

# Specifikacija softverskih sistema

Predavanje br. 5 – Dijagram klasa, 2. deo

Veze generalizacije, implementacije interfejsa i zavisnosti

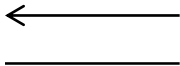
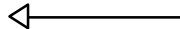
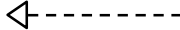
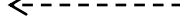
Gordana Milosavljević

Katedra za informatiku, FTN, Novi Sad

2022.

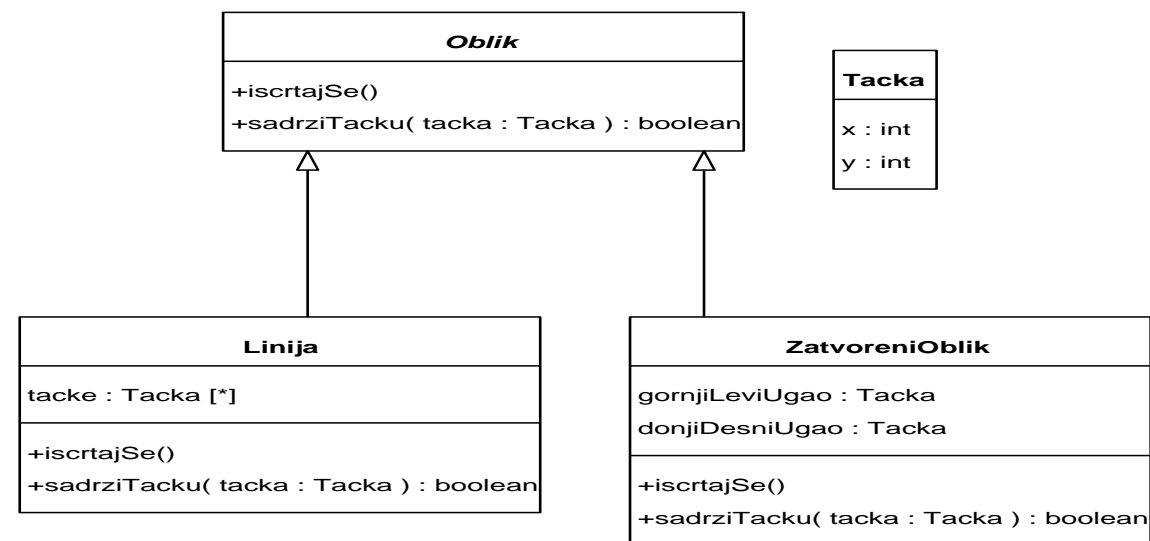
# Veze u dijagramima klasa

- Specificiraju saradnju klasa

Vrsta veze	Ilustracija
Asocijacija	
Generalizacija	
Implementacija interfejsa	
Veza zavisnosti	

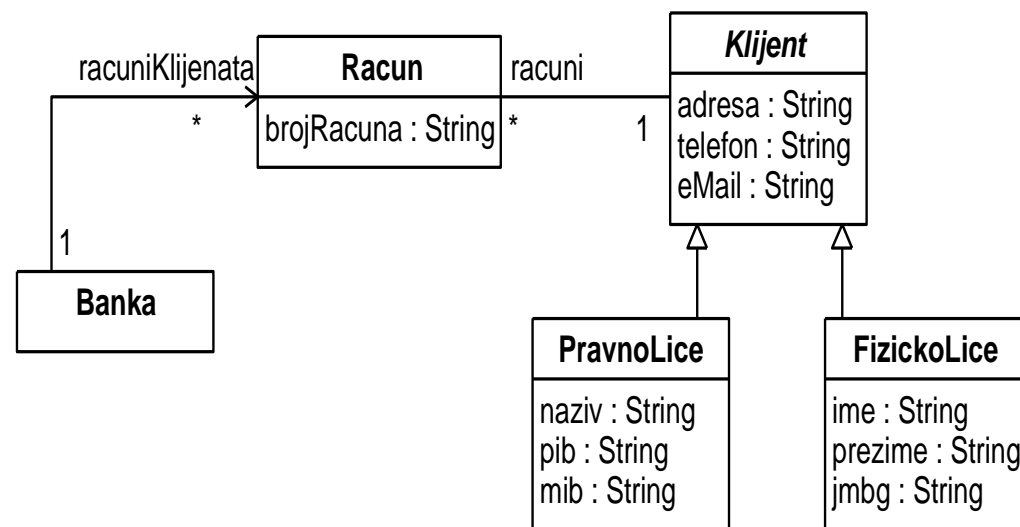
# Veza generalizacije (nasleđivanja)

- Povezuje opštije elemente modela (predak, roditelj) sa specijalizovanim (naslednik, dete)
- Klasa `Oblik` je predak, klase `Linija` i `ZatvoreniOblik` su naslednici
- Pazite na smer!
- Dozvoljeno je višestruko nasleđivanje



