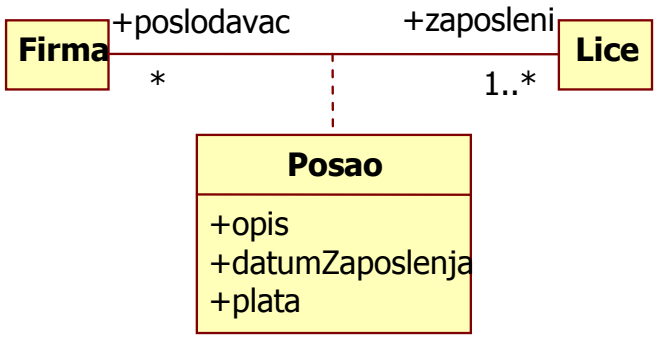
# Projektovanje softvera

Dijagrami klasa –
napredniji pojmovi

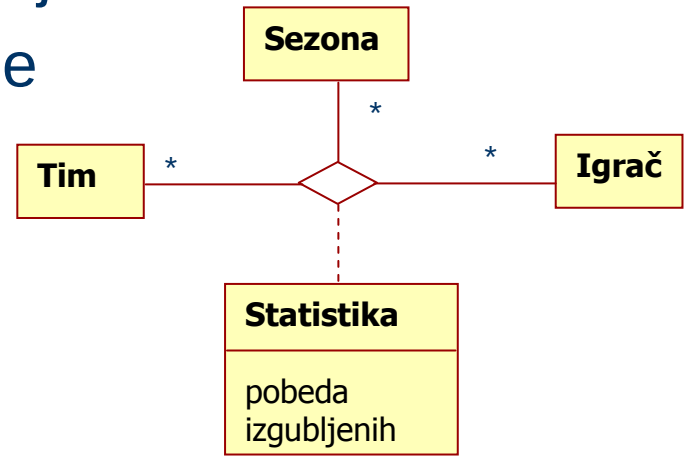
# Klasa asocijacije

- Sama asocijacija može imati svoje atribute
- Ti atributi pripadaju klasi koja opisuje asocijaciju (slično veznoj tabeli kod relacionih baza)
- Primer:



# N-arna asocijacija

- Asocijacija može povezivati više od 2 klase
  - takva asocijacija se naziva n-arnom
- Svaka pojava n-arne asocijacije je n-torka pojava odgovarajućih klasa
- Primer ternarne asocijacije (sa klasom asocijacije):



# Kvalifikacija (1)

- Koristi se da označi ključ (kvalifikator) koji se koristi za selekciju objekta iz neke strukture
- Objekat strukture za datu vrednost ključa selektuje jedan ili grupu elemenata
- Učesnici u relaciji su `Struktura` i njen `Element`
  - element(i) strukture se izdvaja(ju) uz pomoć ključa
- Kvalifikator može imati više atributa, oni su atributi asocijacije
- Atributi kvalifikatora imaju istu sintaksu kao i atributi klasifikatora, sem inicijalne vrednosti
- Grafička notacija:

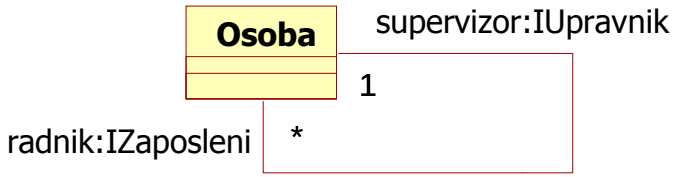

  - ukras je deo asocijacije, ne klase

# Kvalifikacija (2)

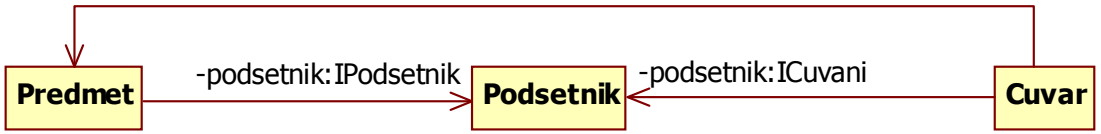
- Primeri struktura kojima se pristupa pomoću ključa su
  - heš tabela, B-stablo, …
- Multiplikativnost na kraju asocijacije kod klase `Element`:
  - moguće kardinalnosti skupa selektovanih objekata uparivanjem sa objektom strukture uz zadatu vrednost ključa:
    - `0..1` – može da bude selektovan jedinstven objekat, ali postoje i vrednosti ključa za koje se ne selektuje ni jedan objekat
    - `1` – svaka moguća vrednost kvalifikatora selektuje jedinstven objekat elementa
    - `*` – vrednost kvalifikatora deli skup elemenata u podskupove
- Primeri:

# Specifikator interfejsa (UML 1)

- Uloge mogu implementirati samo neke od interfejsa koje realizuju klase u asocijaciji
- Ime interfejsa koji zadovoljava uloga - iza dvotačke koja sledi naziv uloge
- Primer 1:




- Primer 2:

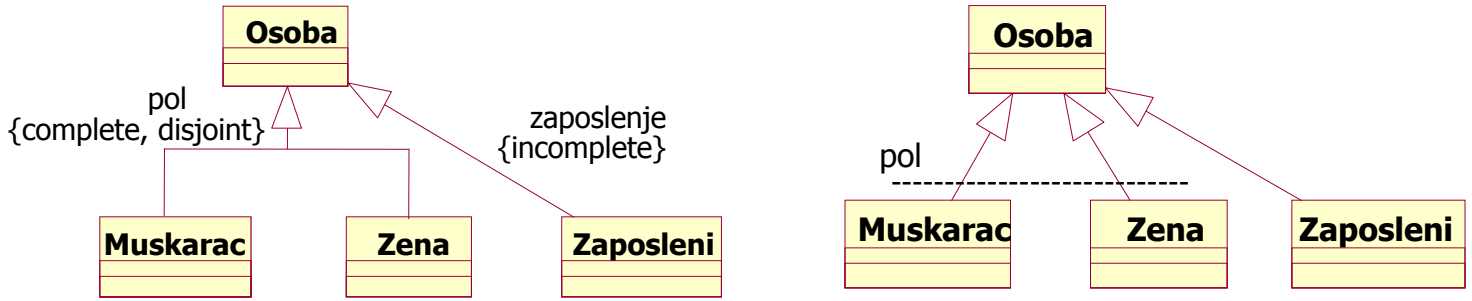


# Generalizacioni skupovi

- Relacija generalizacije/specijalizacije uspostavlja odnos između posebnog i opšteg
- Ponekad se specijalizacija vrši po nekom klasifikacionom kriterijumu
  - kriterijum klasifikacije određuje podtipove
  - često objekti podtipova ne mogu biti i jedne i druge klase (ograničenje `disjoint`)
- Generalizacioni skup definiše poseban skup relacija generalizacije koji opisuje na koji način se natklasa specijalizuje
- Ograničenja generalizacionog skupa (pišu se u zagradama `{}`):
  - `disjoint/overlaping`
    - da li objekti podtipova mogu biti isključivo jednog od podtipova generalizacionog skupa
  - `complete/incomplete`
    - da li podtipovi predstavljaju potpuni skup mogućih podtipova

# Notacija i primer

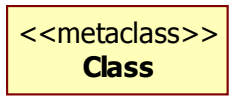
- Primer generalizacionog skupa `pol`:




  - potklase klase `Osoba` mogu biti `Muskarac`, `Zena` i `Zaposleni`
  - generalizacije prema potklasama `Muskarac` i `Zena` pripadaju generalizacionom skupu `pol`

# Metaklasa

- Metaklasa je klasifikator čiji su primerci klase
- Notacija: stereotip klase `<<metaclass>>`
- Primer:
  - metaklasa `Class` u jeziku UML opisuje apstrakciju klase




