

I - Dijagrami slučajeve korišćenja

- Dijagram slučajeve korišćenja (*use-case*) prikazuje skup slučajeva korišćenja i aktera i relacije između njih

Modeluju: kontekst sistema i zahteve sistema

- Tipično se koristi da specificira neku funkcionalnost i ponašanje aplikativnog sistema
- Dijagram vizuelizuje ponašanje: sistema, podsistema ili čak klase i interfejsa
- Služi korisniku da razume **ŠTA** sistem radi, a verifikatoru da proveriti funkcionisanje

provera korisnika



Elementi dijagrama su:

- 1) slučajevi korišćenja
- 2) akteri
- 3) relacije: - asocijacije (komunikacija),
- zavisnosti (uključivanje i proširivanje) i
- generalizacija
- 4) paketi



1) Ponašanje slučaja korišćenja

Ponašanje slučaja korišćenja se opisuje tokom događaja:

- kada slučaj korišćenja počinje i kada završava
- kada slučaj korišćenja interaguje sa akterima
- kada se razmenjuju objekti
- postoje: **primarni** (osnovni) i **alternativni tokovi** događaja

Tok događaja se može opisati na sledeće načine:

- neformalan strukturirani tekst
- formalni strukturirani tekst (sa pred- i post-uslovima)
- pseudokod
- dijagramima interakcije (jedan za primarni i dodatni za alternativne tokove)



Struktura Use Case specifikacije

Ime

Akteri

Trigeri

Preduslovi

Post uslovi

Ispravan scenario

Alternativni tokovi (scenariji)

Ubačena kartica

Ima račun korisnik

Izdat novac, logovan user



Primer opisa ponašanja slučaja korišćenja

Glavni tok događaja za validaciju korisnika pri bankarskoj transakciji:

- slučaj korišćenja počinje kada sistem ispiše prompt za PIN broj
- korisnik unosi PIN broj preko numeričke tastature
- korisnik potvrđuje unos pritiskom na taster Enter
- sistem proverava PIN broj da vidi da li je validan
- ako je PIN broj validan sistem prihvata unos i time završava slučaj korišćenja



Prvi alternativan tok događaja (ponišćavanje transakcije)

- korisnik poništava transakciju pritiskajući Cancel taster, čime se restartuje slučaj korišćenja
- izmene nisu napravljene na korisnikovom računu

Drugi alternativan tok događaja (editovanje PIN-a)

- korisnik može u bilo kom trenutku pre pritiska Enter da obriše PIN broj i unese drugi

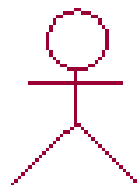
Treći alternativan tok događaja (pogrešan PIN)

- ako korisnik unese pogrešan PIN broj slučaj korišćenja se restartuje
- ako se ovo ponovi tri puta za redom sistem poništava celu transakciju i sprečava korisnika da ponovo pokuša 60s



2) Akteri

- Akter predstavlja neki koherentan skup uloga
- Akter može biti čovek (korisnik) ili neki sistem sa kojim modelirani sistem interaguje
- Aplikativni sistem interaguje sa jednim ili više spoljašnjih aktera
- Akter je standardni stereotip klase sa posebnim grafičkim simbolom



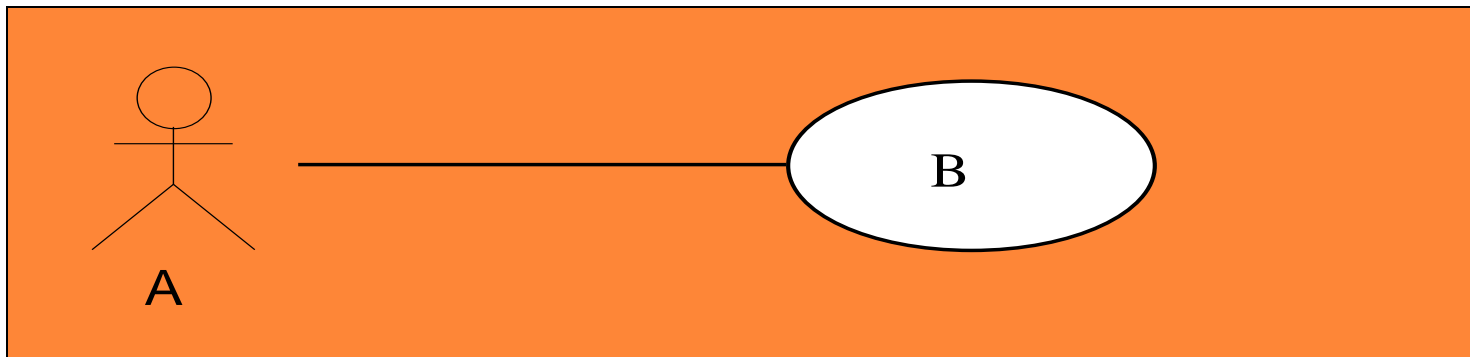
Actor



3) Relacije

A) Relacija komunikacije (asocijacija):

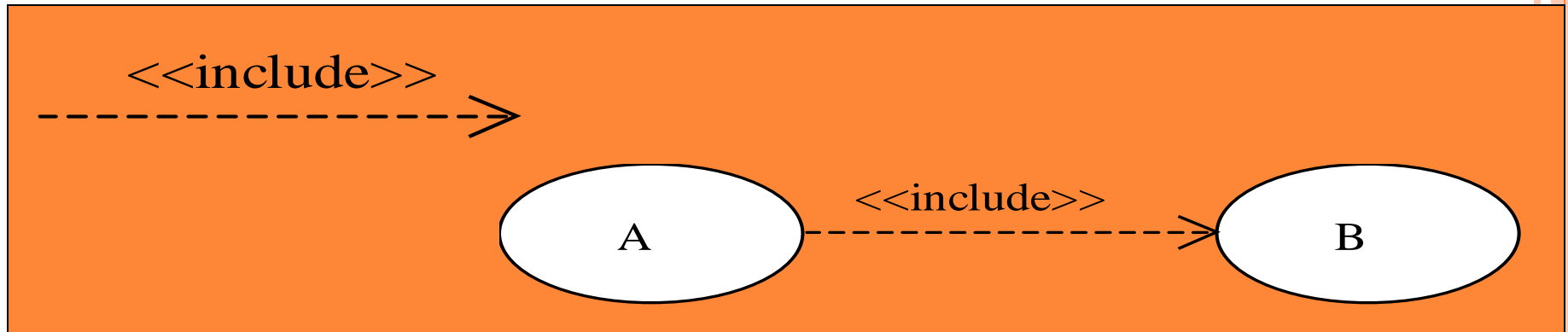
- Prikazuje se punom linijom (asocijacija):



- Na strani A je akter
- Na strani B je slučaj korišćenja
- Komunikaciju može inicirati i strana A i strana B (bidirekciona veza)

B) Relacija uključivanja:

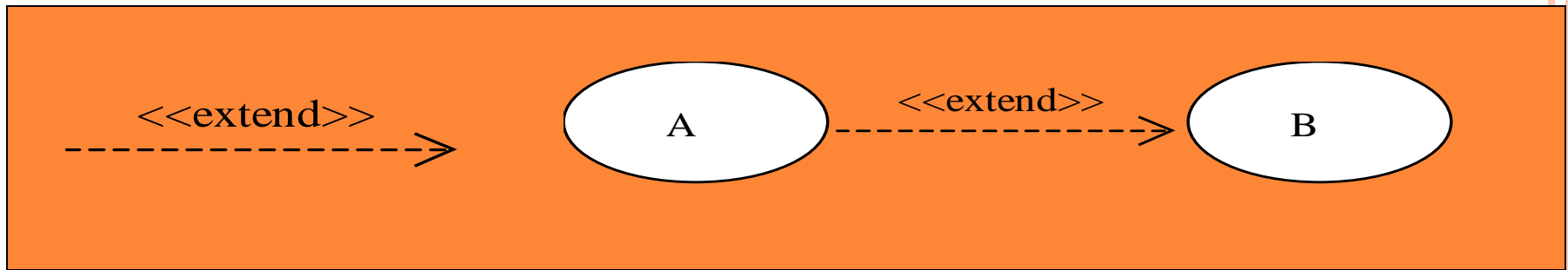
- Prikazuje se isprekidanom linijom sa strelicom i natpisom <<include>>
- Relacija je stereotip **relacije zavisnosti**
- U alatu RR se prikazuje stereotipom unidirekcionane asocijacije (puna linija)



- Relacija uključivanja od slučaja korišćenja A prema slučaju korišćenja B ukazuje da će slučaj korišćenja A uključiti i ponašanje slučaja korišćenja B
- Ponašanje opisano u B je obavezno za A
- Koristi se da opiše zajedničko ponašanje između dva ili više slučaja korišćenja