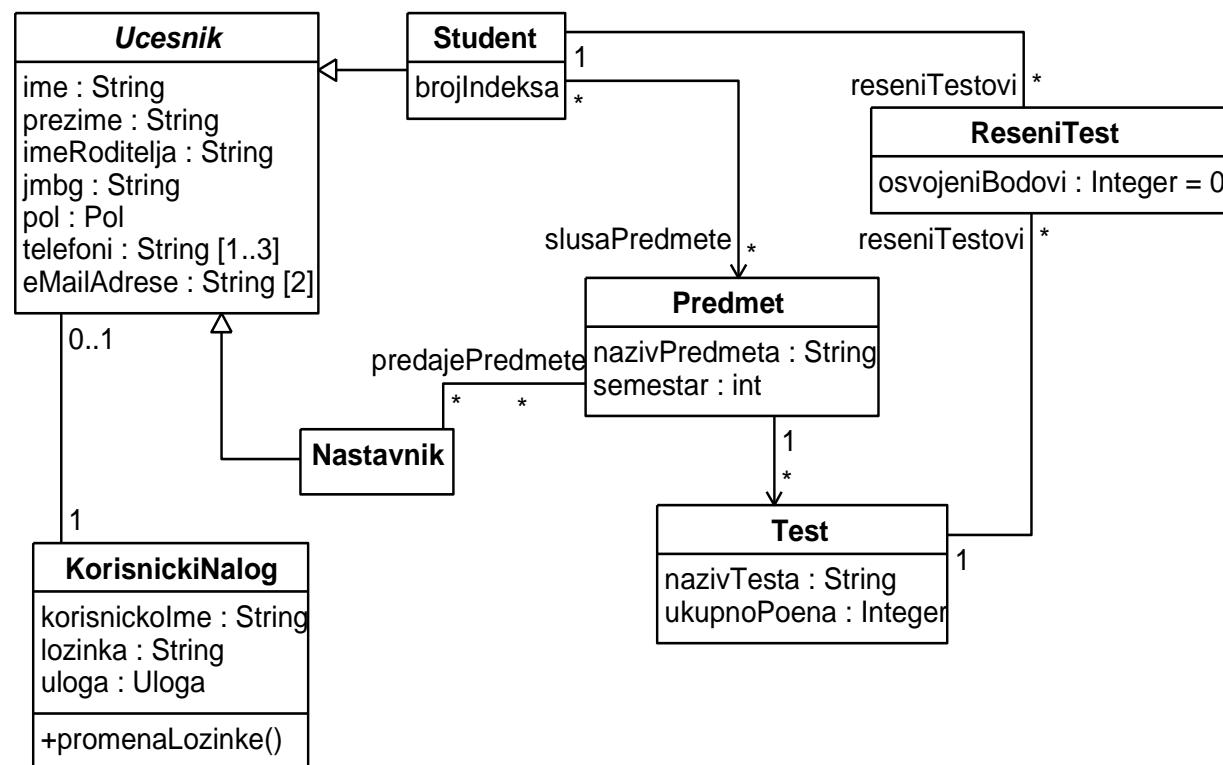
# Primer 1 - nasleđivanje u domenskim modelima

- Klijent banke **može biti** fizičko lice (osoba) ili pravno lice (preduzeće, agencija, zanatska radnja....)



# Primer 2 - nasleđivanje u domenskim modelima

- Student i nastavnik **su** učesnici u nastavnom procesu



# Prepoznavanje generalizacija u specifikaciji zahteva

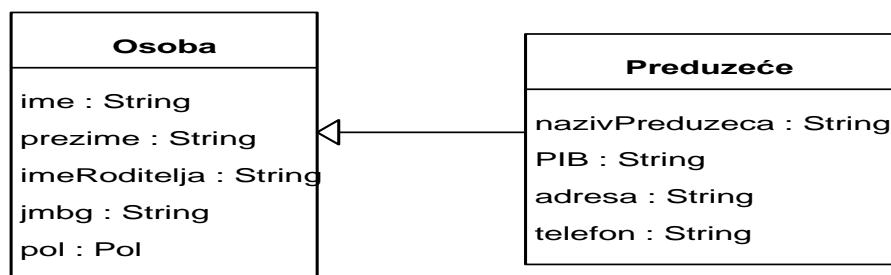
- Nastavnik i student **su** učesnici u nastavnom procesu.
  - Klijenti banke **mogu biti** fizička i pravna lica
  - Linija i zatvoreni oblik **su** geometrijski oblici.
- 
- Klase dele zajedničke osobine ili ponašanje i *iste su vrste!*

