

# Dijagrami klasa

# Izrada potpunog dijagrama klasa

- Iterativnim postupkom potrebno je izvršiti identifikaciju klasa i pri tom dodati operacije.
- Dijagram klasa sadrži skup klasa i saradnji i njihove relacije.
- Dijagram klasa specificira logičke i statičke aspekte modela.

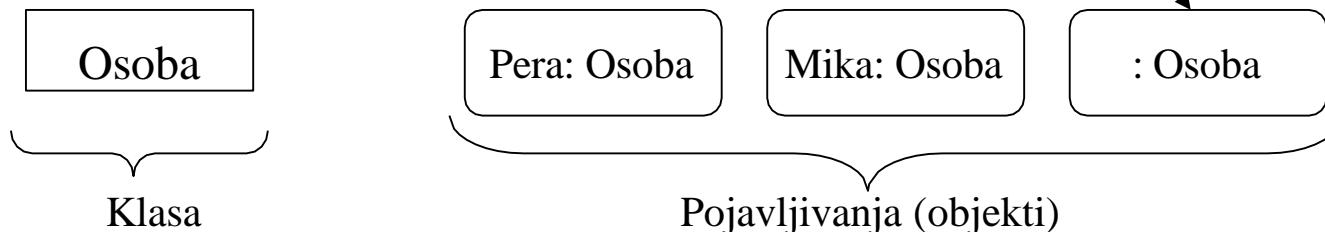
# Izrada potpunog dijagrama klase

- **Klasa** – skup objekata sa zajedničkim atributima, operacijama, metodama, vezama i semantikom
- **Operacija** – definiše ponašanje klase
- **Metoda** - implementirana operacija specificirana u okviru operacionog algoritma ili procedure
- **Tip** - opisuje skup sličnih objekata sa atributima i operacijama, bez uključivanja metoda
- **Interfejs** - skup spoljnih vidljivih operacija

# Definisanje klase

- Klasa predstavlja skup objekata koji imaju zajedničku strukturu i ponašanje
- Granica između objekta i klase je izuzetno tanka:
  - ❖ **Klasa** – apstrakciju objekata – sposobnost prepoznavanja zajedničkih osobina objekata neke klase i formiranje apstraktnog skupa tih svojstava (osmišljavanje klase)
  - ❖ **Objekat** – primerak klase koji ima sva njena ponašanja i specifične vrednosti za svako svojstvo

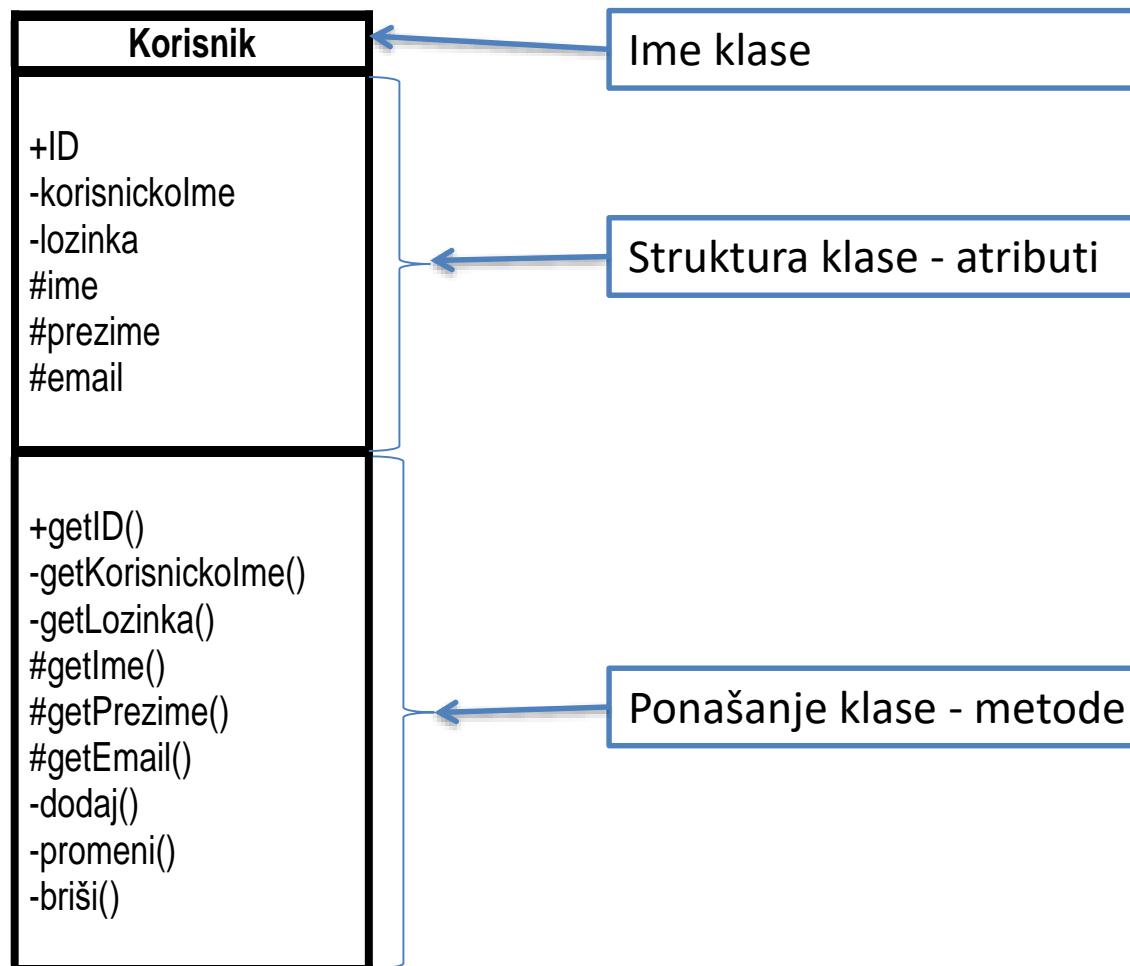
objekat - anonimno pojavljivanje klase



# Notacija klase

Notacija klase	Primer						
<table border="1"><thead><tr><th>Naziv klase</th></tr></thead><tbody><tr><td>-atribut</td></tr><tr><td>-atribut : tip podatka</td></tr><tr><td>-atribut : tip_podatka = inicij. vred</td></tr><tr><td>+operacija ()()</td></tr><tr><td>+operacija (argum.lista):tip rezul()</td></tr></tbody></table>	Naziv klase	-atribut	-atribut : tip podatka	-atribut : tip_podatka = inicij. vred	+operacija ()()	+operacija (argum.lista):tip rezul()	<p><b>GeometrijskaFigura</b></p> <p>← naziv klase</p> <p>karakterističnaTačka</p> <p>← atributi</p> <p>pomeri() promeniVeličinu() prikaži()</p> <p>← operacije</p> <p>Responsibilities geometrijska svojstva figura u ravni</p> <p>← odgovornosti</p>
Naziv klase							
-atribut							
-atribut : tip podatka							
-atribut : tip_podatka = inicij. vred							
+operacija ()()							
+operacija (argum.lista):tip rezul()							

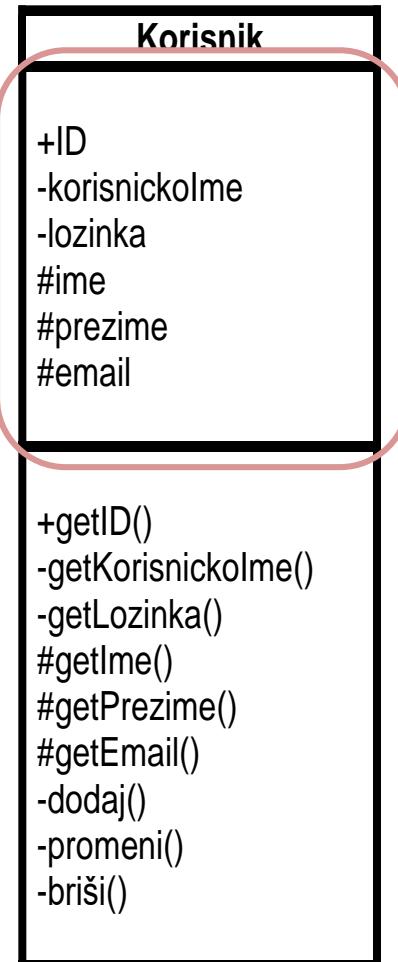
# Elementi klase



# Definisanje klasa

- Naziv klase piše se velikim slovom
- Naziv atributa je kratka imenica ili fraza
- Naziv operacije je glagol ili glagolska fraza
- Vidljivost je pravo pristupa, čiji se znak piše ispred atributa/operacije
  - ❖ javni domen: + pristup nije ograničen
  - ❖ privatni domen: - pristup je ograničen na članove iste klase i podklase
  - ❖ zaštićeni domen: # prošireno pravo pristupa i na klase izvedene iz date klase
  - ❖ implementacioni domen: ? opseg tog trenutka nije definisan, definisće se u okviru implementacije

# Atribut



- **Atribut** je svojstvo klase. Opisuje opseg vrednosti tog svojstva klase.
- Klasa može imati nula ili više atributa.
- Naziv atributa se sastoji od jedne reči napisane **malim slovima**.
- Ako se naziv atributa sastoji od više reči te reči su spojene i svaka reč osim prve počinje sa velikim slovom.

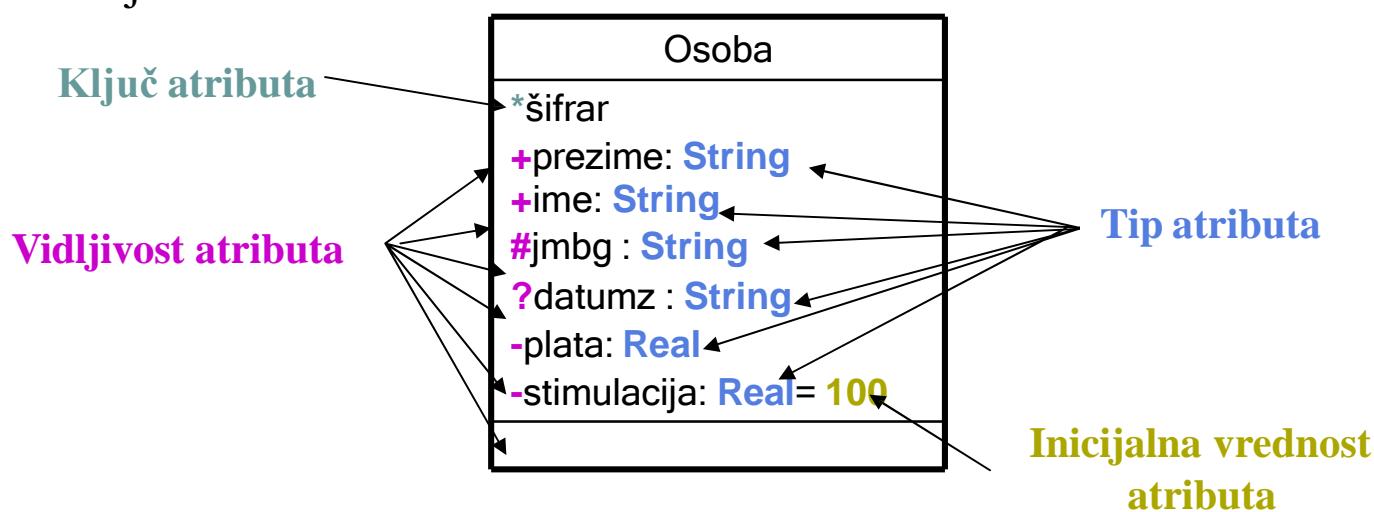
# Definisanje atributa

- ❑ Na najapstraktnijem nivou - kada se modeliraju strukturne osobine klase pišu se samo imena atributa - dovoljna informacija za prosečnog korisnika
- ❑ Na primer, atributi klase Osoba su:
  - ❖ prezime
  - ❖ ime
  - ❖ jmbg
  - ❖ datum
  - ❖ plata
  - ❖ stimulacija

Osoba
prezim
e ime
jmbg
datumz
plata
stimulacija

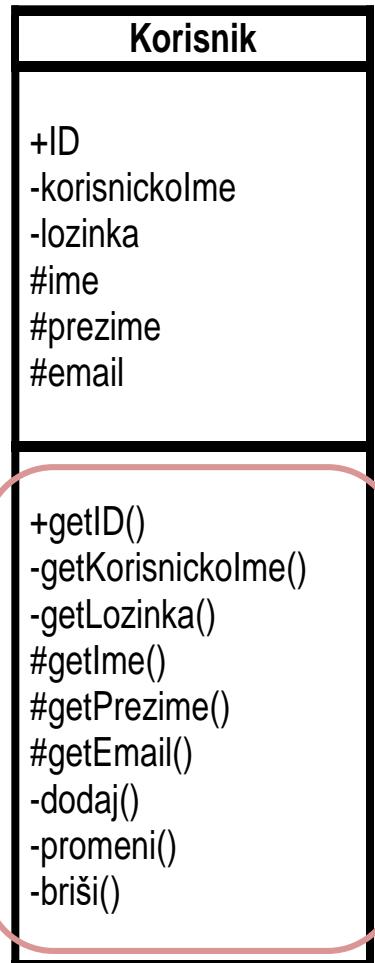
# Osobine atributa

- ❑ Ključ atributa – jedinstvena identifikacija klase - označava se znakom \* ispred atributa
- ❑ Tip atributa: ceo broj (integer), realni broj (real), karakter (char), ...
- ❑ Vidljivost atributa: javni, privatni, zaštićeni ili implementacioni domen
- ❑ Inicijalna vrednost atributa



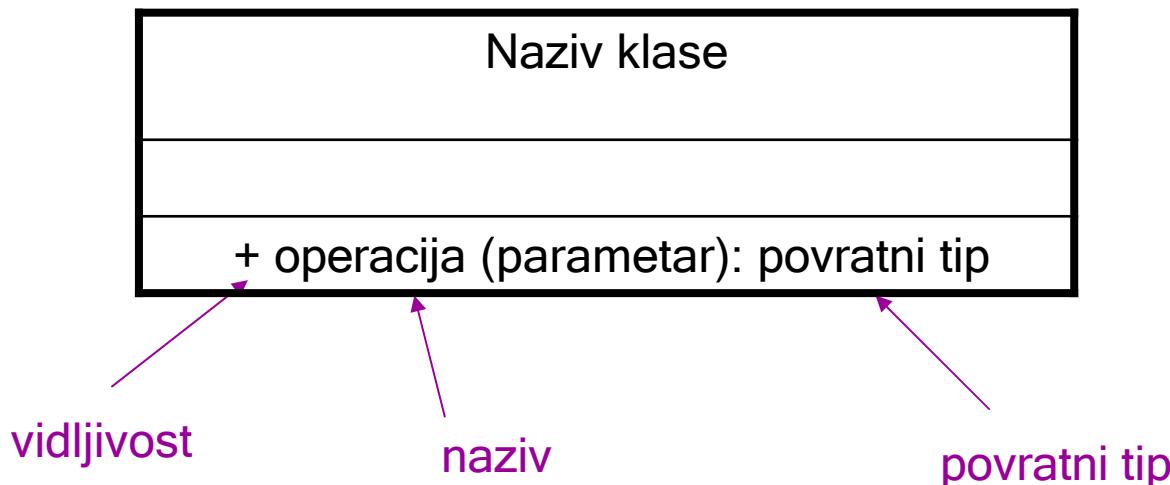
# Metoda

- **Metoda** je nešto što klasa može raditi ili što vi (ili druga klasa) možete raditi toj klasi.
- Za naziv metode važe pravila kao i za naziv atributa.
- I za metode, kao što za attribute mogu da se odrede dodatne informacije.
- U zagradama koje slede iza naziva metode navode se parametri sa kojima metoda radi kao i tip parametara.
- Jedna vrsta metode, funkcija, vraća vrednost nakon što završi sa radom.



# Definisanje operacija

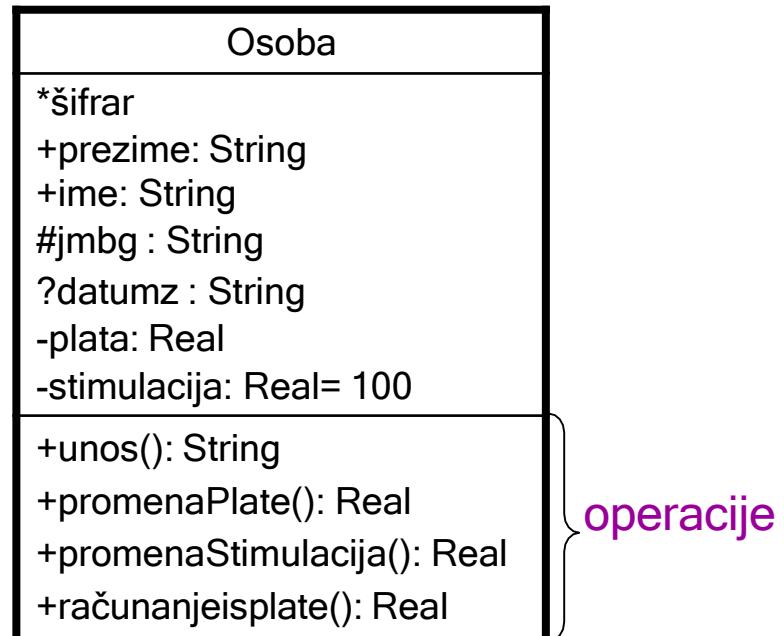
- Operacija se definiše:
  - ❖ nazivom operacije
  - ❖ vidljivošću i
  - ❖ povratnim tipom – tip rezultata koji operacija vraća



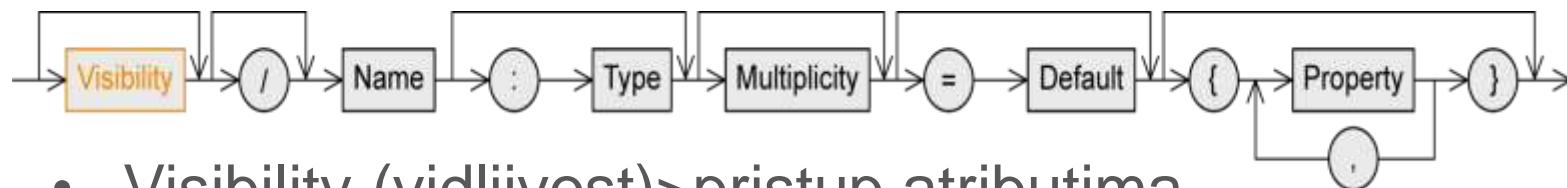
# Definisanje operacija

Za klasu **Osoba**, definišu se sledeće operacije:

- ❖ unos (unos podataka o radniku)
- ❖ promenaPlate (evidentiranje isplate radniku)
- ❖ promenaStimulacije (promena vrednosti atributa stimulacija)
- ❖ računanjeIsplate (računa se isplata na osovu plate i stimulacije)



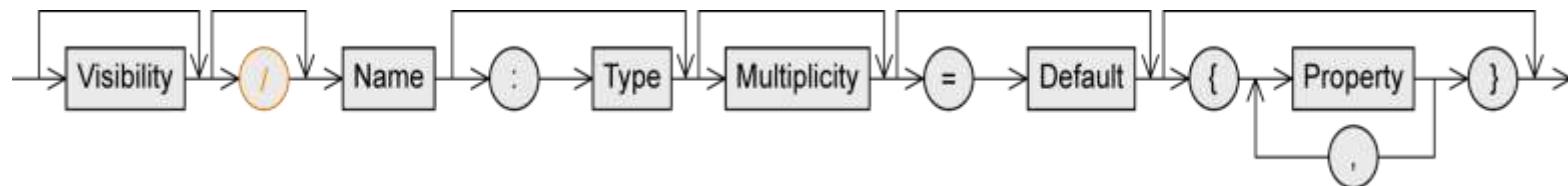
# Sintaksa za atribut



- Visibility (vidljivost)>pristup atributima
  - +.....public: svi
  - -.....private: samo objekat
  - #.....protected: samo klasa i subklase
  - ~.....package: klase koje su u istom paketu

Person
<pre>+ firstName: String + lastName: String - dob: Date # address: String[1..*] {unique, ordered} - ssNo: String {readOnly} - age: int - password: String = "pw123" - personsNumber: int</pre>

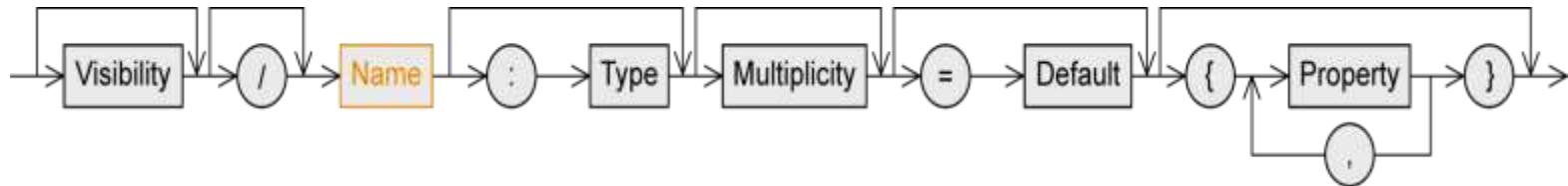
# Sintaksa za atribut



- Derived attribute>potiče od drugih atributa npr age atribut je izračunato na osnovu date of birth

Person
firstName: String lastName: String dob: Date address: String[1..*] {unique, ordered} ssNo: String {readOnly} <b>/age: int</b> password: String = "pw123" <u>personsNumber: int</u>

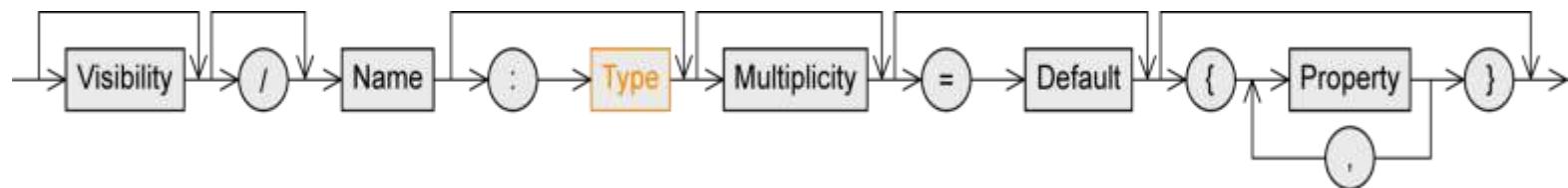
# Sintaksa za atribut



- Name>naziv atributa

Person
firstName: String lastName: String dob: Date address: String[1..*] {unique, ordered} ssNo: String {readOnly} age: int password: String = "pw123" <u>personsNumber: int</u>

# Sintaksa za atribut



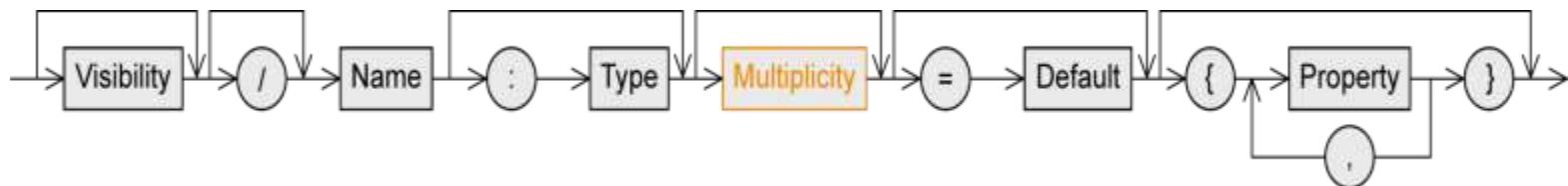
Person
firstName: String lastName: String dob: Date address: String[1..*] {unique, ordered} ssNo: String {readOnly} /age: int password: String = "pw123" <u>personsNumber: int</u>

«primitive»
Float round(): void

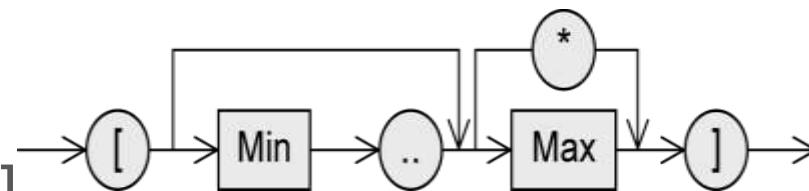
«datatype»
Date day month year

«enumeration»
AcademicDegree bachelor master phd

# Sintaksa za atribut

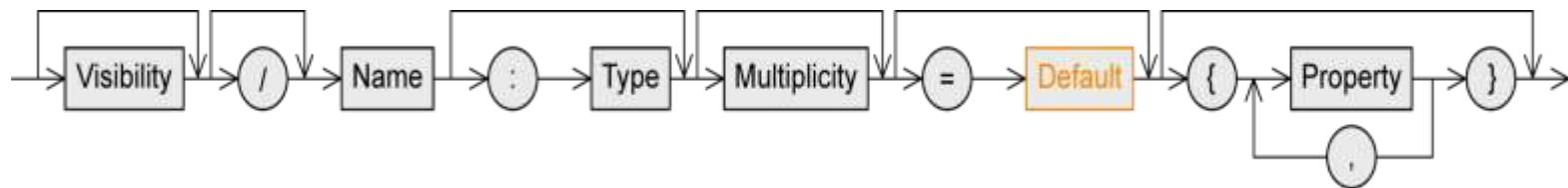


- Multiplicity > broj vrednosti koje atribut može da sadrži
- Default vrednost: 1
- Notacija [min...max]



Person
firstName: String lastName: String dob: Date address: String[1..*] {unique, ordered} ssNo: String {readOnly} /age: int password: String = "pw123" <u>personsNumber: int</u>

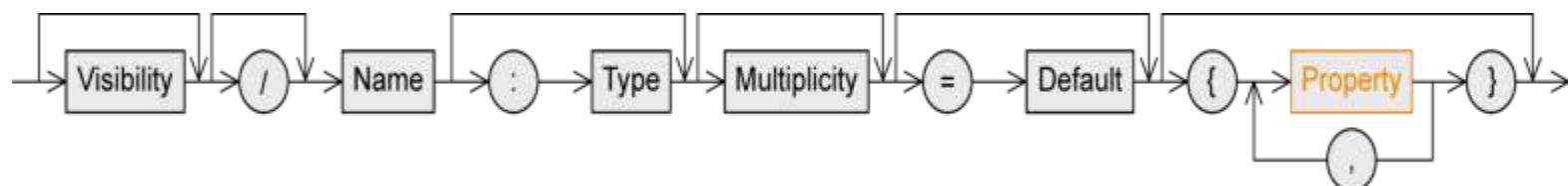
# Sintaksa za atribut



- Default vrednost se koristi ako vrednost atributa nije eksplisitno podesio korisnik

Person
firstName: String lastName: String dob: Date address: String[1..*] {unique, ordered} ssNo: String {readOnly} /age: int password: String = "pw123" <u>personsNumber: int</u>

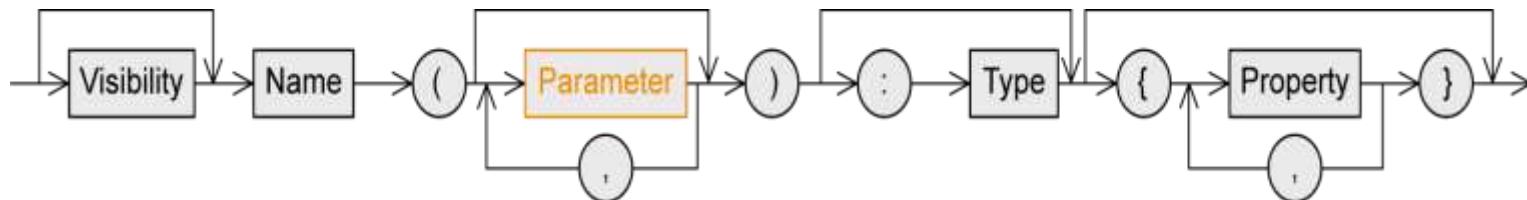
# Sintaksa za atribut



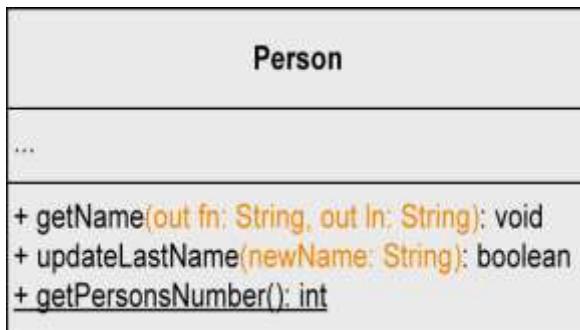
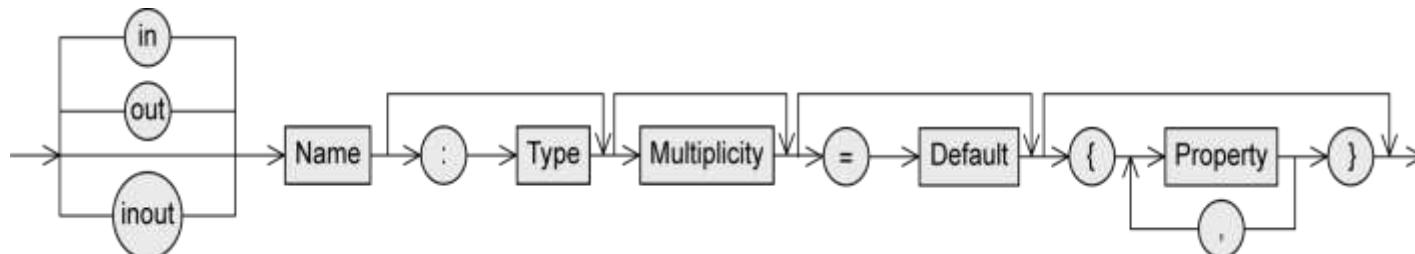
- Pre-definisana podešavanja
  - {readOnly} ... vrednost ne može da se menja
  - {unique} ... nisu dozvoljeni duplikati
  - {non-unique} ... dozvoljeni duplikati
  - {ordered} ... fiksno određen raspored vrednosti
  - {unordered} ... nije fiksno određen raspored vrednosti

Person
firstName: String lastName: String dob: Date address: String[1..*] {unique, ordered} ssNo: String {readOnly} /age: int password: String = "pw123" <u>personsNumber: int</u>

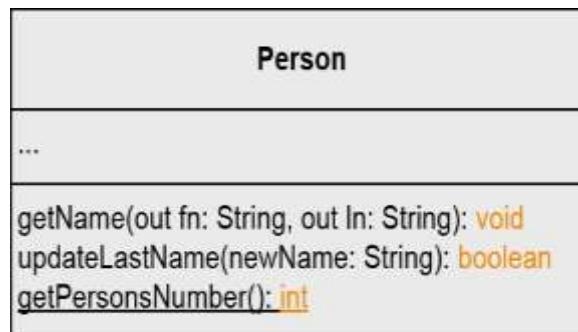
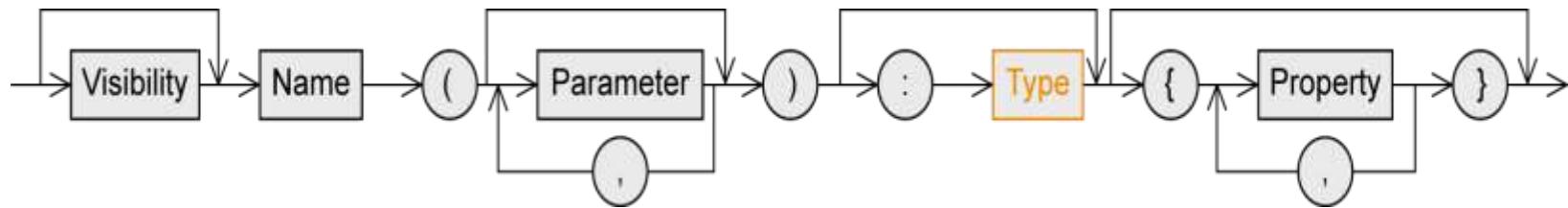
# Sintaksa za operacije



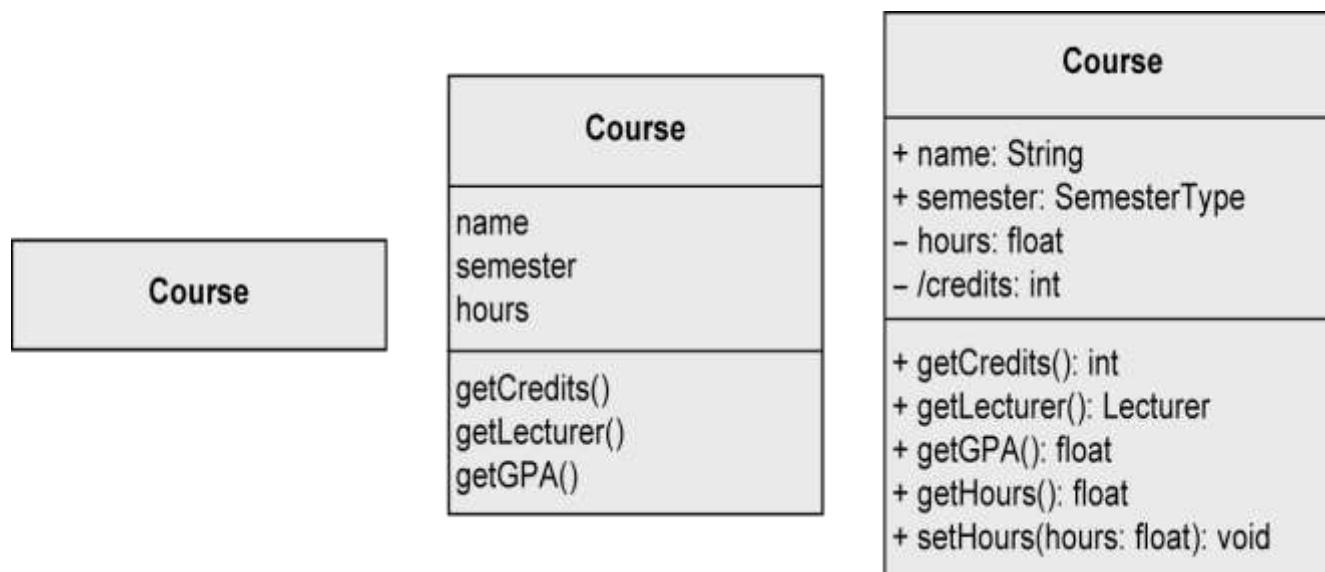
- Input parametar: in....kada se pokrene operacija vrednost se unosi kroz input
- Output parametar: out....nakon izvršavanja operacije, parametar je usvojio novu vrednost
- Input/Output parametar: inout....kombinacija



# Sintaksa za operacije



# Dijagam klase



# Identifikovanje klase

Primer razgovora sa klijentima (naručiocem posla):

- U vašem razgovoru sa klijentima, obratite pažnju na imenice koje koriste da bi opisali entitete u svom poslu.
- Te imenice **će biti klase u modelu**.
- Takođe obratite pažnju na **glagole koje čujete jer će** oni stvarati **metode unutar klase**.
- **Atributi se javljaju kao** imenice koje su u vezi sa imenicama koje određuju klasu.

# Određivanje elemenata klase...Primer:

## Imenice :

- lopta, koš, tim, igrači, odbrana, napadači, centar, trojka, slobodno bacanje, faul, linija slobodnog bacanja, teren.

## Glagoli:

- pucanje, vođenje napada, driblanje, dodavanje, fauliranje.

Imamo i neke dodatne informacije o nekim imenicama

- relativna visina igrača na svakoj poziciji, dimenzije terena, koliko traje napad, trajanje utakmice.

Naravno i mi sami bi se mogli setiti nekih atributa:

- Npr. lopta ima svoj volumen - težinu, prečnik (dimenziju).