  - primerci ove metaklase su korisničke klase u konkretnom modelu

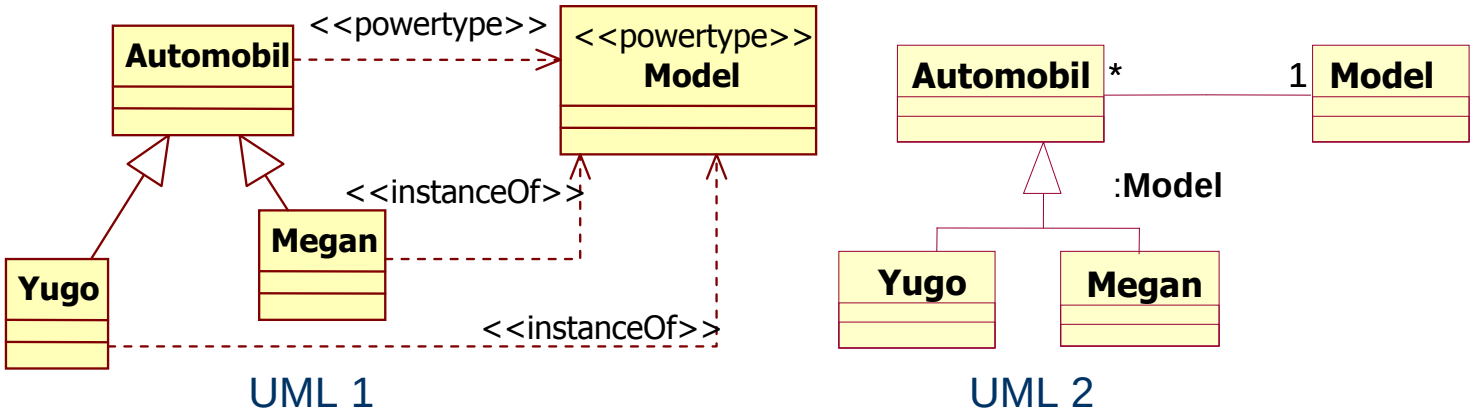

- Metamodel je model kojim se specificira jezik za modeliranje
  - UML je metamodel za konkretan model sistema koji se projektuje

# *Powertype*

- Pojam relevantan za metamodeliranje
- Definicija *powertype:*
  - klasifikator čiji su svi primerci deca (potklase) nekog roditelja
- Formalno:
  - ako je A tip, tada je Power(A) tip čiji su svi primerci podtipovi tipa A
  - ako je tip B primerak tipa Power(A), tada je B podtip A
- *Powertype* je metatip (tip u metamodelu), ali korisnički
  - primerci tog metatipa su tipovi (klase) koji su podtipovi nekog drugog tipa u modelu

# Powertype - primer

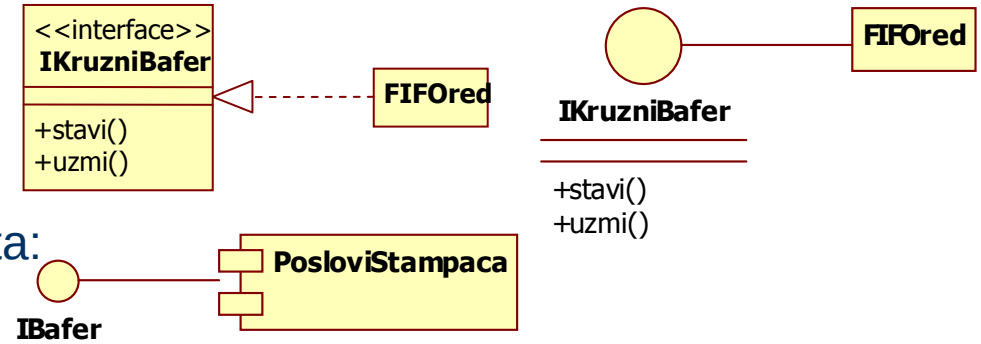
- Primer: instance `Model` (`Yugo`, `Megan`,...) su deca klase `Automobil`