# Česte greške pri nasleđivanju

2/2

- Naslednik nije iste vrste kao predak
- *Primer: Podaci koje je potrebno voditi o zaposlenima su: prezime, ime, ime roditelja, jedinstveni matični broj građana (JMBG) i pol. Podaci koje je potrebno voditi o preduzeću su: naziv, poreski broj (PIB), adresa, telefon, prezime, ime i JMBG rukovodioca. Podaci koje je potrebno voditi o klijentima koji su fizička lica su: prezime, ime, ime roditelja, JMBG, pol, adresa, telefon.*

**Pogrešno!**



**Ispravno!**

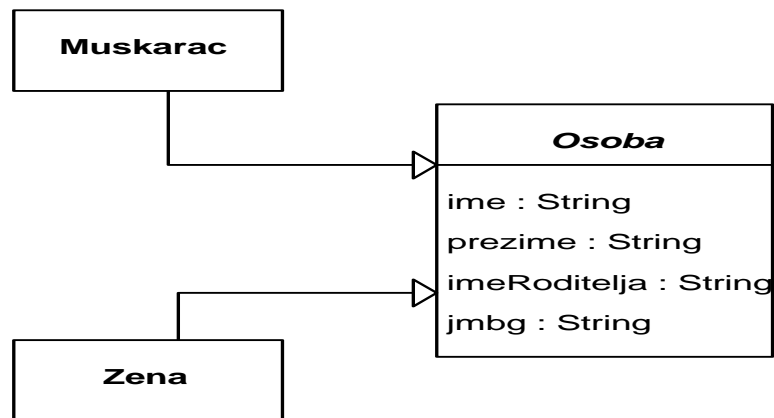


# Česte greške pri nasleđivanju

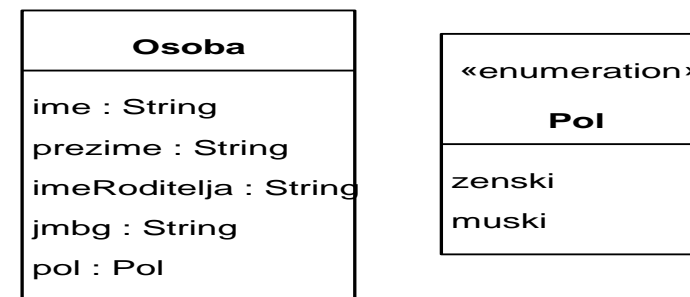
1/2

- Obrnut smer strelice
- Prazne klase-naslednici
  - Ako naslednici nemaju svoje atribute ili metode, verovatno samo nedostaje atribut nabrojanog tipa u pretku

**Pogrešno!**



**Ispravno!**





# Znak da nam treba nasleđivanje

1/2

```
//Metoda za iscrtavanje figure:
```

```
switch(vrstaFigure)
```

```
{
```

```
    case PRAVOUGAONIK:
```

```
        // iscrtavanje pravougaonika
```

```
        break;
```

```
    case DUZ:
```

```
        // iscrtavanje duži
```

```
        break;
```

```
    case TROUGAO:
```

```
        // iscrtavanje trougla
```

```
        break;
```

```
    default :
```

```
        // Statements
```

```
}
```

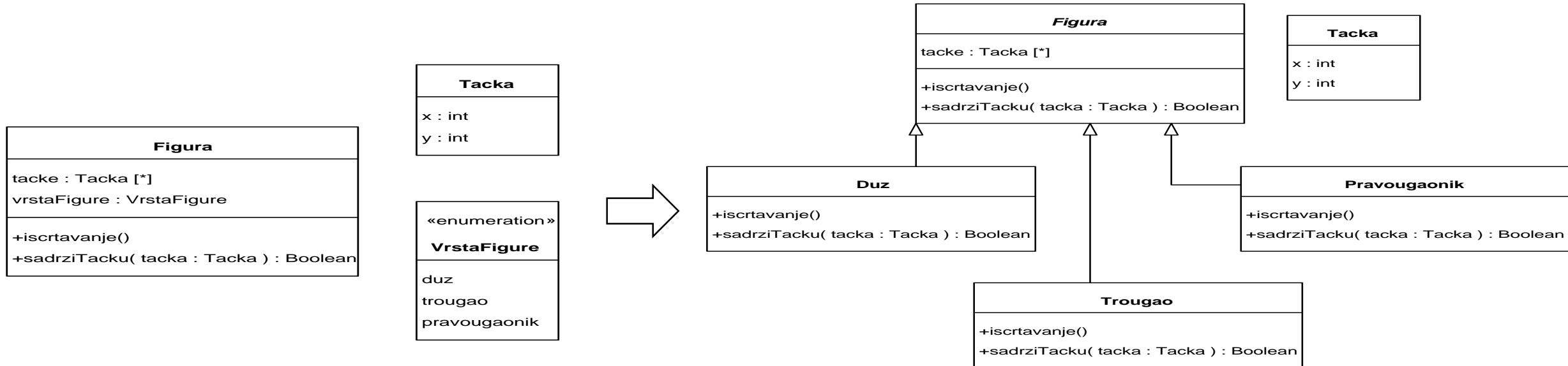
Figura
tacke : Tacka [*] vrstaFigure : VrstaFigure
+iscrtavanje() +sadrziTacku( tacka : Tacka ) : Boolean