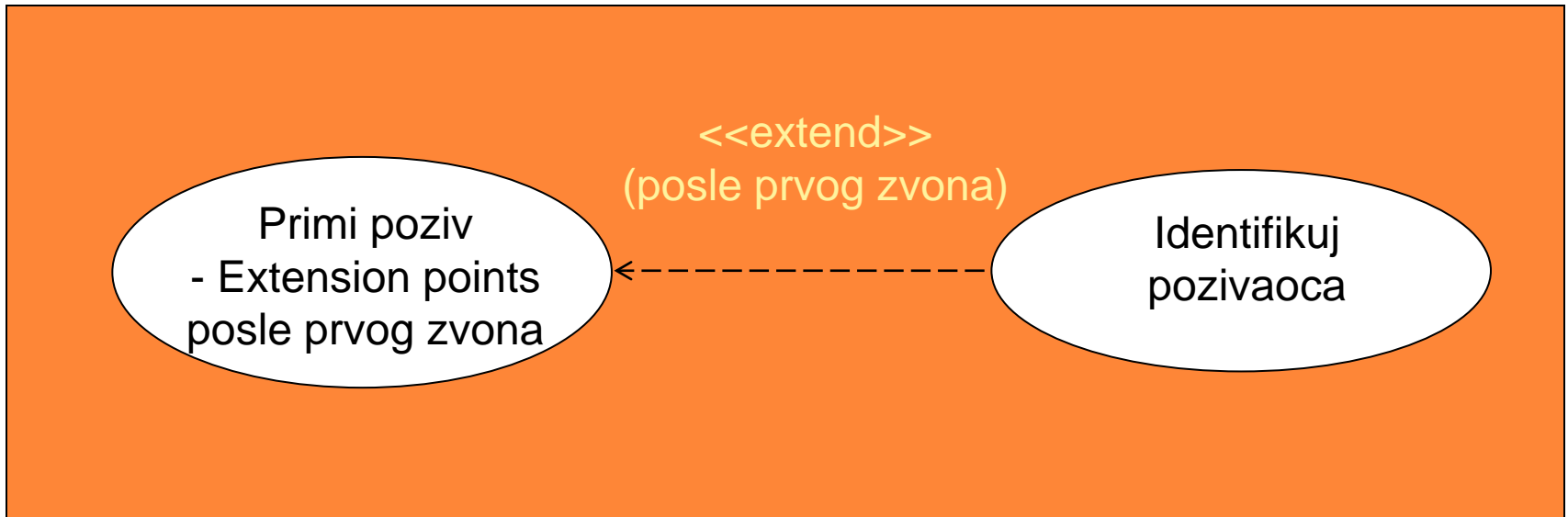
C) Relacija proširivanja:

- Prikazuje se isprekidanom linijom sa strelicom i natpisom <<extend>>
- Relacija je stereotip **relacije zavisnosti**
- U RR-u se prikazuje stereotipom unidirekcione asocijacije (puna linija)



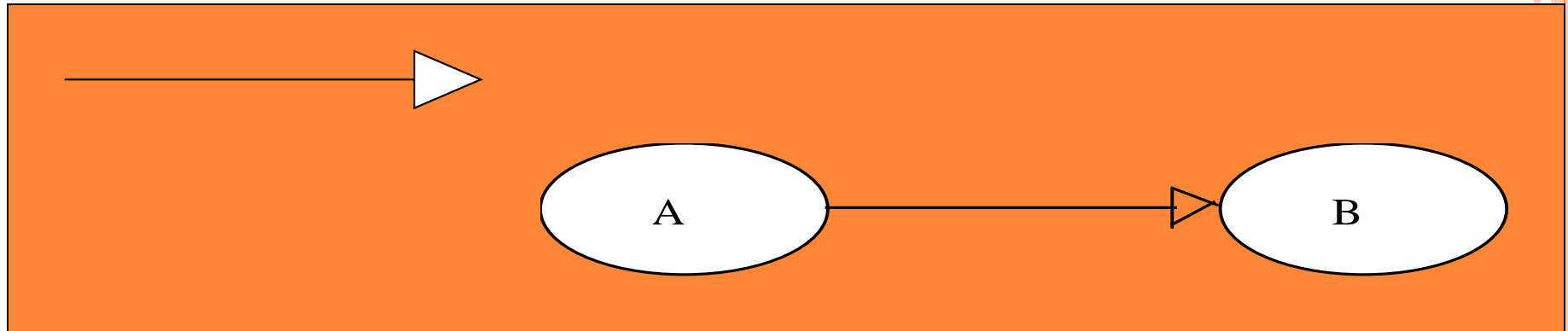
- Relacija proširivanja od slučaja korišćenja A prema slučaju korišćenja B ukazuje da B može obuhvatiti ponašanje specificirano u A
- Praktično B može da se proširi i ispolji celokupno ponašanje opisano u A
- Koristi se da se izrazi opciono ponašanje osnovnog slučaja korišćenja
- Ponašanje opisano u A je opciono, a ono u B osnovno
- Osnovni slučaj korišćenja se proširuje u tačno određenim tačkama
- Tačka u kojoj se slučaj korišćenja proširuje se naziva **tačkom ekstenzije**

Primer:



D) Relacija generalizacije:

- Prikazuje se punom linijom sa trougaonom strelicom
- Relacija je osnovna relacija **generalizacija/specijalizacija**



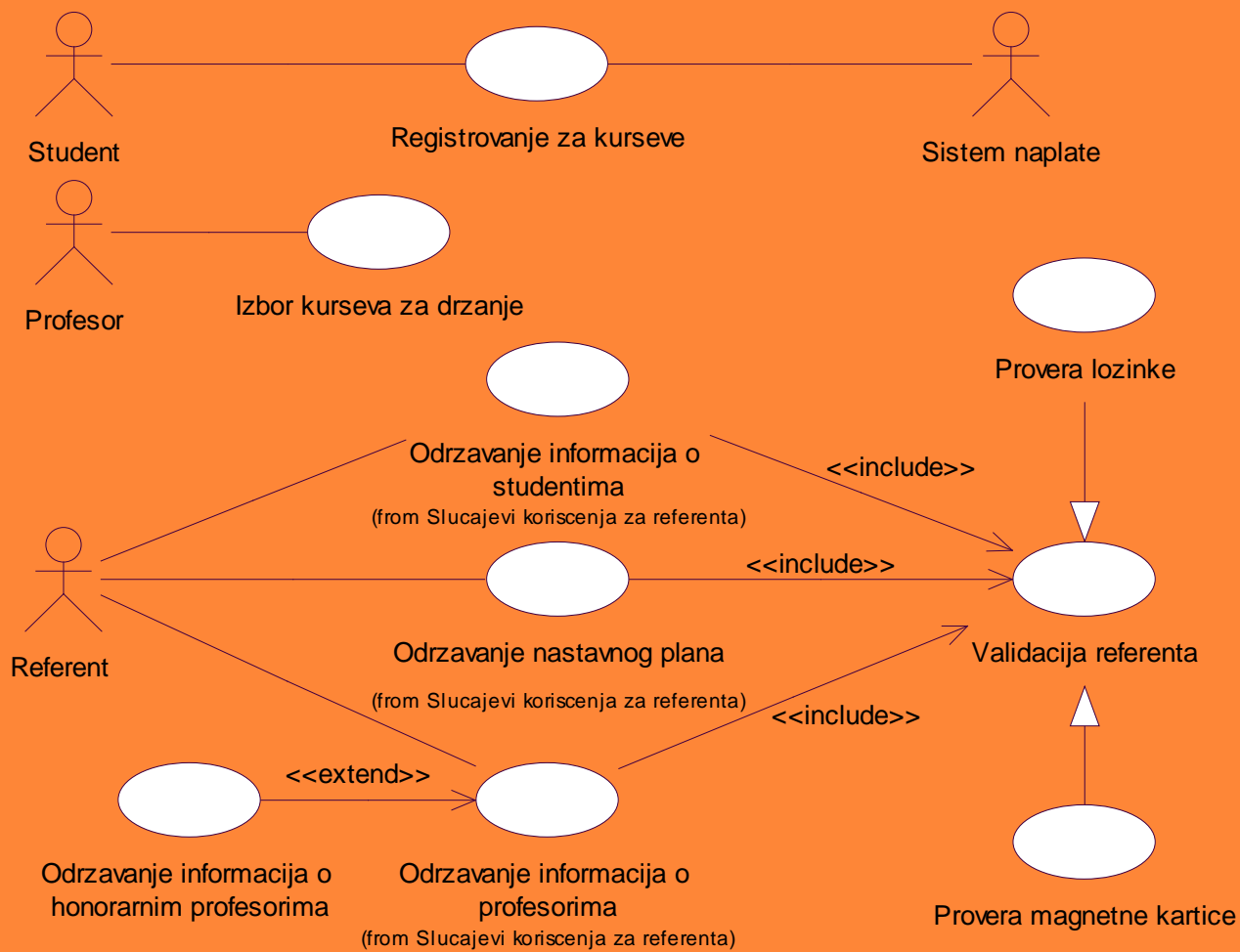
- Relacija generalizacije od slučaja korišćenja A prema slučaju korišćenja B ukazuje da je slučaj korišćenja A specifičan slučaj opštijeg slučaja B



