Projektovanje softvera

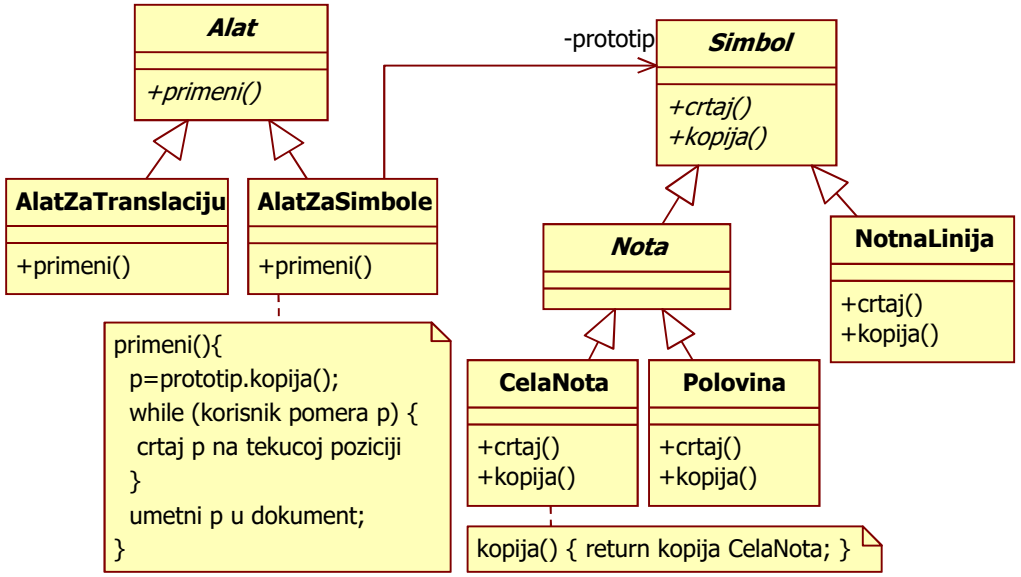
Prototip

Prototip (1)

- Ime i klasifikacija:
  - Prototip (Polimorfna kopija, engl. *Prototype*) – objektni uzorak kreiranja
- Namena:
  - specificira specifične vrste objekata koje se stvaraju
  - stvara nove objekte kopiranjem (kloniranjem) prototipskog primerka
- Motivacija:
  - editor za muzičke partiture bi se mogao realizovati:
    - prilagođenjem opšteg radnog okvira za grafičke editore
    - dodavanjem novih objekata koji predstavljaju note, pauze i druge muzičke simbole
  - radni okvir editora može imati paletu alata za dodavanje simbola dokumentu
    - radni okvir ništa ne zna o konkretnim tipovima simbola koje treba dodavati
    - paleta bi uključila i alate za selektovanje, pomeranje i druge radnje nad simbolima
  - editor za muzičke partiture treba da omogući rad sa muzičkim simbolima
    - korisnik bi kliknuo na alat "četvrtinka" i dodao ovu u partituru
    - korisnik bi koristio alat za pomeranje da premesti notu na drugu notnu liniju

Prototip (2)

- Motivacija (nastavak):
  - apstraktna klasa `Simbol` za proizvoljne grafičke komponente
  - apstraktna klasa `Alat` za alate u paleti
  - konkretna klasa `AlatZaSimbole` za stvaranje i dodavanje objekata u dokument

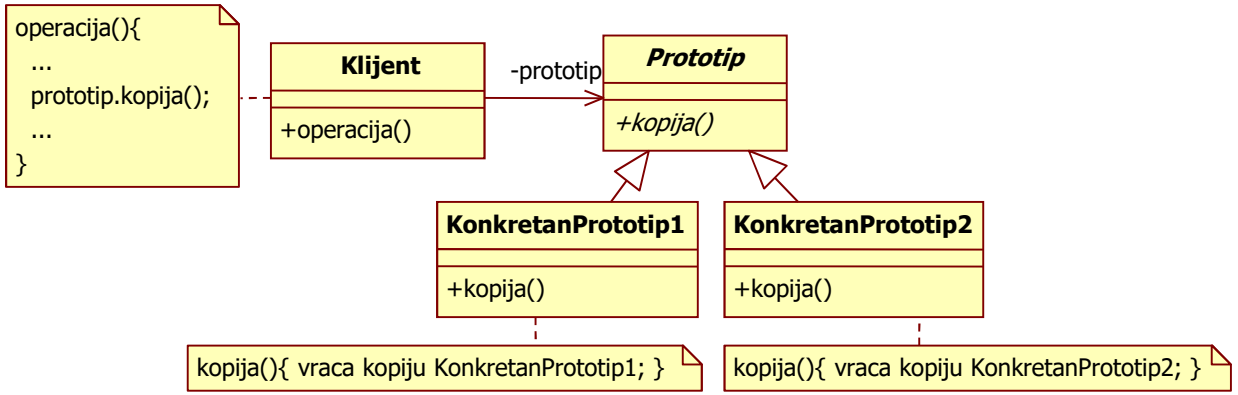


Prototip (3)