Tacka
x : int y : int

«enumeration» VrstaFigure
duz trougao pravougaonik

# Znak da nam treba nasleđivanje

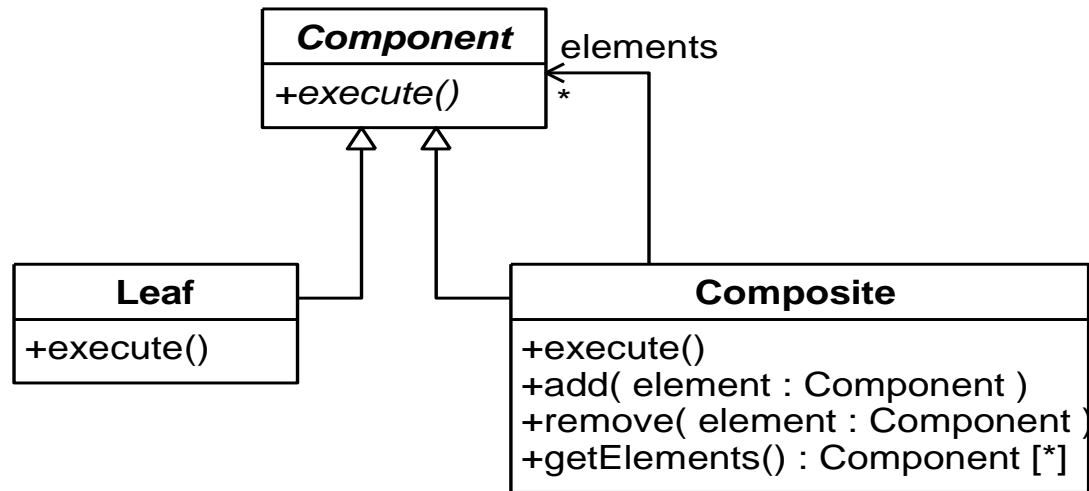
2/2



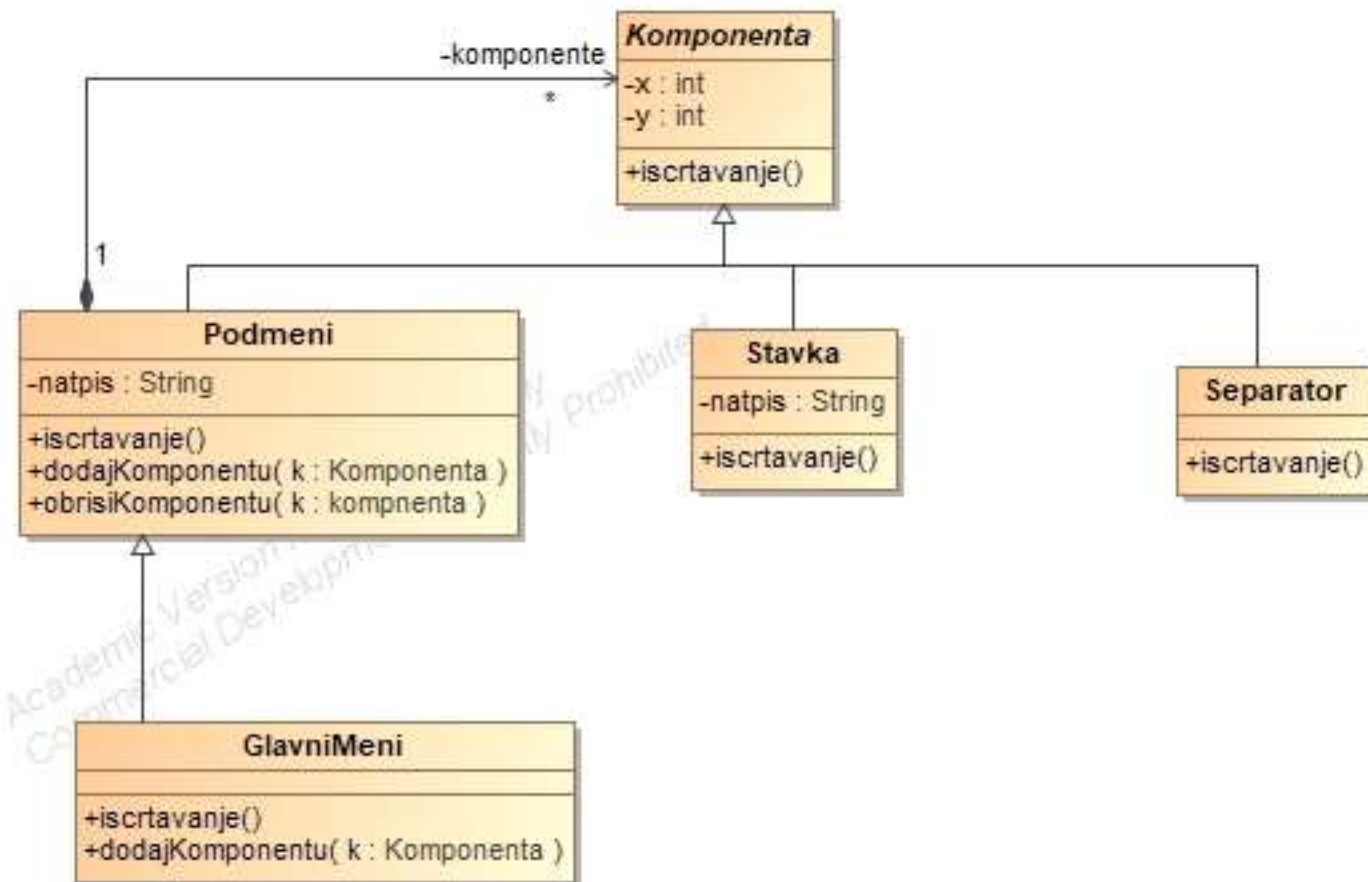
# Zadatak 1

Potrebno je projektovati biblioteku grafičkih komponenti za realizaciju menija aplikacije. Glavni meni aplikacije se može sastojati od podmenija. Podmeni se može sastojati od drugih podmenija, stavki menija i separatora. Glavni meni, podmeni, stavka menija i separatori imaju poziciju na kojoj se iscrtavaju (x i y koordinata). Podmeni i stavka menija imaju i natpis. Sve navedene komponente se mogu iscrtati.