Lopta
promer
volumen
driblanje()
pucanje()
propuštanje()

Igrač
ime
visina
težina
driblanjeLoptom()
vođenjeNapada()
propuštanjeLopte()
pucanjeLoptom()
fauliranjeProtivnika()

Odbojka

Tim
Koš

Napadač

Centar

Pucanje

Faul

VremeZaNapad

LinijaTriBoda

LinijaSlobodnogBacanja

VremeZaIgru

SlobodnoBacanje

Trajanje

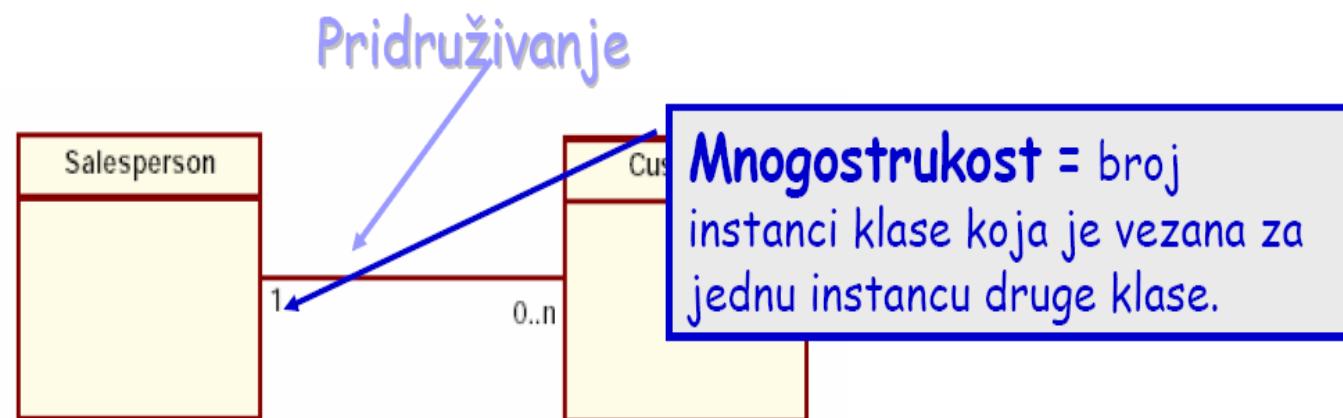
Teren

# Definisanje veza između klasa

- Veze – obezbeđuju komunikaciju između klasa
- UML standard definiše četiri tipa veza:
  - ❖ **asocijacija** (association) – definiše strukturalni odnos između instanci klase;
  - ❖ **zavisnost** (dependency) – koristi odnose;
  - ❖ **generalizacija** (generalization) – povezuje generalizovane klase kao podklase/superklase ili odnos dete/roditelj
  - ❖ **realizacija** - koristi se kad su u pitanju interfejsi ili kada je reč o kolaboraciji

# Asocijacija - Pridruživanje

- Veza između dve klase koja opisuje njihov statički odnos.
- Ona kaže da jedan objekt ima za atribut primerak drugog ili su ti objekti povezani u smislu posedovanja (ali ne i odnosa *sastojati se od*).
- Npr. prodavac je vezan pridruživanjem sa svojim kupcima, ali prodavac se ne sastoji od kupaca.



A

B

Pridruživanje naziva se između A i B  
treba čitati od A prema B.

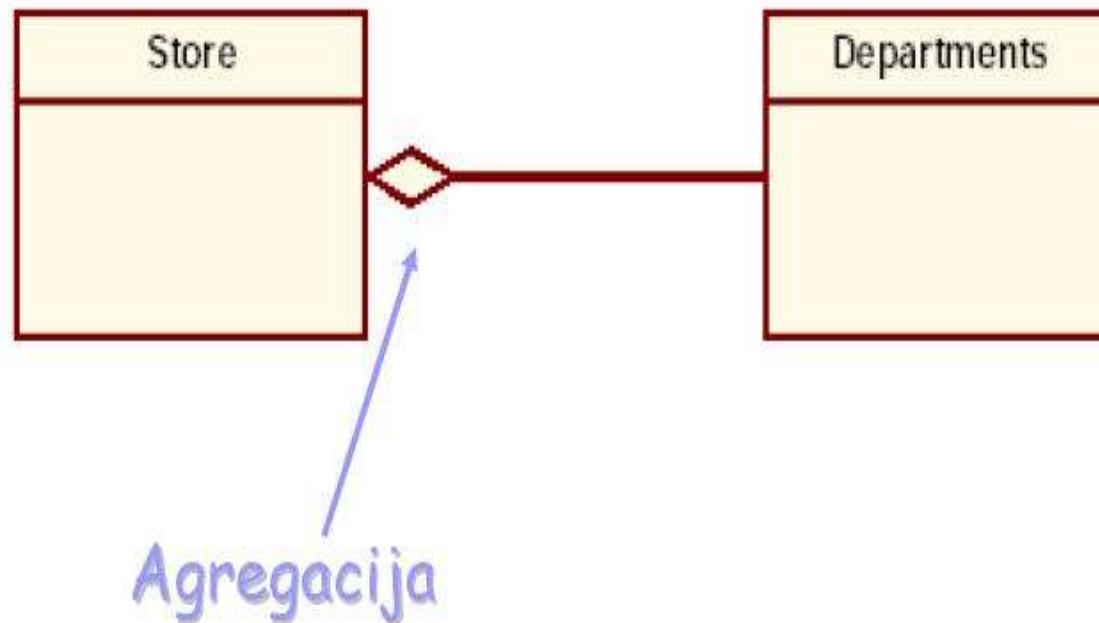
# Asocijacija, Mnogostrukost

- **Mnogostrukost asocijacije** određuje broj primeraka jedne klase u odnosu na drugu klasu

Nije određeno	
Višestruko	2, 4..6
Određeno	4..6
Nula ili jedan	0..1
Jedan ili više	1..*
Nula ili više	0..*
Tačno jedan	1

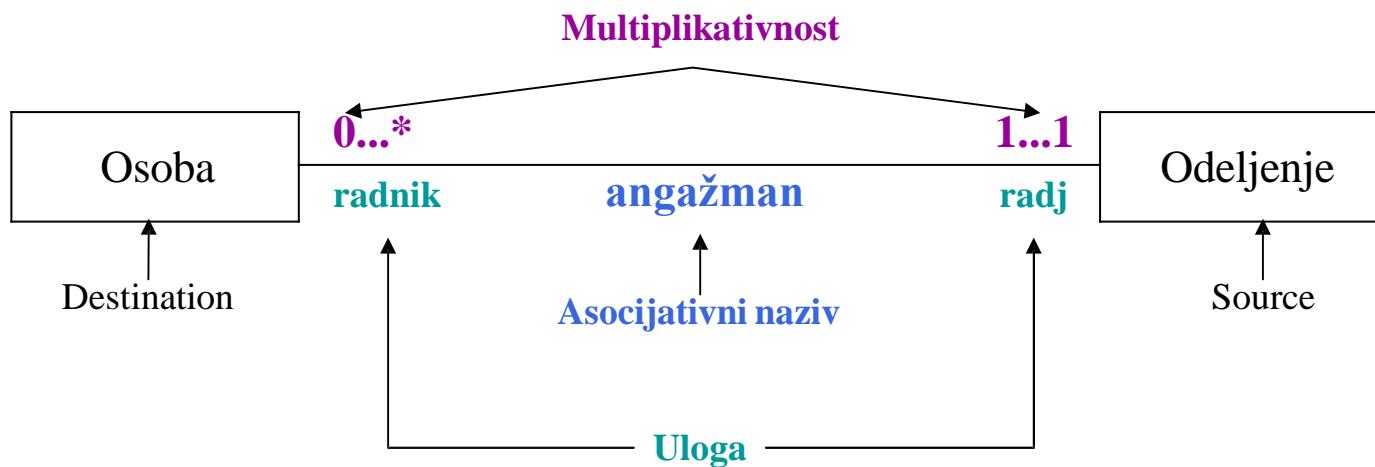
# Agregacija, gomilanje

- Jači oblik pridruživanja koje označava povezanost celine sa nekim njenim delom.
- Npr. putovanje vozom se sastoji od niza putovanja između stanica.
- Prodavnica agregira više odeljenja, tj. prodavnica se sastoji od odeljenja.



# Asocijacija

- Bidirekciona veza između klasa – prikazuje se kao linija koja spaja klase



# Naziv asocijacija

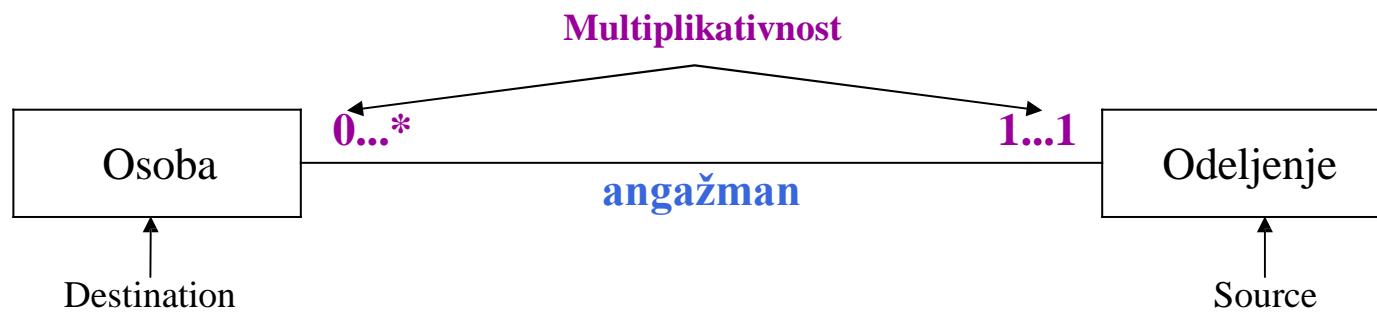
- Asocijacija između klase **Osoba** i klase **Odeljenje** je imenovana rečju **angażman** što znači: Osoba je angažovana u Odeljenju



- Klasa Odeljenje (Source) je u asocijaciji pod nazivom **angażman** sa klasom **Osoba** (Destination)

# Multiplikativnost asocijacija

- Multiplikativnost – kardinalnost- interval koji definiše najmanji i najveći broj pojavljivanja

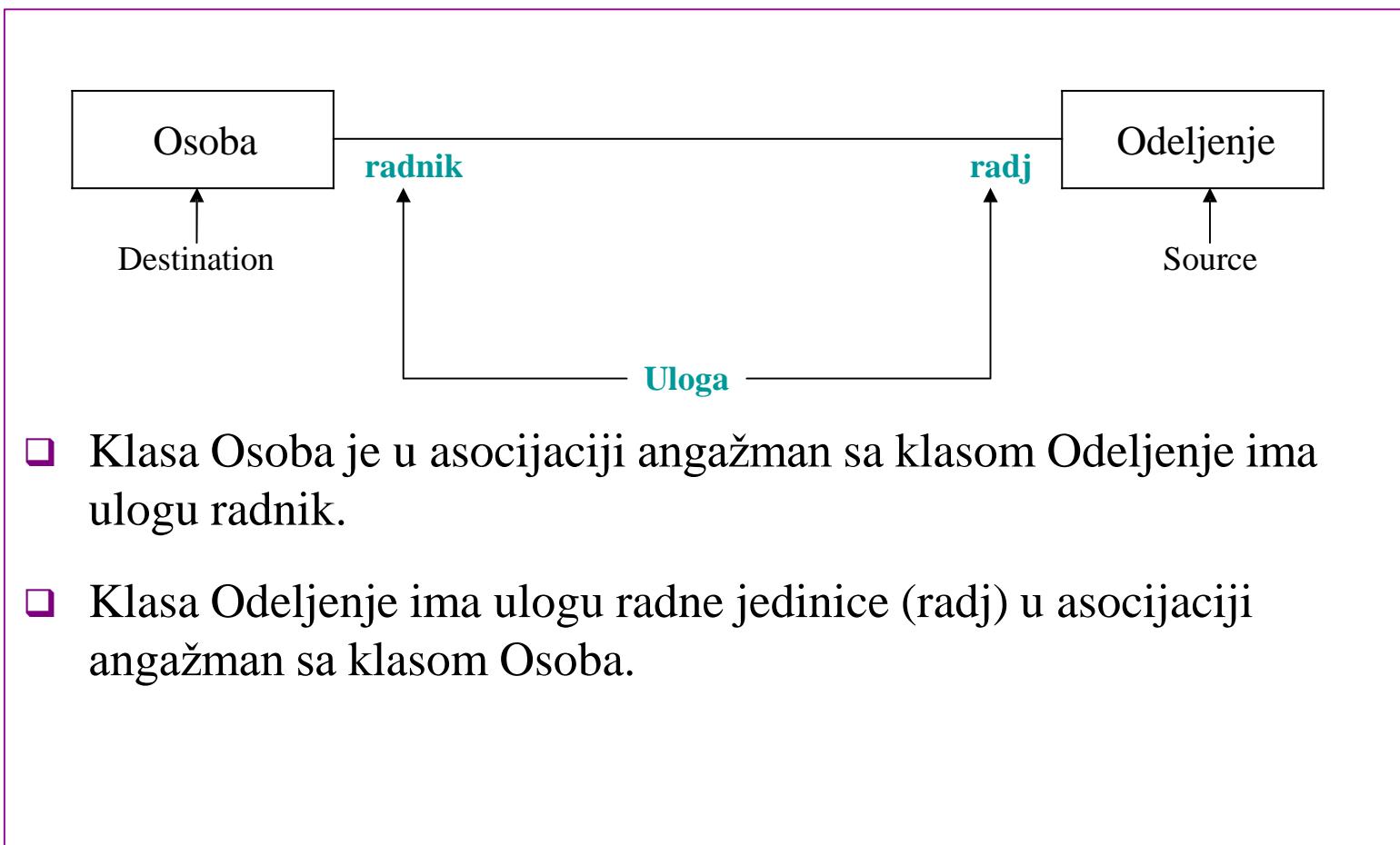


- Jedna pojava klase Odeljenje može da angažuje nijednu ili više pojava klase Osoba ( $0..*$ ), a jedna pojava klase Osoba može da bude angažovana u jednoj i samo jednoj pojavi klase Odeljenje ( $1..1$ )

# Indikatori multiplikativnosti asocijacija

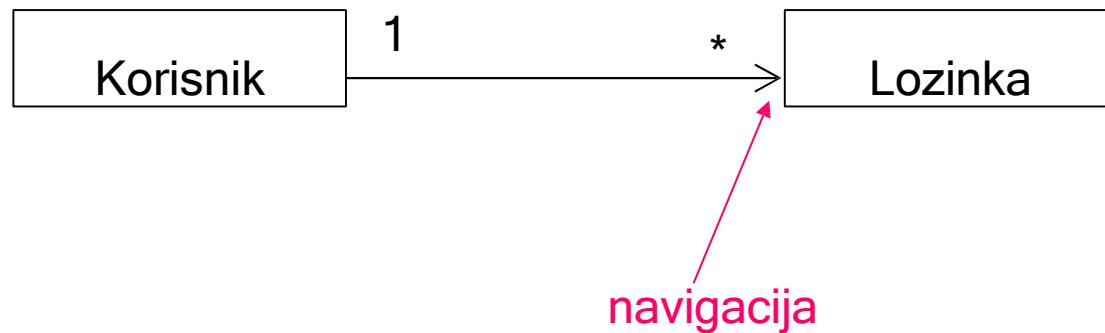
Indikator	Značenje
0..1	Nula ili jedan
1 ili 1..1	Samo jedan
0..* ili *	Nula ili više
1..*	Jedan ili više
n	Samo n (gde je $n > 1$ )
0..n	Od nule do n (gde je $n > 1$ )
1..n	Od jedan do n (gde je $n > 1$ )

# Uloga u asocijaciji



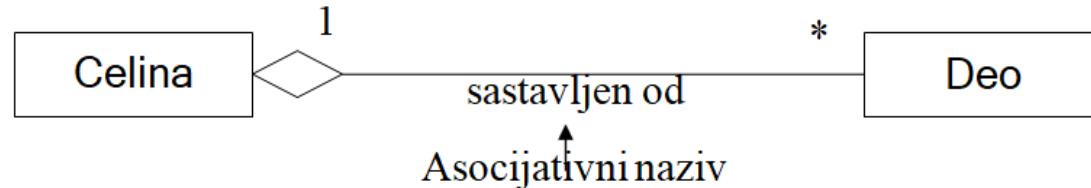
# Ograničenja u navigaciji asocijacija

- ❑ Po definiciji asocijacija i agregacija su bidirekcionе, ali je često potrebno ograničiti navigaciju samo u jednom smeru
- ❑ Ako je navigacija ograničena dodaje se strelica da bi se označio smer navigacije
- ❑ Za zadatog Korisnika – moguće je naći Lozinku, ali nije moguća identifikacija Korisnika na osnovu Lozinke – izraz efikasnosti puta

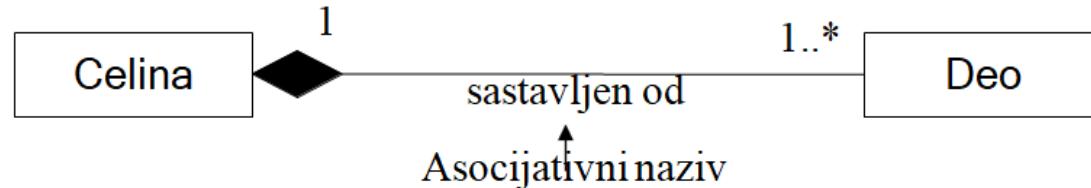


# Agregacija (Aggregation)

- ❑ Agregacija – specijalni oblik asocijacije – jači oblik veze – uspostavlja veza između celine i dela (celina sastavljena od delova)
- ❑ Agregacija je veza tipa ”deo-celina”, ”deo od”, ”ima” ili ”sadrži”
- ❑ **Agregacija po referenci** između celine i dela – uništavanje i kreiranje klase je nezavisno

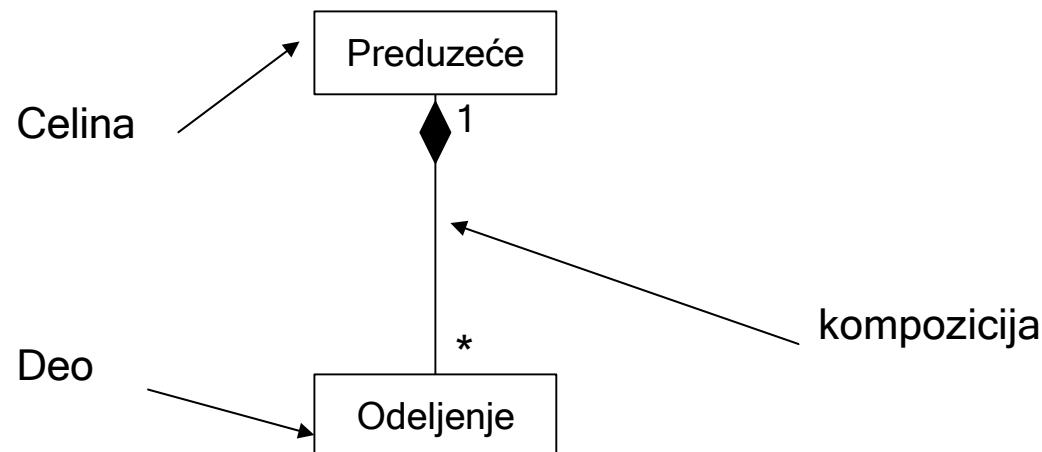


- ❑ **Agregacija po vrednosti – kompozicija** - fizičko zadržavanje celine i dela – deo ne postoji bez celine čiji je deo



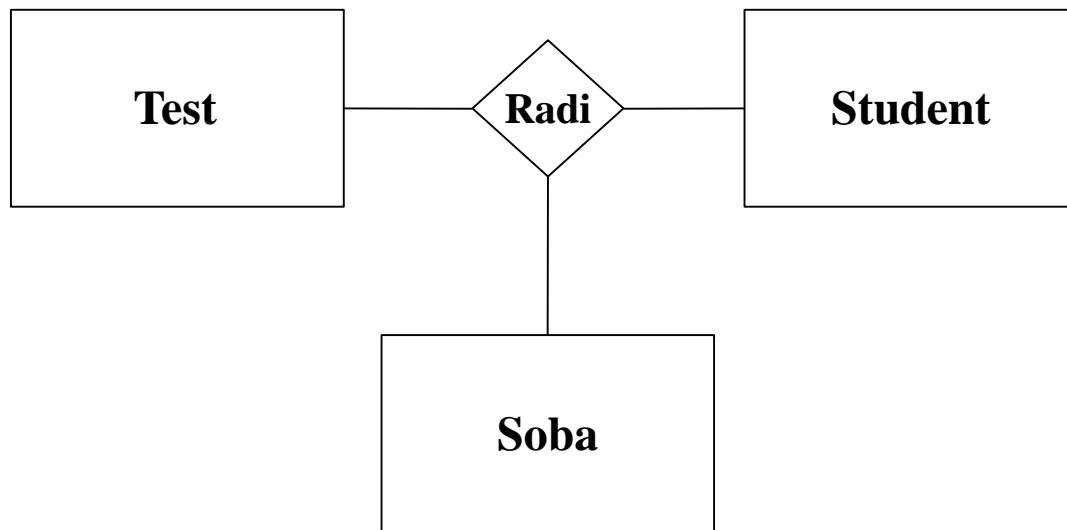
# Primer kompozicije

- ❑ Odeljenje pripada samo određenom preduzeću i nestankom preduzeća nestaje i odeljenje



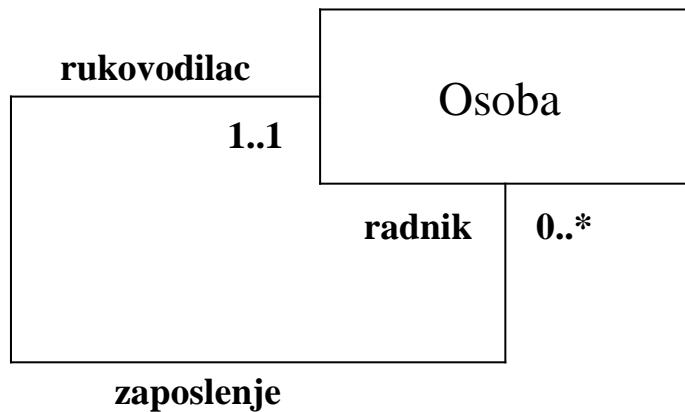
# N-arna asocijacija

- ❑ Mogućnost – više od dve klase u vezi



# Rekurzivna asocijacija

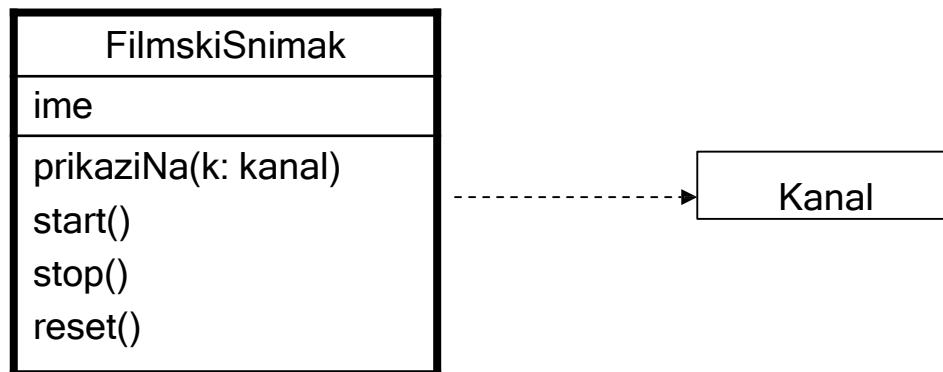
- ❑ Asocijacija koja povezuje klasu samu sa sobom



- ❑ Rekurzivna asocijacija klase Osoba:
  - ❖ Osoba može da bude i Radnik i Rukovodilac
  - ❖ Radnik ima samo jednog Rukovodioca, a Rukovodilac rukovodi nijednim ili sa više radnika

# Zavisnost

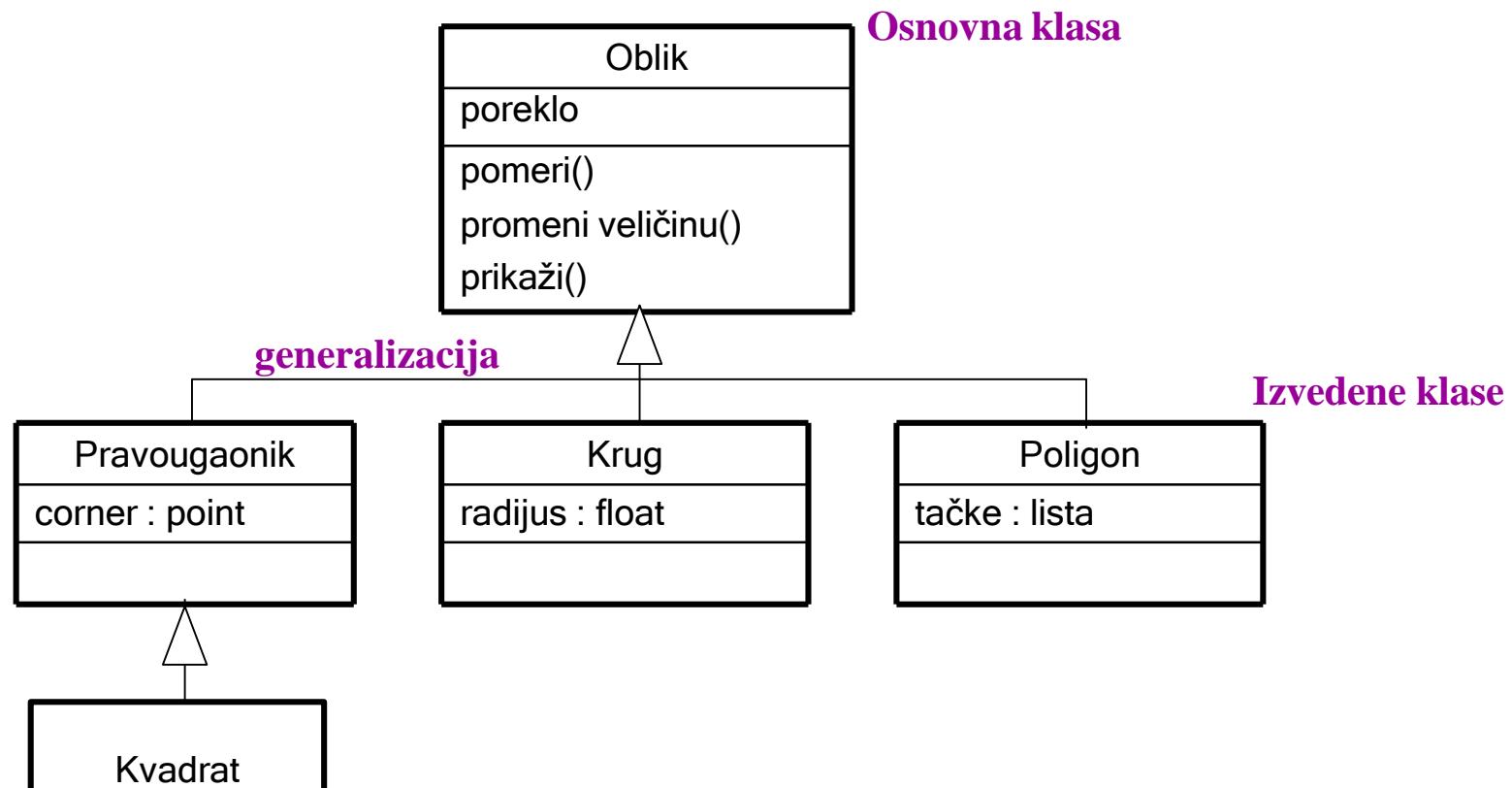
- ❑ Zavisnost se prikazuje kao isprekidana linija usmerena od klijenta (zavisni element) ka drugoj klasi (nezavisni element) – izmena u nezavisnom delu utiče na zavisni
- ❑ Najčešće se koristi – jedna klasa koristi drugu kao argument u oznaci operacije



# Generalizacija

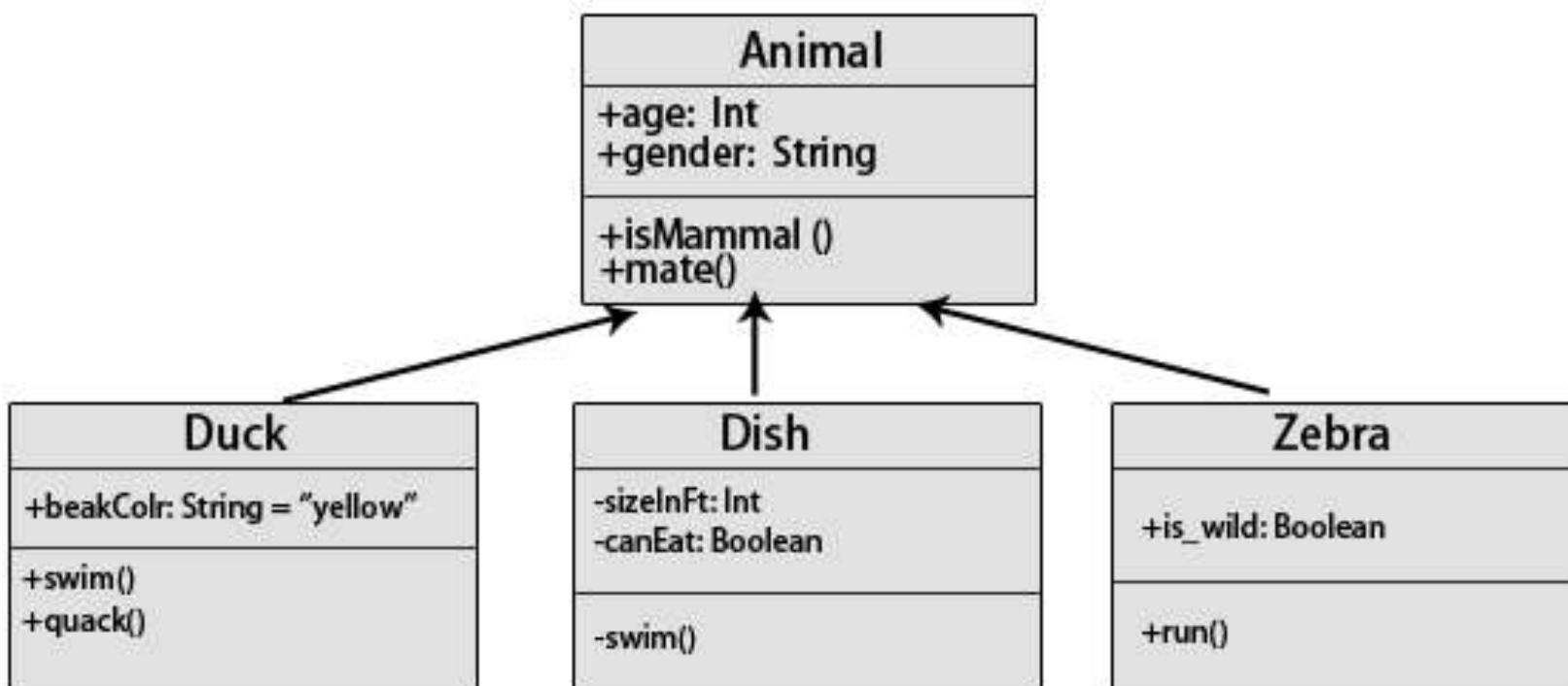
- Generalizacija – nasleđivanje koristi se za prikazivanje odnosa roditelj/dete
- Grafička predstava puna linija sa velikom strelicom u obliku trougla koja pokazuje na roditelja
- Podklase (dete) nasleđuju sve atribute, operacije i asocijacije od superklase (roditelja)

# Primer korišćenja generalizacije



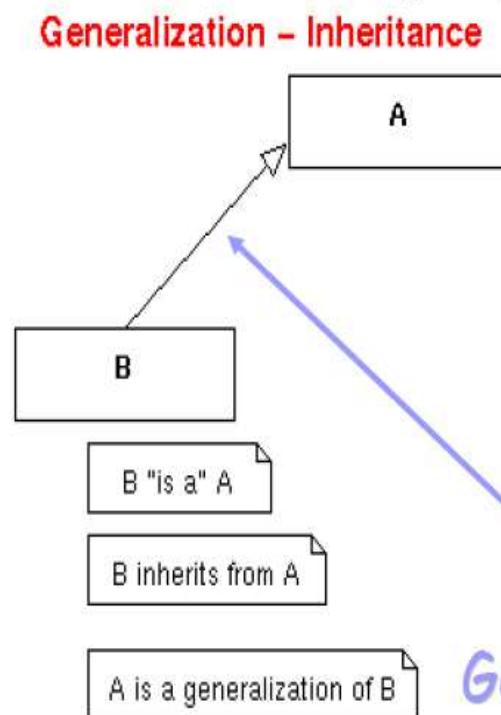
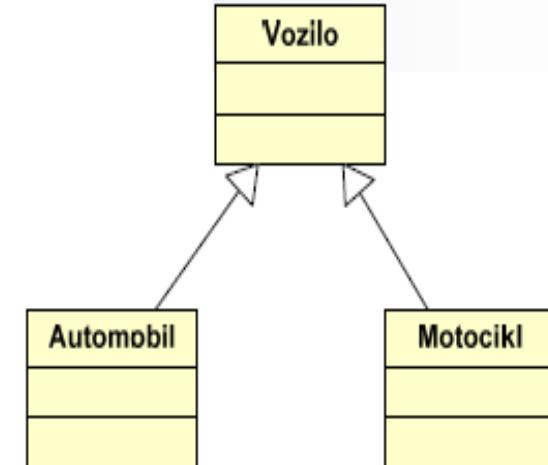
# Primer korišćenja generalizacije

## Class Diagram

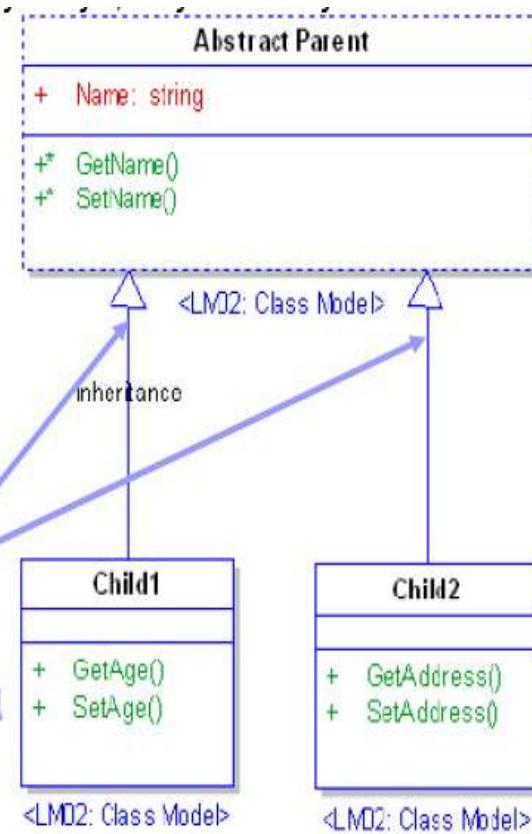


# Generalizacija

- Veza između nadklase i njenih podklasa.
- Opisuje hijerarhijski odnos među klasama. Klase mogu naslediti atribute i ponašanje od nadklasa koje mogu biti podklase drugih klasa.