- Motivacija (nastavak):
  - problem za projektanta radnog okvira je klasa `AlatZaSimbole`
    - klase za note i notne linije su specifične za aplikaciju, nepoznate radnom okviru
    - mogla bi se praviti potklasa `AlatZaSimbole` za svaki muzički objekat, ali ima ih mnogo
    - potklase bi se razlikovale samo po vrsti muzičkog objekta koje stvaraju
  - kompozicija objekata je česta fleksibilna alternativa izvođenju i nasleđivanju
  - radni okvir parametrizuje primerke `AlatZaSimbole` objektima klase `Simbol`
  - rešenje:
    - `AlatZaSimbole` kreira novi `Simbol` kopiranjem objekta potklase `Simbol`
    - dotični objekat (primerak potklase `Simbol`) se naziva prototipom
    - objekat `AlatZaSimble` je parametrizovan prototipom koji klonira i dodaje u dokument
    - potrebno da sve potklase `Simbol` implementiraju operaciju `kopija()`
  - u muzičkom editoru:
    - alati za kreiranje muzičkih objekata su objekti klase `AlatZaSimbole`
    - svaki objekat `AlatZaSimbole` se inicijalizuje različitim prototipom muzičkog objekta
    - objekat `AlatZaSimbole` proizvodi muzički objekat kloniranjem njegovog prototipa

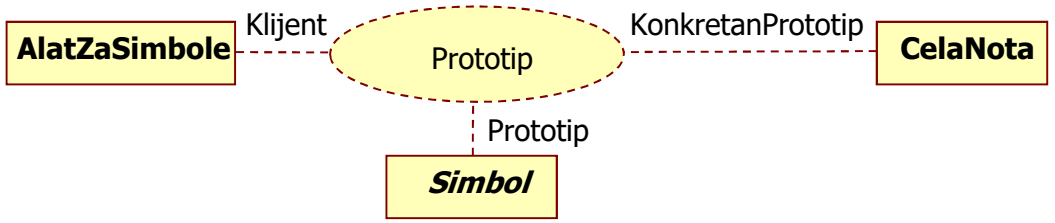
Prototip (4)

- Primenljivost: uzorak treba koristiti
  - kada sistem treba da bude nezavisan od toga kako se njegovi proizvodi stvaraju i predstavljaju
  - kada treba izbeći izgradnju hijerarhije fabrika paralelno sa hijerarhijom proizvoda
  - kada se klase specificiraju u vreme izvršenja, npr. dinamičkim učitavanjem
- Struktura:



Prototip (5)

- Učesnici:
  - `Prototip` (klasa `Simbol`)
    - deklariše interfejs za sopstveno kloniranje
  - `KonkretanProtorip` (klase `CelaNota`, `Polovina`, `NotnaLinija`)
    - implementira operaciju za sopstveno kloniranje
  - `Klijent` (klasa `AlatZaSimbole`)
    - stvara novi objekat zahtevom prototipu da se klonira
- Saradnja:
  - klijent zahteva od prototipa da se klonira
- UML notacija:



Prototip (6)

- Posledice:
  - dobre strane:
    - smanjivanje potrebe izvođenja
    - dodavanje i uklanjanje proizvoda u vreme izvršenja (registracija prototipskog primerka)
    - dinamičko konfigurisanje aplikacije klasama
    - smanjivanje broja imena koje klijent treba da poznaje
  - loša strana:
    - svaka potklasa mora implementirati operaciju `kopija()` što je komplikovano
      - ako klasa već postoji
      - ako objekat klase sadrži (pod)objekte koji ne podržavaju kloniranje
- Povezani uzorci:
  - često zajedno sa uzorcima *Sastav (Sklop*, *Kompozicija)* i *Dekorater (Dopuna)*
  - *Fabrički metod* takođe služi za stvaranje objekta čiji tip ne poznaje klijent
    - stvaranje objekta se delegira potklasi, a ne drugom objektu
  - *Apstraktna Fabrika* se implementira *Fabričkim metodom* ali može i *Prototipom*:
    - fabrika može sadržati skup prototipa na osnovu kojih klonira i vraća proizvode