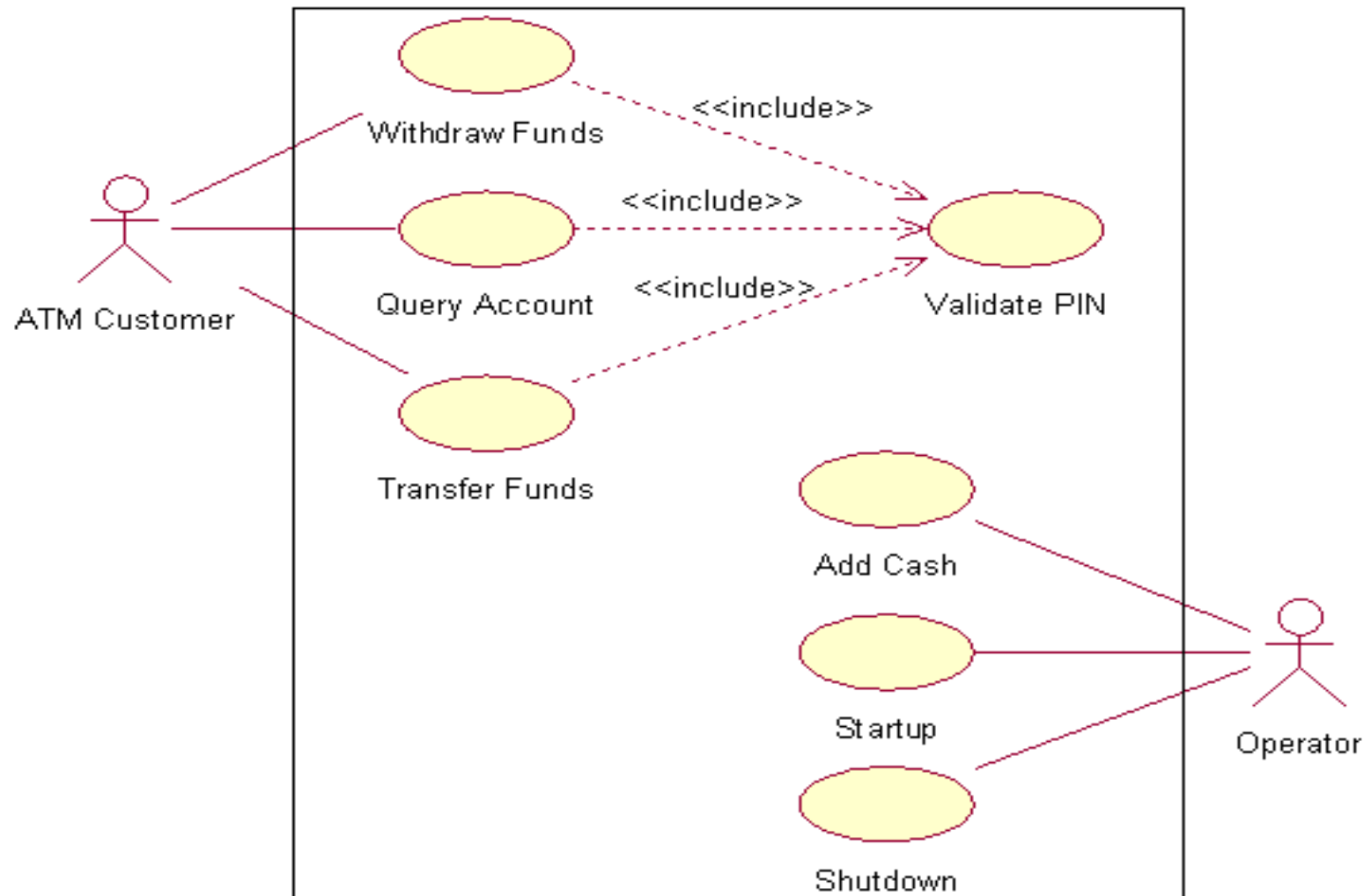
Primer dijagrama slučaja korišćenja

- Student može da izabere određeni broj kurseva koje želi da sluša
- Školarina zavisi od broja i trajanja kurseva
- Profesor daje ponudu za kurs koji namerava da drži u semestru
- Referent održava informacije o studentima, profesorima i nastavnom planu
- Neki profesori nisu redovno zaposleni, već su honorarni
- Pre nego što se referentu dozvoli rad mora se izvršiti provera prava pristupa
- Provera prava pristupa se može temeljiti na unosu lozinke, ili na magnetnoj kartici