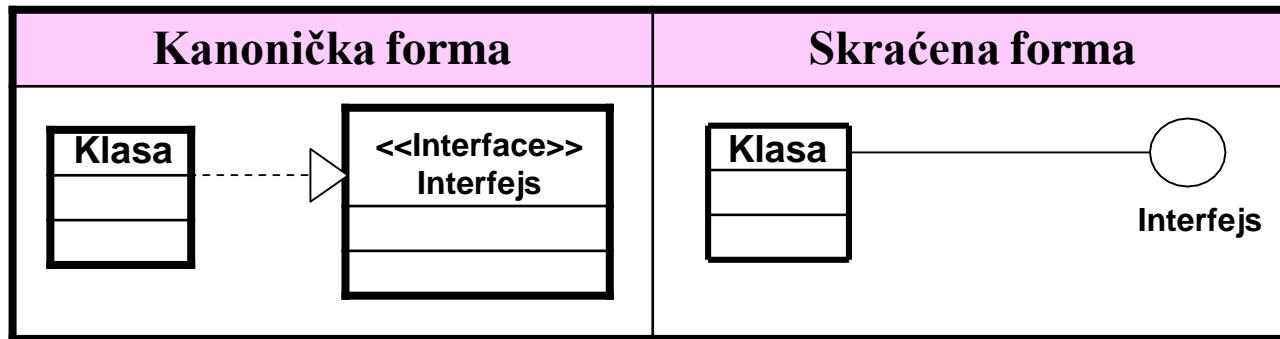


Generalizacija



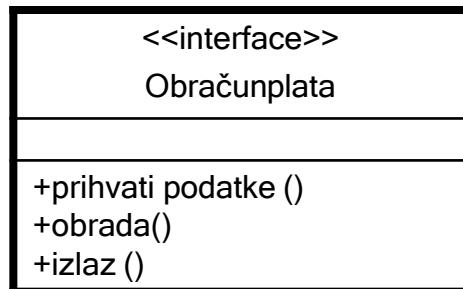
# Realizacija

- ❑ Koristi se za:
  - ❖ označavanje relacije između interfejsa i klase ili
  - ❖ komponente koja obavlja operacije ili servis za interfejs
- ❑ Može se reprezentovati na dva načina:
  - ❖ U kanoničkoj formi i
  - ❖ U skraćenoj (elided) formi

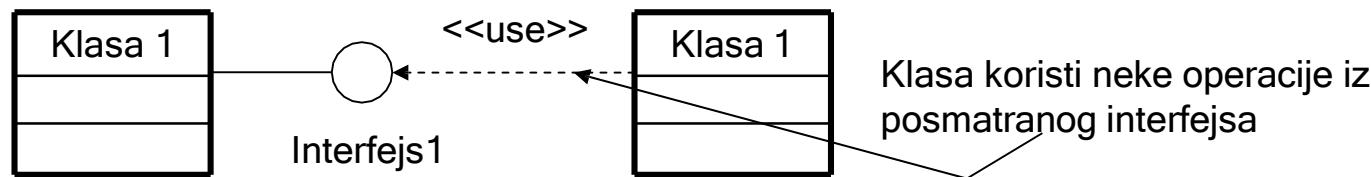


# Definisanje interfejsa klase

- Interfejs se na dijagramu klasa može prikazati na dva načina:
  - ❖ **Stereotipom <<interface>>** iznad oznake klase, gde je popunjen samo odeljak za operacije



- ❖ **Krugom**, uz koji se navodi ime interfejsa i koji je pripojen punom linijom klasama koje ga podržavaju



# Definisanje dijagrama klasa

- ❑ Konceptualni model – “dijagram klasa bez operacija”
- ❑ Dijagram klasa sadrži:
  - ❖ klase,
  - ❖ interfejse,
  - ❖ saradnje,
  - ❖ veze zavisnosti, generalizacije i asocijativnosti

# Dijagram klasa (eng. Class Diagram)

## Definicija

- Dijagram klase opisuje statičku strukturu sistema (modela) i daje pregled sistema pokazujući njegove klase i odnose među tim klasama.
- Dijagram klasa je statički prikaz – on pokazuje šta uzajamno deluje, ali ne pokazuje šta se događa tokom tog uzajamnog delovanja.

## Semantika

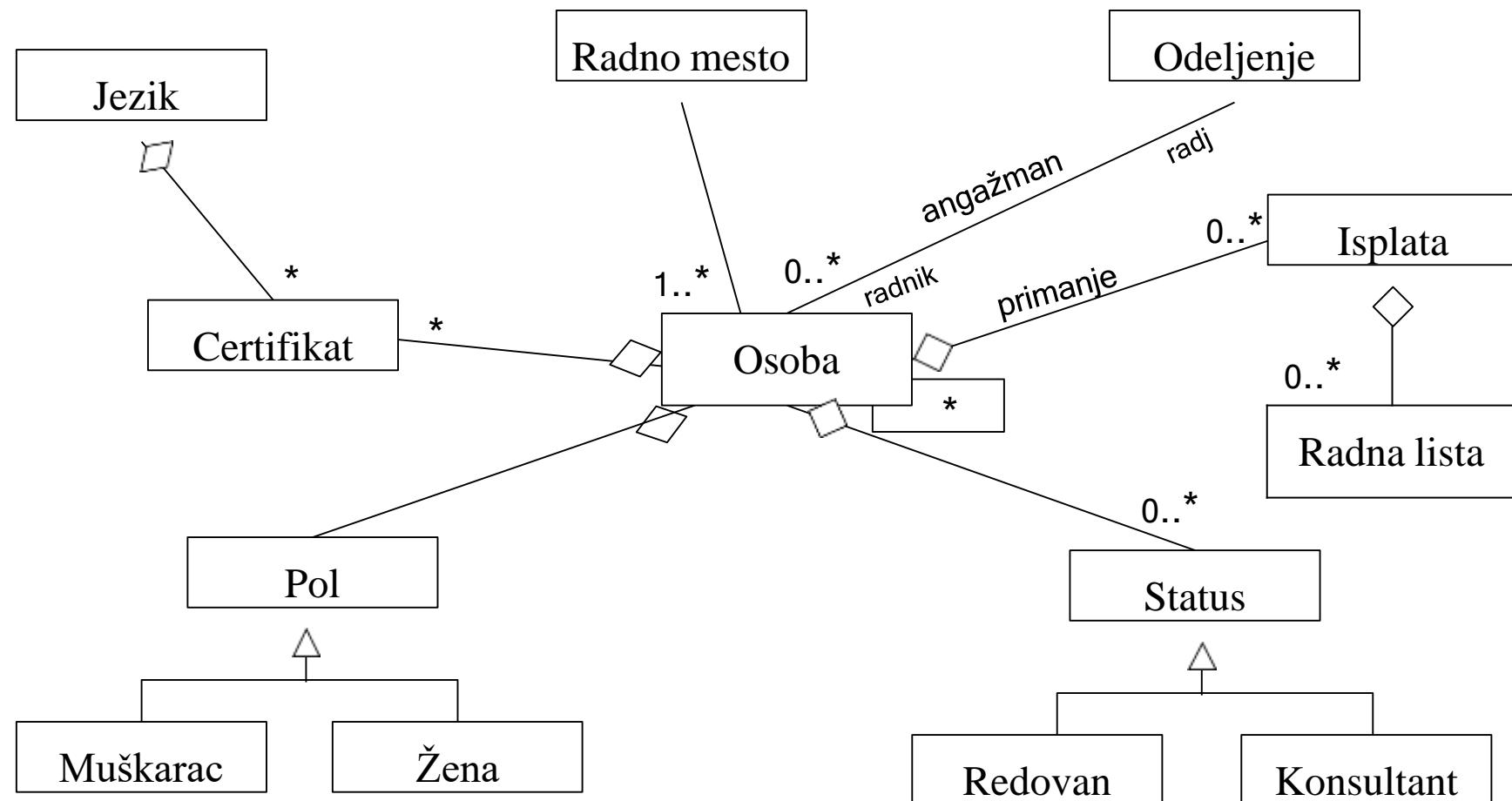
- Dijagram klase je grafički prikaz statičke strukture sistema.
- Dijagram klasa pokazuje egzistenciju klasa i njihovih relacija u logičkom dizajnu sistema.

# Dijagram klasa (eng. Class Diagram)

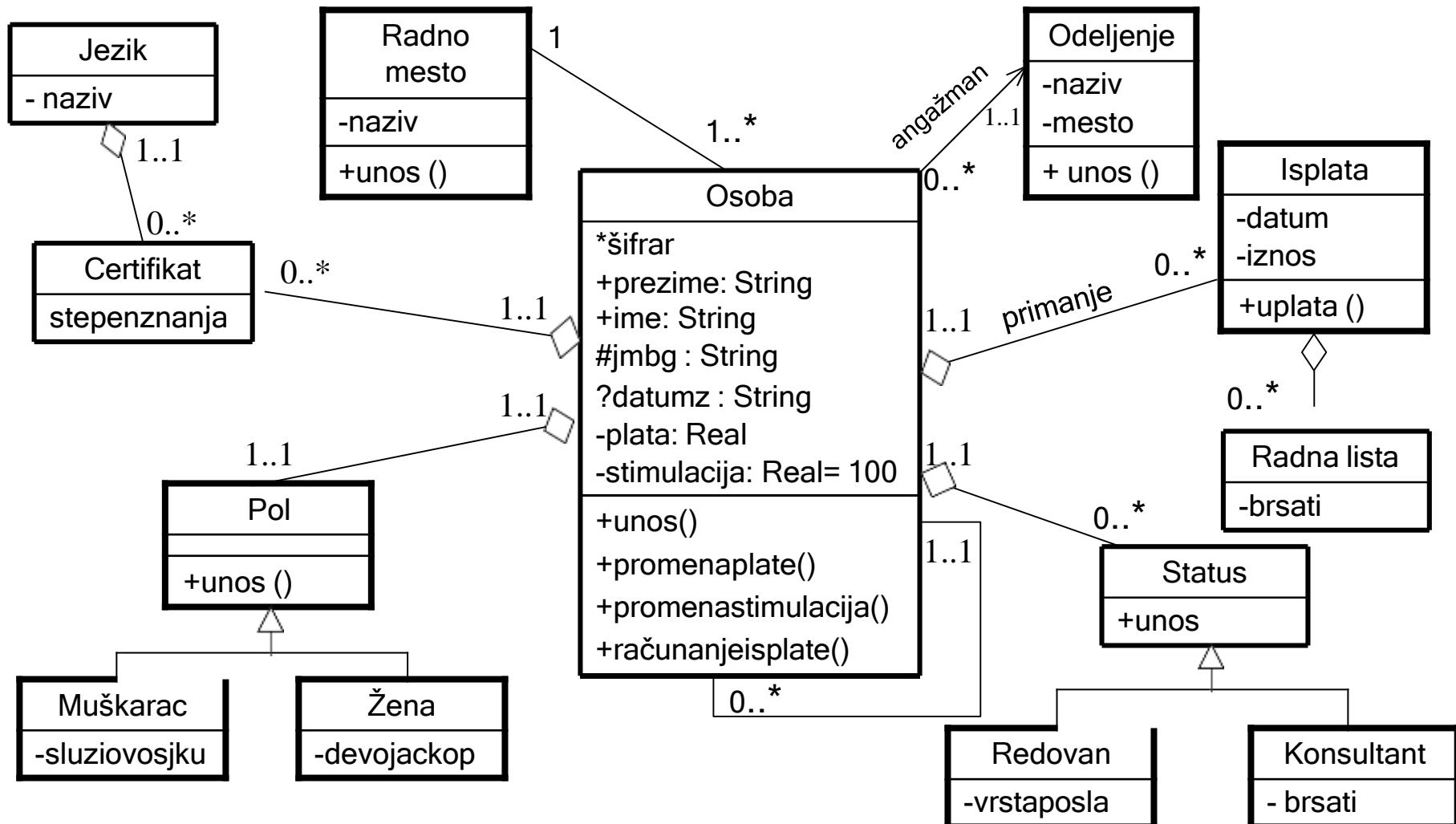
## Notacija

- Dijagram klasa može predstavljati sve ili samo neke klase strukture sistema.
- Njime se oslikavaju moguća stanja sistema.
- Strukturno, dijagram klasa je kolekcija statičnih elemenata sistema, kao što su:
  - paketi koji organiziraju klase i relacije u posebne jedinice,
  - klase kao uzorci izgradnje objekata,
  - njihove međusobne relacije i povezanosti koje mogu biti asocijacija, zavisnost i generalizacija.

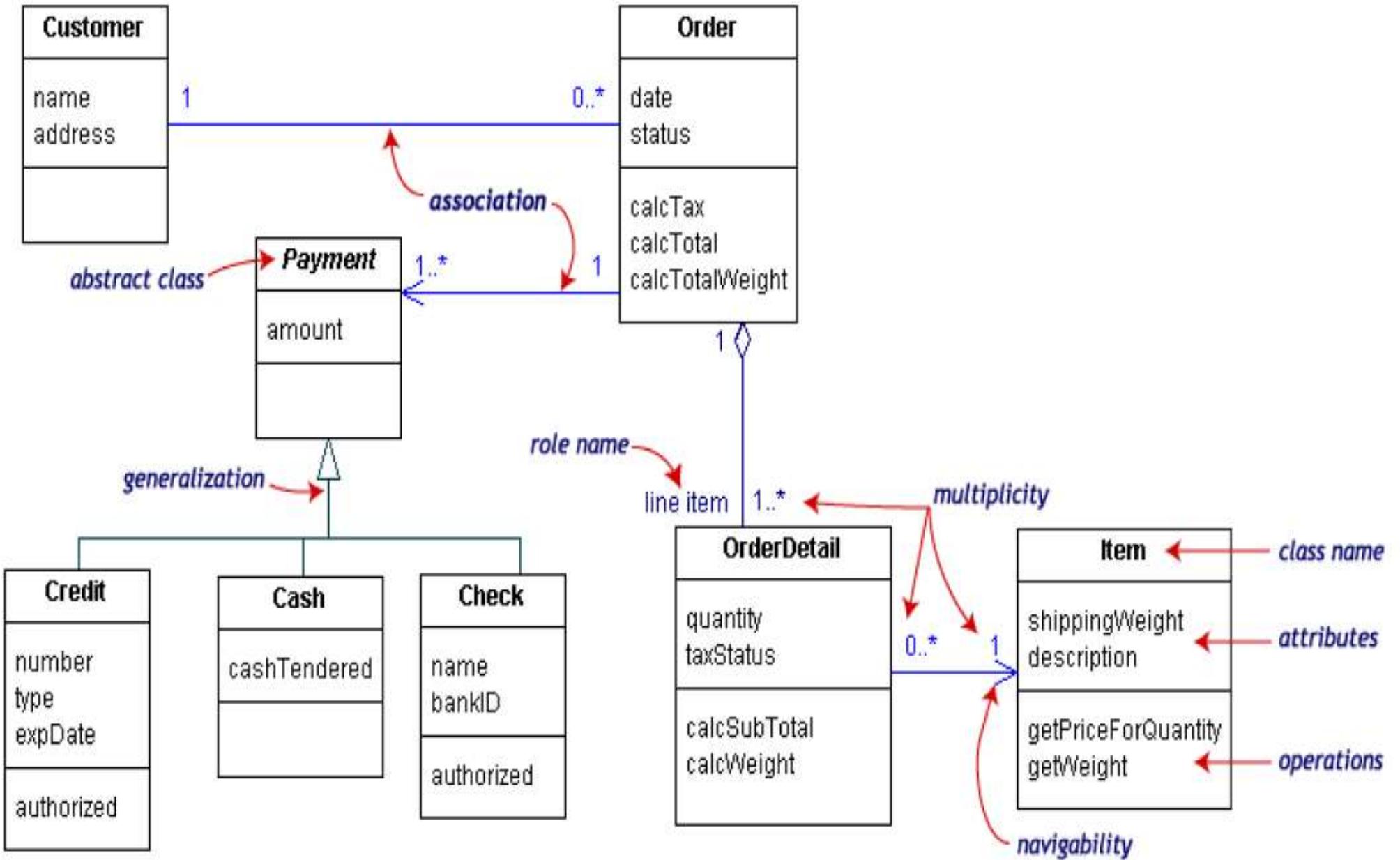
# Konceptualni model



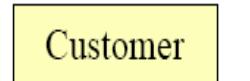
# Primer definisanja dijagrama klasa



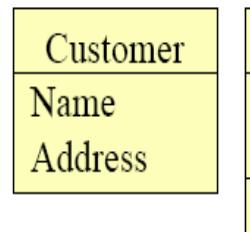
# Dijagram klasa (eng. Class Diagram)



# UML dijagram klasa – sažetak notacije



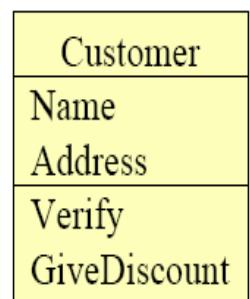
Class (name only)



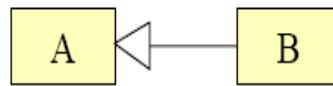
Class with attributes



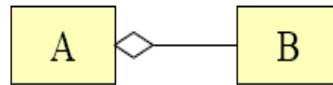
Class with operations (methods)



Class with attributes and operations (methods)



B is a specialization of A  
A is a generalization of B



A is an aggregate of Bs



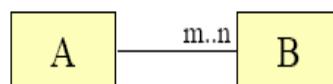
A contains B



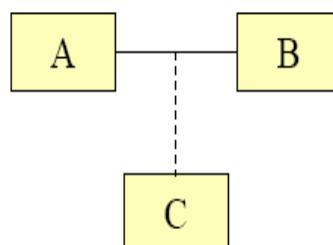
A is associated with exactly one B



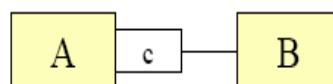
A is associated with many (zero or more) Bs



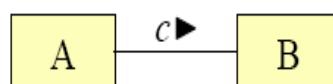
A is associated with minimum m and maximum n Bs; n can be \*.



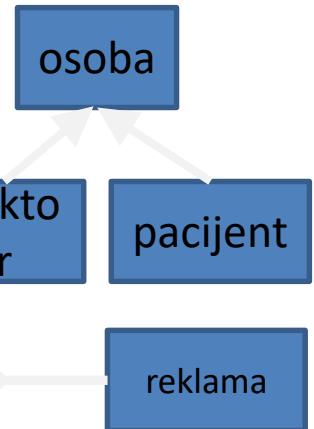
Class C represents the properties assigned to the relationship between A and B.



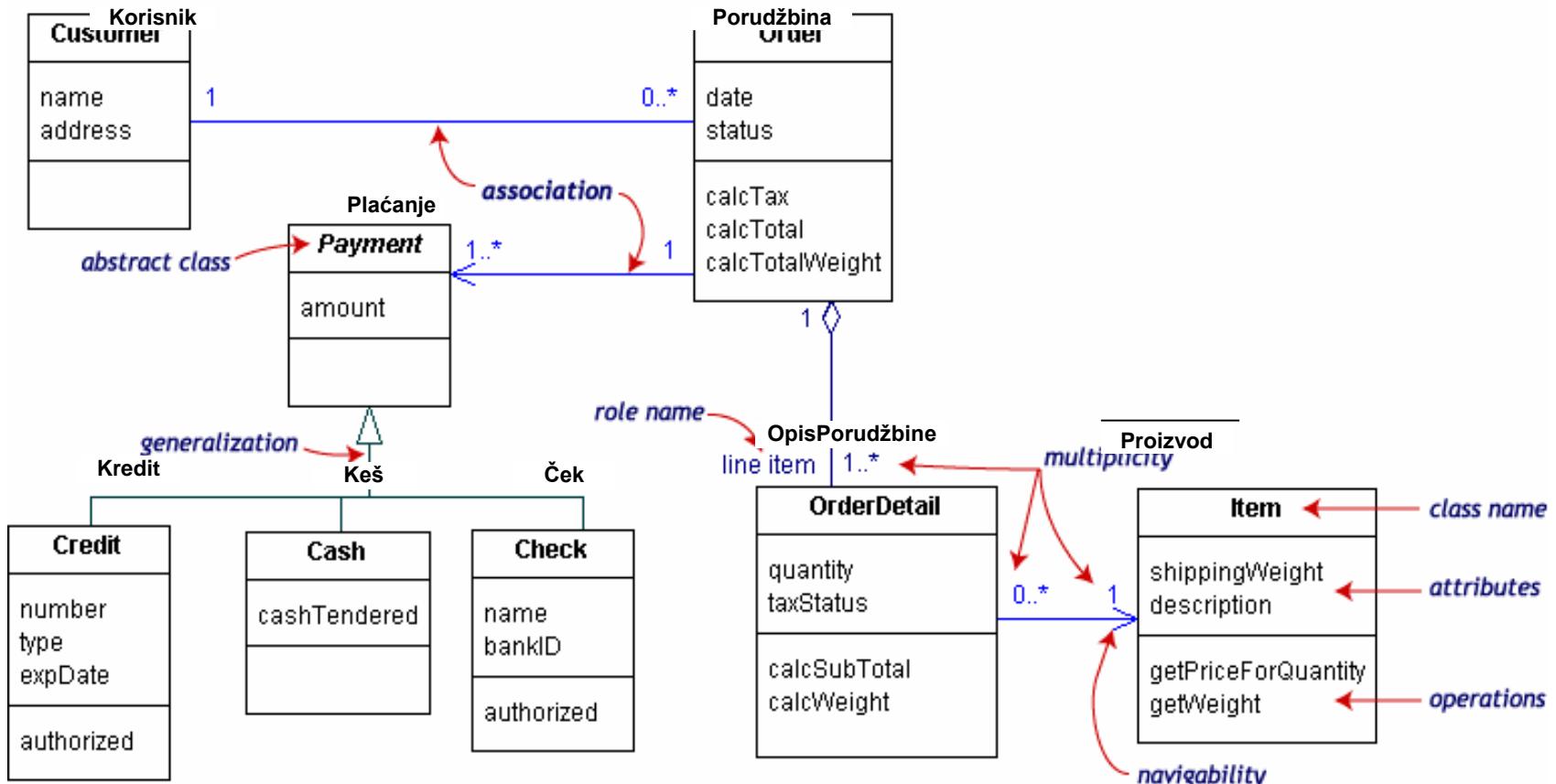
Association between A and B is qualified by attribute c.



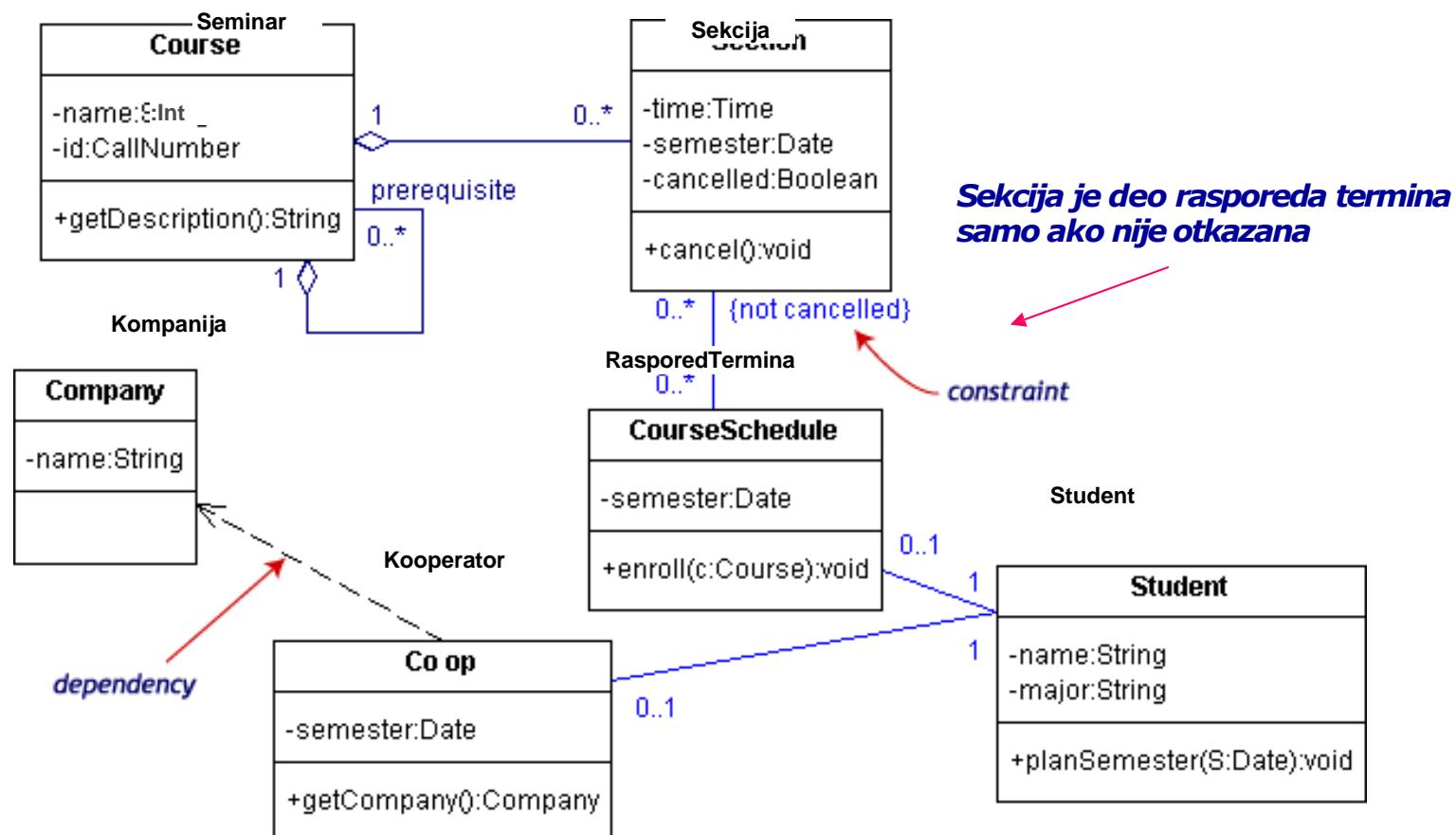
Association name c between A and B is to be read from A to B.



# Primer

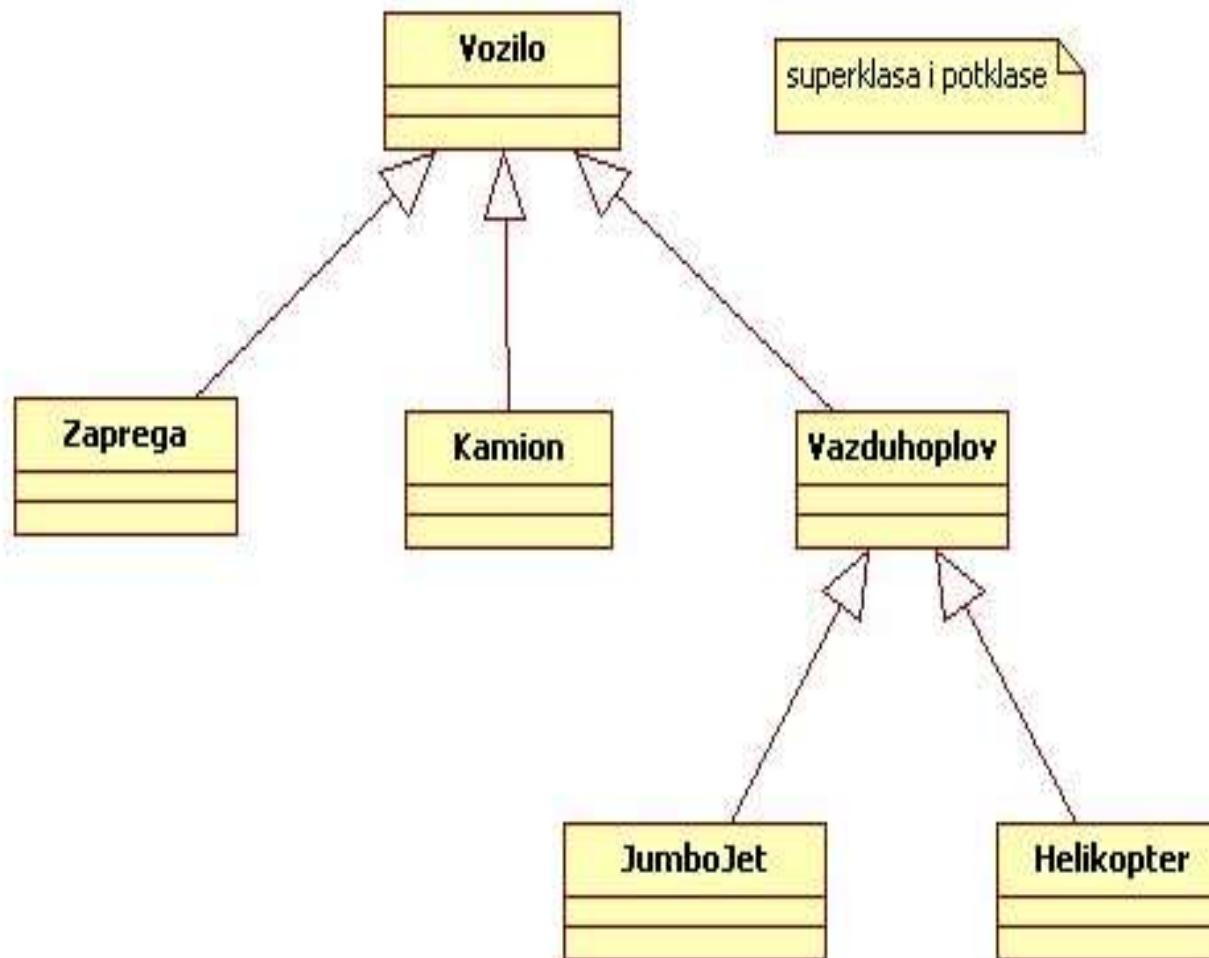


# Primer



# Primer:

- Nacrtati dijagram klasa koji opisuje sledeći sistem: helikopter i jumbo-jet su vrsta vazduhoplova, a vazduhoplov, kamion i zaprega su vrsta vozila.



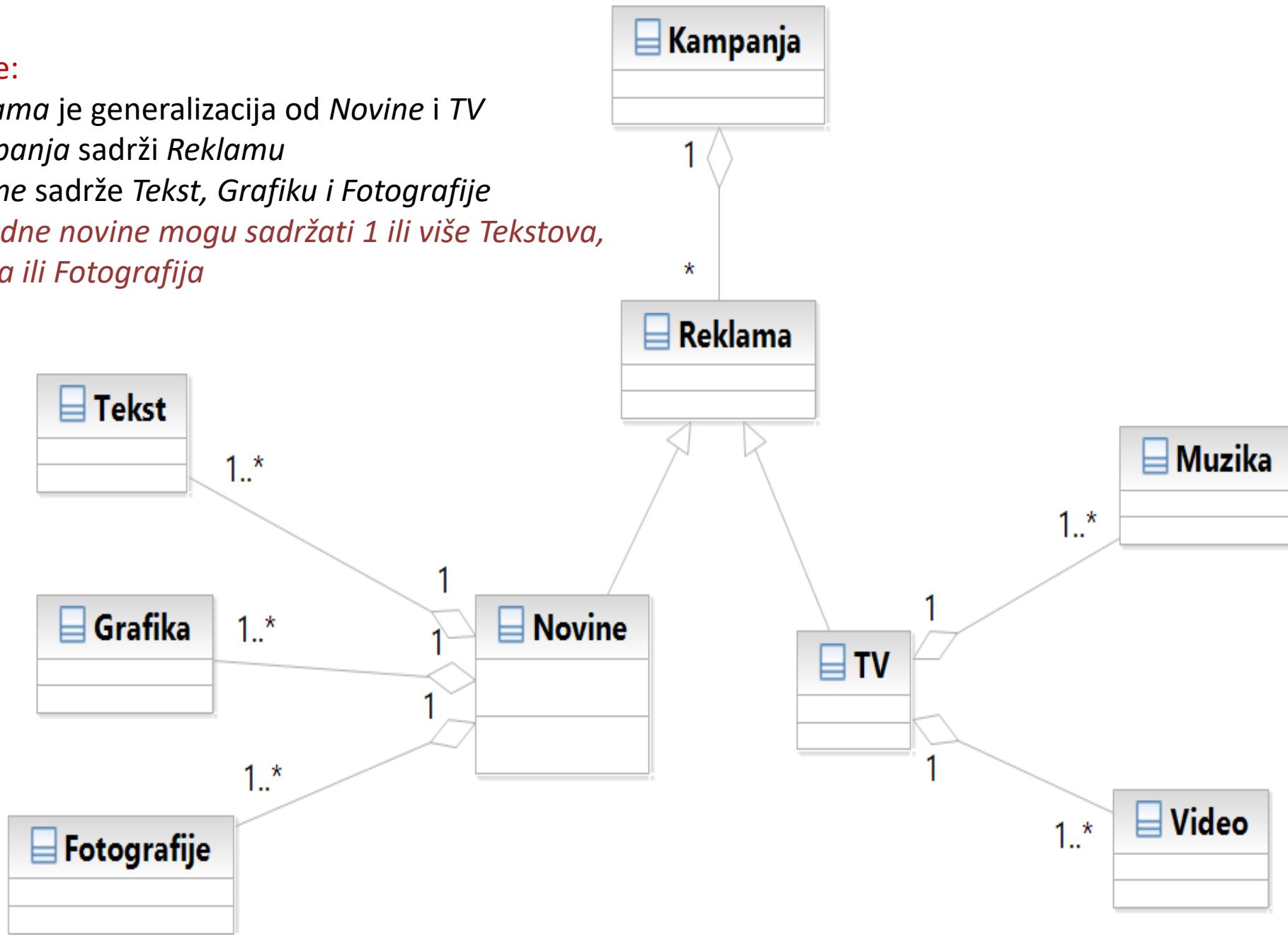
# Primer:

- Nacrtati dijagram klasa koji opisuje deo reklamne kampanje koji se odnosi na način reklamiranja.
- U okviru reklamne kampanje može se koristiti jedan ili više načina reklamiranja:
  - novine – uključuje, pisanu, grafičku formu oglašavanja kao i fotografije
  - TV – uključuje korišćenje audio i video zapisa, filmova, muzike, učešće glumaca i slično.

