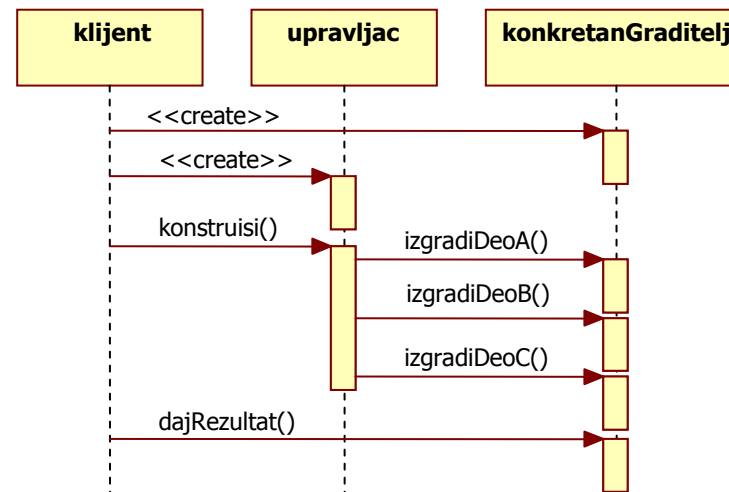


- **Učesnici:**

- Graditelj (klasa TekstKonvertor)
 - specificira apstraktan interfejs za kreiranje delova objekta Proizvod
- KonkretanGraditelj (klase ASCIIKonvertor, TeXKonvertor)
 - konstruiše i sestavlja delove proizvoda implementiranjem interfejsa Graditelj
 - definiše i čuva proizvod koji kreira
 - obezbeđuje interfejs za dohvanjanje proizvoda
- Upravljac (klasa RTFCitac)
 - konstruiše objekat koristeći interfejs Graditelj
- Proizvod (klase ASCII Tekst, TeX Tekst)
 - predstavlja složeni objekat koji se konstruiše
 - uključuje klase koje definišu sastavne delove
 - uključujući interfejs za sestavljanje delova u finalan rezultat

Graditelj (7)

- Saradnja:

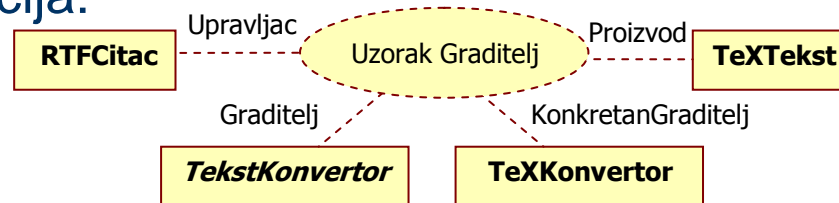


Graditelj (8)

- Posledice:
 - dopušta izmene interne reprezentacije i načina sklapanja složenog proizvoda
 - graditelj pruža upravljaču apstraktan interfejs za konstrukciju proizvoda
 - reprezentacija i interna struktura, kao i način sklapanja su sakriveni
 - za promenu interne reprezentacije potrebna je samo nova vrsta graditelja
 - izolovanje koda za konstrukciju i reprezentaciju
 - bolja modularnost kroz kapsuliranje načina konstrukcije složenog objekta
 - klijenti ne treba da znaju ništa o klasama koje definišu unutrašnju strukturu proizvoda
 - te klase se ne pojavljuju u interfejsu graditelja
 - daje finiju kontrolu nad procesom konstrukcije od drugih fabrika
 - drugi uzorci kreiranja konstruišu proizvod u jednom potezu
 - graditelj konstruiše proizvod korak-po-korak, pod kontrolom upravljača

Graditelj (9)

- UML notacija:



- Povezani uzorci:

- Druge fabrike (npr. *Apstraktna fabrika*) takođe mogu da konstruiše kompleksne objekte
 - *Graditelj* vrši konstrukciju kompleksnog objekta korak-po-korak, druge fabrike u jednom koraku
 - *Graditelj* vraća proizvod kao finalni korak, druge fabrike vraćaju proizvod odmah
- *Graditelj* često gradi objekat *Kompozicije*
- *Upravljac* uzorka *Graditelja* se parametrizuje objektom klase *Graditelj*, kao što se *Kontekst* parametrizuje objektom *Strategija* u odgovarajućem uzorku
 - osim u nameni, razlika je što se *Strategija* najčešće implementira u jednoj metodi dok su kod *Graditelja* implementirani pojedini koraci gradnje kao posebne metode
- *Šablonska metoda* poziva apstraktne metode sopstvene klase, dok kod uzorka *Graditelj*, *Upravljac* sadrži konkretnu metodu koja poziva apstraktne metode klase *Graditelj*