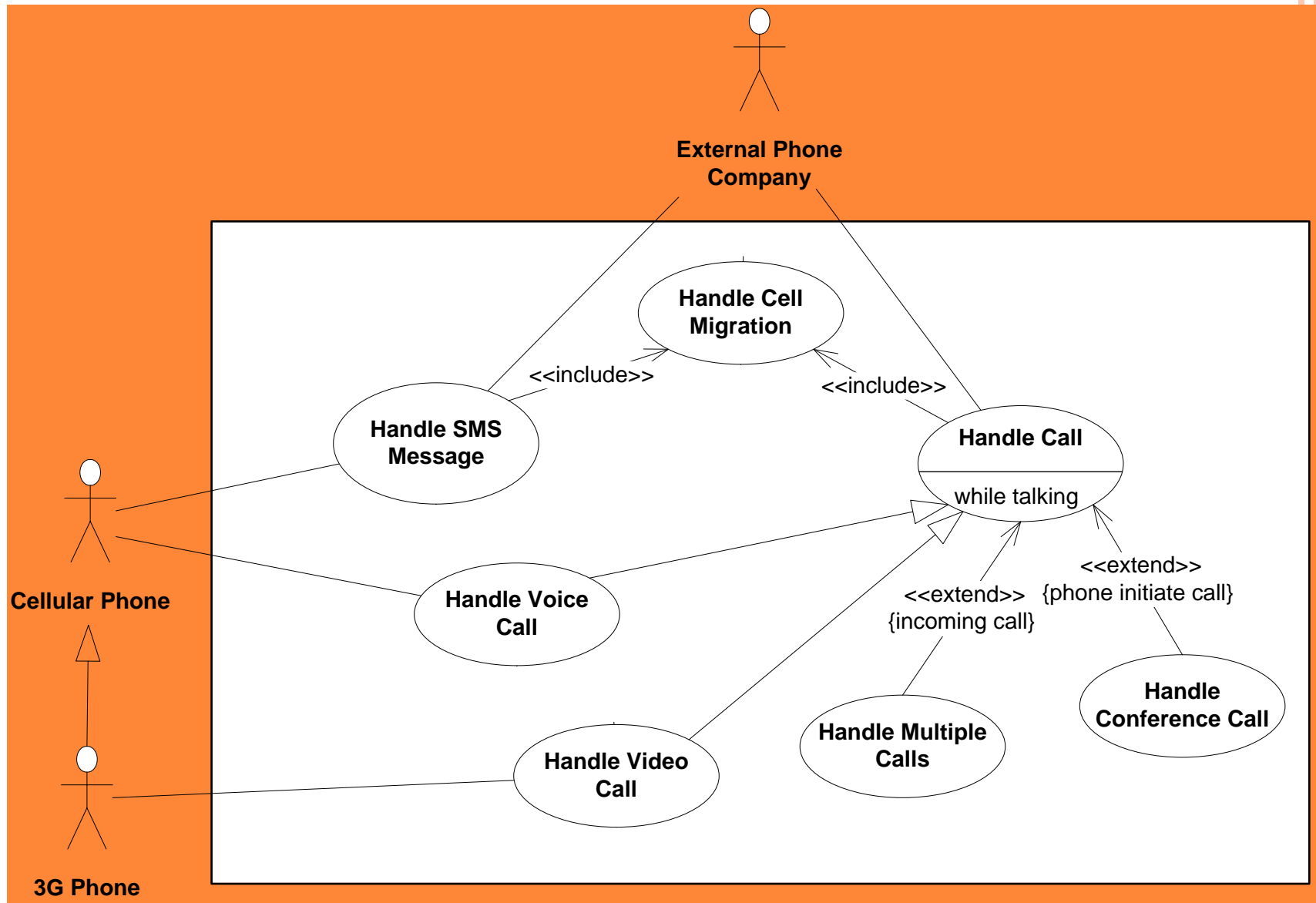




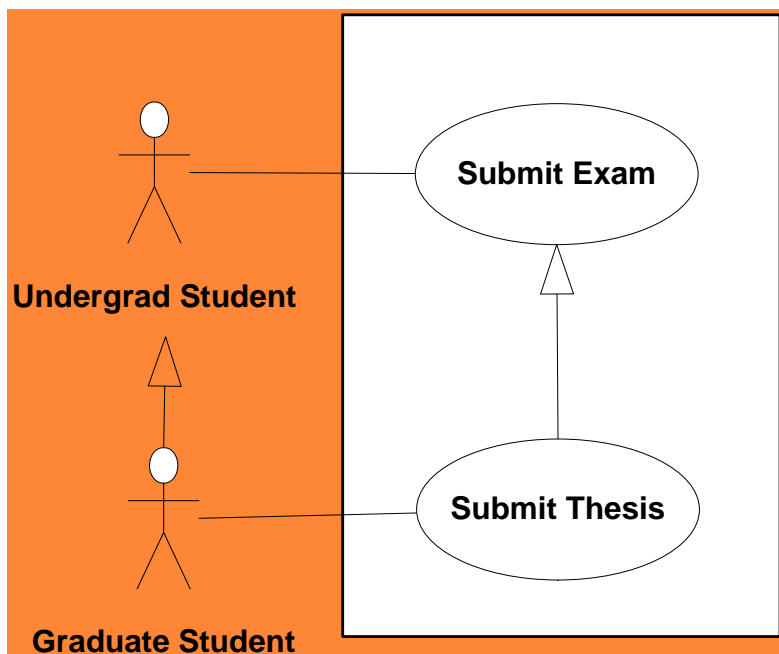
PRIMER: BANKOMAT



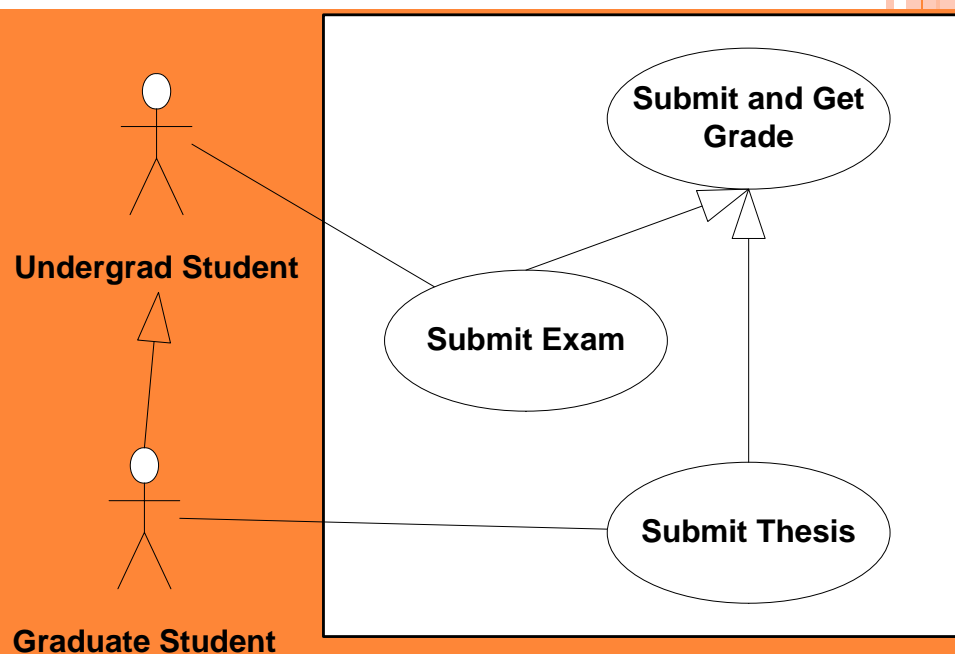
PRIMER: MOBILNA TELEFONIJA



GENERALIZACIJA: OPASNOST!!!



Pogrešno: Undergraduate može da radi tezu



Dobro: Samo diplomirani može da radi tezu

PREPORUKE

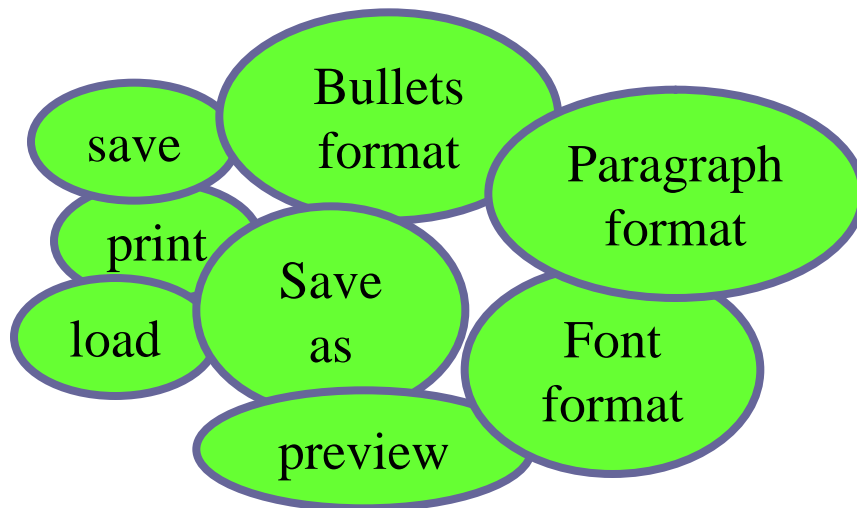
- Svaki use case dijagram mora da ima:
 - Bar jedan use case
 - Bar jednog aktera
 - Bar jedan use case koji je u asocijaciji sa akterom
 - Nema asocijacija izmedju aktera
 - Nema asocijacija izmedju use case
 - Imenovati svaki use case i aktera
 - Ne označavati asocijacije
 - Ne više od 10 do 15 use case
 - Svi use case na istom nivou apstrakcije
 - Obratiti pažnju na problem kod generalizacije aktera



KAKO MODELOVATI?

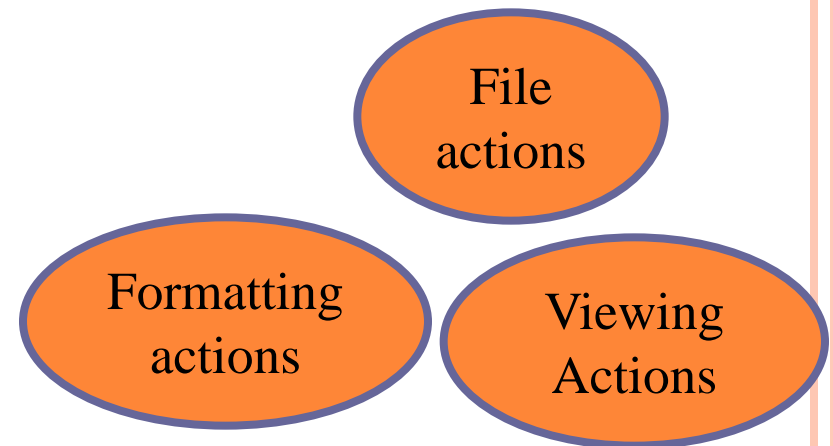
Bottom-up pristup

Polazi se od identifikovanja svih scenarija, crtanja na strani i postupnoj kombinaciji



Top-down pristup

Polazi se od pregleda funkcionisanja sistema i podele na više use case



Kombinovanje

II – Dijagrami klasa (Klasni dijagrami)

- **Dijagram klasa prikazuje skup klasa, interfejsa i kolaboracija i njihove relacije**
- **Klasni dijagram je graf sačinjen od temena (stvari) povezanih granama (relacijama)**