# Primer:

Čitanje:

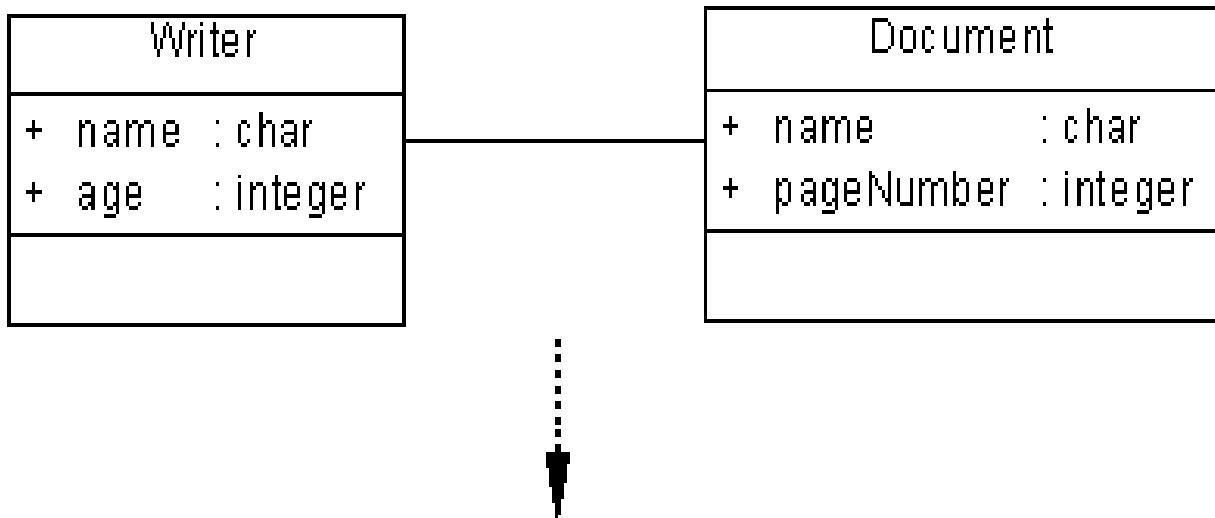
- \*Reklama je generalizacija od Novine i TV
- \*Kampanja sadrži Reklamu
- \*Novine sadrže Tekst, Grafiku i Fotografije
- I to: jedne novine mogu sadržati 1 ili više Tekstova, Grafika ili Fotografija



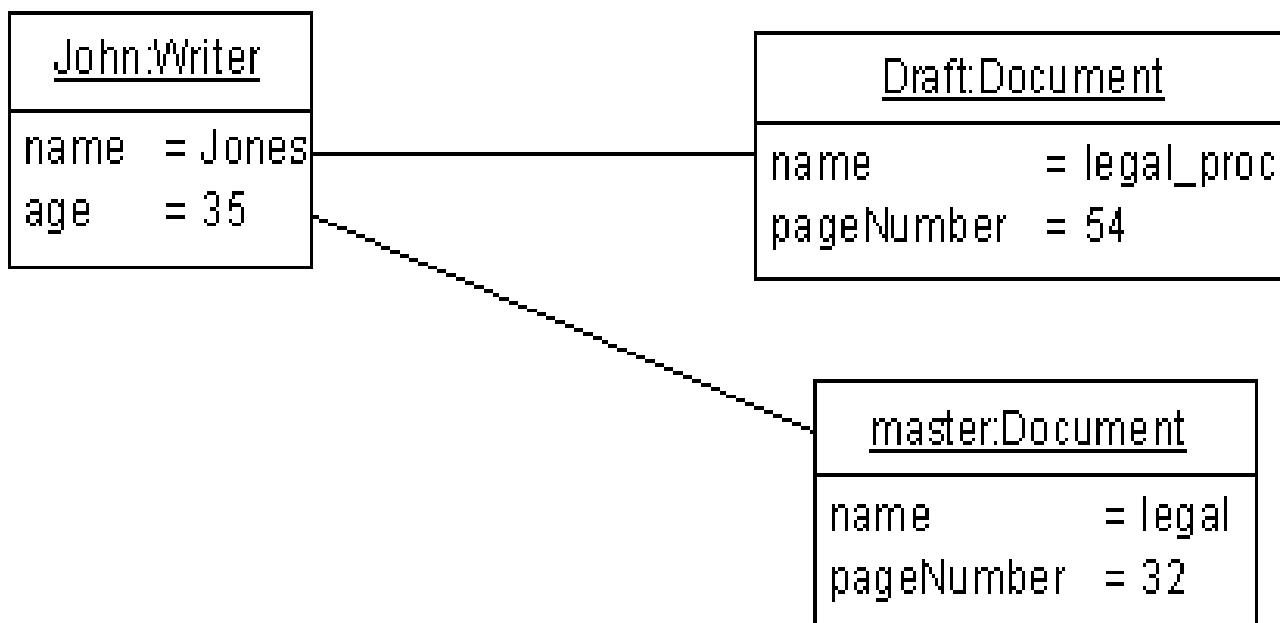
# Dijagrami objekata

- Dijagrami objekata su izvedeni iz klasnih dijagrama pa su objektni dijagrami zavisni od klasnih dijagrama.
- Objektni dijagrami predstavljaju instancu dijagrama klase.
- Osnovni koncepti su slični i za klasni dijagram i za dijagram objekata. Dijagrami objekata takođe prikazuju staticki pogled na sistem, ali ovaj staticki pogled predstavlja sliku sistema u određenom momentu.
- Dijagrami objekata se koriste da iscrtaju skup instanci objekata i njihove međusobne veze.

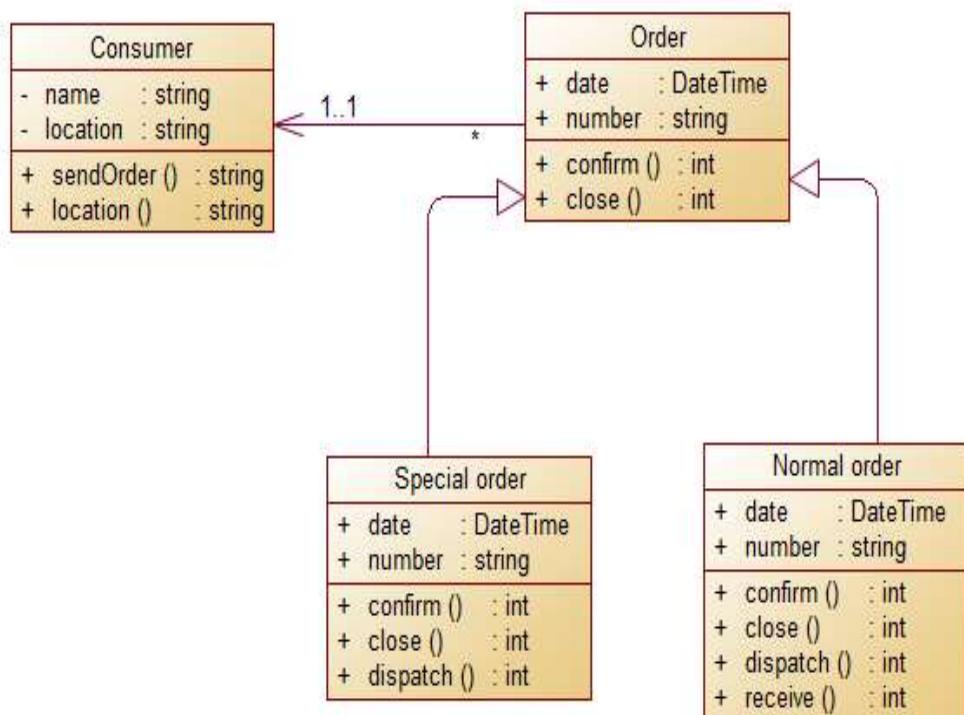
*Class diagram*



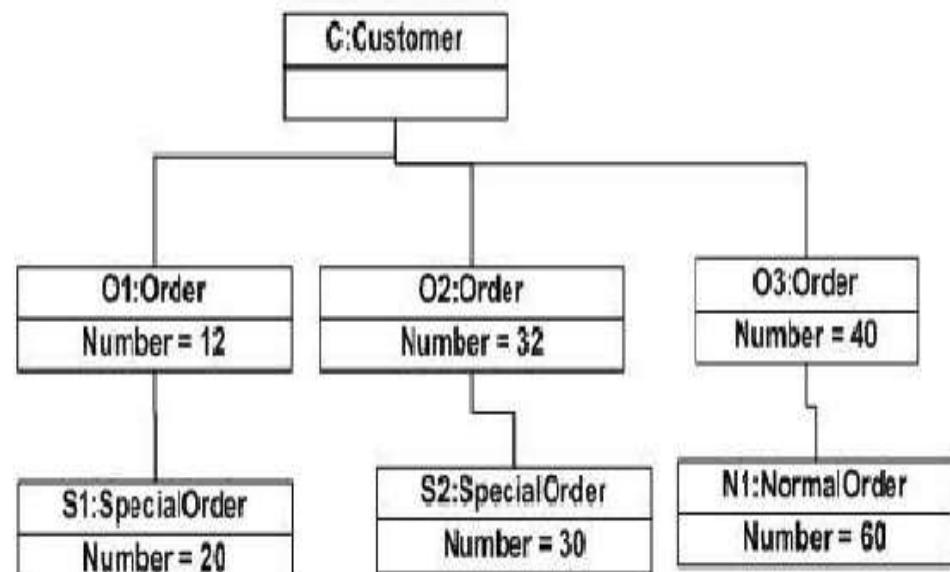
*Object diagram*



KLASNI DIJAGRAM



Object diagram of an order management system



## C# kod za definisanje klase *Special order*

Class Properties - Special order (SpecialOrder)

General Detail Attributes Operations Ports Parts Annotations C# Definition Preview

Ln 2, Col 20

```
// File:      SpecialOrder.cs
// Author:    Dragana
// Purpose:   Definition of Class SpecialOrder

using System;

public class SpecialOrder : Order
{
    public new int Confirm()
    {
        throw new NotImplementedException();
    }

    public new int Close()
    {
        throw new NotImplementedException();
    }

    public int Dispatch()
    {
        throw new NotImplementedException();
    }

    public int Receive()
    {
        throw new NotImplementedException();
    }

    public new DateTime date;
    public new string number;
}
```

Source

More >> OK Cancel Apply Help

## C# kod za definisanje klase *Consumer* .....

Class Properties - Consumer (Consumer)

General Detail Attributes Operations Ports Parts Annotations C# Definition Preview

Ln 2, Col 20

```
// File:      Consumer.cs
// Author:    Dragana

// Purpose:  Definition of Class Consumer

using System;

public class Consumer
{
    private string name;
    private string location;

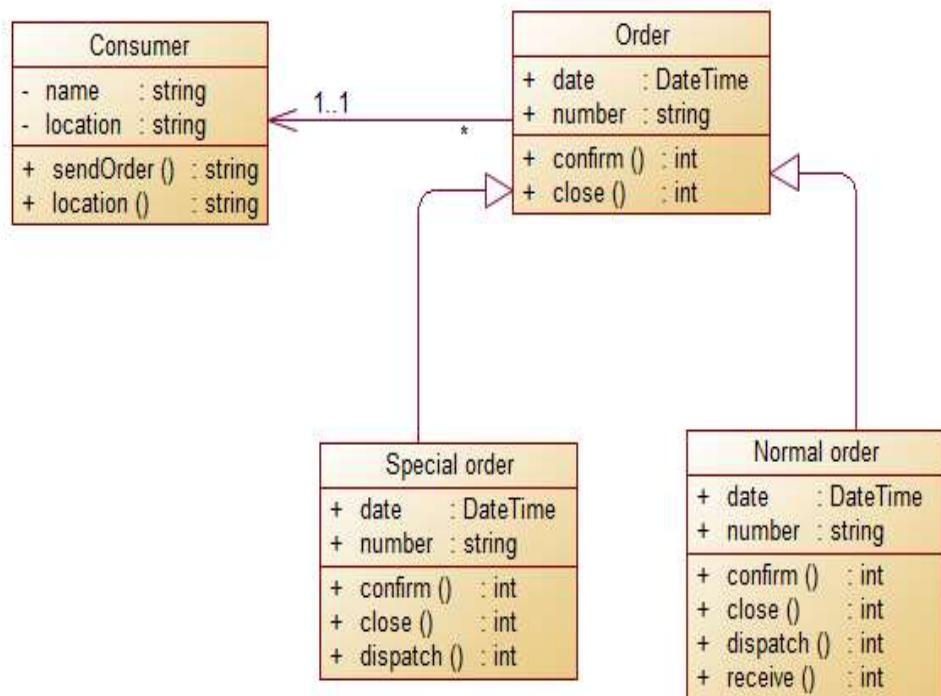
    public string SendOrder()
    {
        throw new NotImplementedException();
    }

    public string Location()
    {
        throw new NotImplementedException();
    }
}
```

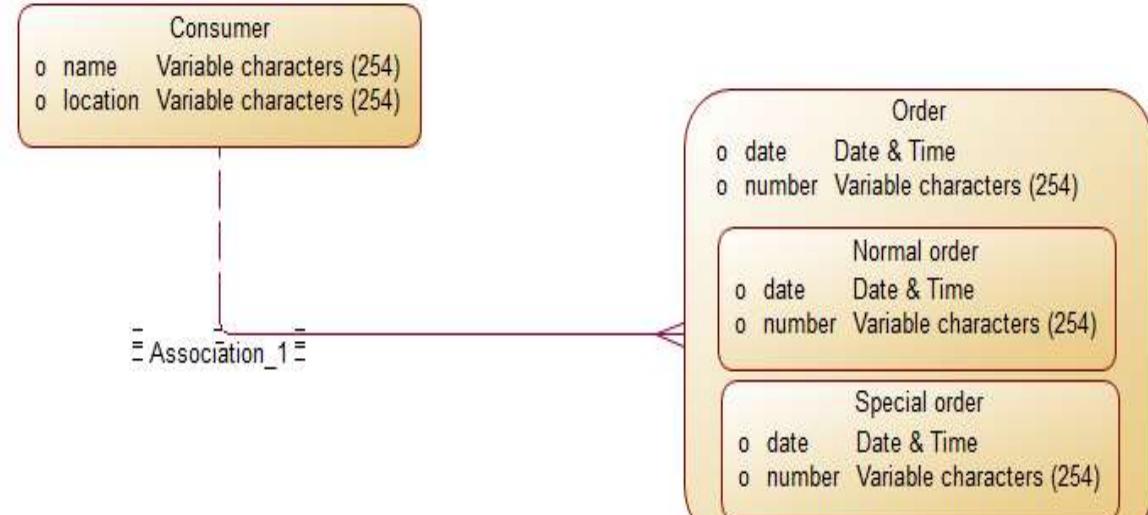
Source

More >> OK Cancel Apply Help

### KLASNI DIJAGRAM



### KONCEPTUALNI MODEL

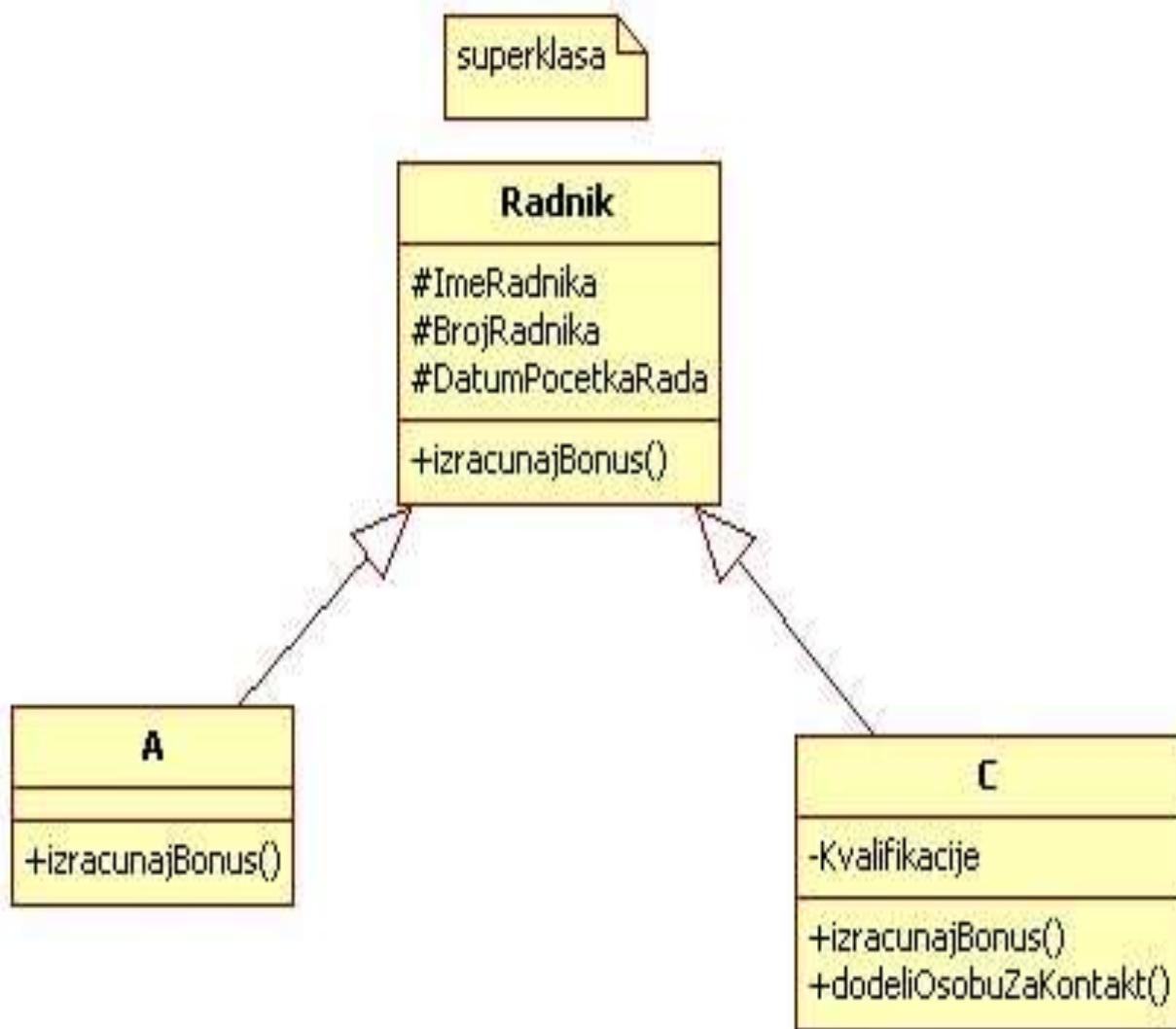


# Primer:

## Opis dela poslovnog sistema jedne kompanije

- Nacrtati dijagram klasa koji opisuje deo poslovnog sistema jedne kompanije.  
Svi radnici kompanije se mogu podeliti u dve grupe:
  - prvu, koju čine zaposleni u administraciji (A) i
  - drugu, koju čine članovi projektnih timova tj. zaposleni koji rade na reklamnim kampanjama (C).
- Za sve zaposlene neophodno je zapamtiti osnovne podatke tj. ime i prezime, jedinstveni broj radnika kao i datum početka rada. Ovi atributi su zaštićeni (*protected*).
- Jedna od razlika zaposlenih je računanje godišnjih bonusa:
  - bonus kod C se računa na osnovu profita kampanja na kojima su radili
  - bonus kod A se računa na osnovu prosečnog profita svih kampanja u prethodnoj godini.
- Razlike među zaposlenima su još i da:
  - je kod C neophodno zapamtiti kvalifikacije (privatni atribut)
  - C može biti osoba za kontakt sa klijentom
  - A se ne može dodeliti ni jednoj kampanji.

# Primer:



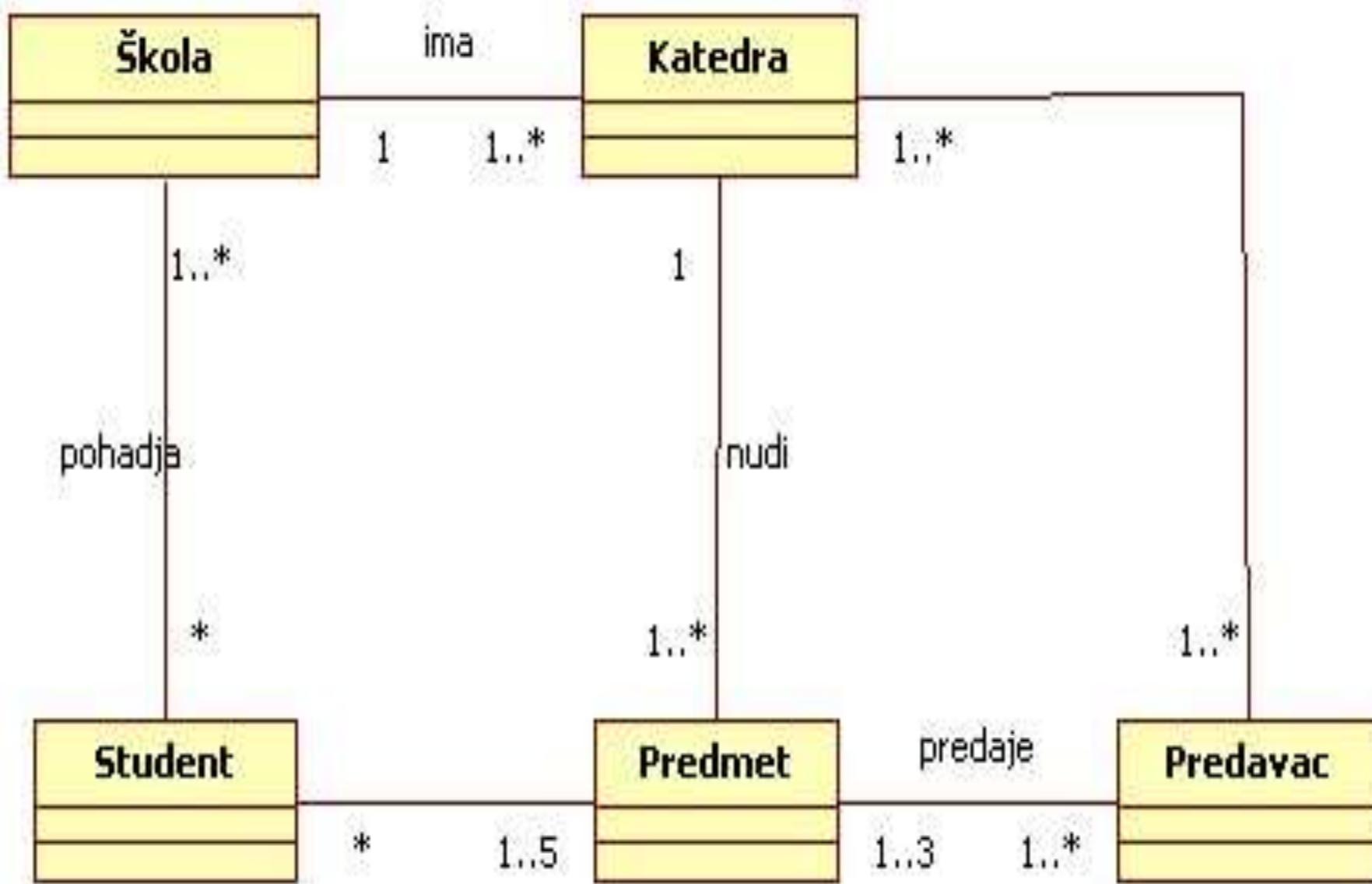
dve potklase - nasledjuju sve atribute i operacije superklase

# Primer:

## Opis sistema za škole

- Nacrtati dijagram klasa na jeziku UML sledećeg sistema:
- Škola ima jednu ili više katedri
- Svaka katedra nudi jedan ili više predmeta
- Određeni predmet može biti ponuđen samo od strane jedne katedre
- Svaka katedra ima predavače, a svaki predavač može raditi na jednoj katedri ili na više njih
- Student može prijaviti do 5 predmeta u školi
- Predavač može predavati ne više od tri kursa
- Svaki predmet može biti predavan od strane više predavača
- Student može pohađati više od jedne škole

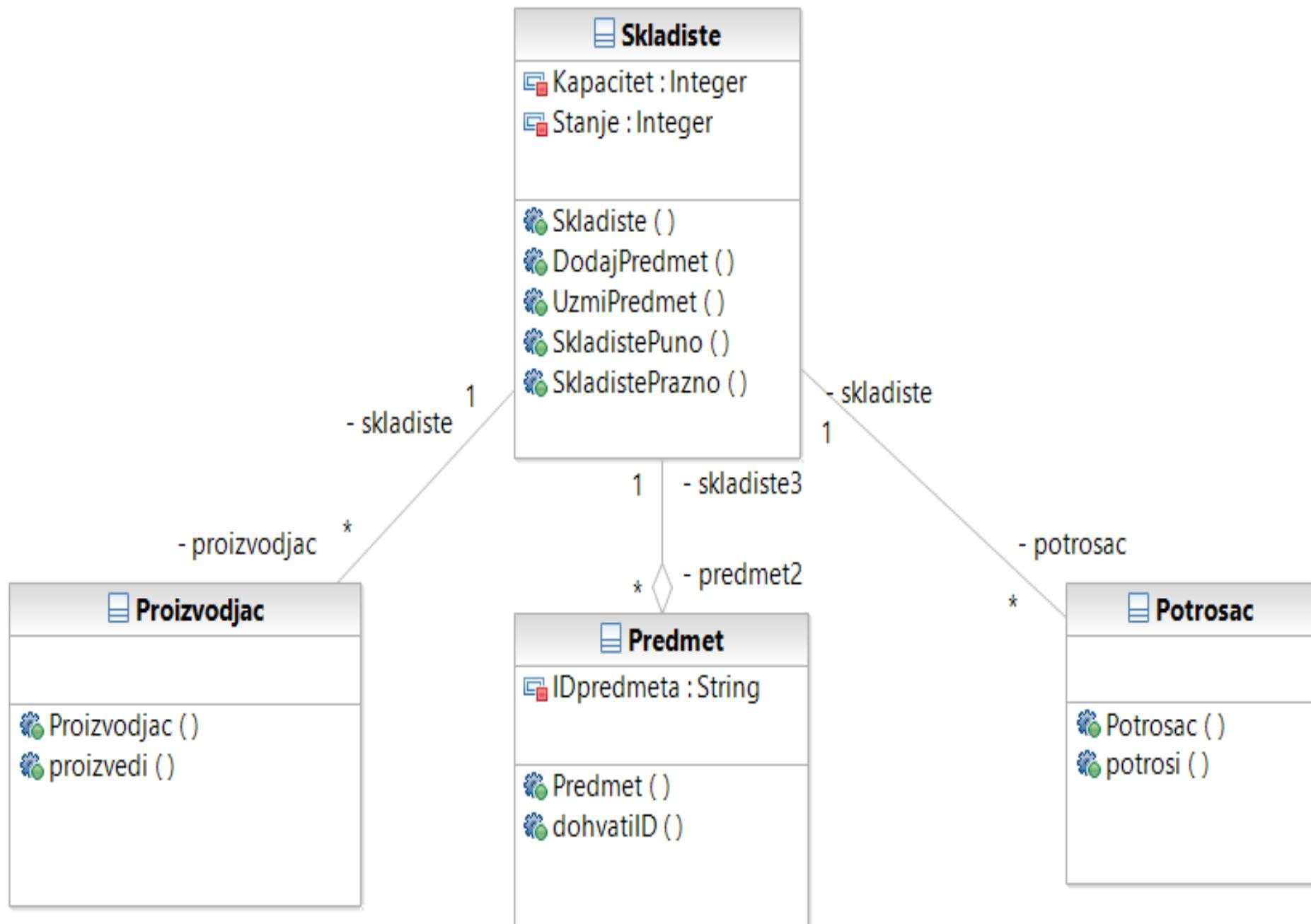
# Primer:



# Primer:

- Nacrtati dijagram klasa na jeziku UML sledećeg sistema klasa:
- **Predmet** – ima jedinstven, automatski generisan ID, koji može da se *dohvati*
- **Skladište** – može da sadrži zadati broj predmeta, otvara se prazno, posle čega predmeti mogu da se stavlju i uzimaju jedan po jedan. Može se *dohvatiti* broj predmeta u skladištu da se ispita da li je skladište puno ili prazno.
- **Proizvođac** – može da napravi jedan predmet i da ga stavi u skladište, koje se zadaje prilikom stvaranja proizvođača.
- **Potrošač** – može da uzme jedan predmet iz skladišta, koje se zadaje prilikom stvaranja potrošača.
- atributi su privatni (-) a operacije javne (+)

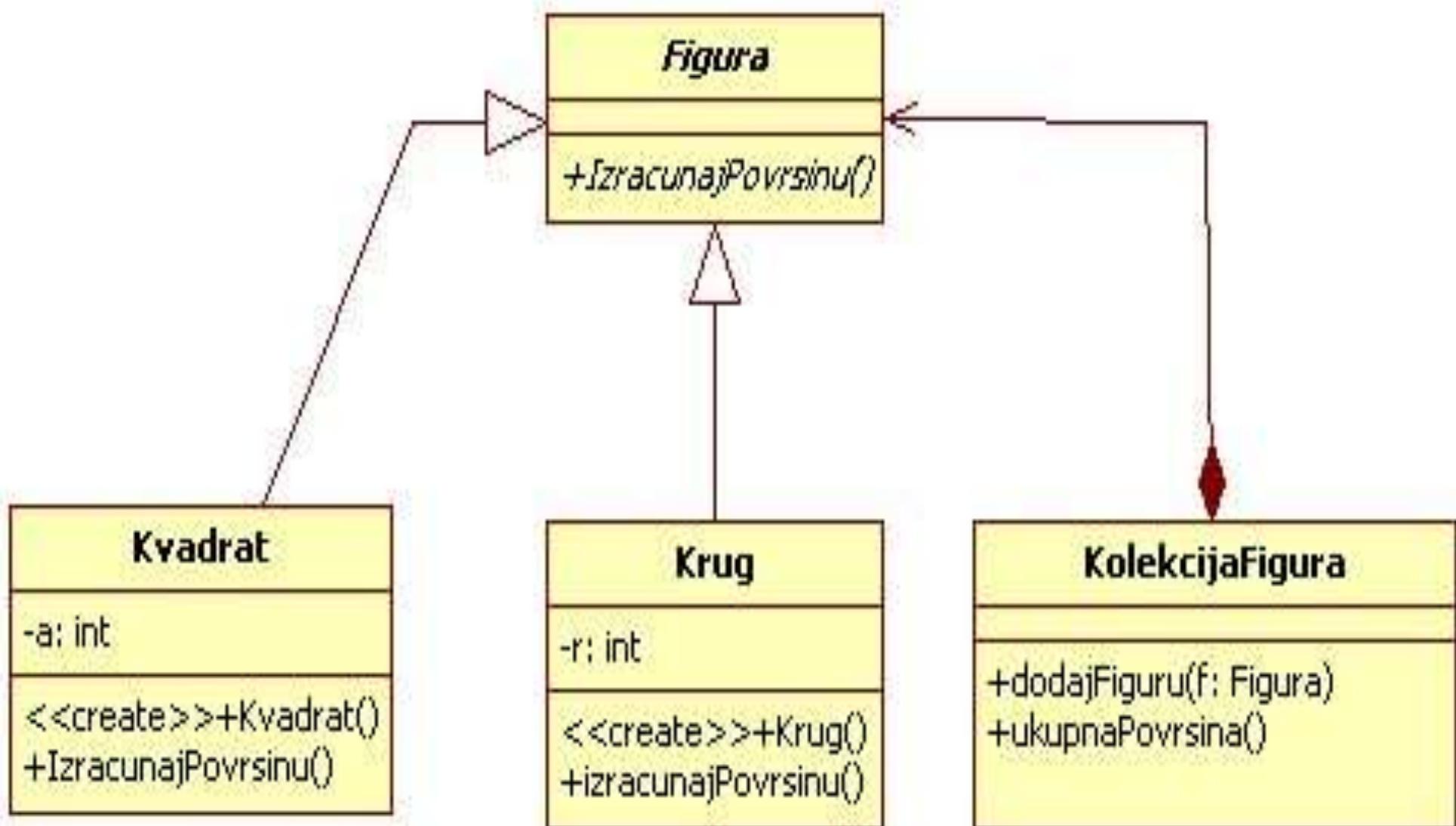
# Primer:



# Primer:

- Napraviti sistem za rad sa figurama u 2D ravni.
- Od figura za sada postoje Kvadrat i Krug.
- Kvadrat je opisan dužinom stranice, dok je Krug opisan poluprečnikom.
- Ove veličine korisnik zadaje prilikom konstrukcije odgovarajućeg objekta.
- Svakoj figuri se može izračunati površina. Dodatno korisniku je na raspolaganju Kolekcija figura koja može sadržati proizvoljan broj 2D figura.
- Kolekciji se takođe može izračunati površina kao suma površina svih figura koje sadrži.
- Nacrtati dijagram klase opisanog sistema.

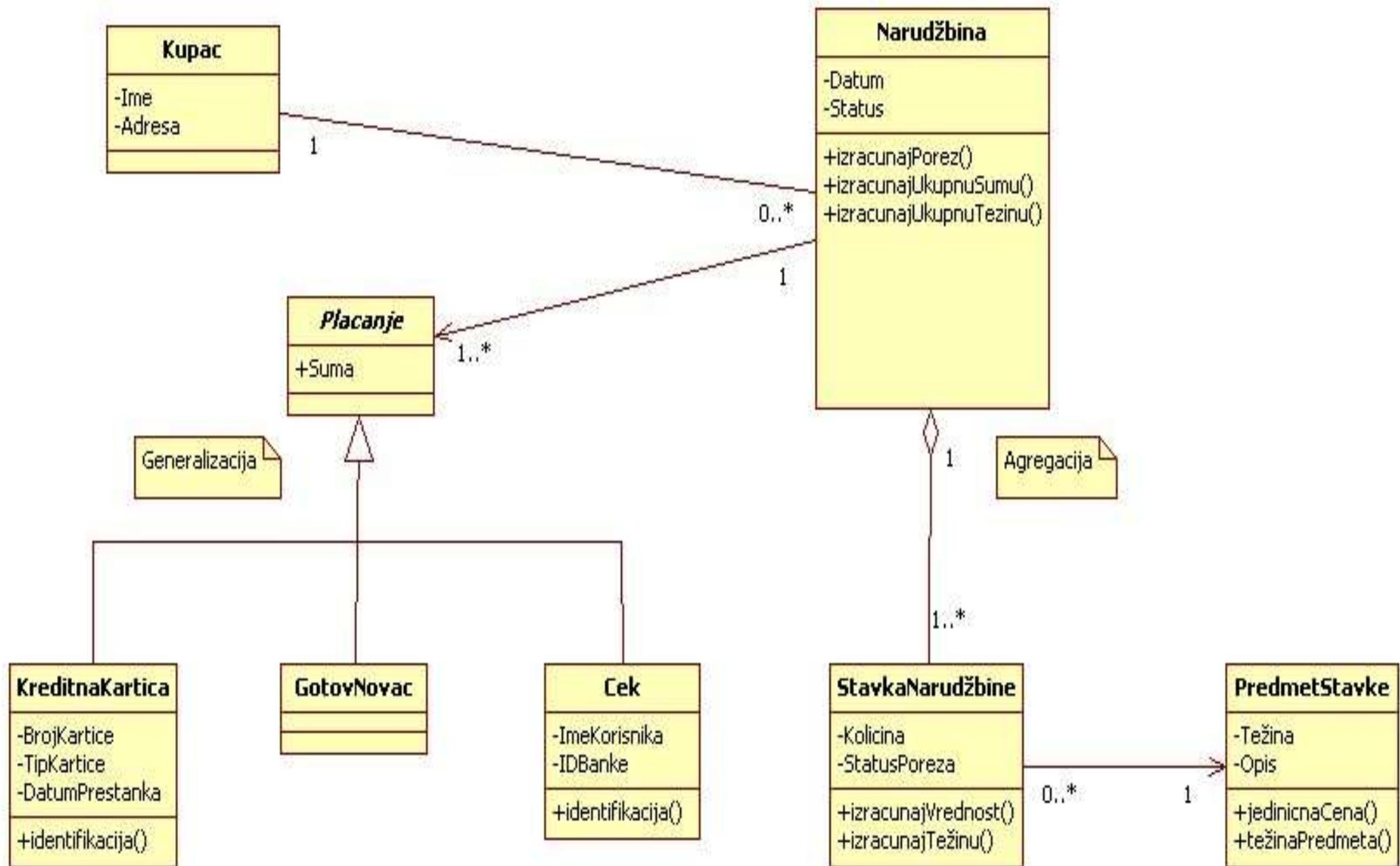
# Primer:



# Primer:

- Nacrtati dijagram klasa na jeziku UML sledećeg sistema klasa:
  - Kupac može da napravi određeni broj porudžbina. Da bi napravio porudžbinu treba da ostavi svoje ime i adresu.
  - Osnovni podaci o narudžbini su datum i status, i mogu se **dobiti** informacije o porezu, ukupnoj sumi i težini date porudžbine.
  - Ona se može platiti (i to određeni iznos) pomoću više vrsta plaćanja: kreditnom karticom (bitni podaci su broj, tip i datum prestanka važenja, a može se proveriti da li je izvršena identifikacija), novcem ili čekom (pamti se ime korisnika i ID banke, a proverava se da li je izvršena i identifikacija).
  - Svaka narudžbina ima svoje stavke (pamti se količina i status poreza, a može se izračunati vrednost stavke i težina robe).
  - Svaka stavka može imati jedan predmet koji je definisan težinom i opisom, a može se pristupiti jediničnoj ceni i težini datog predmeta.