- U gronjem primeru `Model` je metatip za klase `Yugo`, `Megan`
- UML 1:
  - odredište stereotipa relacije zavisnosti `<<powertype>>` je *powertype* klasifikator
  - stereotip zavisnosti `<<instanceOf>>` se koristri za veze klasa-->*powertype* (metatip)

# Konteksti relacije realizacije

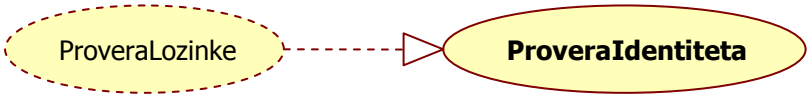
- Realizacija se koristi u dva konteksta:
  - kontekst interfejsa (realizuje ga klasa ili komponenta)
  - kontekst slučajeva korišćenja (realizuje ga kolaboracija)
- Klasni dijagram:




- Dijagram komponenata:

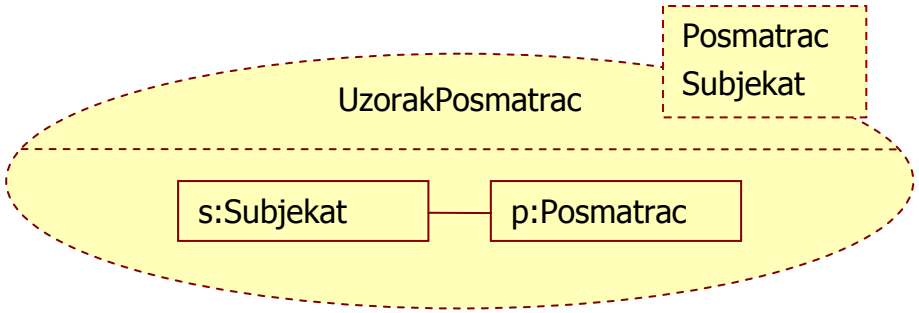



- Dijagram slučajeva korišćenja:

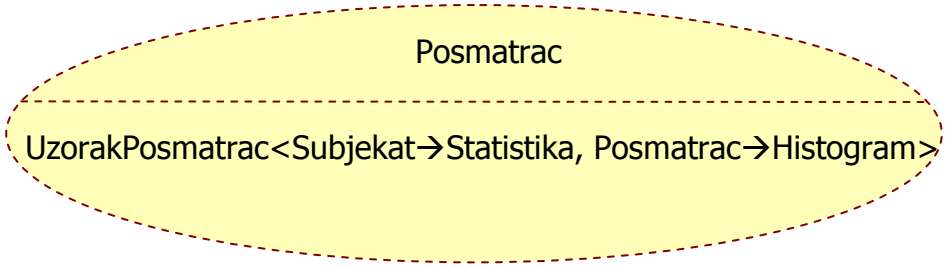


# Parametrizovana kolaboracija

- Koristi se za opis projektnih uzoraka
- Definicija uzoraka "posmatrač“ – parametrizovana saradnja:

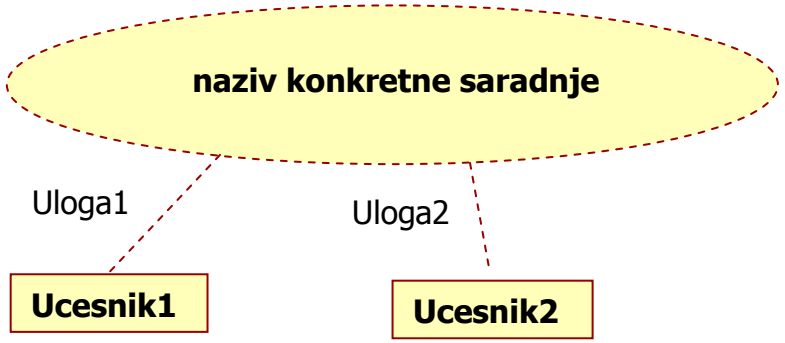



- Konkretna saradnja koja realizuje uzorak "posmatrač":