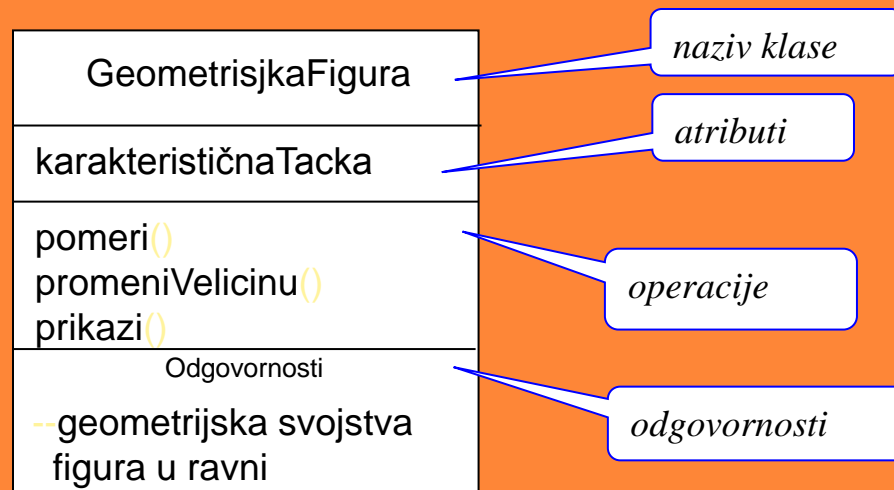
Elementi dijagrama klasa:

- **Stvari**: klase, interfejsi, kolaboracije, paketi, objekti
- **Relacije**: zavisnosti, generalizacije, asocijacije i realizacije
- **Specificira logičke i statičke aspekte modela**
- **Klasni dijagrami su najčešći u objektnom modeliranju**
 - **Alat RR podržava generisanje koda iz klasnih dijagrama**



Klasa

- Klasa je opis skupa objekata koji imaju iste atribute, operacije, relacije i semantiku
- Klasa implementira jedan ili više interfejsa
- Klase opisuju apstrakcije iz domena problema kao i apstrakcije iz domena rešenja
- Koriste se da reprezentuju softverske stvari, hardverske stvari i konceptualne stvari



Naziv klase:

- jednostavan – samo tekst (string)
- sa putanjom - <naziv paketa> :: <jednostavan naziv>,
npr: `C++::graphics::Circle`

Atributi:

- Atributi su imenovana svojstva klase koja opisuju opsege vrednosti koje instance tog svojstva mogu sadržati
- Drugi nazivi: članovi podaci (C++), polja (Java)
- Mogu se pisati sa tipom i podrazumevanom vrednošću

Primer: `izbor:Boolean=false`



Operacije:

- Operacije su implementacije servisa koji se mogu zahtevati od bilo kog objekta klase
- Može se pisati sa potpisom koji sadrži listu argumenata sa eventualnim tipovima i podrazumevanim vrednostima, kao i tipom rezultata

```
transliraj(novaPoz:Poz=koordPocetak):Poz
```

Odgovornosti:

- Odgovornosti klase predstavljaju stavku njenog ugovora
- Pišu se kao slobodan tekst u zasebnom odeljku (svaka počinje sa --)
- Svaka dobro strukturirana klasa bi trebalo da ima barem jednu i ne više od nekoliko odgovornosti



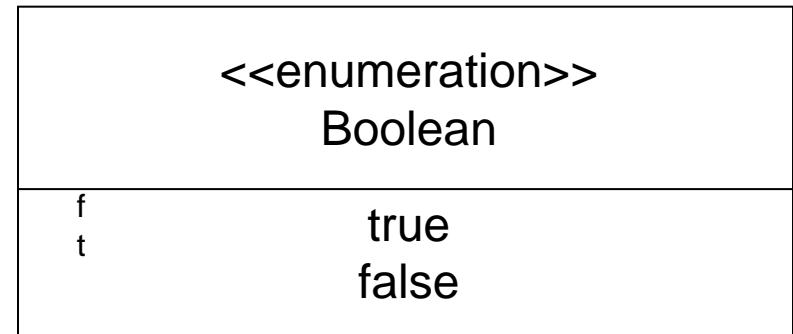
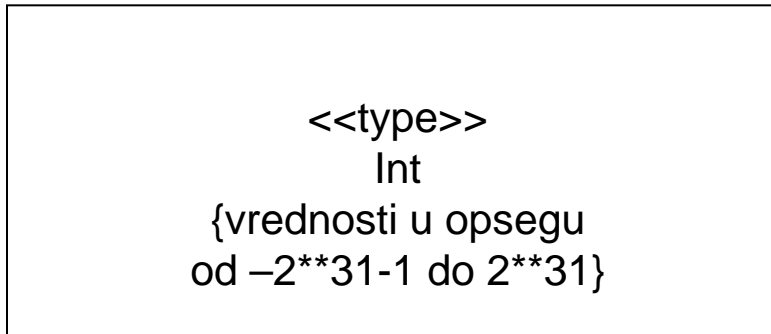
Dodatne mogućnosti:

- Klasa može sadržati prazne odeljke, a može biti i bez odeljaka
- Prazan odeljak atributa/operacija ne znači da ih klasa nema, već da nisu relevantni za dati pogled (dijagram)
- Mogu se koristiti i tri tačke (...) da naznače postojanje dodatnih atributa/operacija
- Atributi/operacije se mogu grupisati a svakoj grupi može prethoditi deskriptivni prefiks
- Deskriptivni prefiks se piše kao stereotip,
npr: <<constructor>>, <<query>>
- Apstraktna klasa i apstraktna operacija – naziv se piše *italicom*
- Zajednički članovi (atributi i operacije) klase – pišu se podvučeno
- **Vizibilitet** (prava pristupa): znak se piše ispred atributa/operacije:
 - javni član: + (podrazumevano)
 - zaštićeni član: #
 - privatni član: -



Modeliranje primitivnih tipova

Primitivni tipovi se modeliraju stereotipovima klasa



Elementi dijagrama klasa:

- **Stvari:** klase, interfejsi, kolaboracije, paketi, podsistemi, objekti
- **Relacije:** zavisnosti, generalizacije, asocijacije, realizacije



Osobine klasa

- Koren hijerarhije klasa (*root*) je klasa koja nema roditelje
- List hijerarhije klasa (*leaf*) je klasa koja nema potomke, tj. ona iz koje se ne može izvoditi
- Osobine "koren" i "list" se pišu unutar zagrada { } u odeljku naziva klase
- Multiplikativnost - osobina klase koja ograničava broj njenih instanci
- Specifične vrednosti multiplikativnosti:
 - 0 – uslužna klasa, sadrži samo zajedničke (klasne, statičke) attribute i operacije
 - 1 – *singleton* klasa
- Podrazumevani slučaj je proizvoljan broj instanci
- Multiplikativnost se navodi u gornjem desnom uglu



Sintaksa atributa

- Sintaksa:

[vizibilitet] ime [multiplikativnost][:tip][=inicijalna vrednost][{osobina}]

- Multiplikativnost se primenjuje i na attribute klase,

npr. `consolniProzor[2..*]:Prozor`

- Neke od osobina atributa:

- `changeable` – nema restrikcija za promene vrednosti
(podrazumevana osobina)
- `addOnly` – samo za `multip.>1`: vrednosti se mogu dodavati,
ali se ne mogu uklanjati ni menjati
- `frozen` – vrednost atributa se ne može menjati nakon
inicijalizovanja
- `composite` – atribut složene strukture
- `readOnly` – ne može se menjati, dodavati ni uklanjati nakon inicijal.



Sintaksa operacije

- Sintaksa:

[vizibilitet] ime ([lista argumenata]) [: tip rezultata] [{osobina}]

- Sintaksa argumenta: [smer] ime : tip [= podrazumevana vrednost]

- Smer može biti:

- **in** – ulazni argument, ne sme biti modifikovan (default vrednost)
- **out** – izlazni argument, mora se inicijalizovati
- **inout** – ulazno-izlazni argument, može se modifikovati

Osobine operacije:

- **query** – izvršenje ne menja stanje objekta, operacija je čista funkcija bez bočnih efekata
- **leaf** – operacija nije polimorfna i ne može se redefinisati u izvedenoj klasi
- **concurrency = vrednost** – šta se garantuje pri izvršenju u konkurentnoj sredini
- **sequential** – pozivaoci moraju obezbediti da je samo jedna nit u jednom trenutku poziva
- **guarded** – operacija garantuje sinhronizaciju svih niti koje pristupaju datom objektu (monitor)
- **concurrent** – operacija se izvršava u konkurentnoj sredini kao atomska