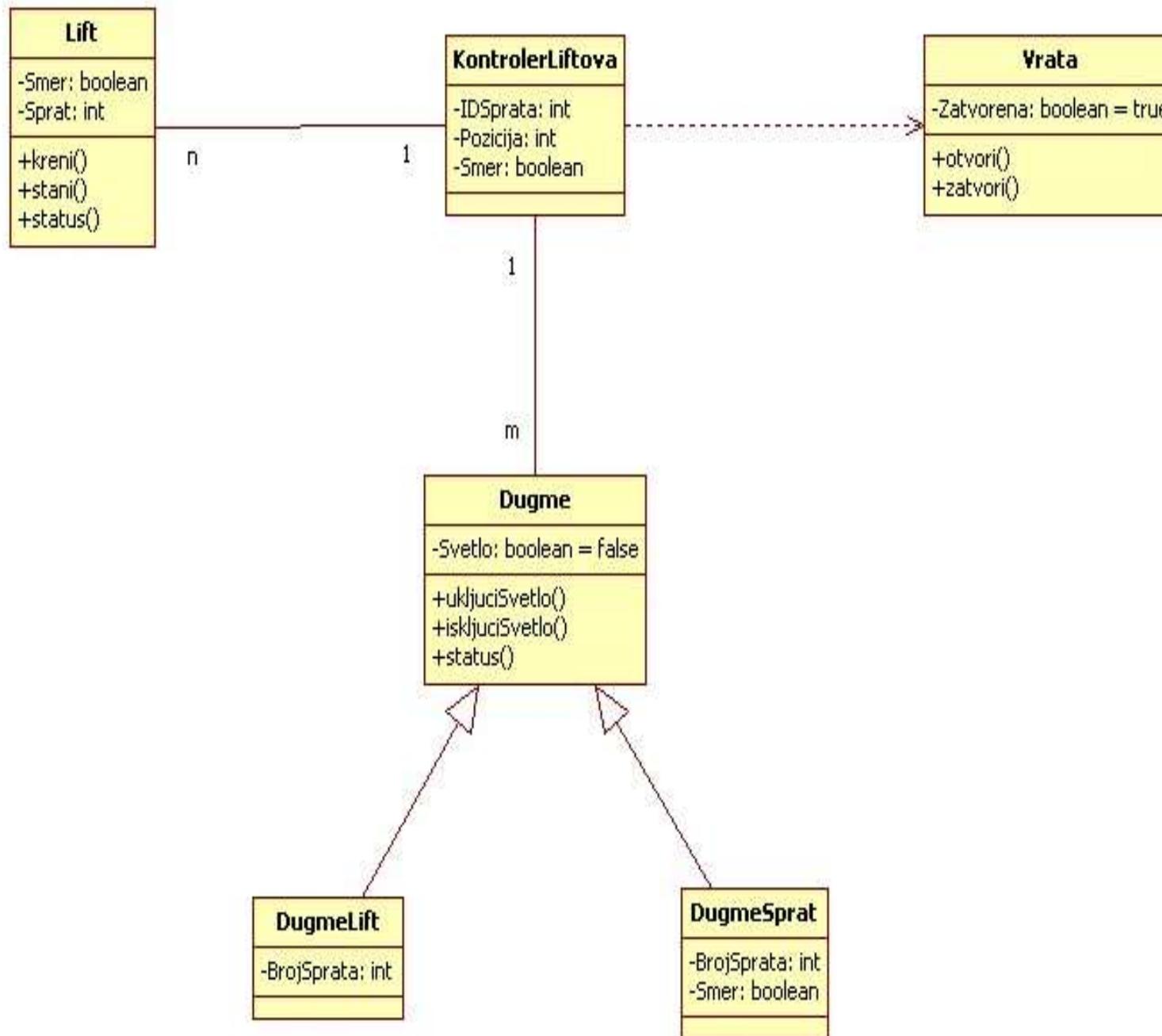
# Primer:



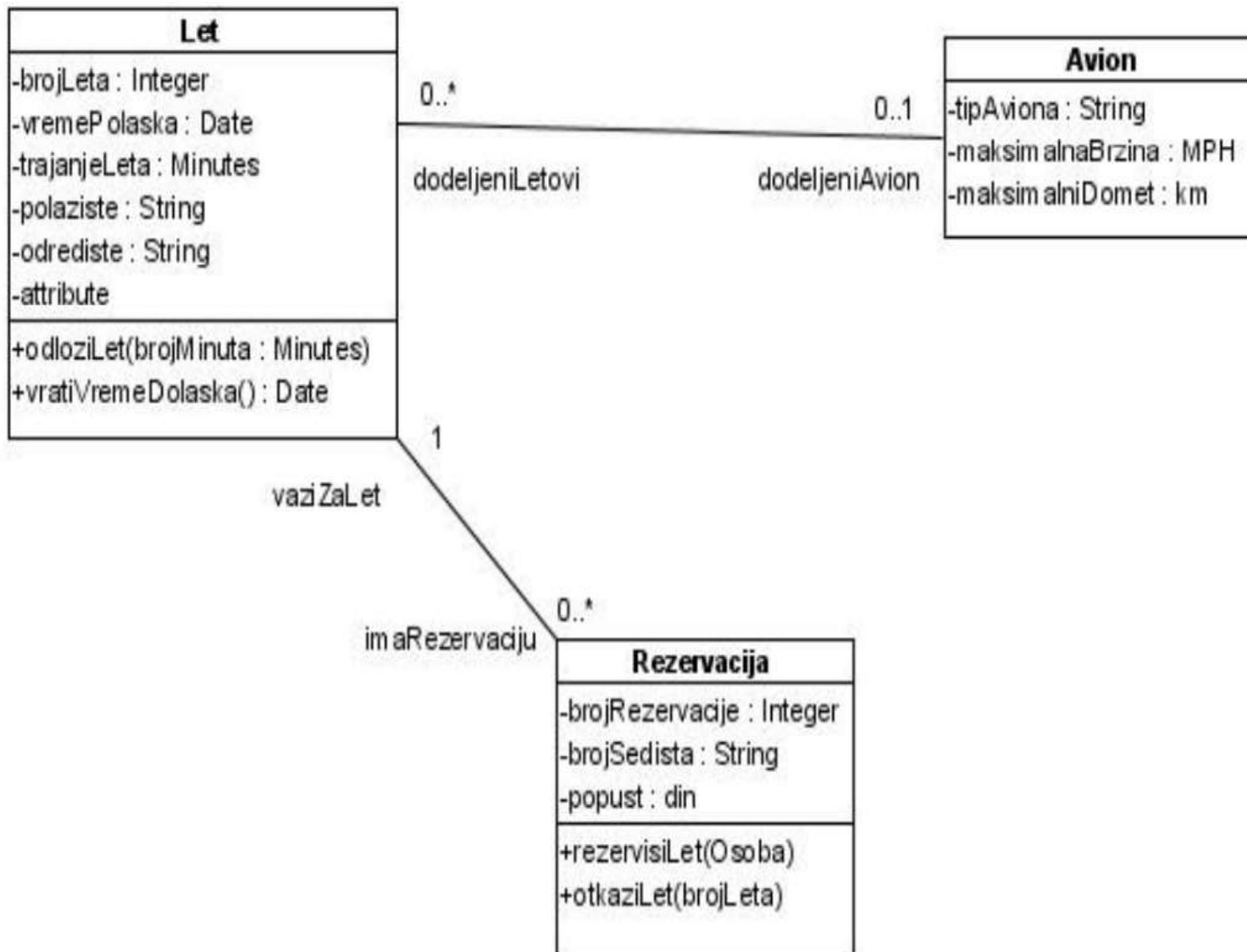
# Primer:

- Nacrtati dijagram klasa na jeziku UML sledećeg sistema klasa:  
Opisati rad lifta.
  - Lift ide u određenom smeru (gore ili dole), nalazi se na određenom spratu i ima svoj pravac kretanja.
  - Može da se kreće, zaustavi i dobije informaciju o statusu.
  - Postoji i kontroler liftova na određenom spratu koji pamti ID sprata, svoju poziciju i smer lifta. U okviru kontrolera postoji određeni broj dugmadi, koja mogu da svetle. Može im se zadati da svetle, da prestanu da svetle ili pročitati status.
  - Dugme može da bude za rad sa liftom, kada pamti broj sprata ili za rad sa spratom kada se pamti broj sprata i smer lifta. Kontroler lifta upravlja i vratima tako što može da ih otvara ili zatvara. Vrata pamte da li su otvorena ili zatvorena.

# Primer:



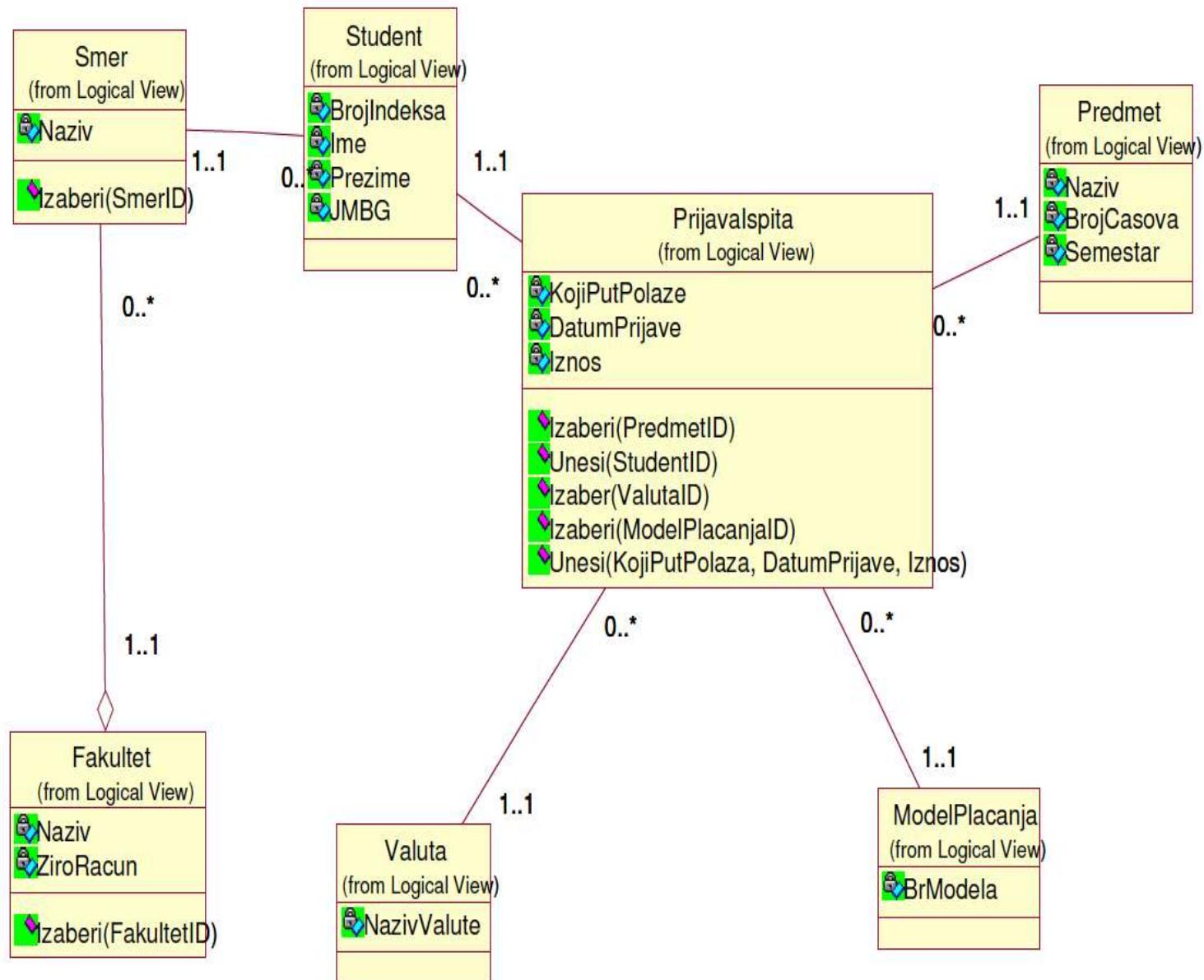
# Primer avio kompanije:



# DK – Primer 1

## Prijavljanje ispita

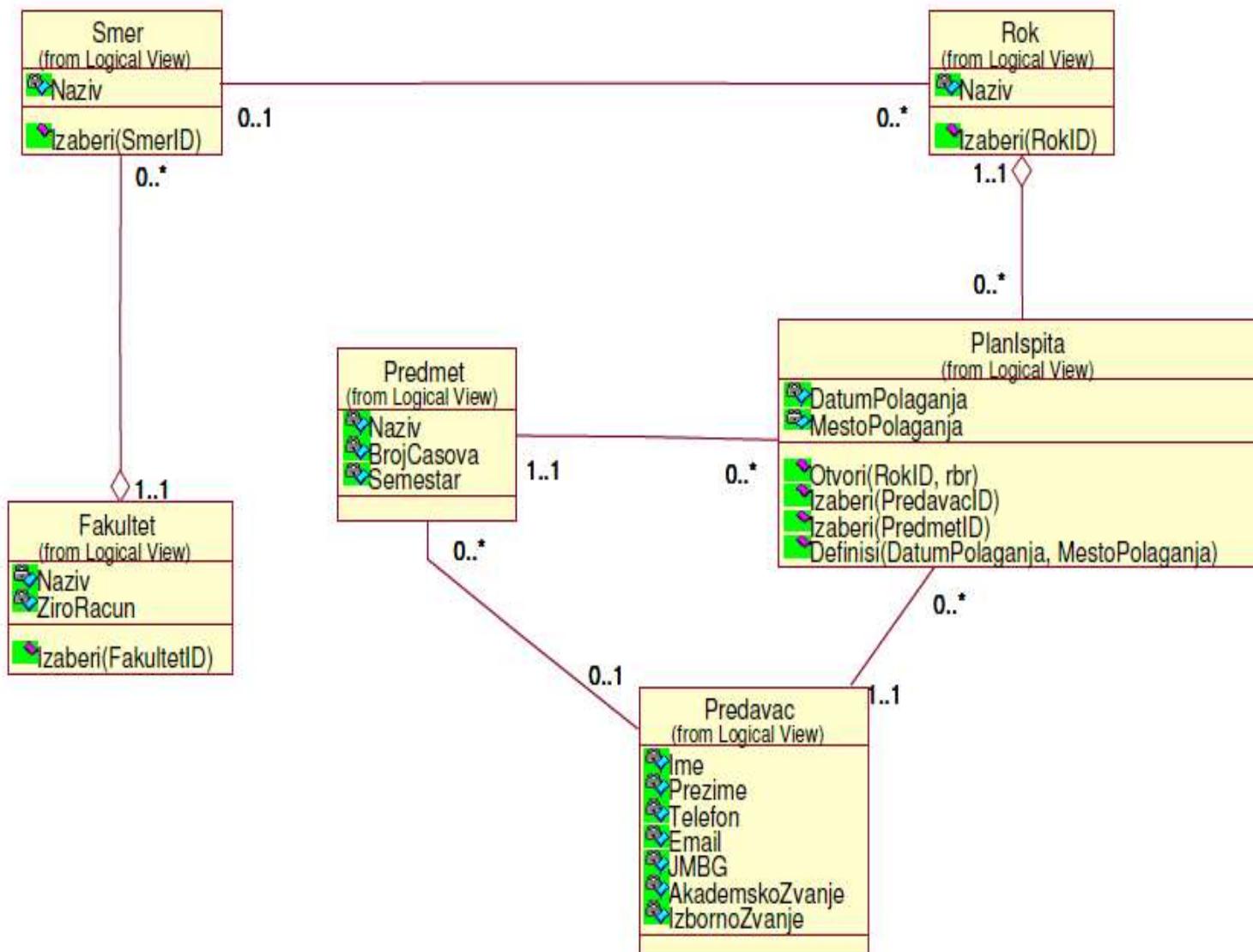
- Dijagram klase Prijavljanje ispita se sastoji iz klasa i veza izmedu njih. Klasa predstavlja složeni tip, kolekciju, struturu koja se sastoji od više atributa (podata clanova) i operacija (metoda, funkcija clanica).



# DK – Primer 2

## Izrada plana polaganja ispita

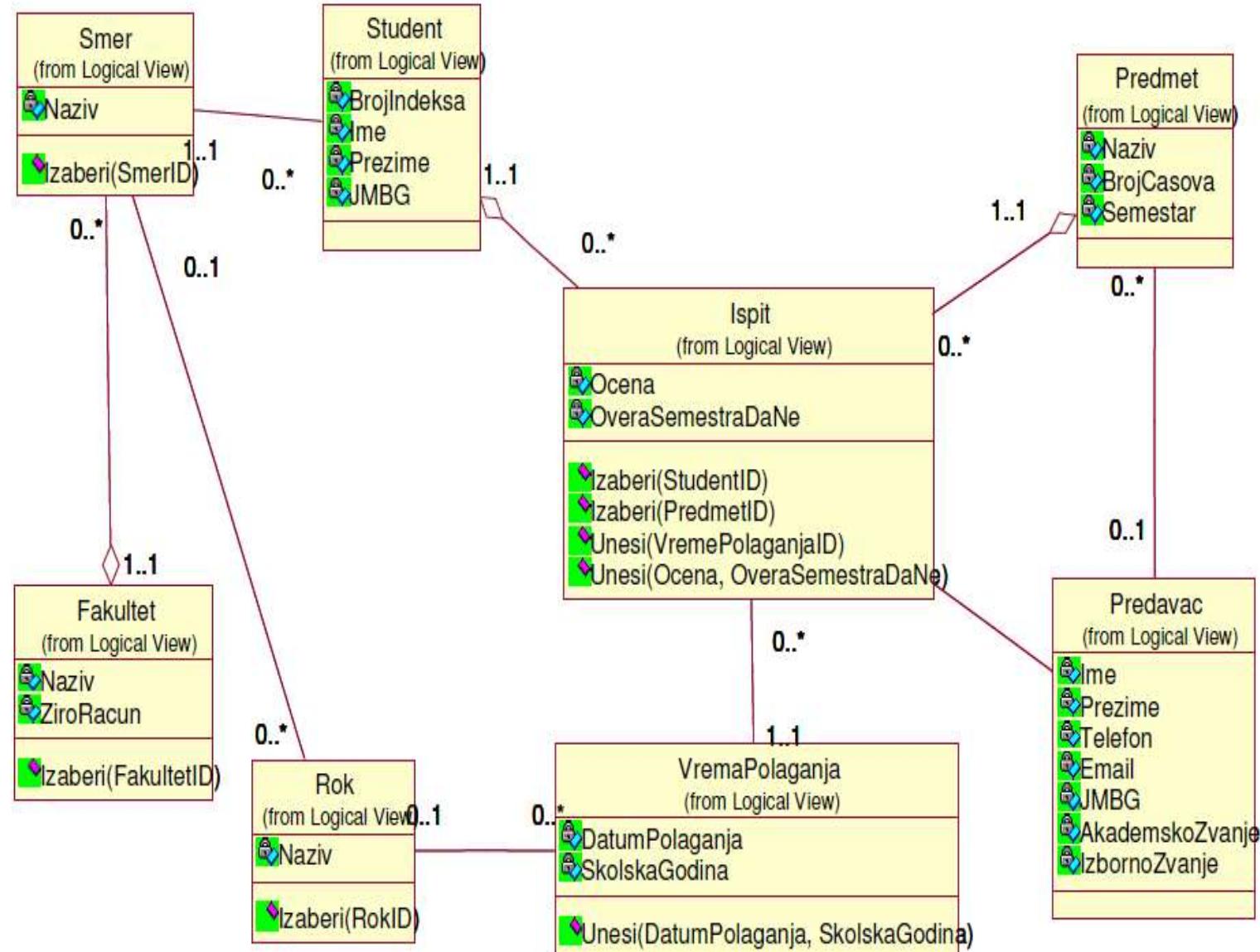
- Dijagram klase Izrada plana polaganja ispita se sastoji iz klasa i veza izmedu njih. Klasa predstavlja složeni tip, kolekciju, struturu koja se sastoji od više atributa (podata clanova) i operacija (metoda, funkcija clanica). Na sledecoj slici prikazan je dijagram klasa za poslove Izrada plana polaganja ispita.



# DK – Primer 3

## Polaganje ispita

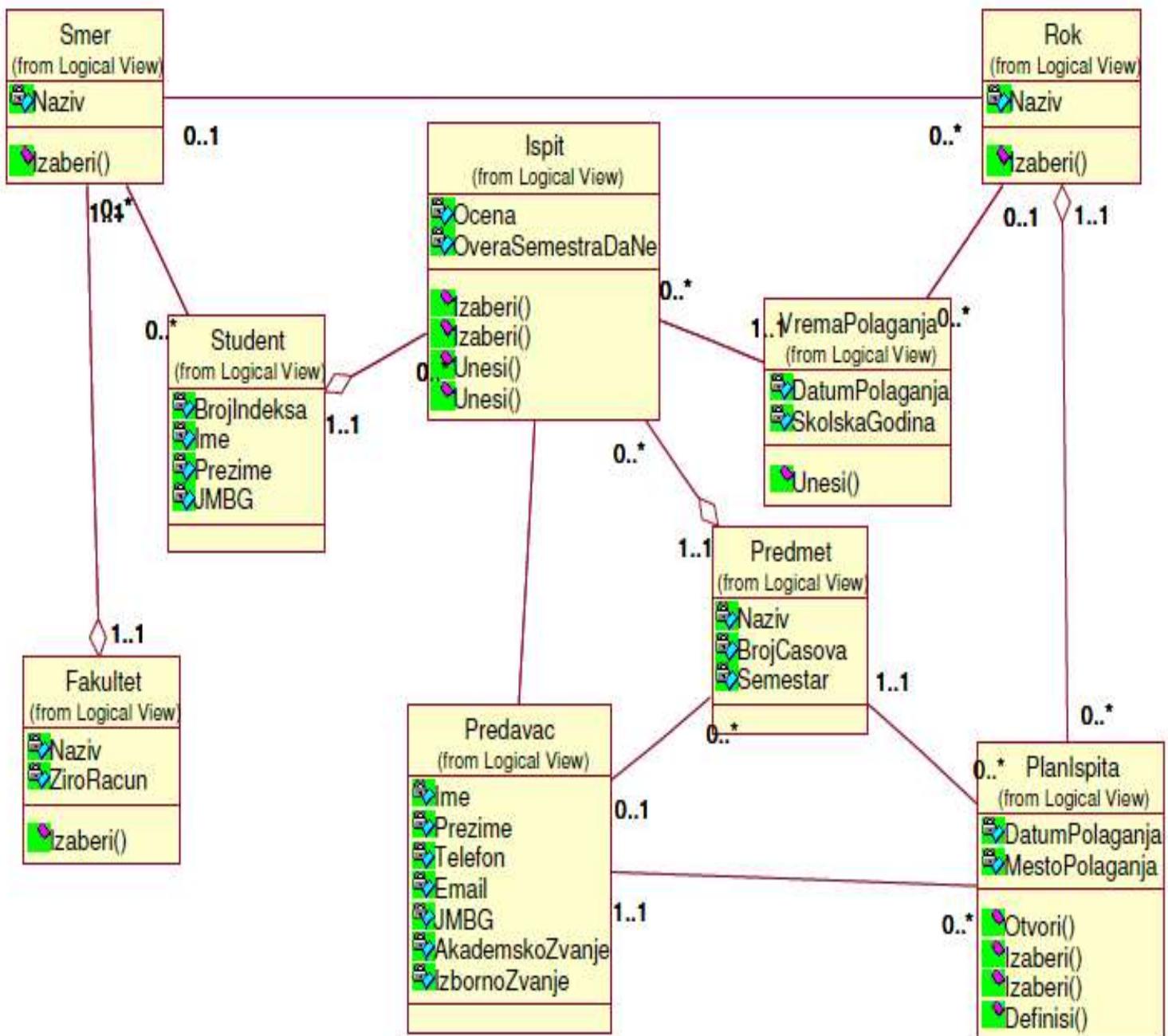
- Prikazace se šta slučaj upotrebe Polaganje ispita treba da radi u realnom vremenu da bi obavio svoju funkciju.
- Vremenskim redosledom poruka u sekvenčijalnom dijagramu opisace logiku odvijanja poslova Polaganje ispita za slučaj upotrebe Izrada plana polaganja ispita.



# DK – Primer 4

## Analiza ispita

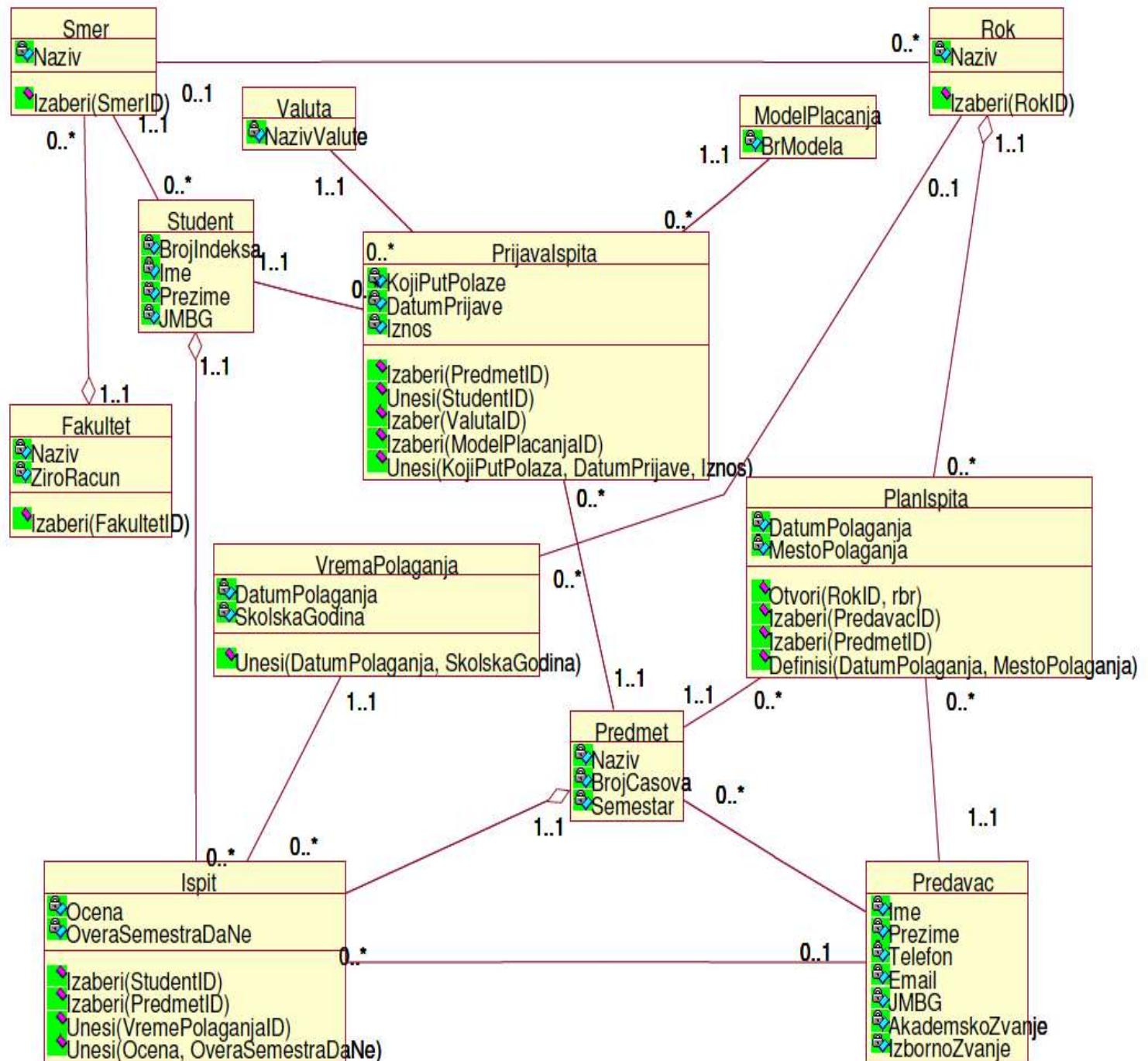
- Dijagram klase Analiza ispita se sastoji iz klasa i veza izmedu njih. Klasa predstavlja slozeni tip, kolekciju, strukturu koja se sastoji od više atributa (podata clanova) i operacija (metoda, funkcija clanica). Na sledecoj slici prikazan je dijagram klasa za poslove Analiza ispita.



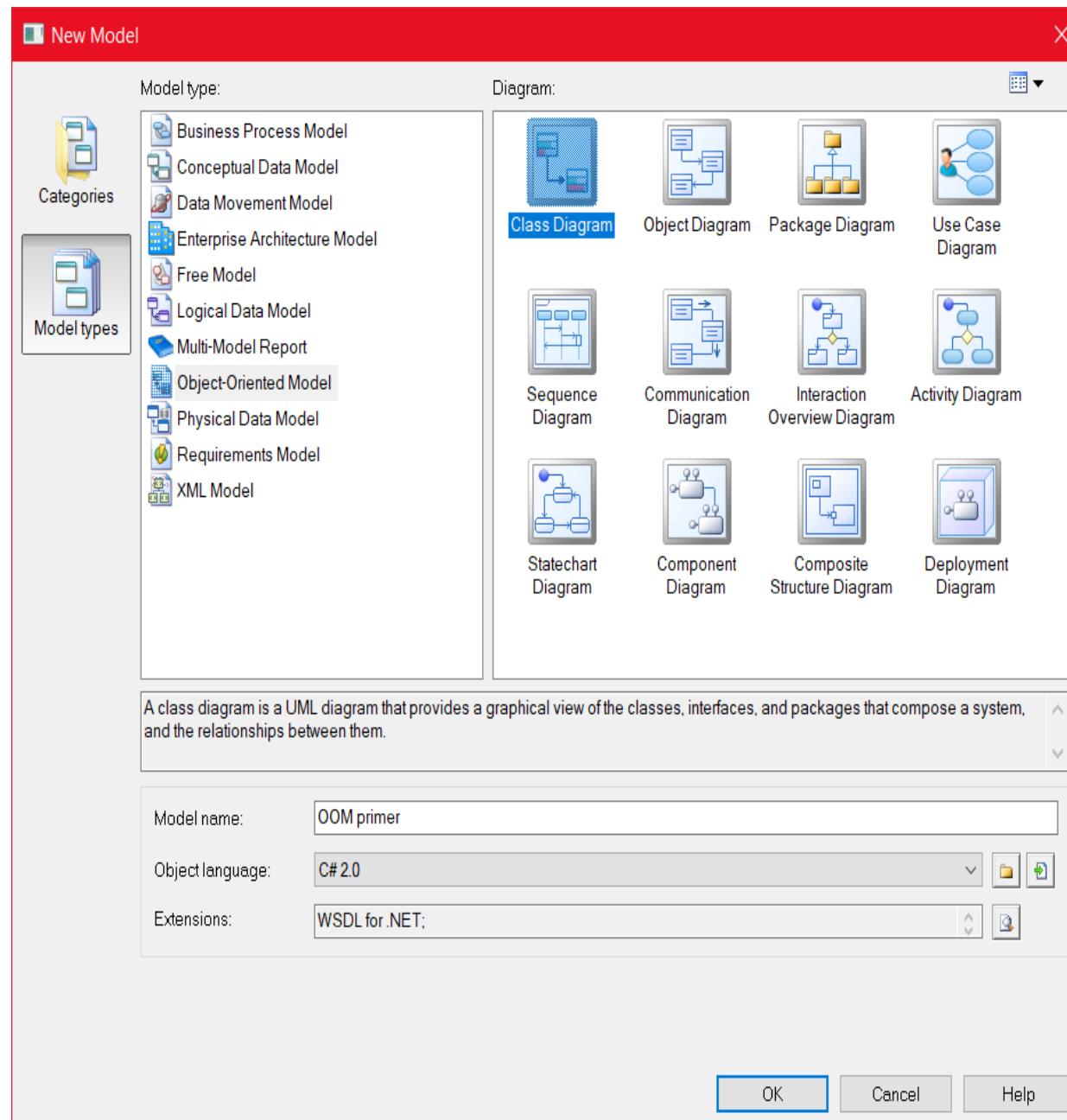
# DK – Primer 5

## Praćenje ispita

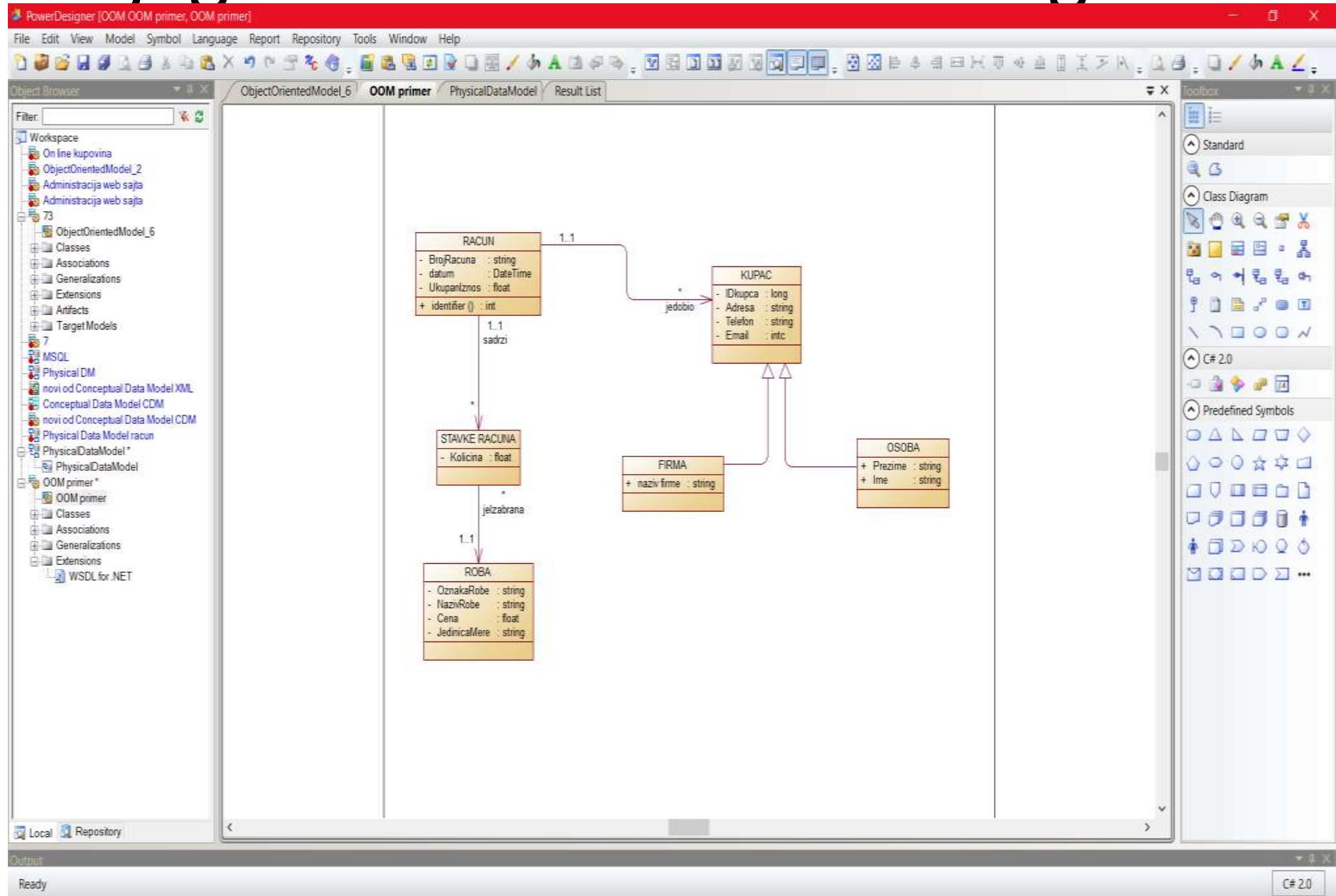
- Dijagram klase se sastoji iz klasa i veza izmedu njih. Naziva se još i dijagram staticke strukture. Klasa predstavlja složeni tip, kolekciju, struturu koja se sastoji od više atributa (podata clanova) i operacija (metoda, funkcija clanica). Na sledecoj slici prikazan je potpuni dijagram klasa za poslove pracenja ispita.