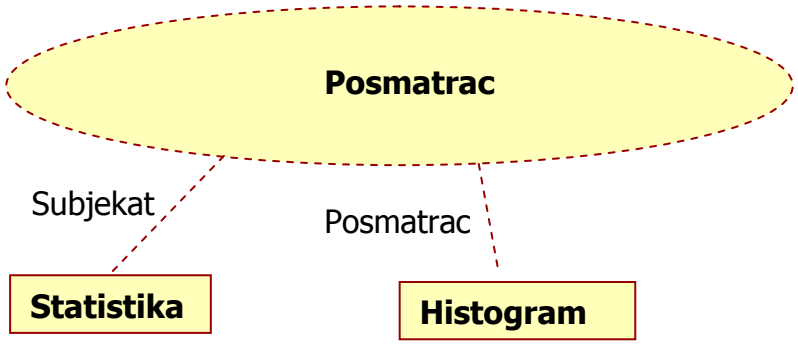


# Projektni uzorci

- Drugi način za opis konkretne saradnje




- Na prethodnom primeru:



# Standardni stereotipovi klasifikatora

- **utility** – klasa čiji su atributi i operacije zajednički za sve instance klase
- **focus** – klasa koja implementira glavnu (poslovnu) logiku
- **auxiliary** – klasa koja pomaže focus klasi u implementaciji logike
- stereotype – klasifikator je stereotip koji se može primeniti na druge elemente
- **metaclass** – klasifikator čije su instance klase
- *powertype* – klasifikator čije su instance deca datog roditelja (UML1)
- *thread* – klasa čiji su objekti aktivni sa deljenim adresnim prostorom (UML1)

- **process** – **komponenta** čiji primerci imaju vlastite procese (u UML1 process je bio stereotip klase)

- Legenda:
  - **naglašeno** – stereotip specificiran kao standardni i u UML 2 (profili L2 i L3)
  - obično – nije naveden kao zastareo, ali nije naveden ni kao std. stereotip UML 2
  - *kurziv* – stereotip eksplicitno naveden kao zastareo

# Standardni stereotipovi relacije zavisnosti

- Između klasa i/ili objekata na klasnom dijagramu:
  - **`instantiate`** – izvor stvara primerke odredišta
  - `instanceOf` – izvor je objekat koji je primerak odredišnog klasifikatora
  - **`send`** – izvor (operacija) šalje signal koji je odredište relacije (odredište relacije nije primalac signala)
  - **`derive`** – izvor se može izračunati na osnovu odredišta; relacija između dva atributa ili dve asocijacije: jedan je konkretan, a drugi konceptualan (npr: DatumRodj i Starost)
  - **`refine`** – izvor je finiji stepen apstrakcije od odredišta; na primer: klasa u projektu je na finijem stepenu apstrakcije od odgovarajuće klase u analizi
  - **`permit`** – izvoru se daju posebna prava pristupa odredištu
  - **`bind`** – izvor je generisan iz parametrizovanog šablona koji predstavlja odredište
  - *`friend`* – izvoru se daju posebna prava pristupa odredištu (samo UML 1)
  - *`powertype`* – odredište je *powertype* izvora (samo UML 1)
- Između paketa:
  - *`access`* – izvorni paket ima pravo pristupa elementima odredišnog (privatni uvoz)
  - **`import`** – javni sadržaj odredišnog paketa ulazi u prostor imena izvornog paketa (kao da su imena odredišnog paketa deklarisana u izvornom paketu)

# Standardni stereotipovi generalizacije

- Standardni stereotip relacije generalizacije
  `implementation` – dete nasleđuje implementaciju roditelja, ali je čini privatnom i ne podržava interfejs roditelja, pa ne može zamenjivati roditelja