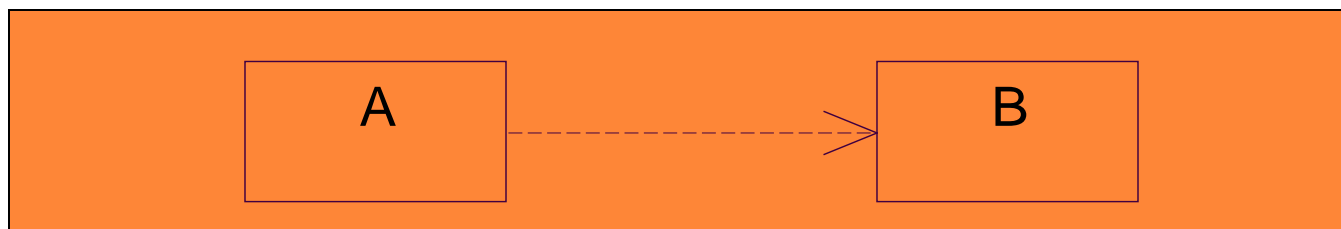
Relacije

Na klasnim dijagramima se pojavljuju sve relacije(1,2,3 su češće):

- 1) zavisnost (*dependency*)
- 2) asocijacija (*association*)
- 3) generalizacija (*generalization*)
- 4) realizacija (*realization*)

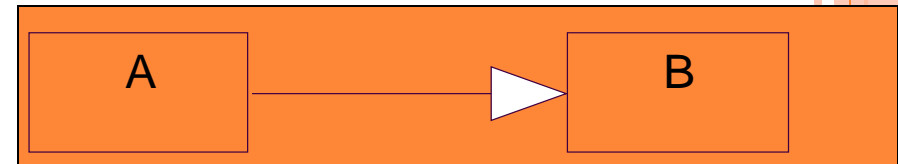
1) Zavisnost

- Povezuje stvari kod kojih izmena nezavisne stvari utiče na ponašanje zavisne stvari
- Zavisna stvar koristi nezavisnu stvar
- Grafička notacija: klasa A zavisi od klase B
- Često se koristi kad je jedna klasa (B) tip parametra operacije druge klase (A)



2) Generalizacija

- Povezuje opštije (superklasa ili roditelj) sa specijalizovanim (subklasa ili dete) stvarima
- Grafička notacija: klasa A je dete, B je roditelj
- Drugi nazivi relacije: vrsta (*is-a-kind-of*), izvođenje A se izvodi iz B), nasleđivanje
- Generalizacija znači da se objekti dece mogu pojaviti gde god se očekuje objekat roditelja
- Deca nasleđuju osobine svojih roditelja, naročito attribute i operacije
- Operacija deteta koja ima isti potpis kao operacija roditelja redefiniše operaciju roditelja
- Redefinicija operacije omogućava njeno polimorfno ponašanje
- Klasa koja ima samo jednog roditelja koristi jednostruko nasleđivanje
- Klasa koja ima više roditelja koristi višestruko nasleđivanje



3) Asocijacija

- Asocijacija je strukturna relacija
- Specificira da li su instance jedne stvari povezane sa instancama druge stvari
- Asocijacija je (ako se drugačije ne kaže) bidirekciona veza (dvosmerna navigabilnost)
- Dozvoljena je i asocijacija sa samim sobom (postoje veze između objekata iste klase)
- Uobičajeno je da asocijacija povezuje dve klase (**binarna asocijacija**)
- Moguće je da asocijacija povezuje i više klasa (**n-arna asocijacija**)
- Grafička notacija: klasa A je u asocijaciji sa klasom B



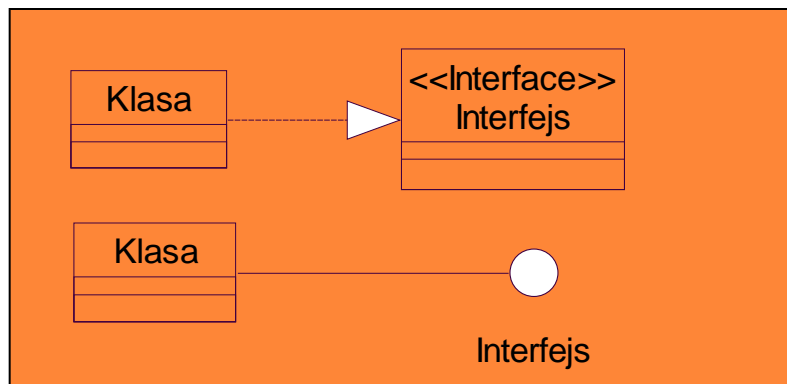
4. Realizacija

- Semantička relacija između klasifikatora (element modela koji poseduje strukturu i ponasanje: klasa, interfejs, tip podatka, signal, komponenta, cvor, slučaj upotrebe, podsistem) od kojih jedan specificira ugovor a drugi garantuje njegovu implementaciju

- Grafička notacija:

kanonička forma

skraćena forma

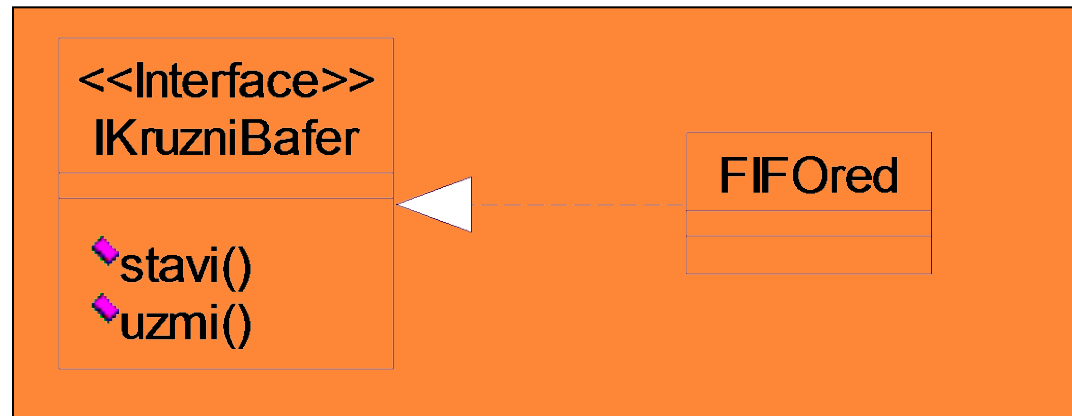


- Realizacija se koristi u dva konteksta:

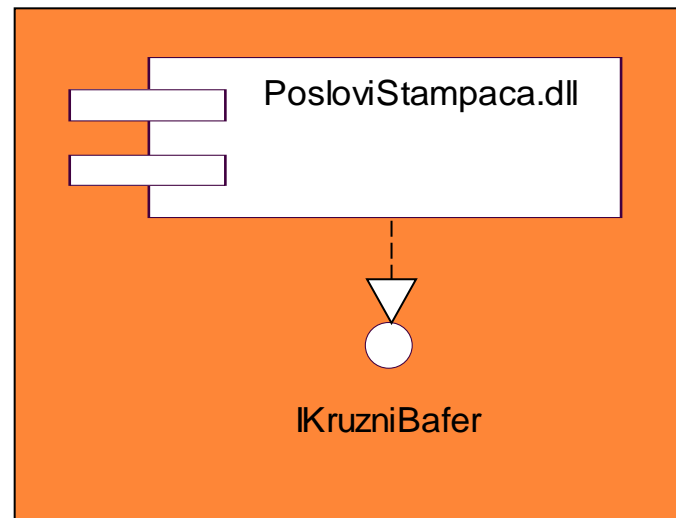
- 1) kontekst interfejsa (realizuje ga klasa ili komponenta)
- 2) kontekst slučajeva korišćenja (realizuje ga kolaboracija)

Primeri realizacije

Klasni dijagram:



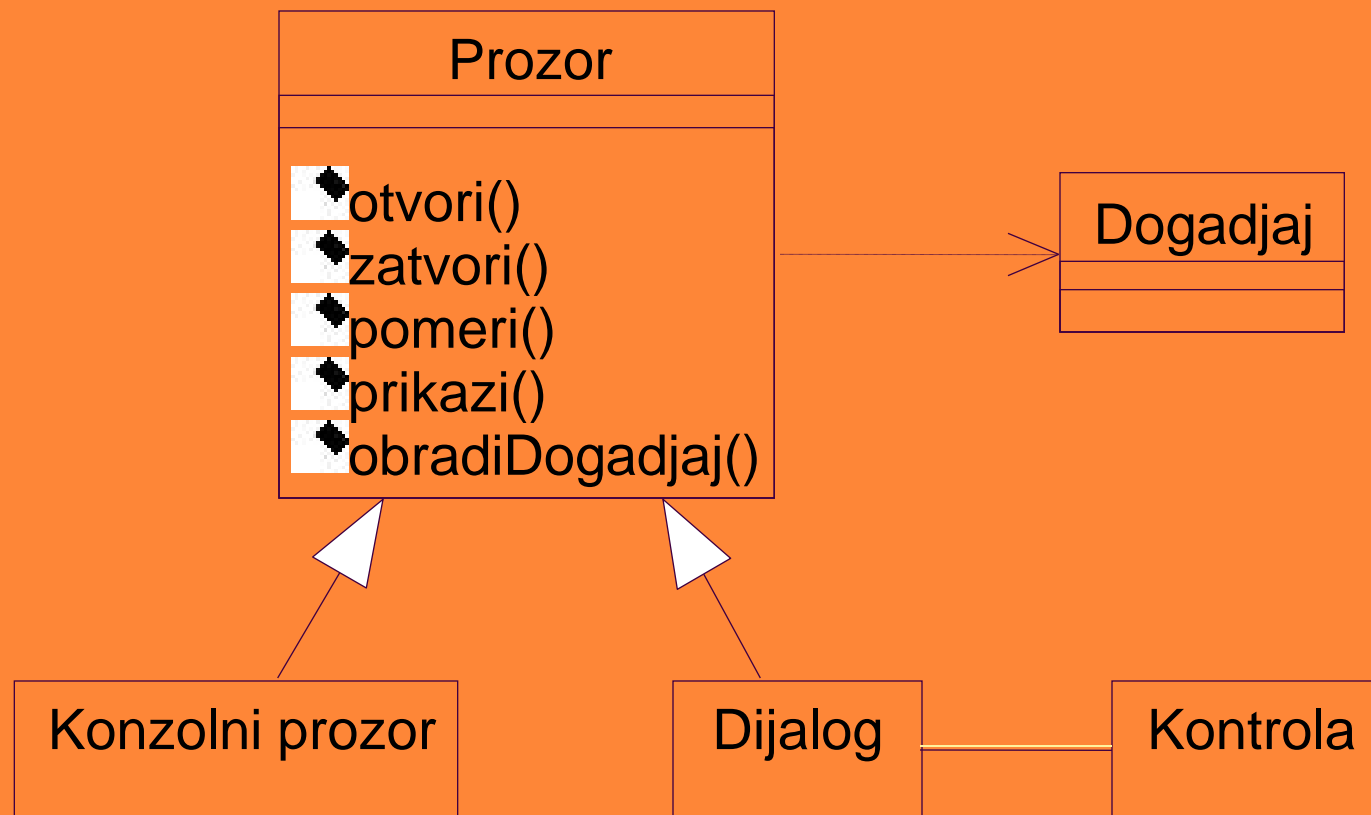
Dijagram komponenata:



Dijagram slučajeva korišćenja:



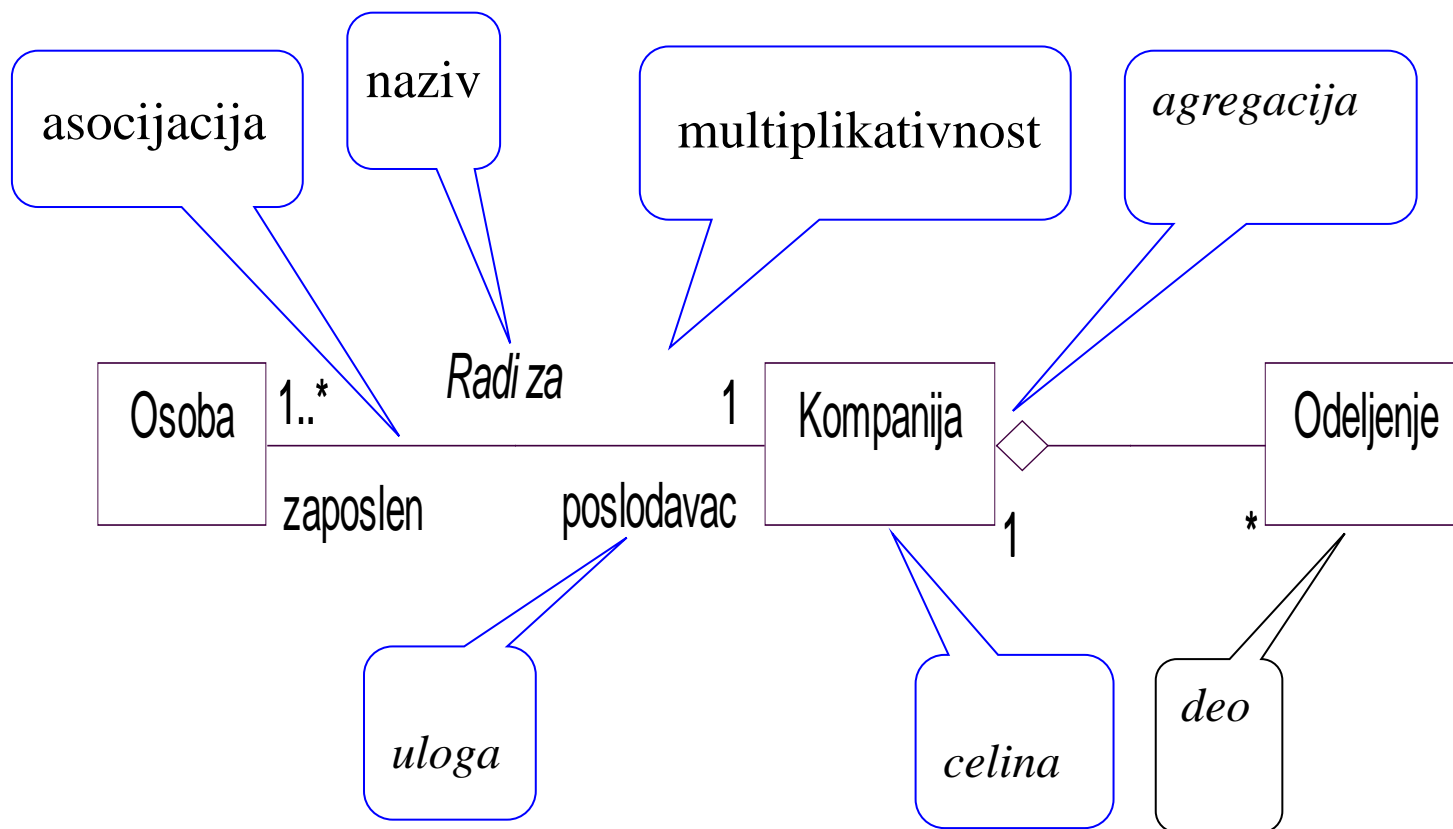
Primer osnovnih relacija



Ukrasi asocijacija:

Na asocijaciji se mogu pojaviti sledeći ukrasi:

ime, uloge, multiplikativnost, agregacija/kompozicija, pravo pristupa



Multiplikativnost

Na jednoj strani asocijacije označava se broj objekata klase sa te strane koji su u vezi sa tačno jednim objektom sa druge strane relacije

Može biti:

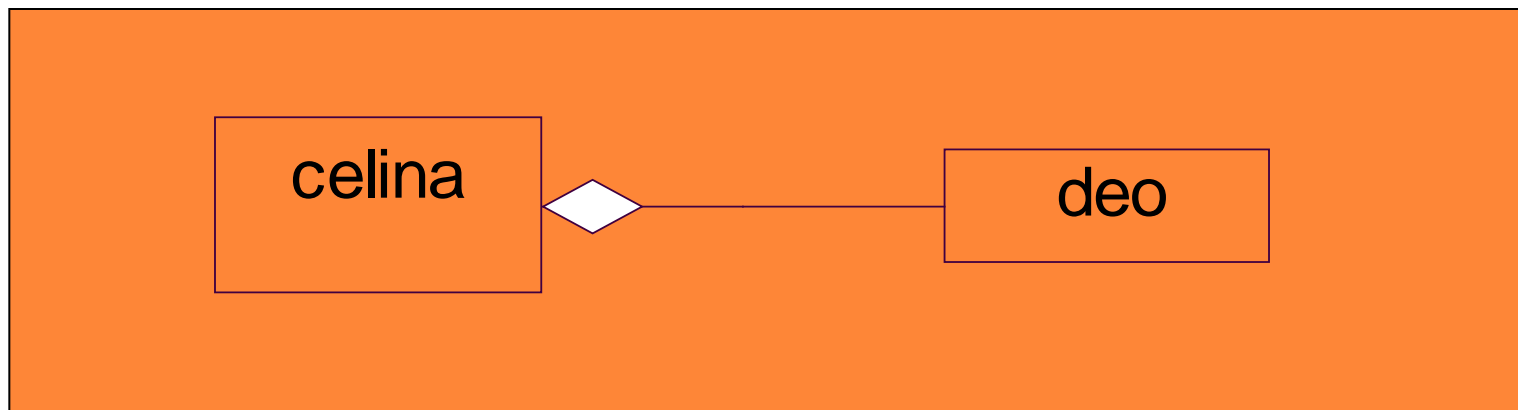
- nijedan (0)
- tačno jedan (1)
- proizvoljno mnogo (*) – može i nula
- opseg (npr. 2..*)
- izraz (npr. 0..1,3..4,6..* - proizvoljan broj osim 2 i 5) (**UML 2.0 ne dozvoljava**)