# Dijagram klasa u Power Designer-u

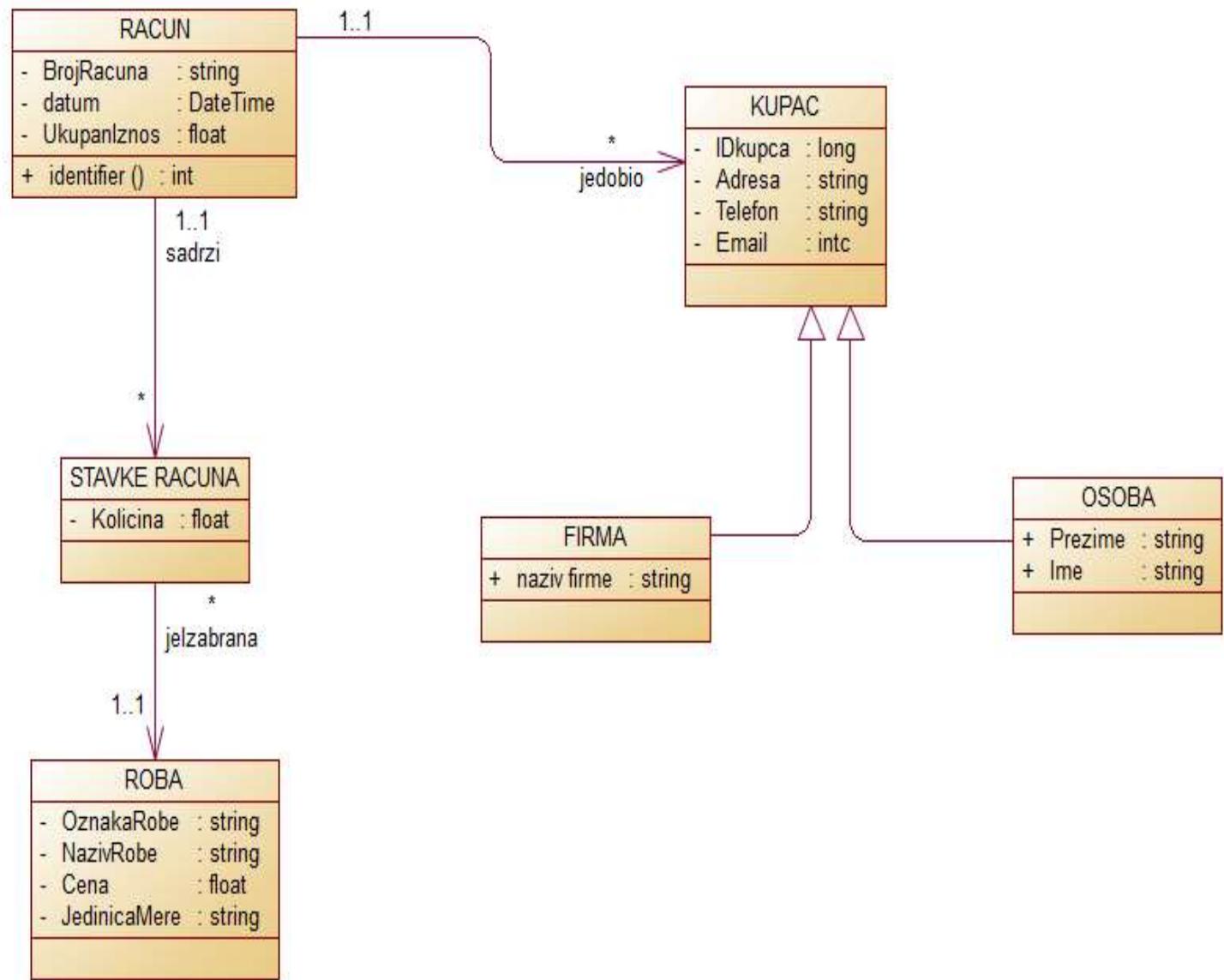
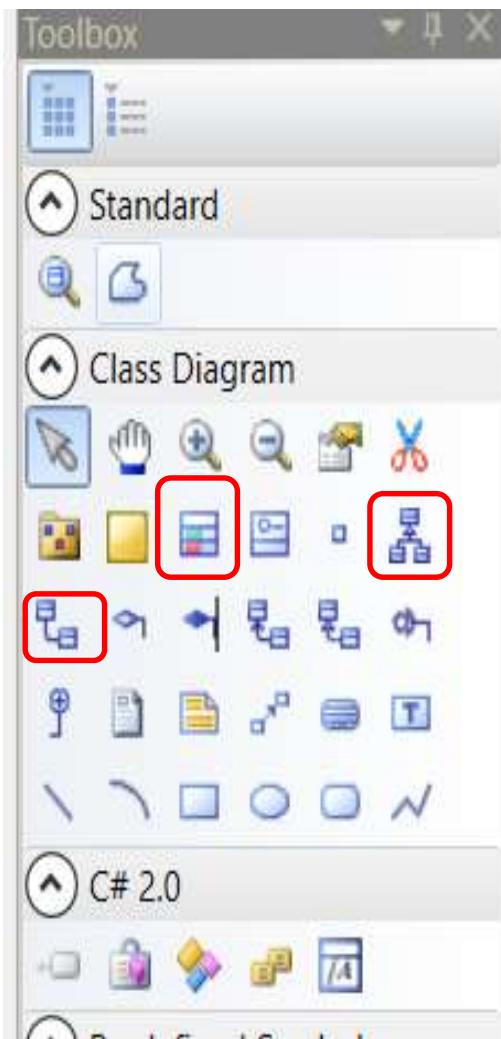


# Dijagram klasa u Power Designer-u



# Dijagram klasa u Power Designer-u

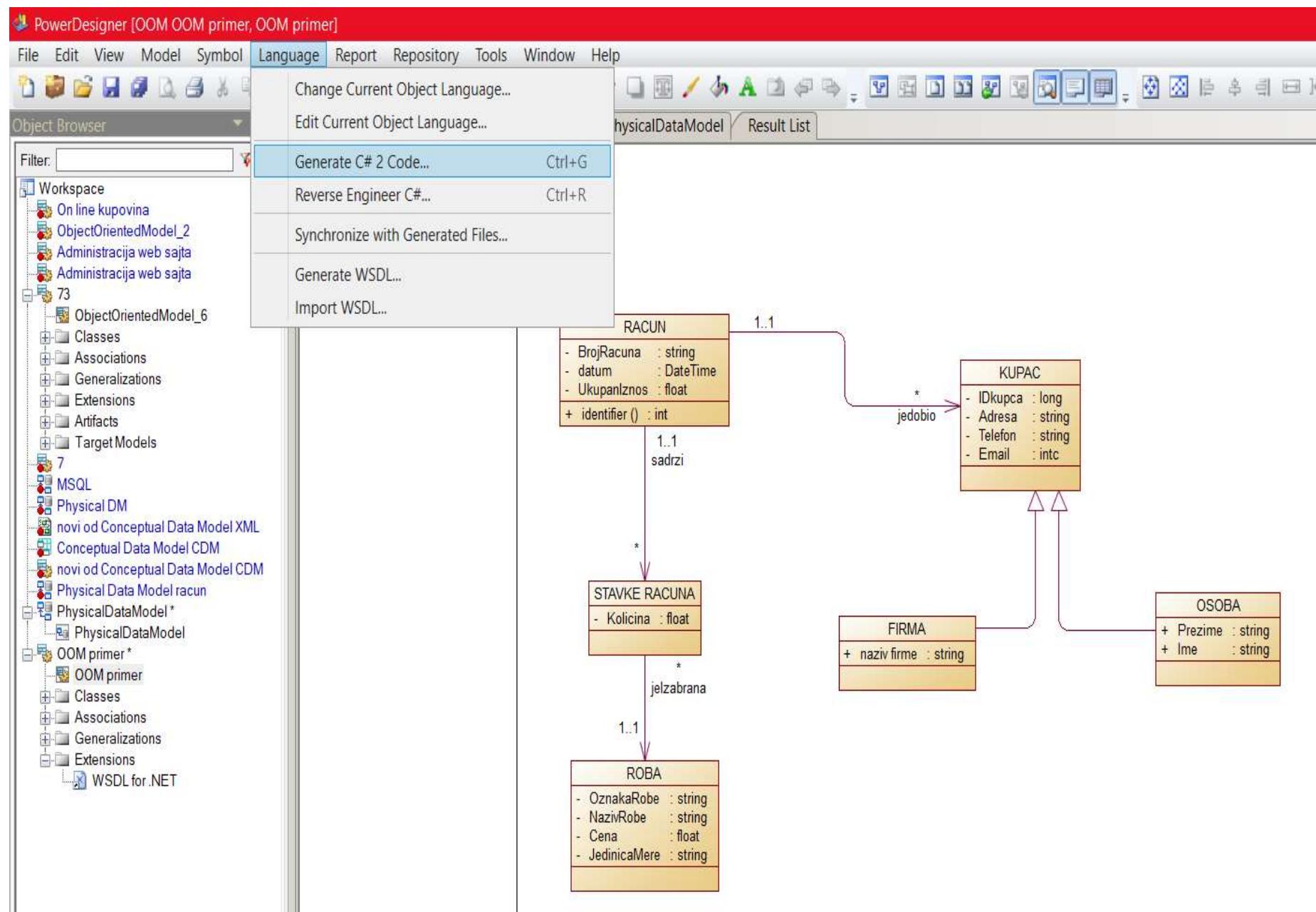
## Osnovni elementi



## Podsetnik...

- Atributi i metode mogu se označiti na sledeći način:
  - Privatna – nije vidljiva izvan klase -
  - Javna – vidljiva je svima +
  - Zaštićena – vidljiva je samo podklasama određene klase #

# Generisanje koda u C#...



# Generisanje koda u C#... (podešavanja po karticama)

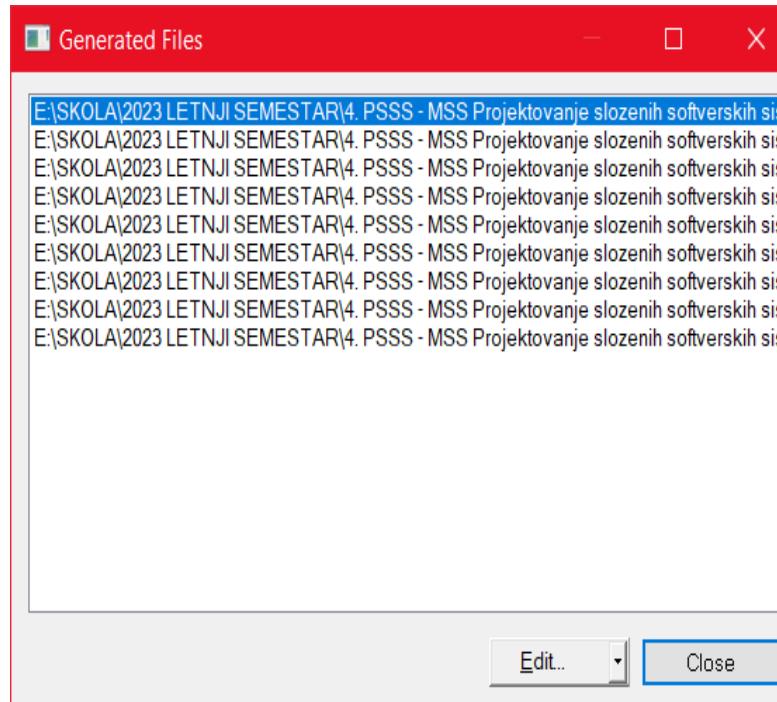
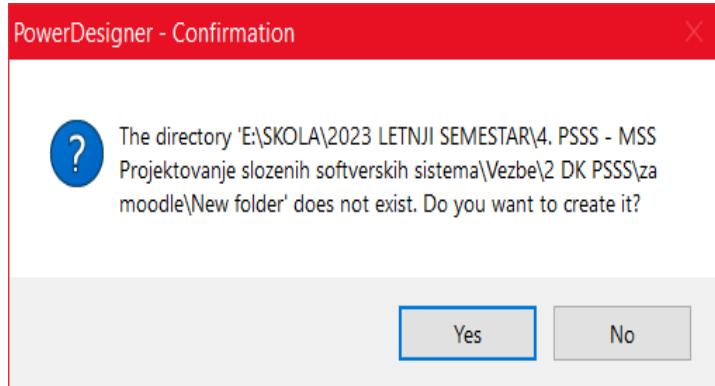
The image displays four windows of the 'Generation' dialog box, each showing a different step in the configuration process:

- Step 1 (Left):** Shows the 'Targets' tab selected. It has a single checked option: 'WSDL' under 'WSDL for .NET'. Buttons at the bottom include OK, Cancel, Apply, and Help.
- Step 2 (Middle Left):** Shows the 'Options' tab selected. A table lists various options with their values:

Target	Option	Value
C# 2	Generate object IDs as documentation tags	false
C# 2	Sort class members primarily by	Visibility
C# 2	Class members type sort	Properties - Methods - Fields
C# 2	Class members visibility sort	Private - Public
C# 2	Generate Visual Studio Project Files	true
C# 2	Generate Assembly Info File	true
C# 2	Generate Visual Studio Solution File	true
C# 2	Visual Studio Version	Visual Studio 2008
C# 2	Generate Web Service C# code in .asmx file	true
C# 2	Generate default accessors for navigable associations	false

Buttons at the bottom include OK, Cancel, Apply, and Help.
- Step 3 (Middle Right):** Shows the 'Generated Files' tab selected. It displays a tree view of generated files: Firma.cs, Kupac.cs, DOM\_primer.csproj, DOM\_primer.sln, Osoba.cs, Properties, AssemblyInfo.cs (checked), Racun.cs, Roba.cs, and StavkeRacuna.cs. Buttons at the bottom include OK, Cancel, Apply, and Help.
- Step 4 (Bottom):** Shows the 'Tasks' tab selected. It lists three checked tasks:
  - WSDLDotNet: Generate Web Service proxy code
  - C# 2: Compile Source Files
  - C# 2: Open the solution in Visual StudioButtons at the bottom include OK, Cancel, Apply, and Help.

# Generisanje koda u C#...



Fajlovi u Folder Explorer-u

Name
Properties
Firma.cs
Kupac.cs
OOM_primer
OOM_primer.sln
Osoba.cs
Racun.cs
Roba.cs
StavkeRacuna.cs

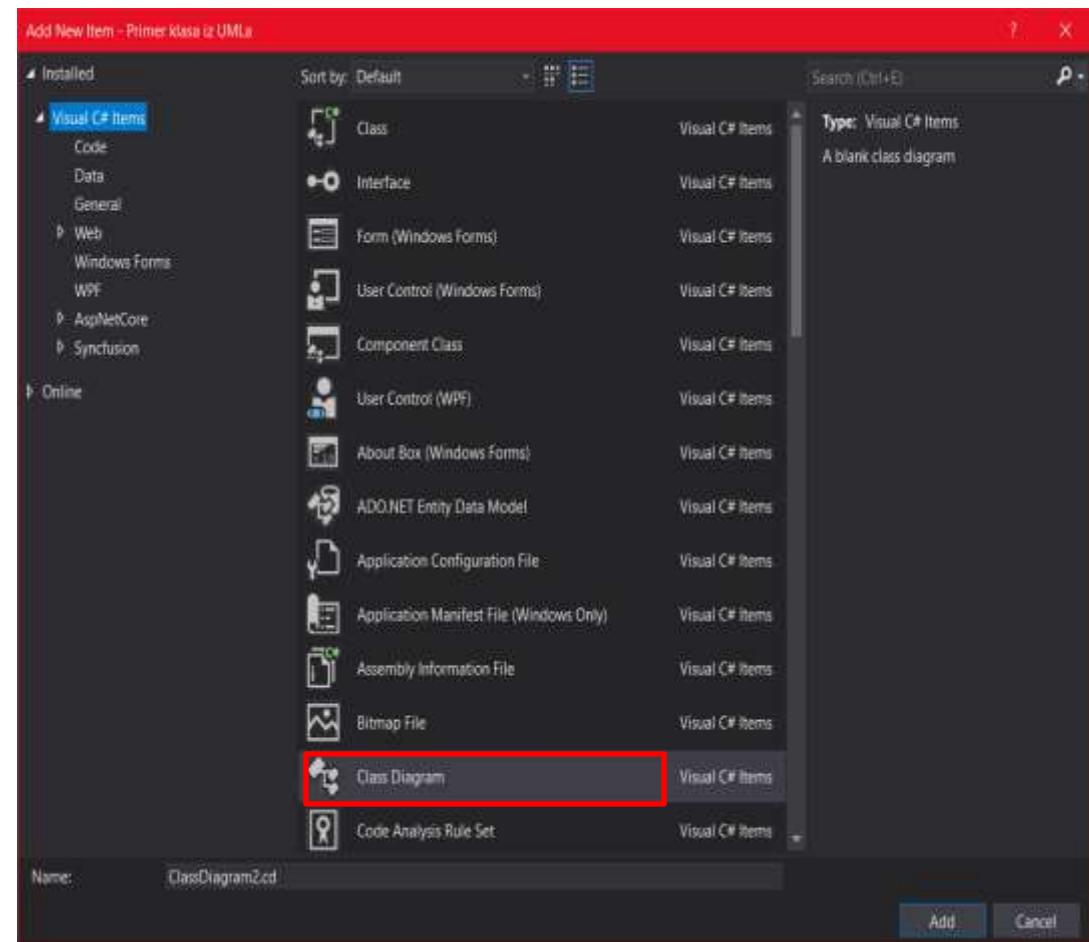
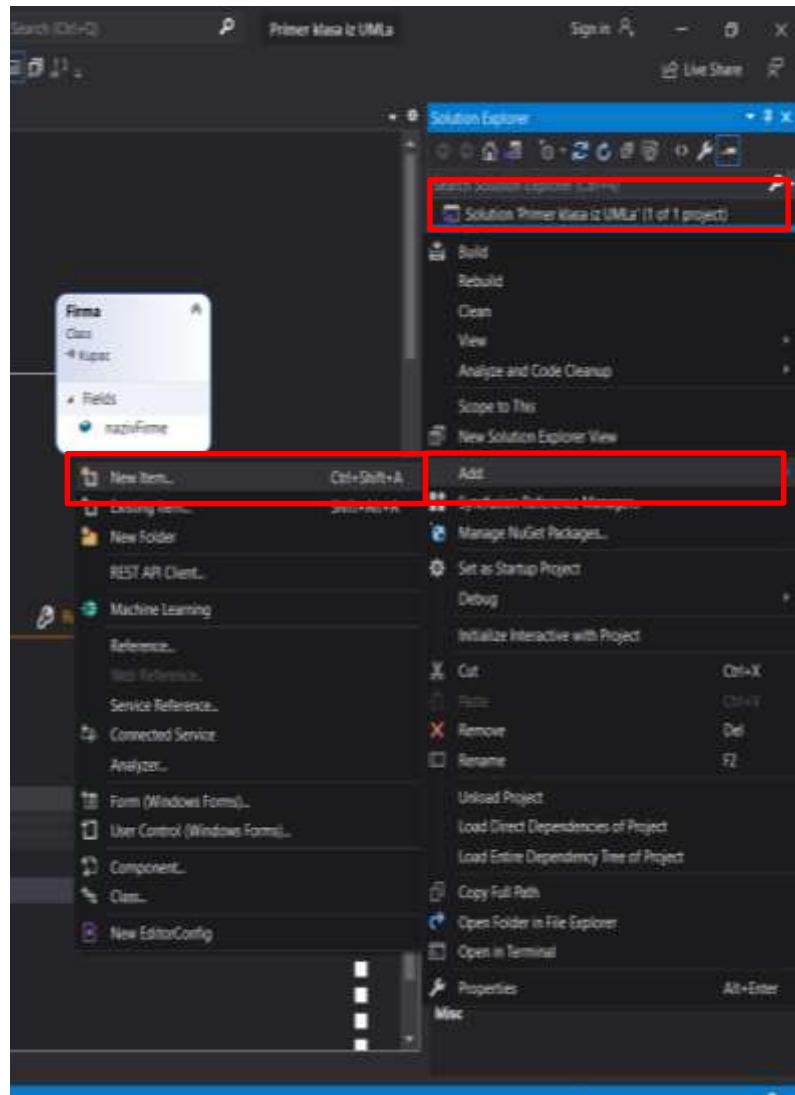
# C# kod u Visual Studio-u

The screenshot shows the Visual Studio IDE interface with the following details:

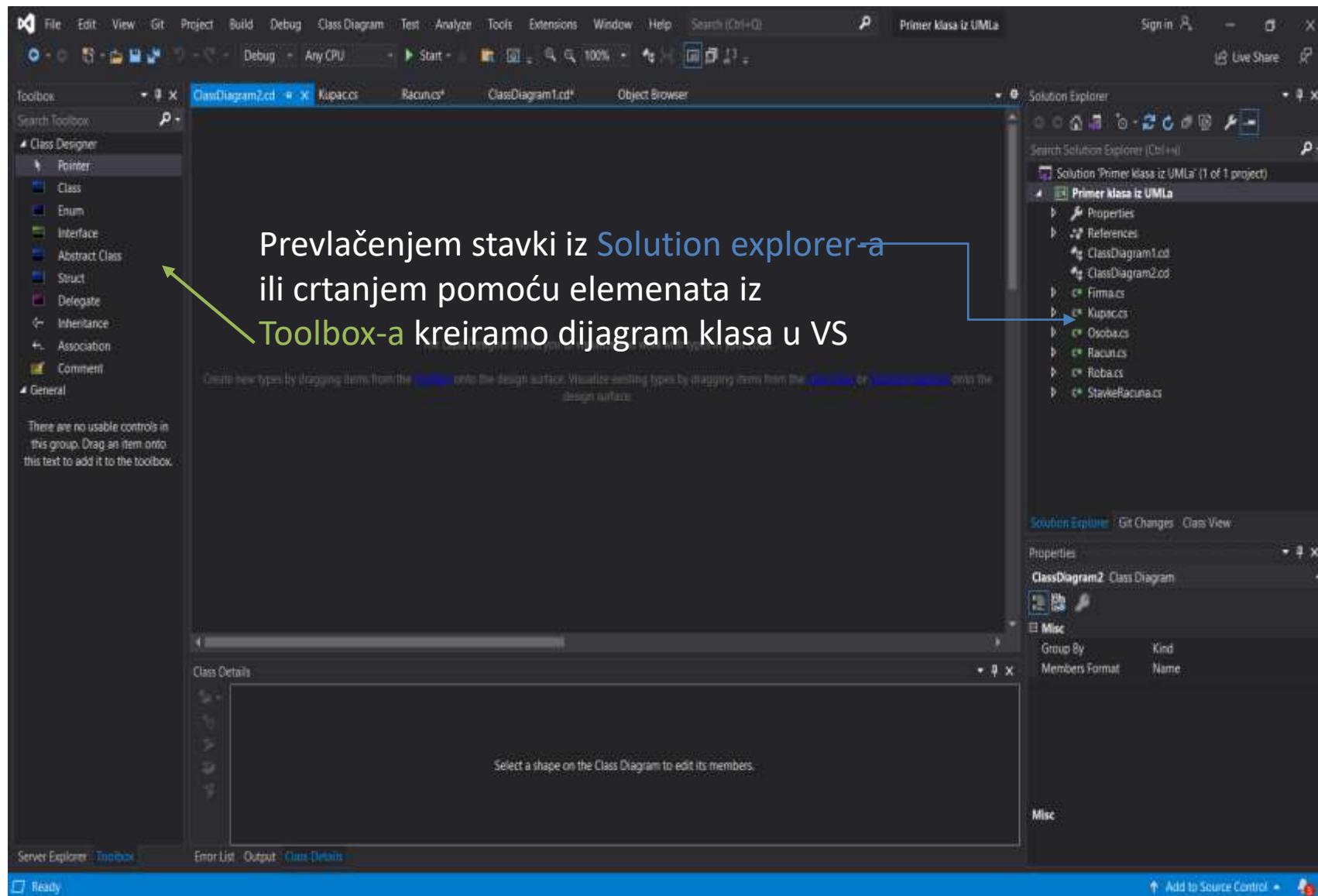
- Top Bar:** File, Edit, View, Git, Project, Build, Debug, Test, Analyze, Tools, Extensions, Window, Help, Search (Ctrl+Q), a magnifying glass icon, and the project name "OOM\_primer".
- Solution Explorer:** Shows the solution "OOM\_primer" with one project "OOM\_primer" containing files: AssemblyInfo.cs, Firma.cs, Kupac.cs, Osoba.cs, Racun.cs, Roba.cs, and StavkeRacuna.cs. It also lists references to System, System.Core, System.Data, and System.Xml.
- Toolbox:** Shows a general category with a note: "There are no usable controls in this group. Drag an item onto this text to add it to the toolbox."
- Code Editor:** Displays the content of the Kupac.cs file:

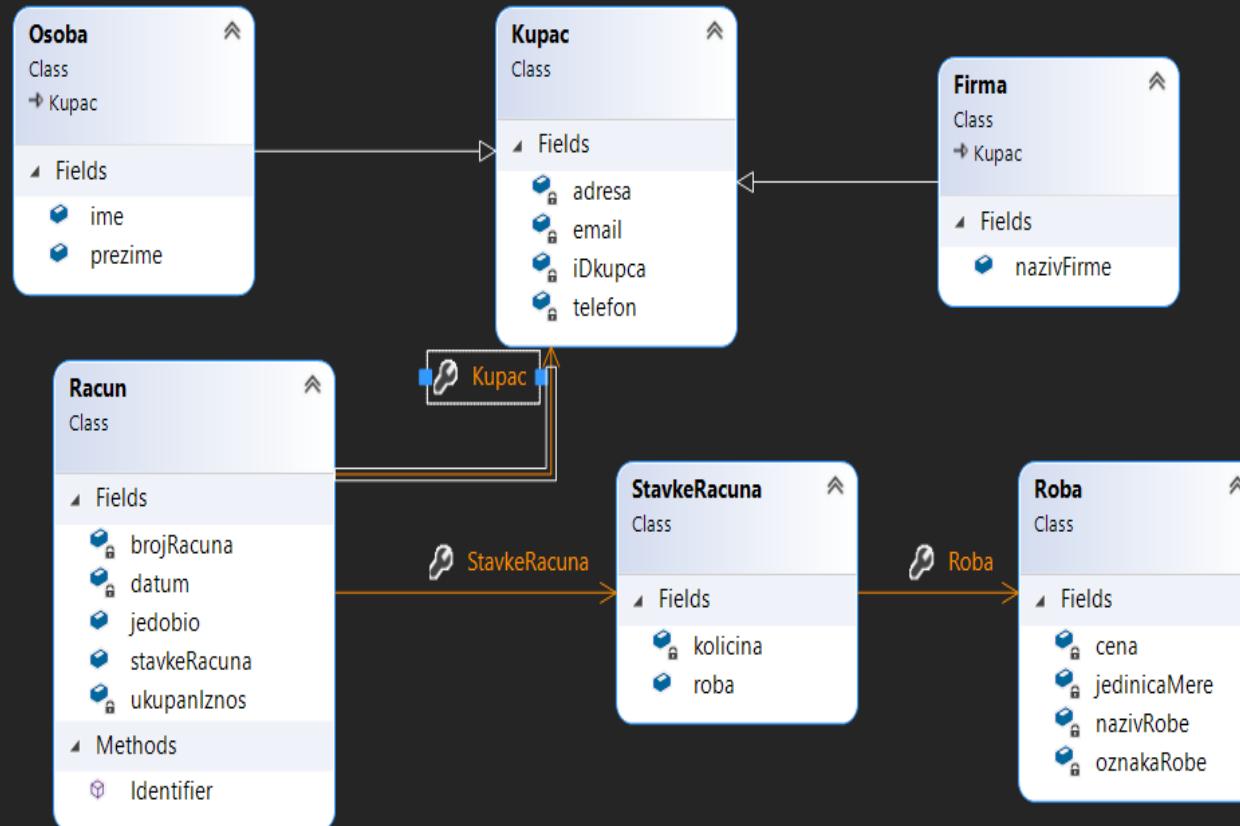
```
1 // File: Kupac.cs
2 // Author: DraganaDesktop
3 // Created: Sunday, February 12, 2023 11:02:36 PM
4 // Purpose: Definition of Class Kupac
5
6 using System;
7
8 public class Kupac
9 {
10     private long iDkupca;
11     private string adresa;
12     private string telefon;
13     private intc email;
14 }
15 }
```
- Status Bar:** Shows "100 %", "No issues found", and "Ln: 1 Ch: 1 SPC CRLF".
- Output Tab:** Shows "Output" and "Show output from:" dropdown.
- Bottom Status Bar:** Shows "Ready" and "Add to Source Control" with a "2" notification.

# Kreiranje dijagrama klase u Visual Studio-u



# Kreiranje dijagrama klasa u Visual Studio-u

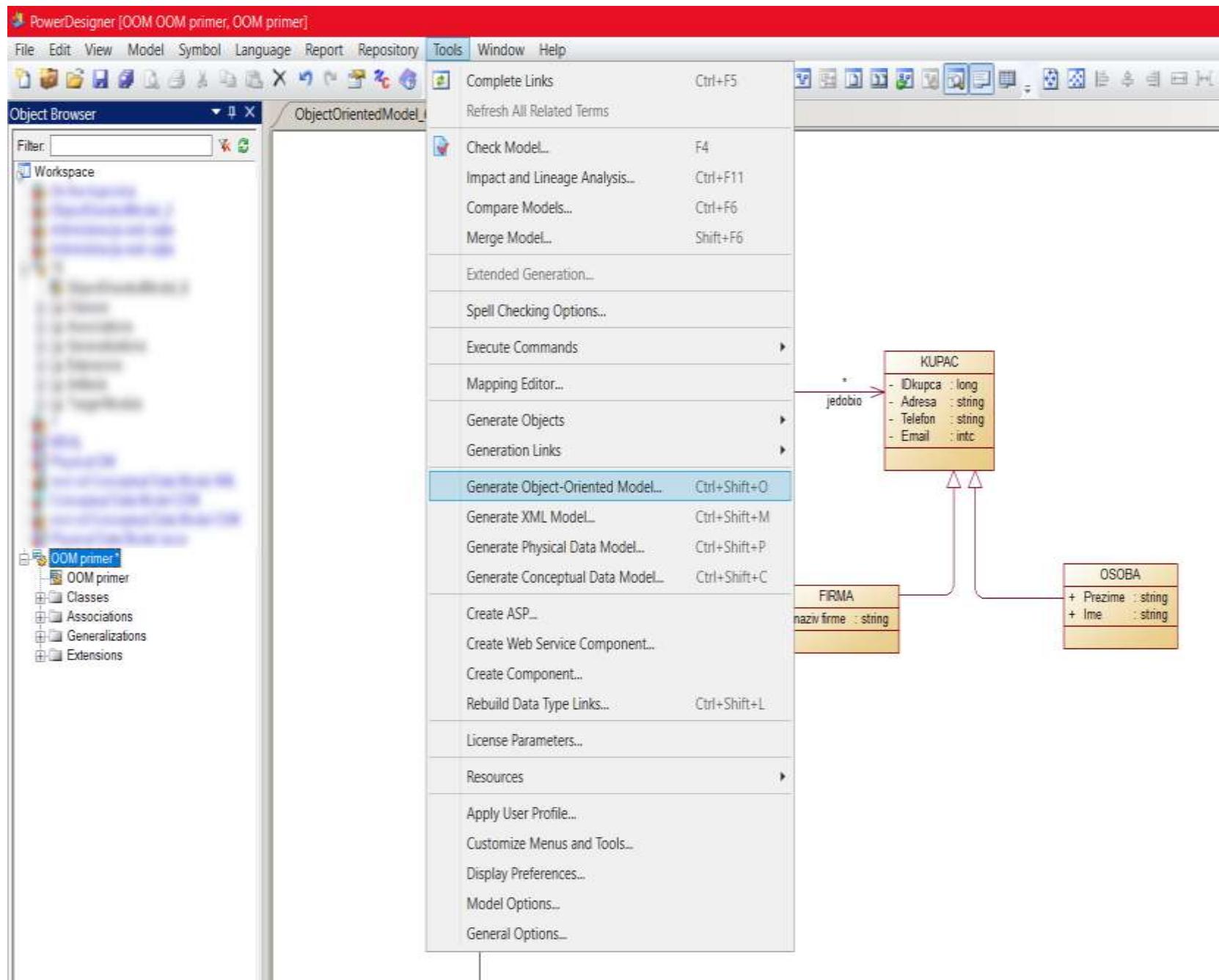




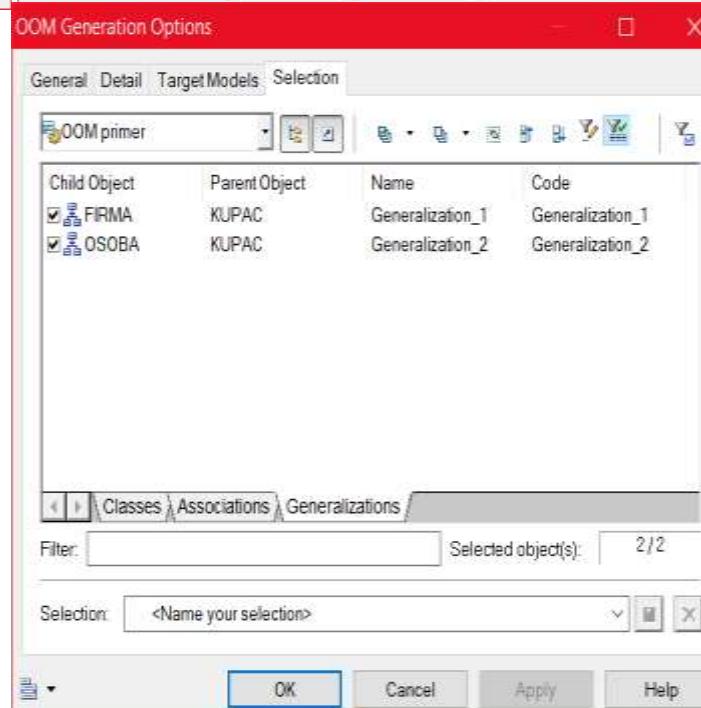
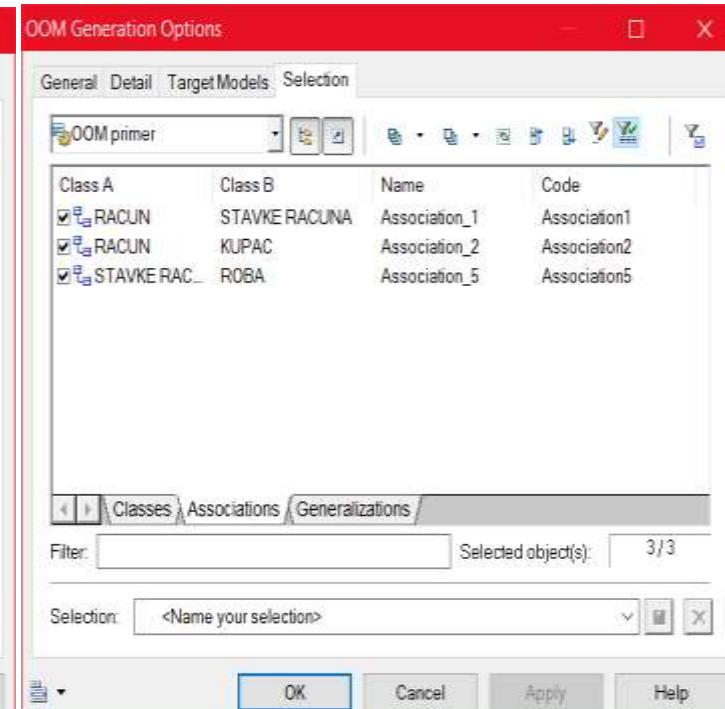
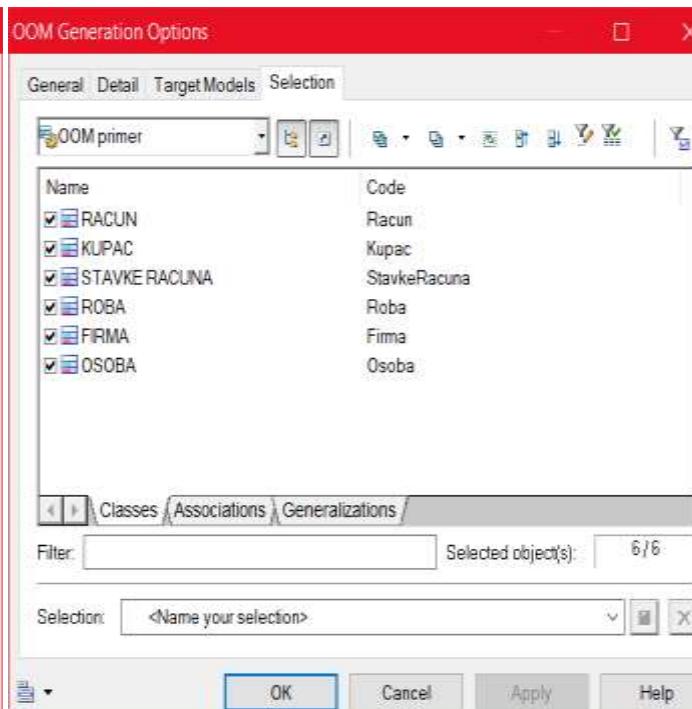
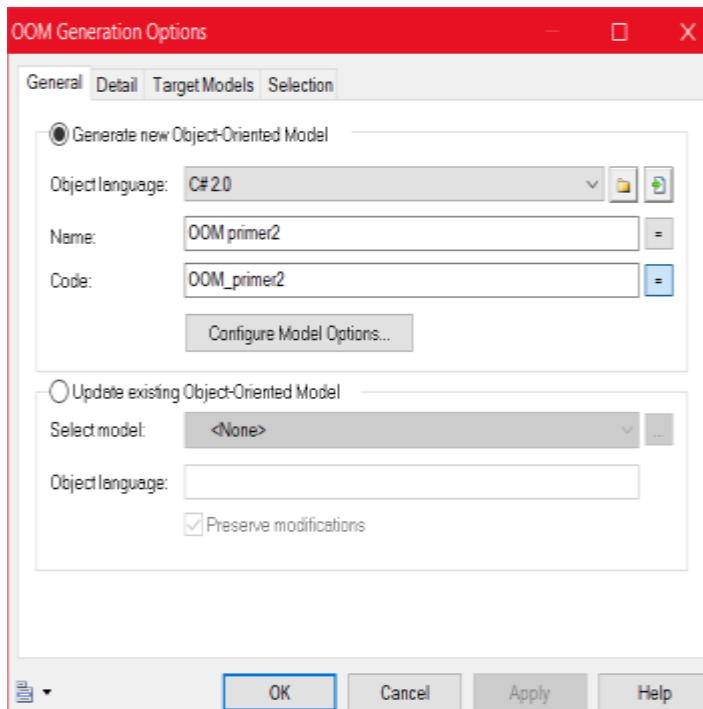
## Class Details - Racun