# Kompozitni šablon (Composite pattern)

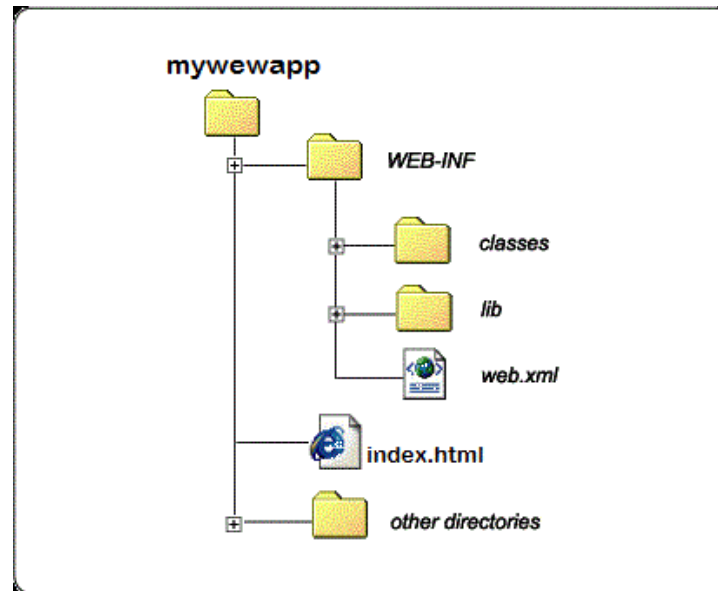


# Rešenje zadatka 1

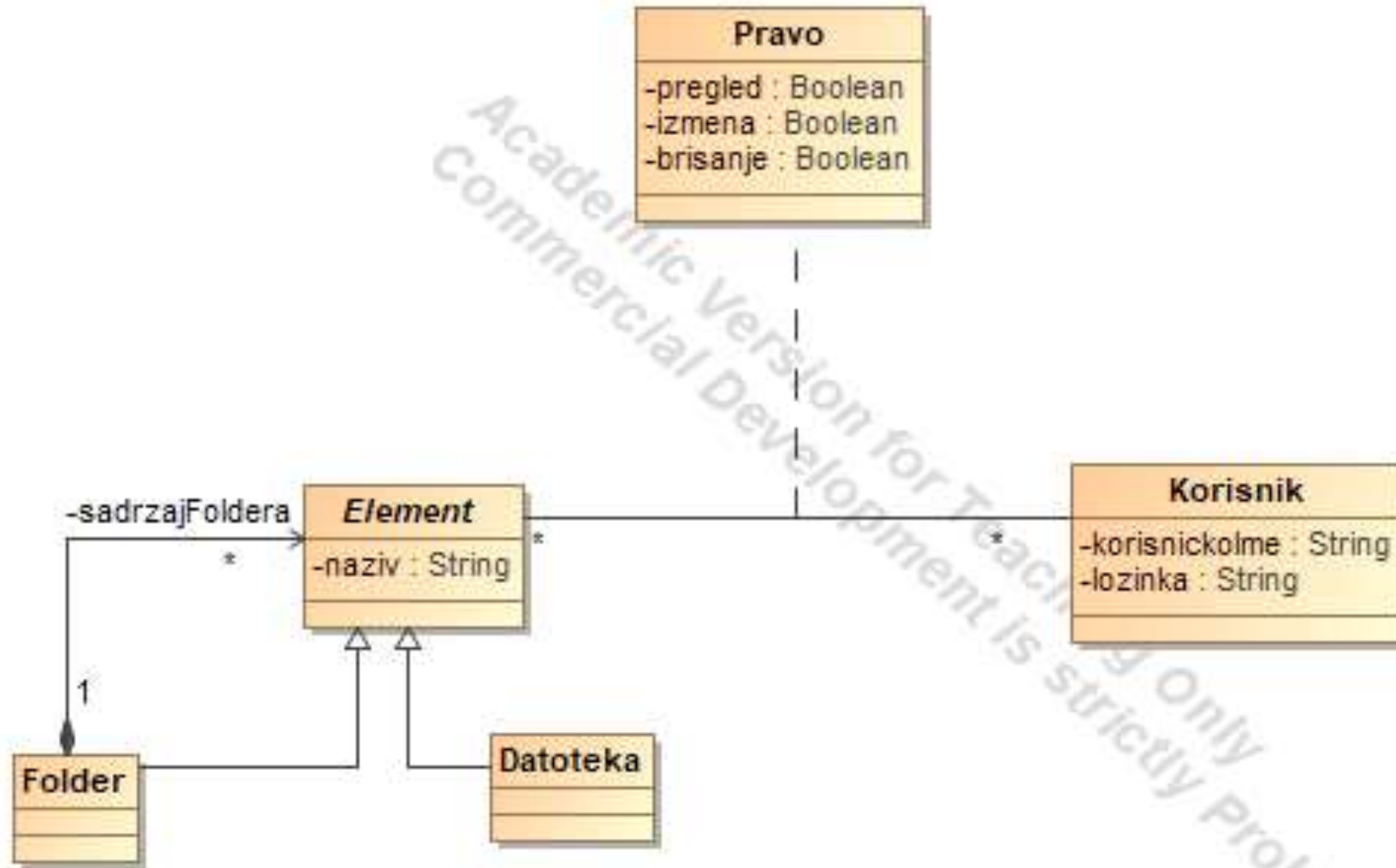


# Zadatak 2

- Projektovati strukturu podataka za podršku rada sistema datoteka - *file system*. U okviru sistema datoteka se mogu naći folderi i datoteke. U folderu se mogu naći datoteke i potfolderi. Svaki element sistema datoteka ima naziv. U folder se mogu dodavati novi elementi i brisati postojeći. Ime datoteke ili foldera se može menjati, pod uslovom da korisnik ima pravo izmene.
- Nad svakim elementom sistema datoteka, za svakog korisnika, može se definisati pravo pregleda, izmene i brisanja. Korisnik od podataka ima korisničko ime i lozinku.