Name	Type	Modifier	Summary	Hide
Kupac	Kupac	public		<input type="checkbox"/>
StavkeRacuna	StavkeRacuna	public		<input type="checkbox"/>
<add property>				<input type="checkbox"/>
<b>Fields</b>				
brojRacuna	char	private		<input type="checkbox"/>
datum	DateTime	private		<input type="checkbox"/>

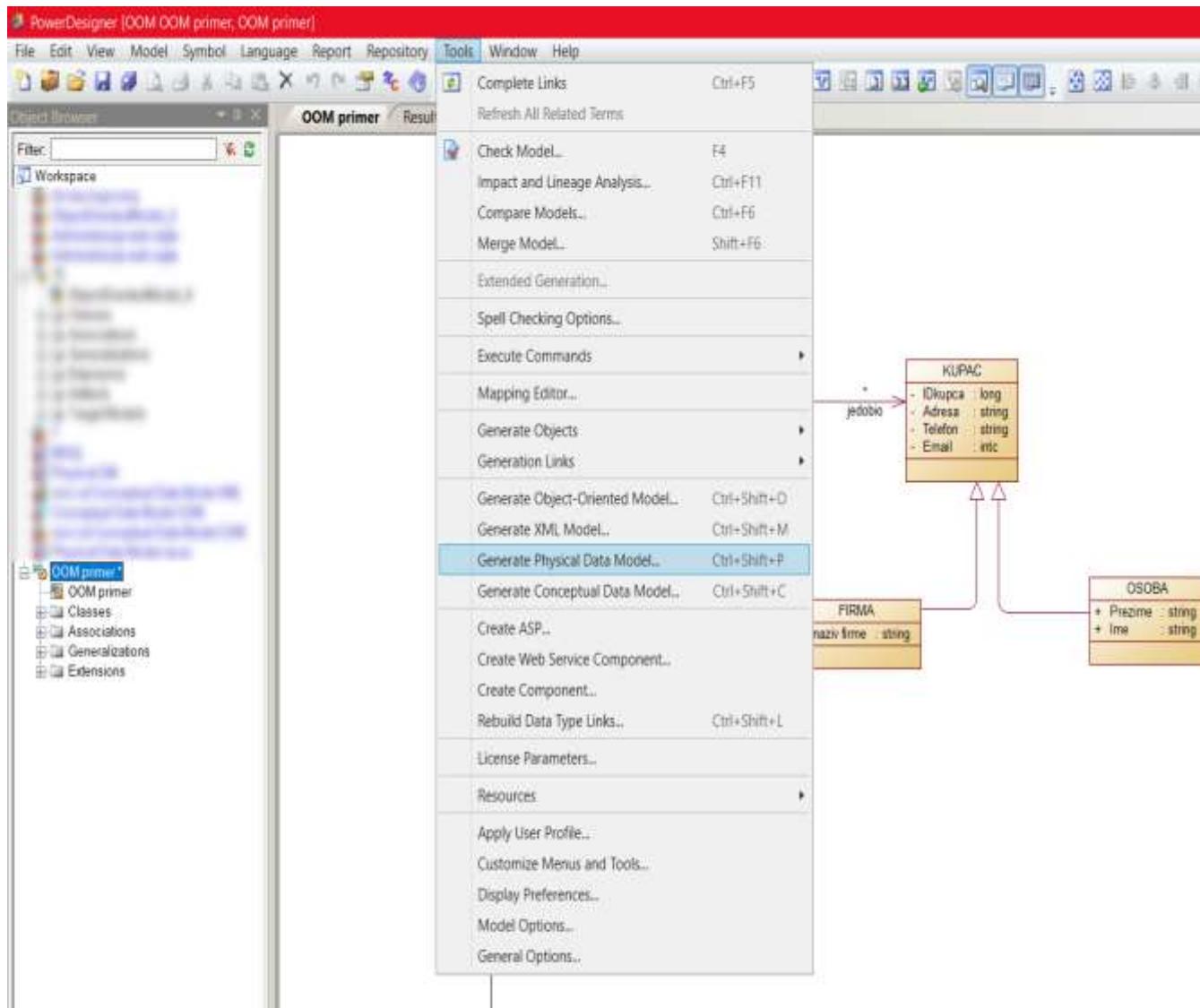
# Generisanje OO Modela (ako ga već niste kreirali kao takvog)



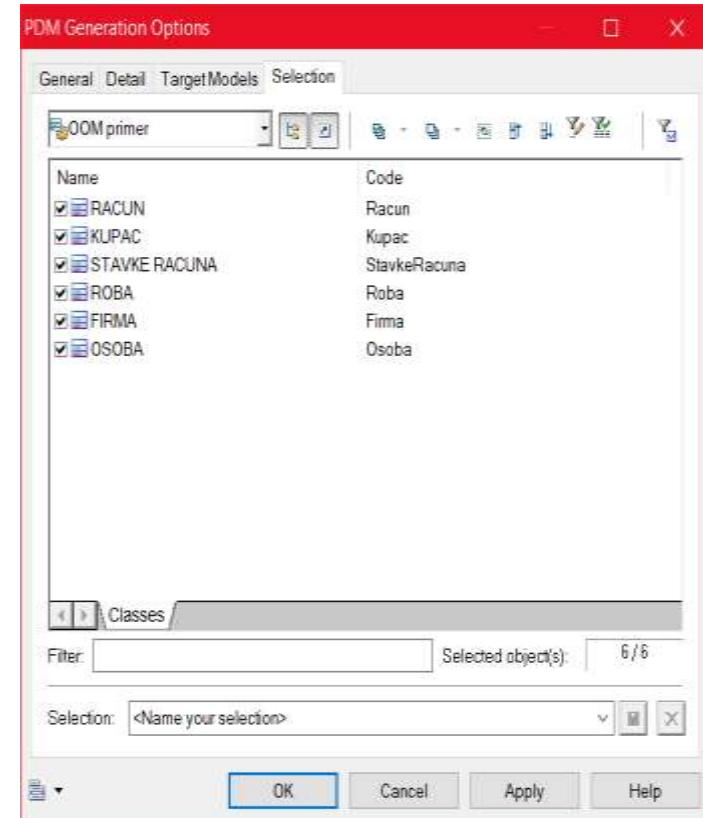
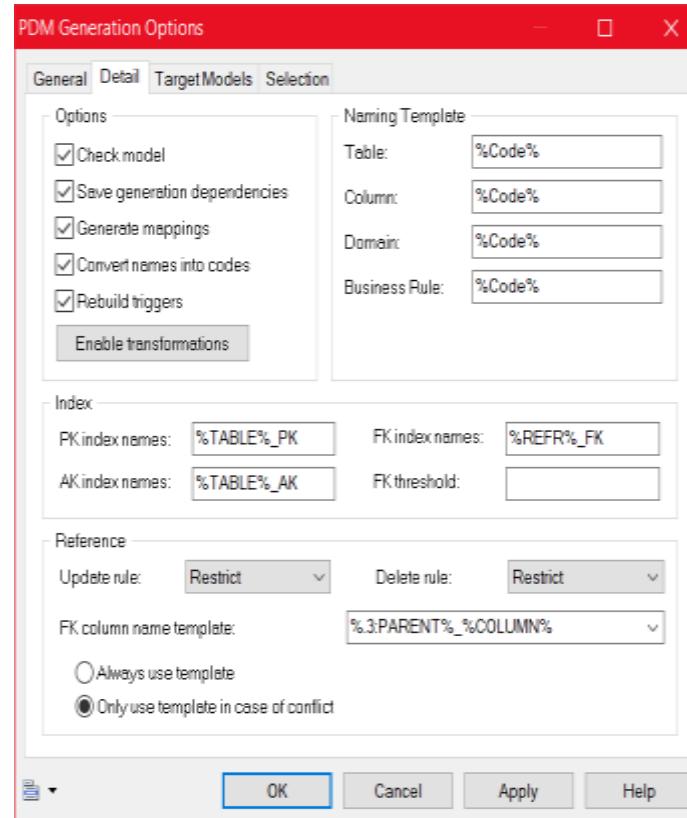
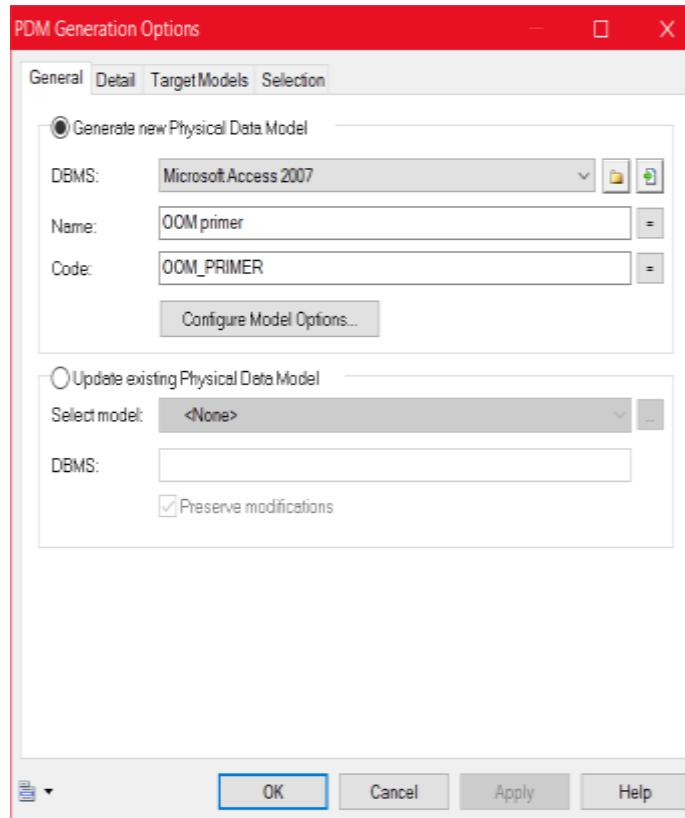
# Generisanje OO Modela (ako ga već niste kreirali kao takvog)



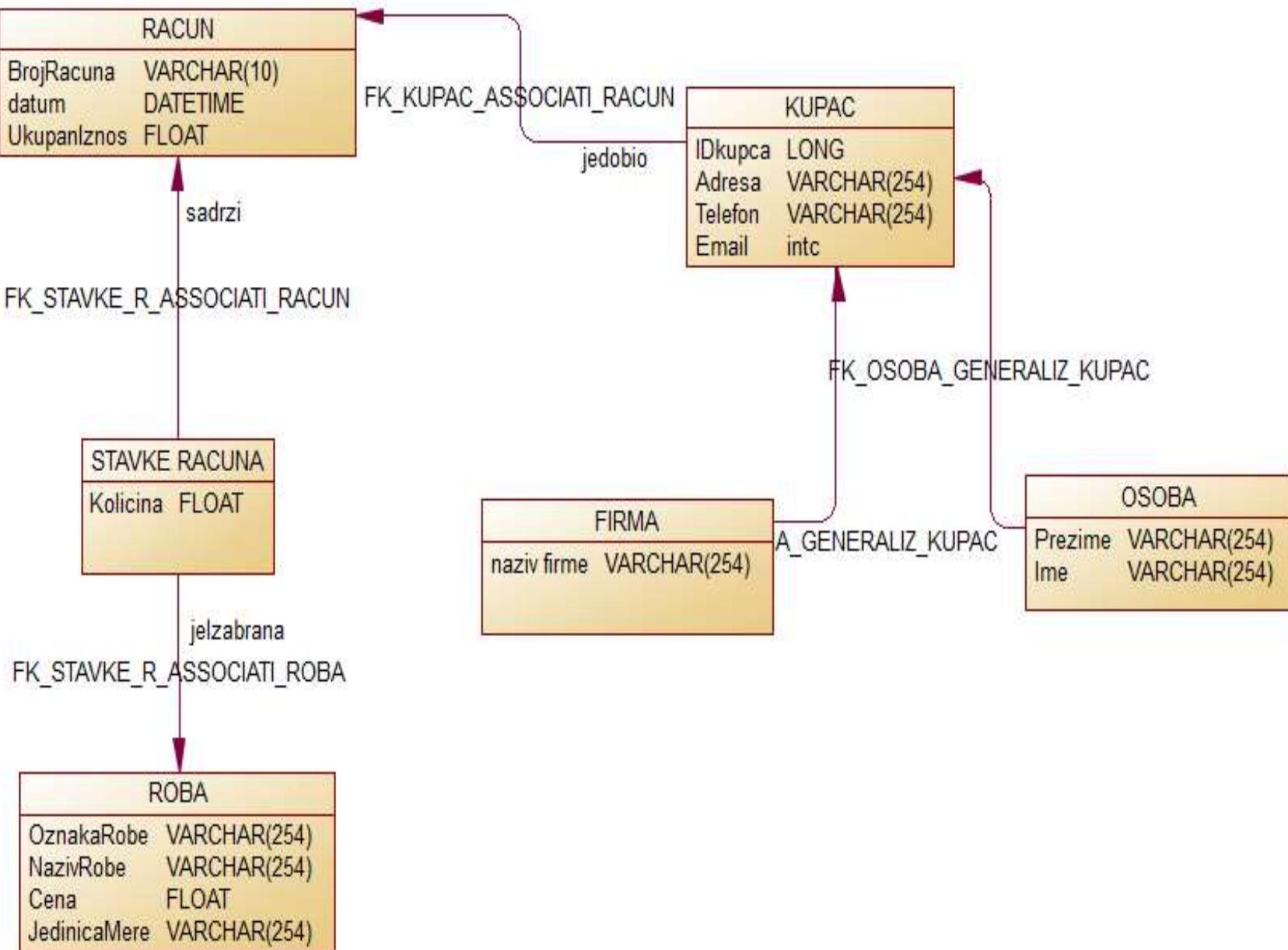
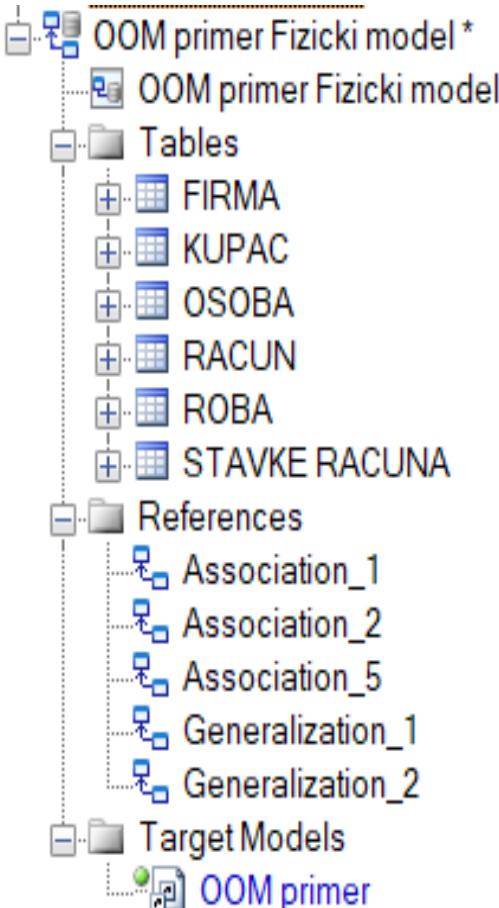
# Generisanje fizičkog modela (Physical DM)



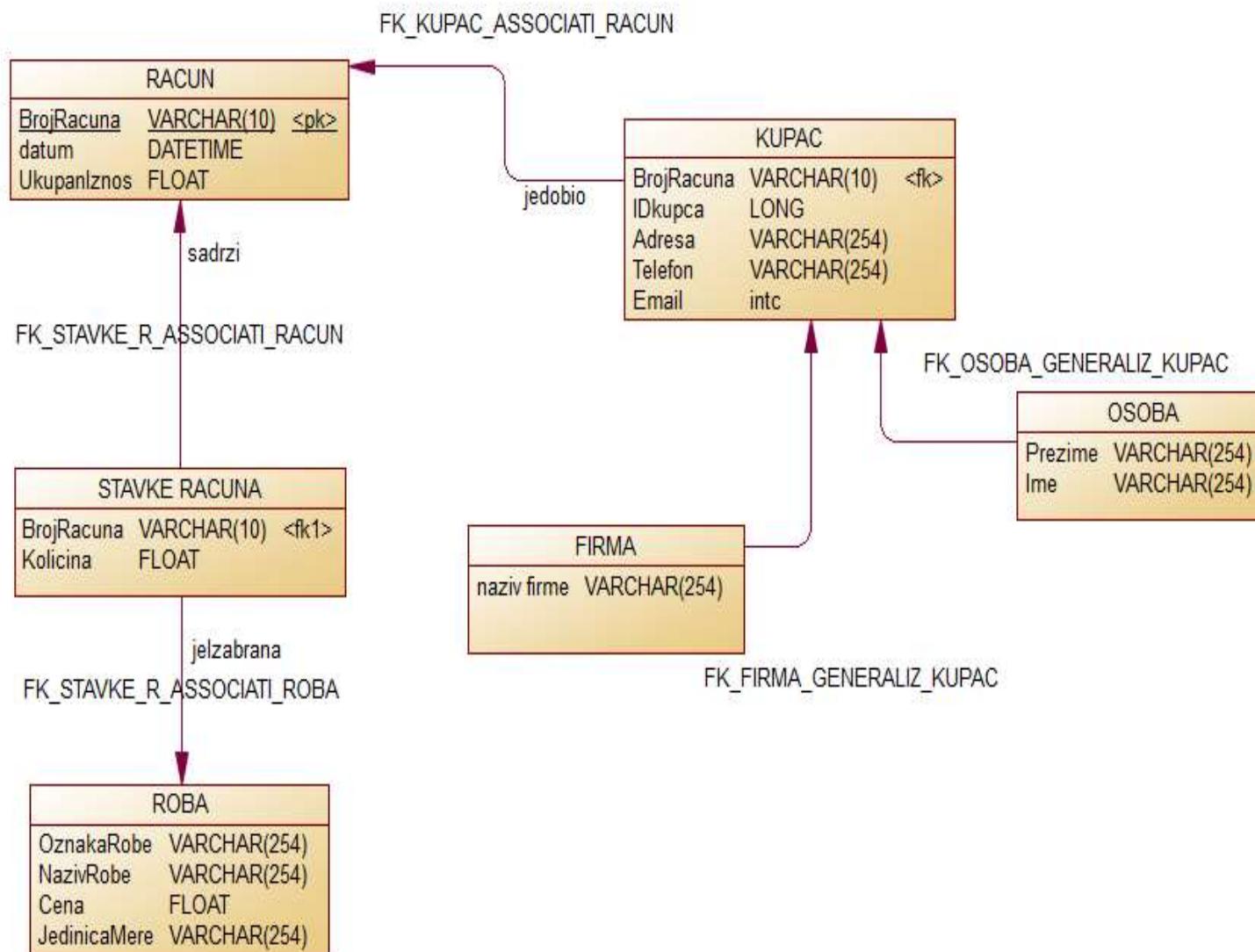
# Generisanje fizičkog modela (Physical DM)



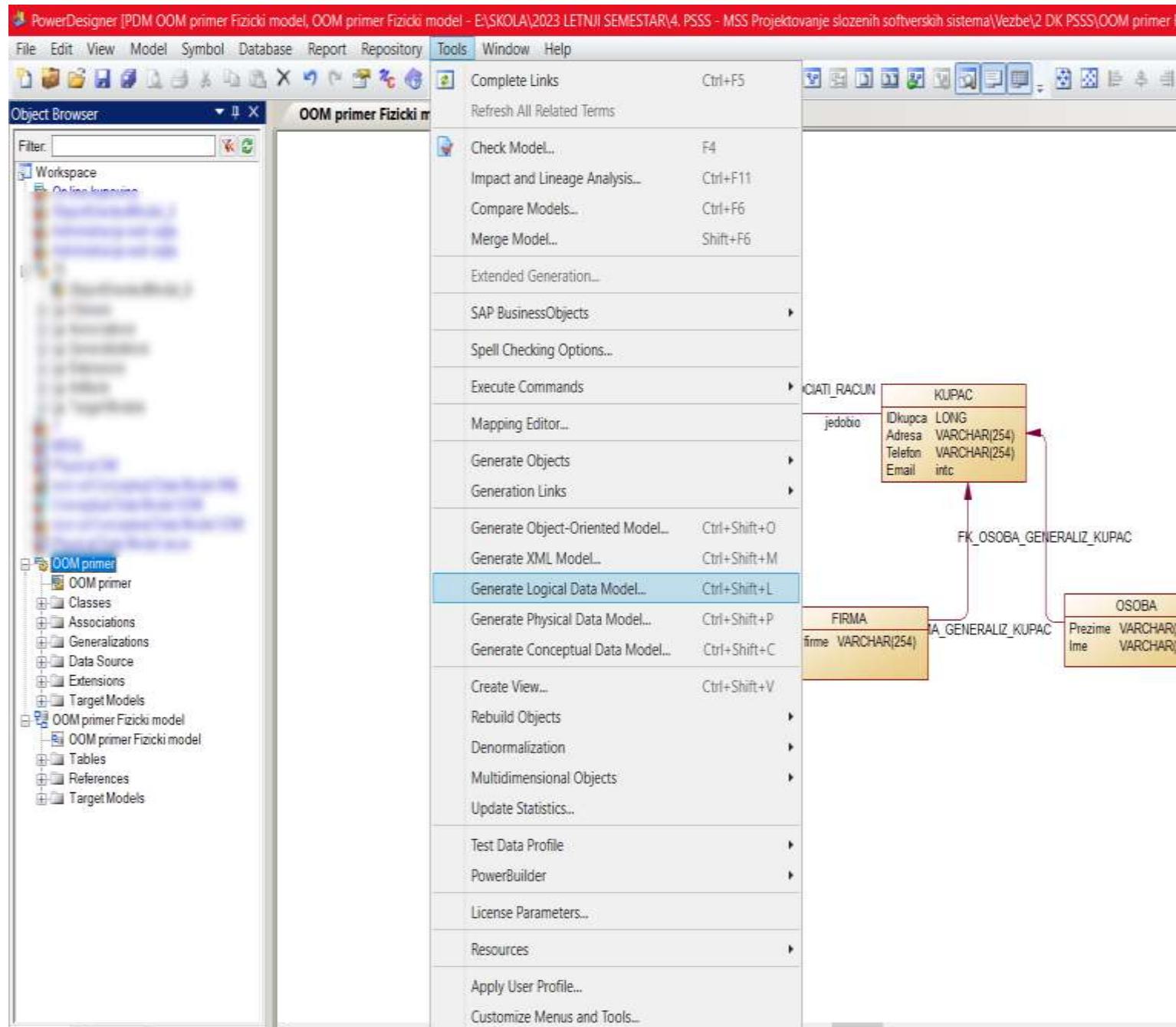
# Generisan fizički model



# Fizički model sa PK u entitetu RACUN

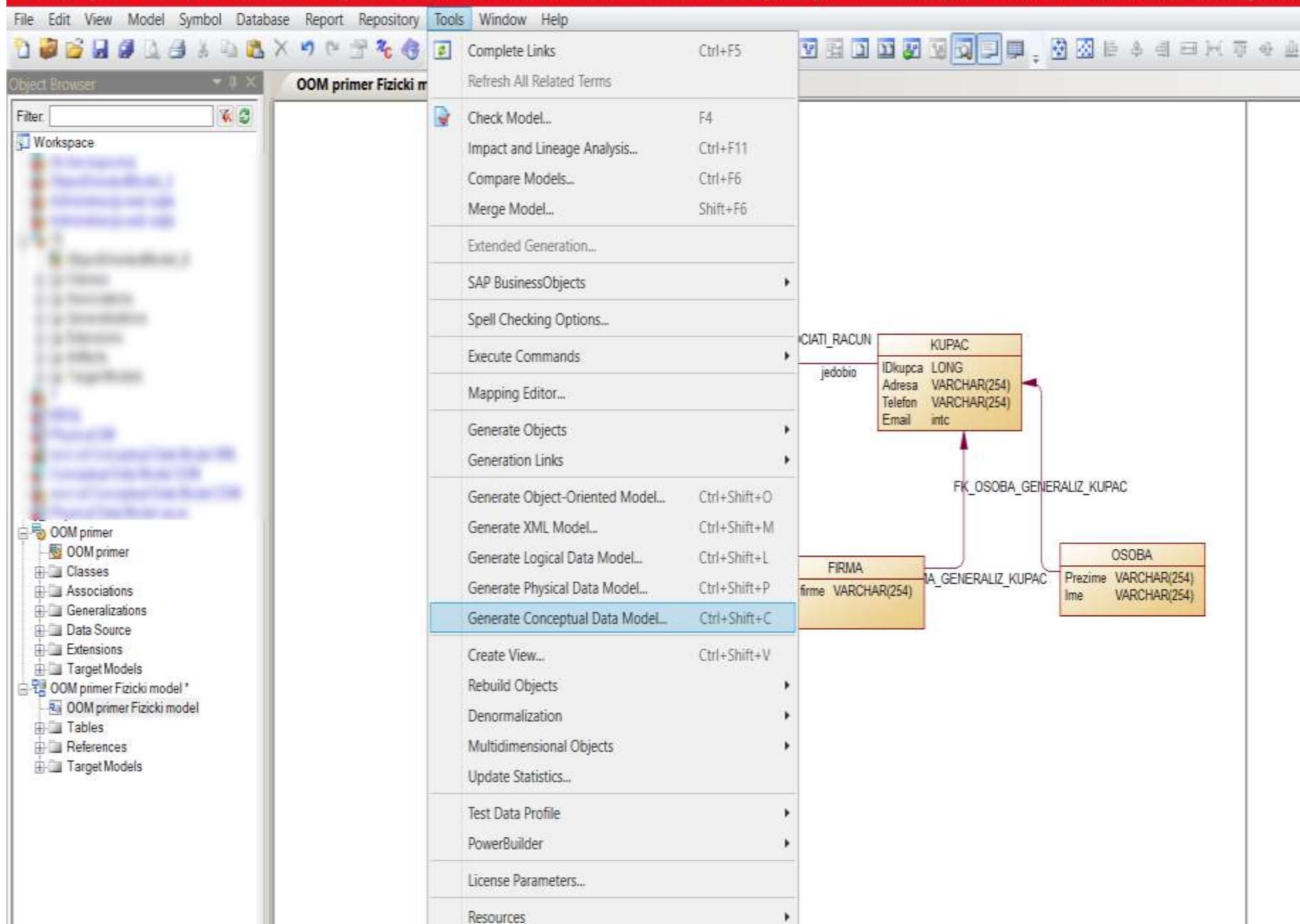


# Generisanje logičkog modela (Logical DM)

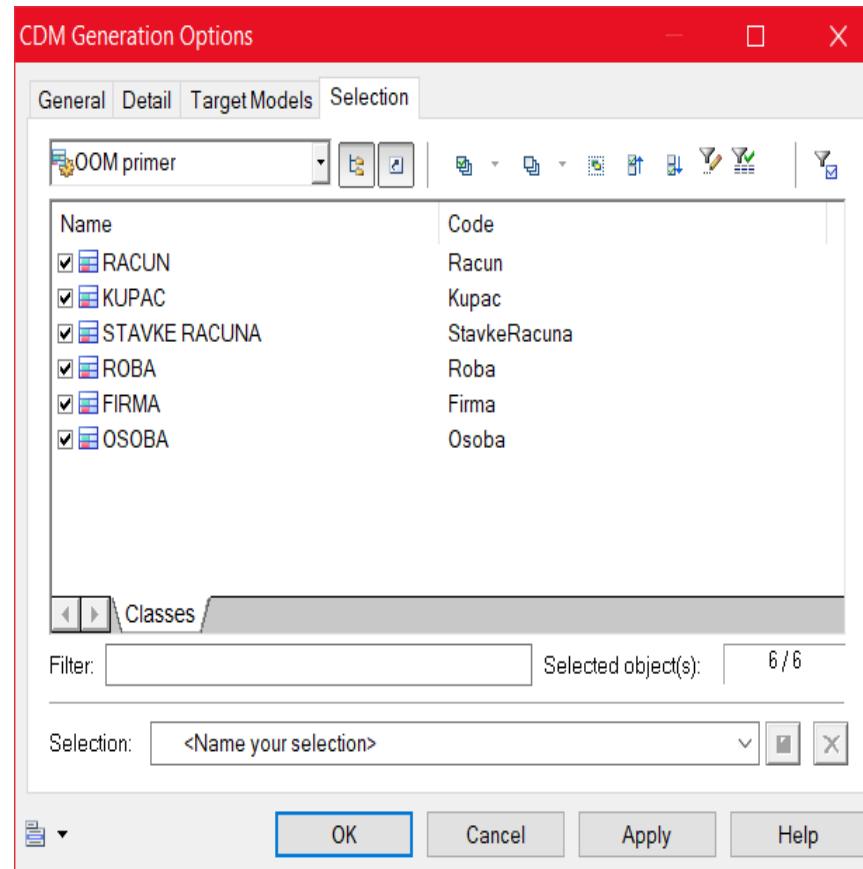
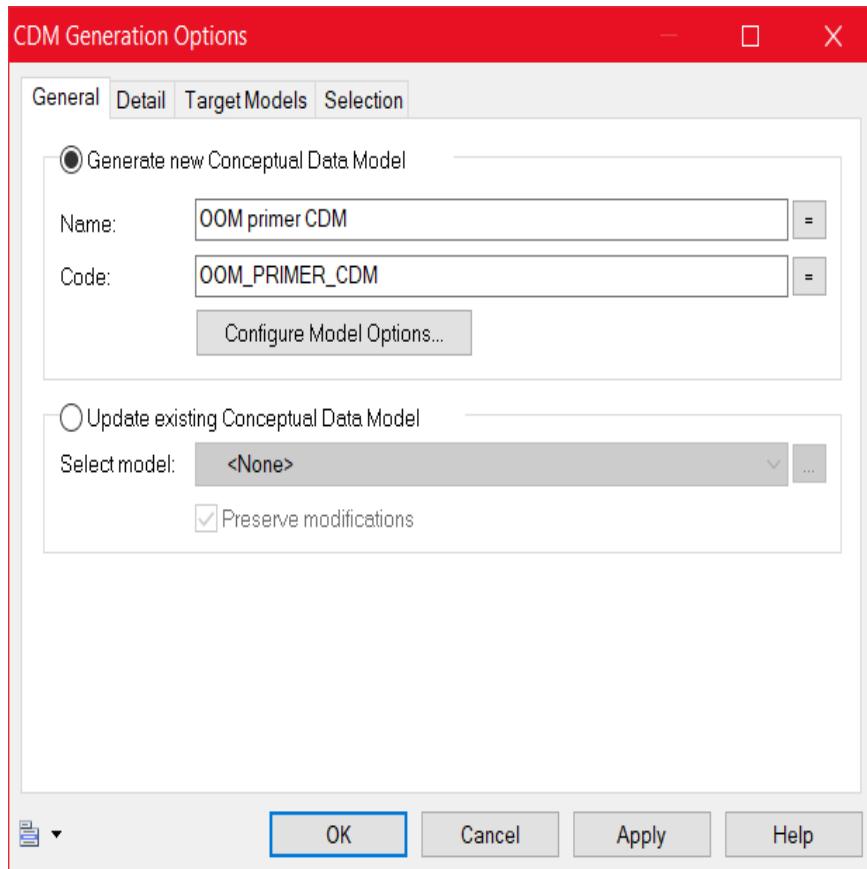