# Rešenje zadatka 2

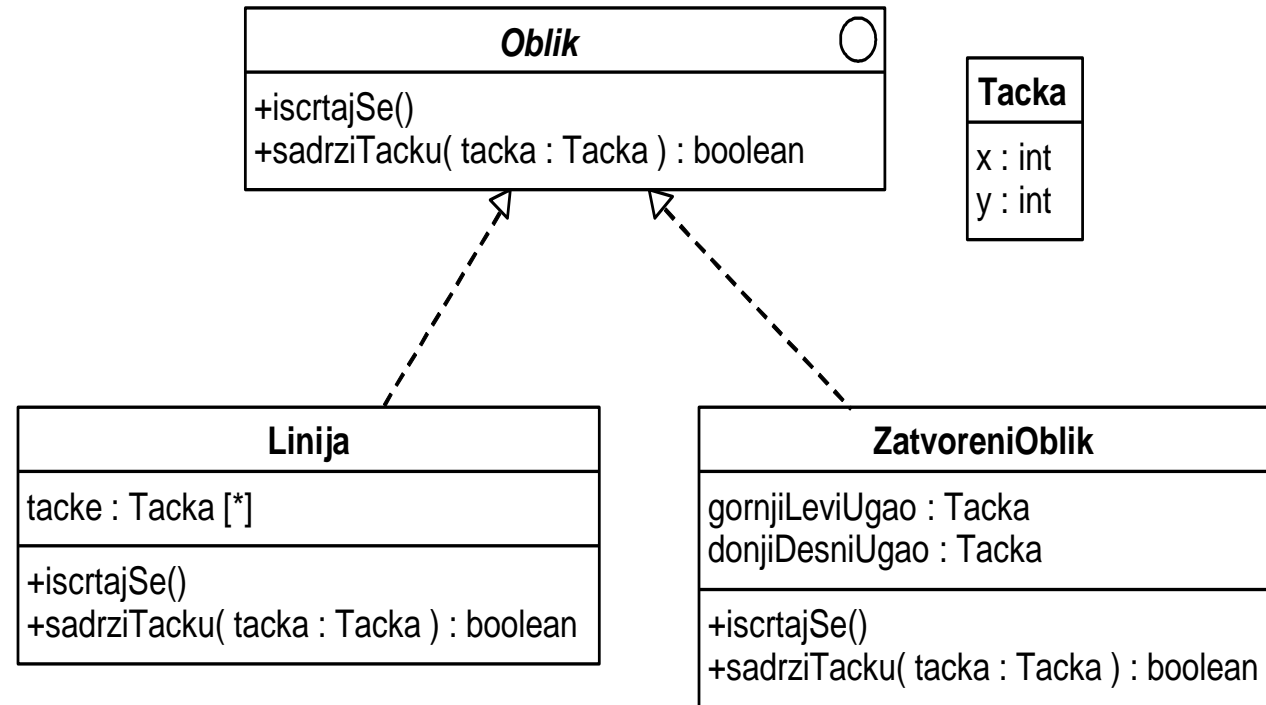


# Implementacija interfejsa

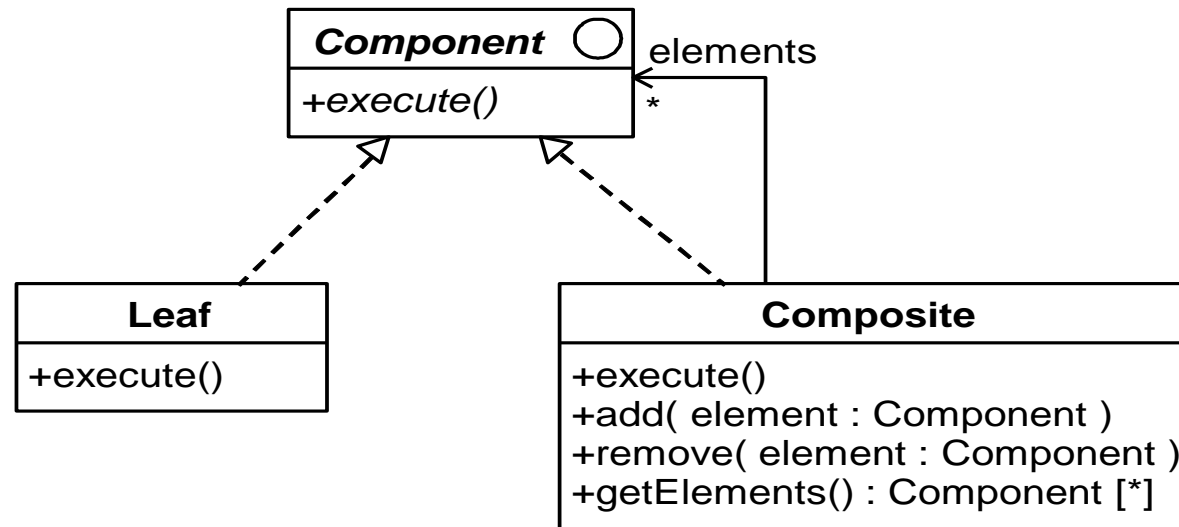
- Specificiraju skup osobina koje svaka klasa, koja implementira dati interfejs, mora da podrži na sebi svojstven način.
- Implementacija interfejsa je manje striktna od veze generalizacije: klase koje implementiraju neki interfejs ne moraju imati ništa zajedničko, osim ponašanja



# Primer



# Kompozitni šablon modelovan korišćenjem interfejsa



# Veze zavisnosti

- Veza zavisnosti označava da jedna klasa zavisi od druge
- Ako se implementacija nezavisne klase promeni, verovatno će biti potrebna i promena implementacije zavisne klase koja koristi njene usluge:
  - jedna klasa kreira instance druge klase,
  - instanca jedne klase se prenosi kao argument metode druge klase, pri čemu se date instance ne čuvaju u obeležjima zavisne klase (što bi rezultovalo vezom asocijacije)
  - ....