# Generisanje konceptualnog modela (Conceptual - CDM)

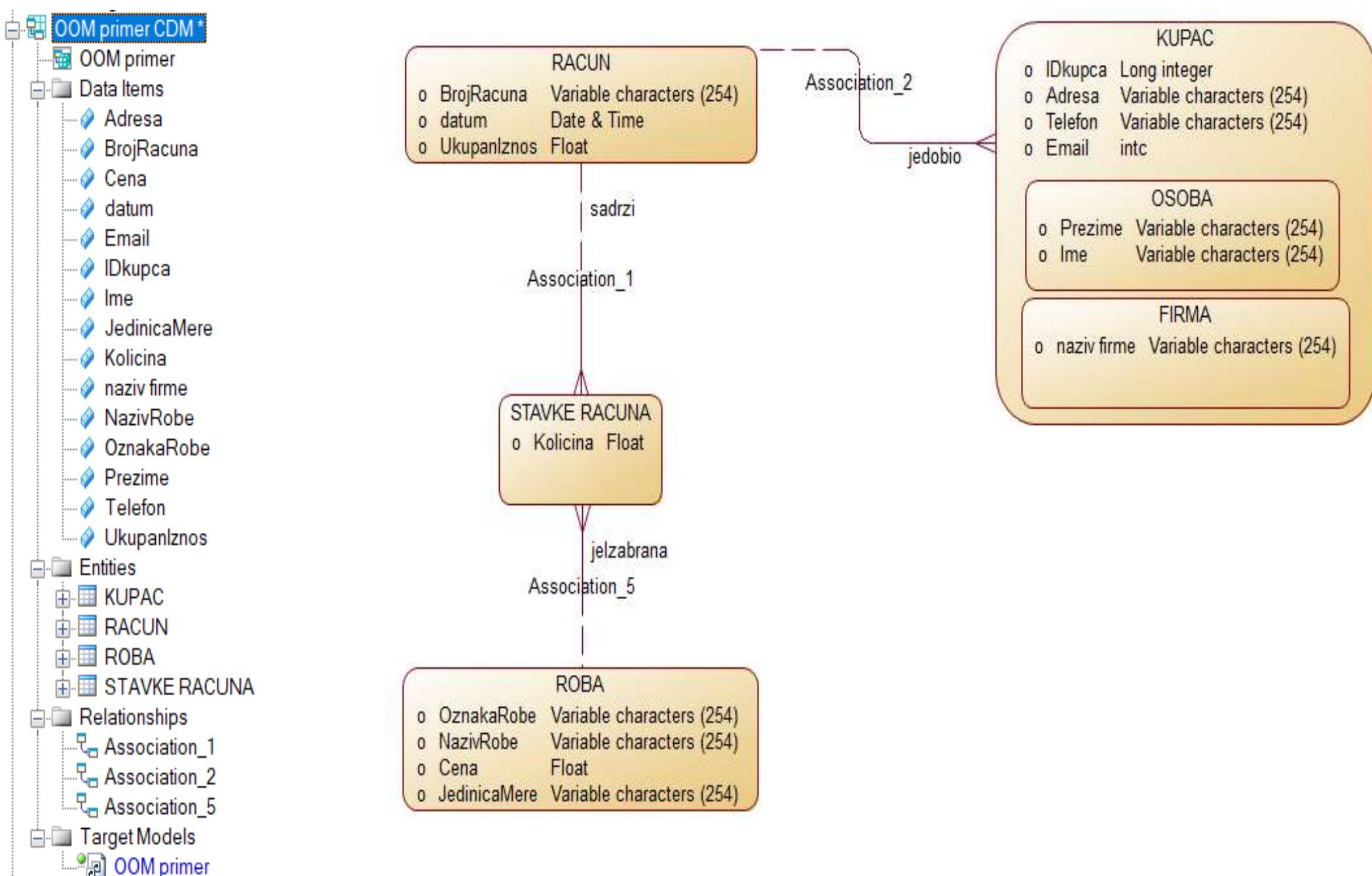
PowerDesigner [PDM OOM primer Fizicki model, OOM primer Fizicki model - E:\SKOLA\2023 LETNJI SEMESTAR\4. PSSS - MSS Projektovanje slozenih softverskih sistema\Vezbe\2 DK PSSS\OOM primer Fizicki model generisan.p



# Generisanje konceptualnog modela (Conceptual - CDM)



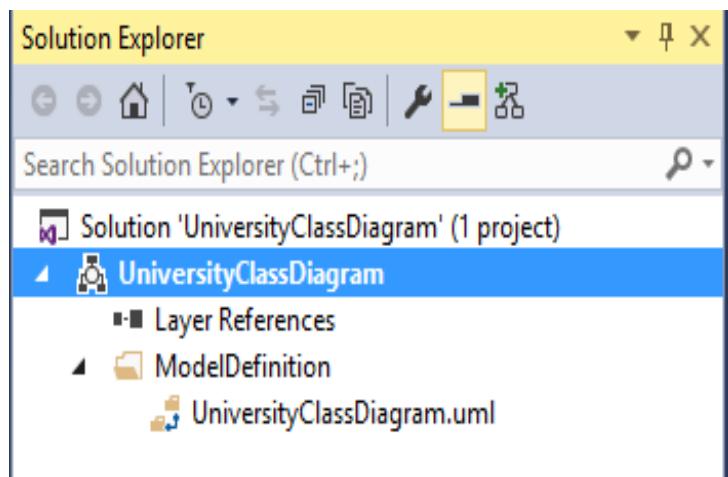
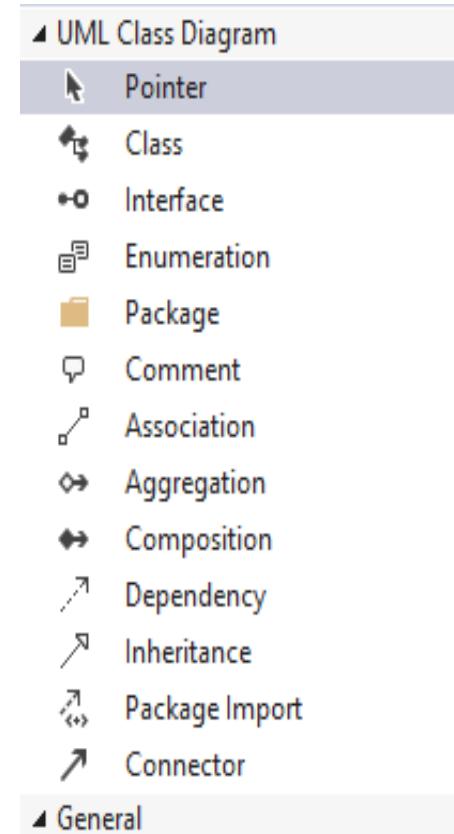
# Generisan konceptualni model (Conceptual - CDM)



# Informacioni sistem jednog univerziteta

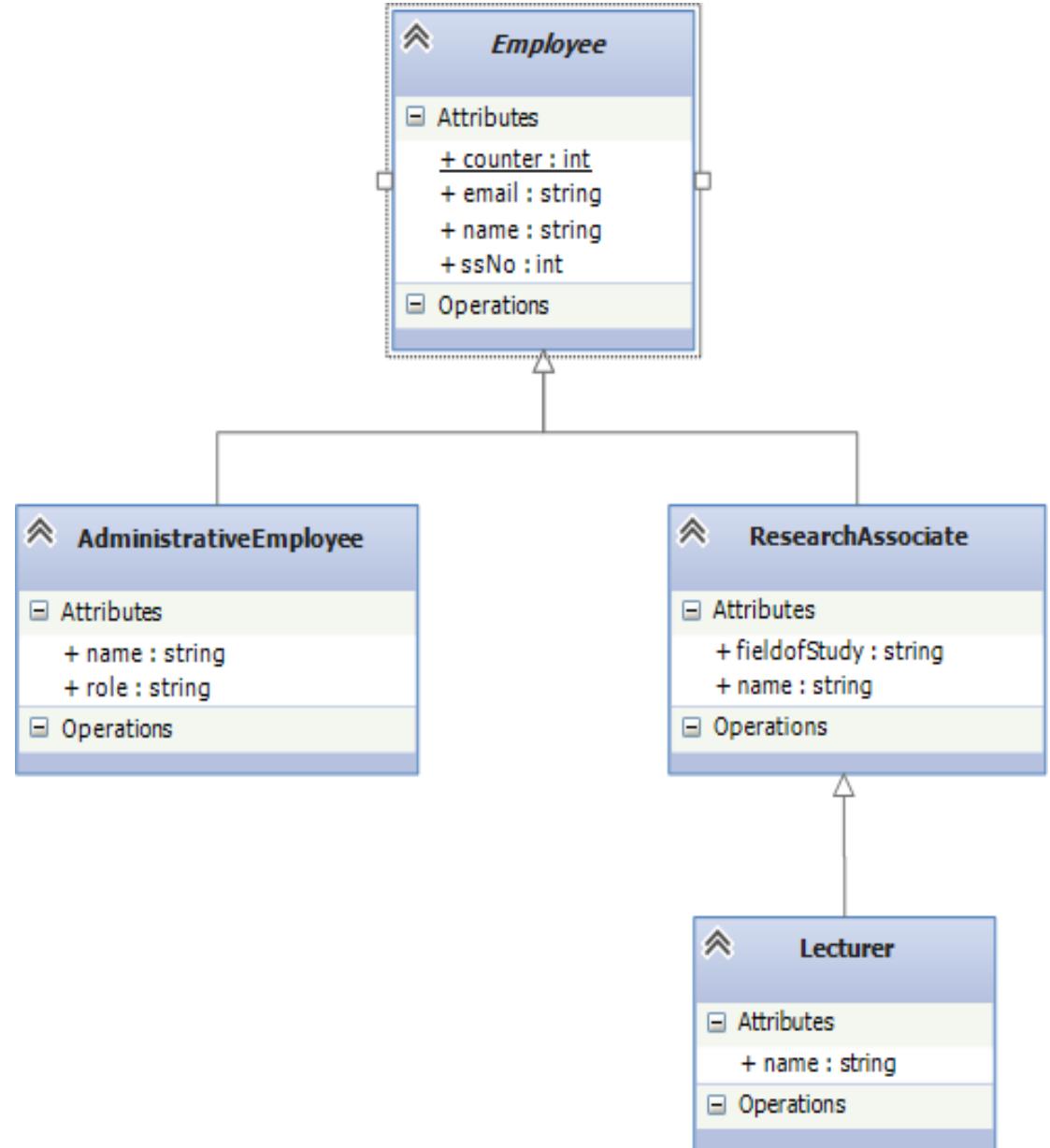
- Univerzitet se sastoji iz više fakulteta koji pored nastave imaju projekte na institutima, koji su deo svakog fakulteta. Svaki fakultet i svaki institut ima naziv. Adresa je poznata za svaki institut.
- Svaki fakultet vodi dekan, koji je zaposlen na univerzitetu.
- Broj zaposlenih je poznat. Zaposleni imaju ime, prezime, email adresu i osigurani su. U ovakvoj strukturi rada, postoji velika razlika između zaposlenih u administraciji i zaposlenih u nastavi/nauci.
- Naučni istraživači moraju da budu prijavljeni na makar jednom institutu. Oblast istraživanja svakog naučnog istraživača je poznat. Takođe, naučni istraživači su angažovani na projektima po određenom vremenu, a naziv, datum za početak i kraj angažovanja su poznati.
- Neki naučni istraživači drže i kurseve. Tada se nazivaju predavači/nastavnici. Kursevi imaju jedinstven identifikator, naziv, vreme trajanja u satima po nedelji.

- Korak 1: Identifikovati klase
- Korak 2: Identifikovati atribute
- Korak 3: Identifikovati operacije
- Korak 4: Identifikovati relacije
- Pokrenuti Visual Studio 15-17 program
- File-new-Project-Modeling project
- Architecture-New UML or Layer Diagram

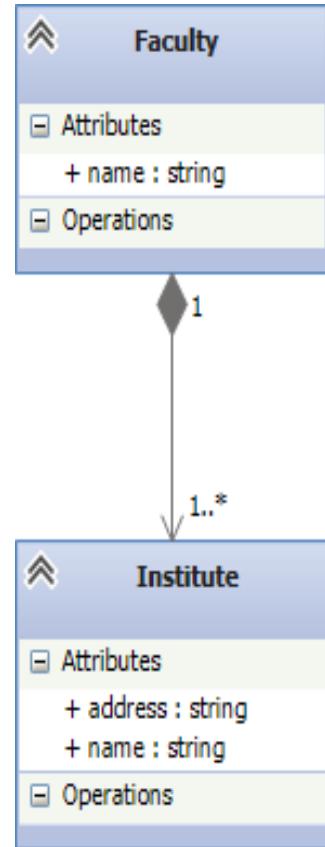


Potražite opširnije na: <https://msdn.microsoft.com/en-us/library/dd409445.aspx>

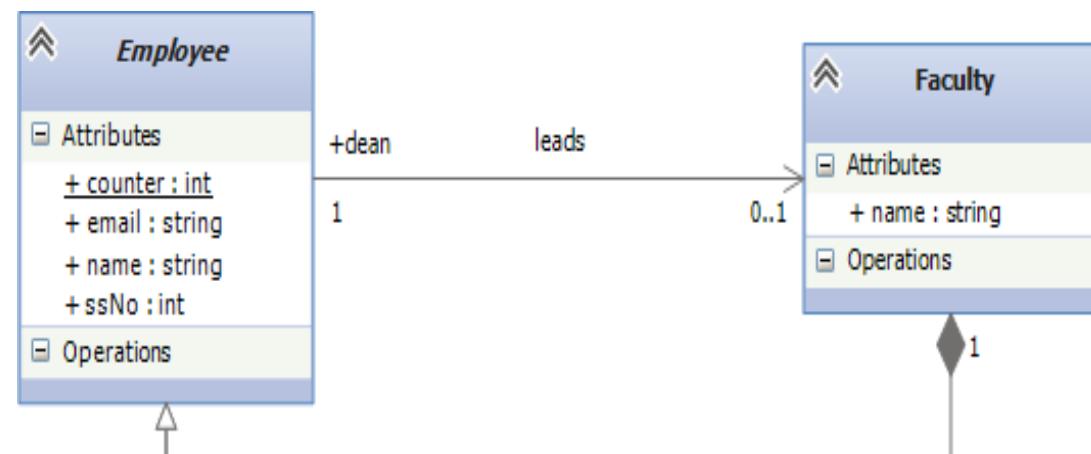
- Indikacija da je generalizacija



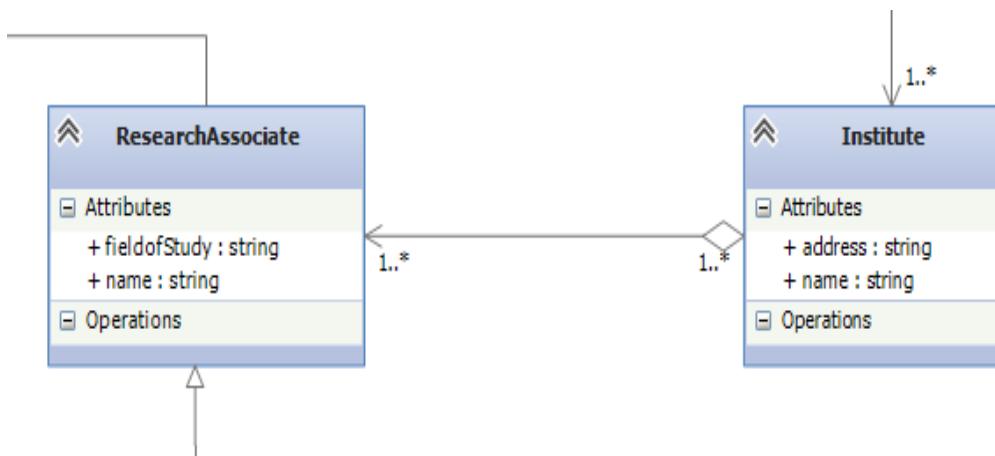
- Univerzitet se sastoji iz više fakulteta koji pored nastave imaju projekte na institutima, koji su deo svakog fakulteta. Svaki fakultet i svaki institut ima naziv. Adresa je poznata za svaki institut.



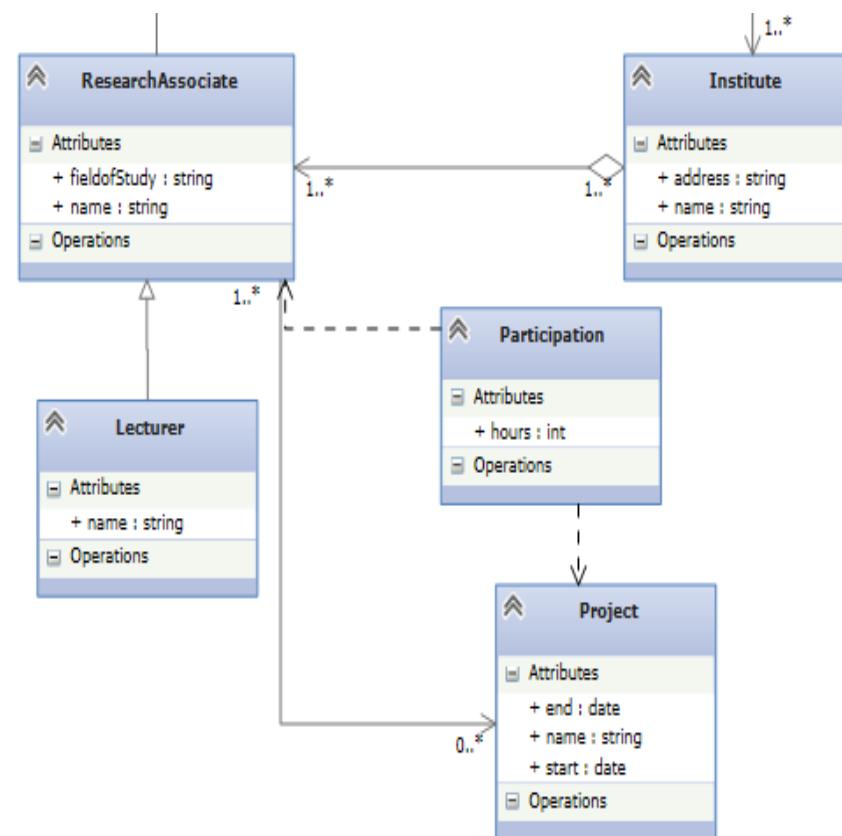
- Svaki fakultet vodi dekan, koji je zaposlen na univerzitetu



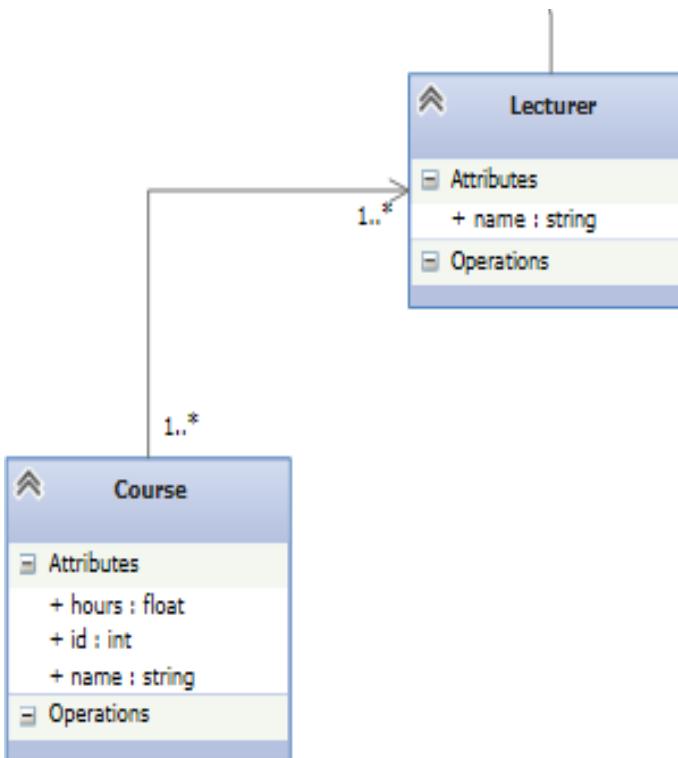
- Naučni istraživači moraju da budu prijavljeni na makar jednom institutu.



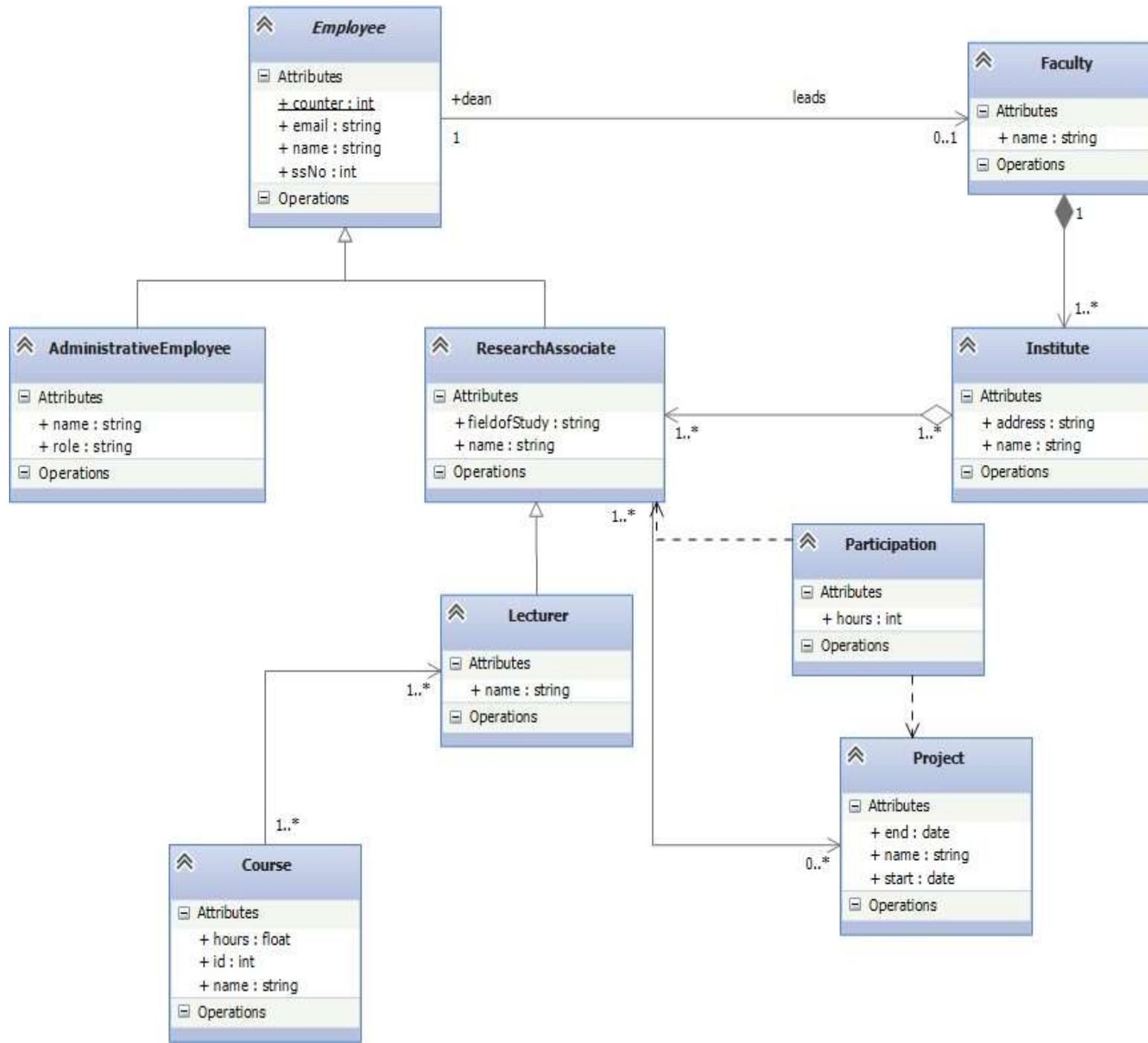
- Takođe, naučni istraživači su angažovani na projektima po određenom vremenu, a naziv, datum za početak i kraj angažovanja su poznati.

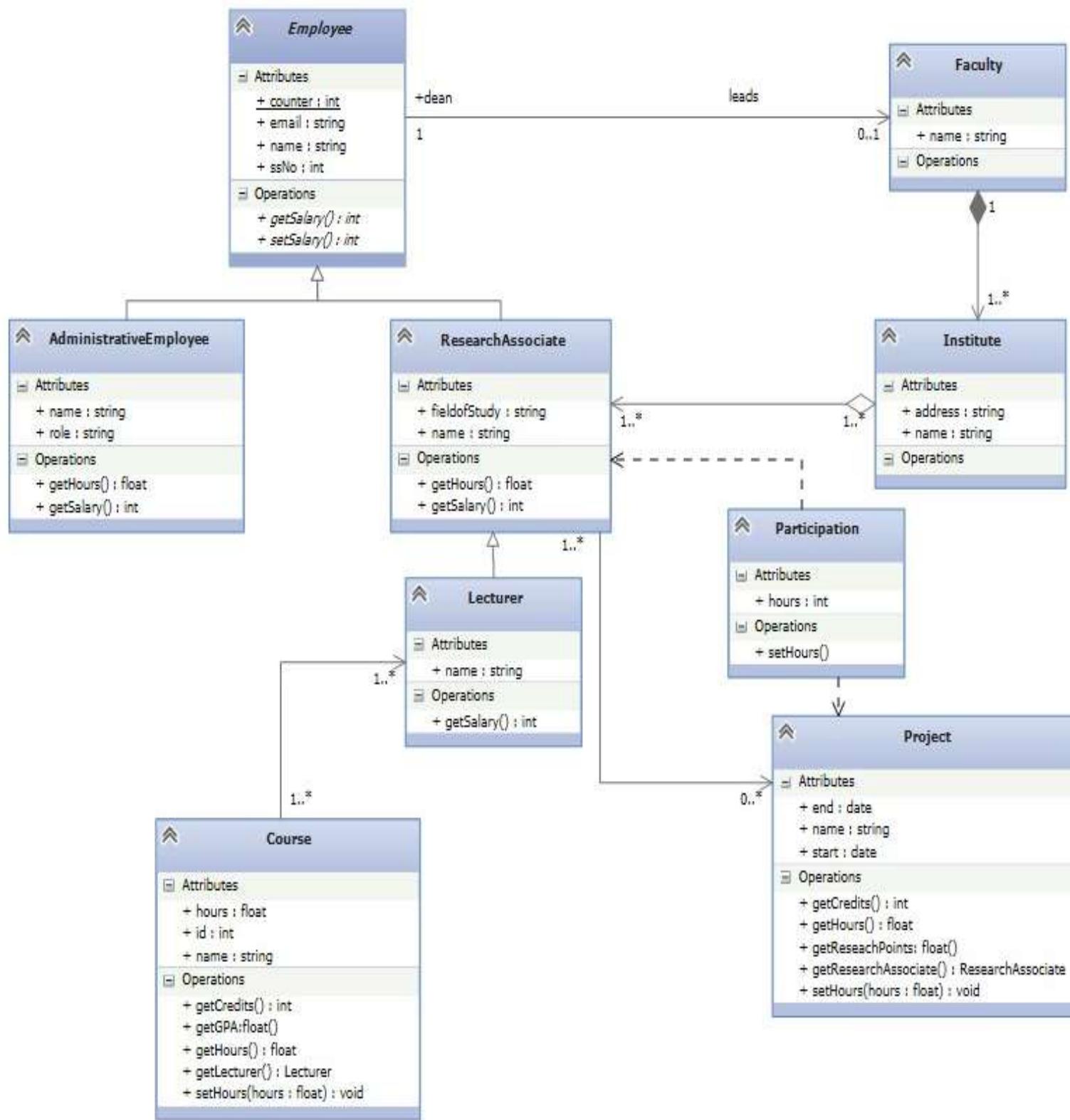


- Neki naučni istraživači drže i kurseve. Tada se nazivaju
- predavači/nastavnici. Kursevi imaju jedinstven identifikator, naziv, vreme trajanja u satima po nedelji
- Nastavnik nasleđuje sve karakteristike, asocijacije i agregacije od naučnog istraživača. Pored ovoga nastavnik ima relaciju asocijacije prema klasi kurs.

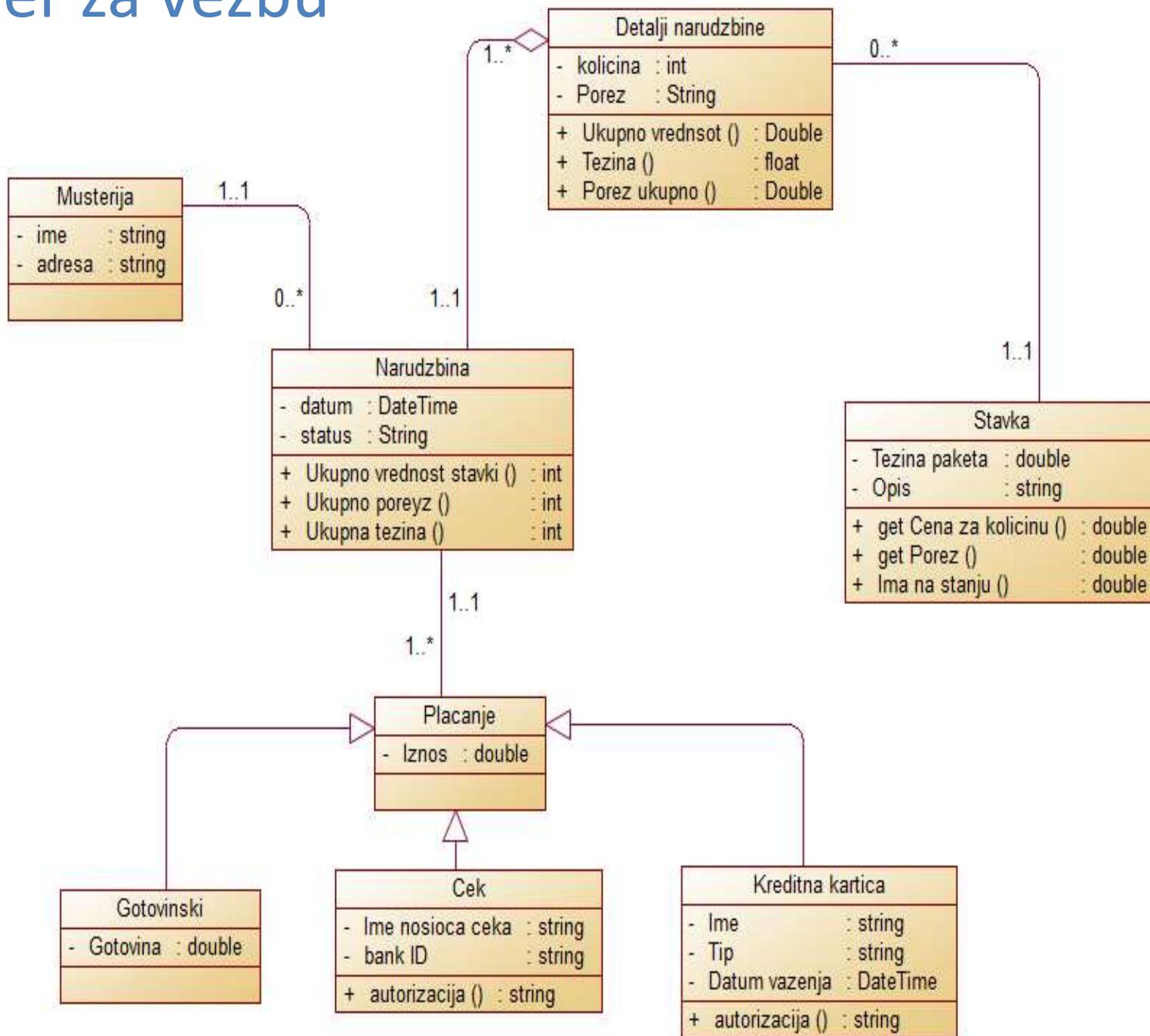


# Kompletan dijagram klasa za IS





# Primer za vežbu



# Primer za vežbu

Napraviti dijagram klasa **sistema rezervacije avio karata**.

Potrebno je da se napravi dijagram klasa koji treba vizuelno da prikaže kupovinu karata i precizne podatke o letu.

Sistem rezervacije avio karata ažurira listu klijenata i putnika, kao i red letenja.

Klase **Klijent** sa atributima idKlijent (private), punolme, adresa, brTelefona može da poruči više avio karata.

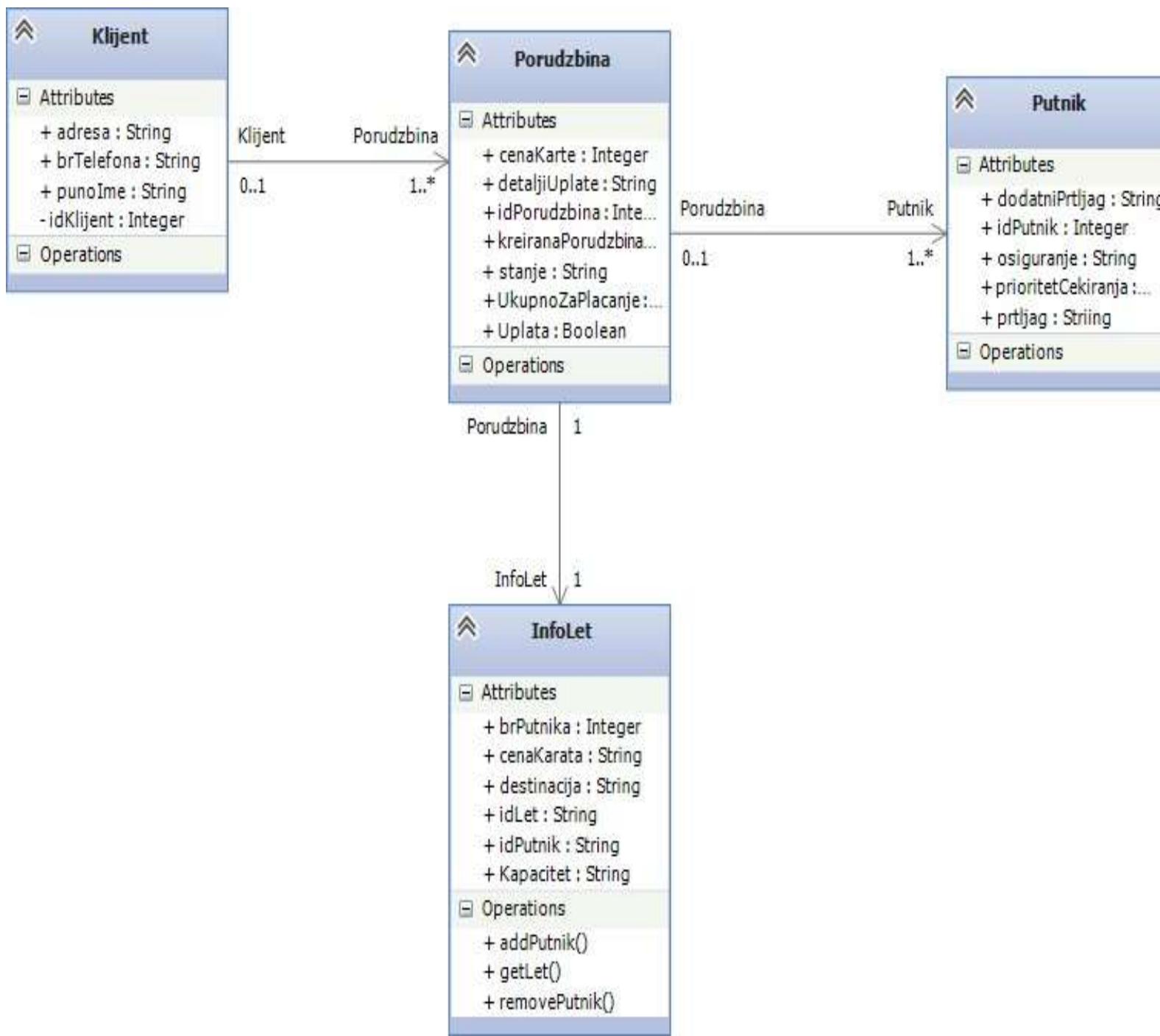
Jedna porudžbina se odnosi na jednog klijenta koji može više avio karata da poruči.

Klase **Porudžbina** sadrži sledeće attribute: idPorudžbina, cenaKarte, stanje, kreiranaPorudžbina, ukupnoPlaćanje, Uplata (Boolean TRUE/FALSE), detaljiUplate.

Nakon uspešne porudžbine i plaćanja, klijent postaje putnik i potrebno je da se definije klasa **Putnik** koja će da se povezuje sa Porudžbinom. Klasa Putnik sadrži attribute idPutnik, osiguranje, prtljag, dodatniPrtljag, prioritetiCekiranja.

Jedan putnik za određeni red letenja se odnosi na 0..1 porudžbina avio karata, a jedna porudžbina za određeni red letenja može imati 1..\* putnika. Na osnovu ove relacije se određuje lista reda letenja koja je krucijalna za avio sistem i za pregled putnika i reda letenja.

# Primer za vežbu



# Primer za vežbu

Klasa Porudžbina mora da ima podatke o letu, tako da je potrebno da se kreira klasa **INFOLet** koja će da sadrži attribute idLet, Kapacitet, brPutnika, idPutnik, cenaKarata, destinacija I određene operacije koje dodaju putnike (addPassenger()) uklone putnike (removePassenger()) I unos leta (getFlight()). INFOLet klasa se dalje povezuje sa klasom **Let** koja sadrži: brLeta, odlasci I dolasci. Relacija je 1:1. Osim reda letenja I unosa putnika nakon kupovine karata, potrebno je da se definiše I aerodrom gde će sleteti avion. U tom slučaju treba da se kreira klasa **Aerodrom** sa atributima naziv, kod, lokacija. Jedan let može da se evidentira na najmanje 2 aerodroma (poletanje I sletanje) ili na maksimalno M.