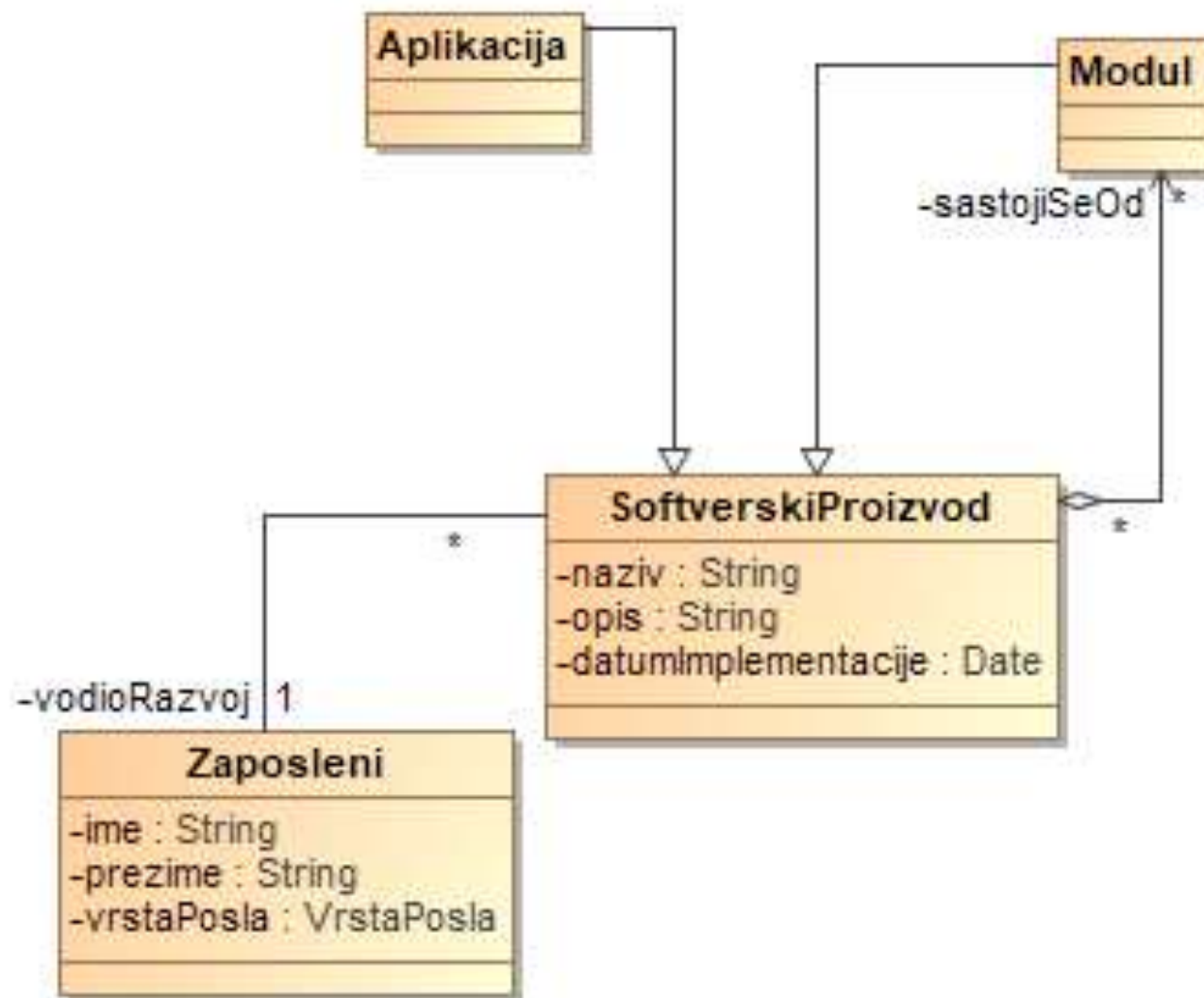
# Korišćenje

- Veze zavisnosti treba stavljati na dijagram samo ako želimo da posebno naglasimo zavisnost dve klase između kojih nema drugih vrsta veza.
- Ako se previše koriste, mogu učiniti dijagram nečitkim, pošto odvlače pažnju od veza koje imaju direktan uticaj na programski kod, odnosno na šemu baze podataka.

# Zadatak 3

- Modelovati dijagram klasa za aplikaciju za podršku rada softverske firme koja treba da omogući ažuriranje i pretragu podataka o softverskim proizvodima koje data firma implementira kao i podataka o njenim zaposlenima.
- Softverski proizvodi **mogu biti** programski moduli ili gotove aplikacije. Gotove aplikacije se sastoje od programskih modula. Programski moduli se mogu sastojati od drugih programskih modula. Svaki softverski proizvod ima svoj naziv, opis, datum kada je implementiran i zaposlenog koji je vodio njegov razvoj. Zaposleni ima attribute: ime, prezime, broj telefona i vrstu posla koju obavlja. Moguće vrste posla su: projektant, programer i tester.

# Rešenje zadatka 3



# Literatura

1. James Rumbaugh, Ivar Jacobson, Grady Booch, The Unified Modeling Language Reference Manual, Second Edition, Addison-Wesley, 2004
2. Scott W. Ambler, The Object Primer: Agile Model-Driven Development with UML 2.0, Cambridge University Press, 2004
3. M. Fowler, UML Distilled - A Brief Guide to the Standard Object Modeling Language, Third Edition, Addison Wesley, Boston, 2004.