Agregacija

- Vrsta asocijacije kod koje jedna strana igra ulogu celine a druga ulogu dela (*whole/part*)
- Celina sadrži delove ("*has-a*" relacija)
- Agregacija ne govori ništa o uzajamnom odnosu životnih vekova celine i dela
- Deo u agregaciji može biti zajednički deo više celina

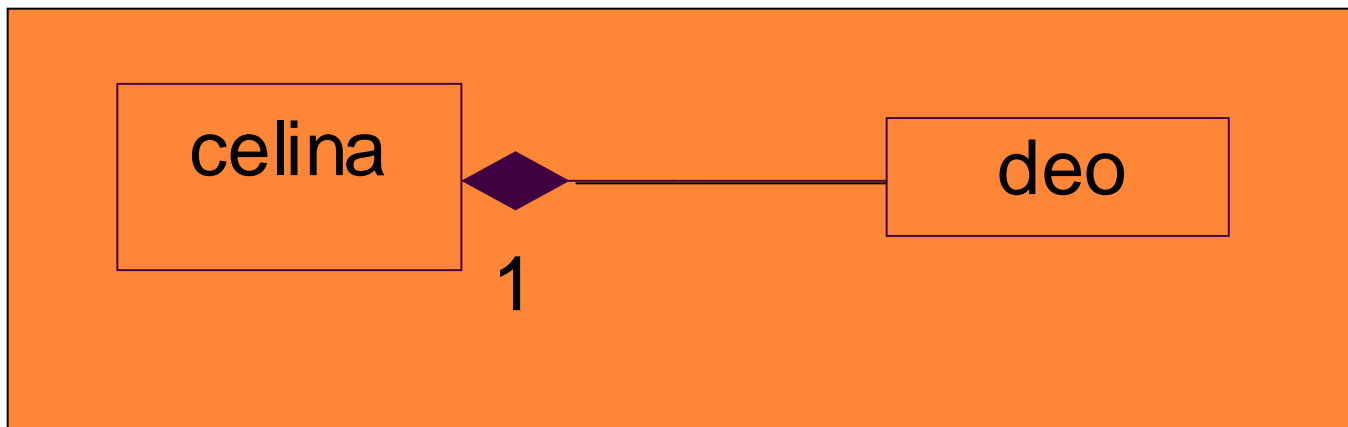
Grafička notacija:



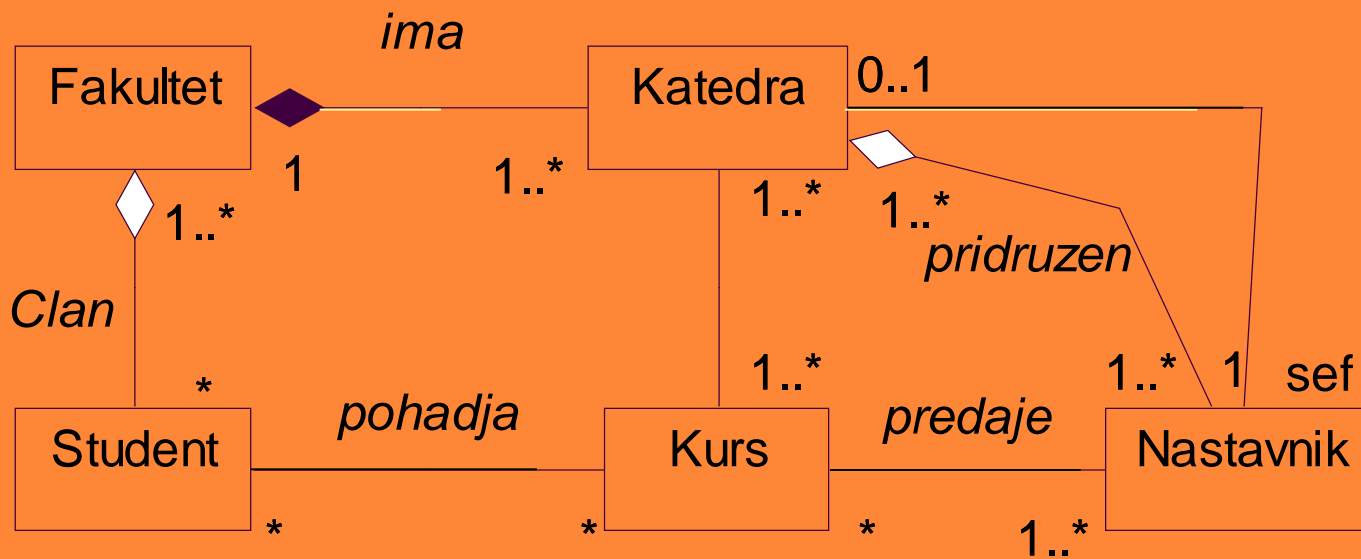
Kompozicija

- Asocijacija kod koje postoji odnos celina/deo, ali je celina odgovorna za životni vek dela
- Agregacija sa strogim vlasništvom i koincidentnim životnim vekom dela u odnosu na celinu
- Deo može nastati u toku života celina i može biti uništen pre uništenja celine
- Deo u kompoziciji može biti deo samo jedne celine

Grafički notacija:

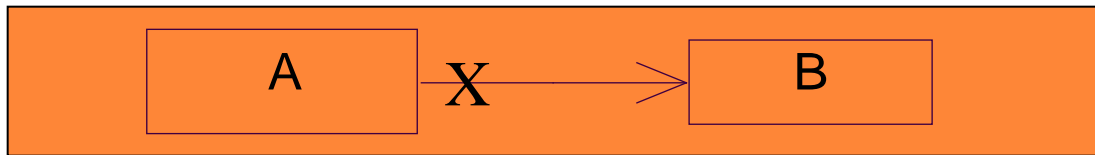


Primer asocijacija



Navigacija

- Asocijacija je podrazumevano bidirekciona
- Kada je potrebna unidirekciona asocijacija (navigabilnost u jednom smeru) – strelica
- Krstić označava da nema navigabilnosti na toj strani
- Za asocijaciju bez ukrasa navigabilnosti se smatra da je navigabilnost neodređena



Vizibilitet

Ograničava vizibilitet objekata u asocijaciji za spoljašnji svet

Označava se sa **+, #, -, ~** ispred imena uloge odgovarajuće strane relacije

+ znači da objektima sa te strane mogu pristupati svi objekti preko objekata sa druge strane (podrazumevana vrednost)

- znači da objektima klase sa te strane mogu pristupati samo objekti klase sa druge strane

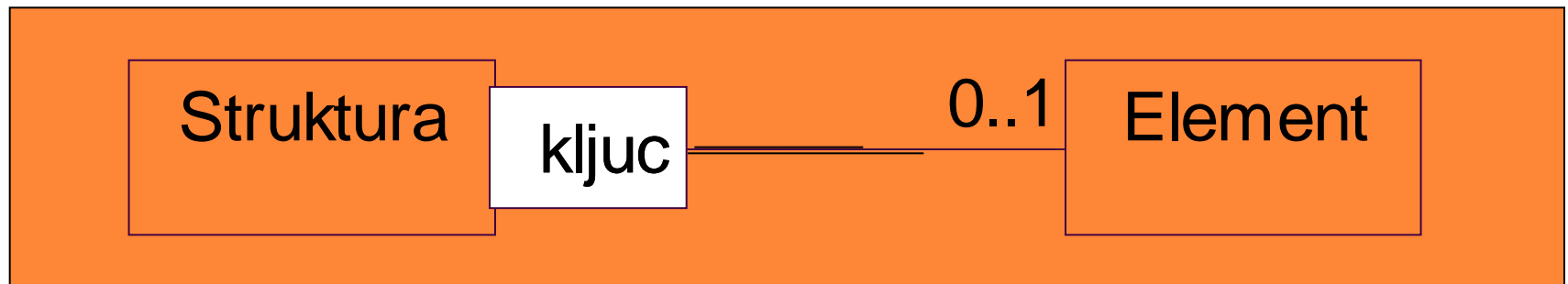
znači da i objekti klase izvedenih iz klase sa drugog kraja asocijacije imaju pristup

~ znači da i objekti klase iz istog paketa kao klasa sa drugog kraja asocijacije imaju pristup



Kvalifikacija

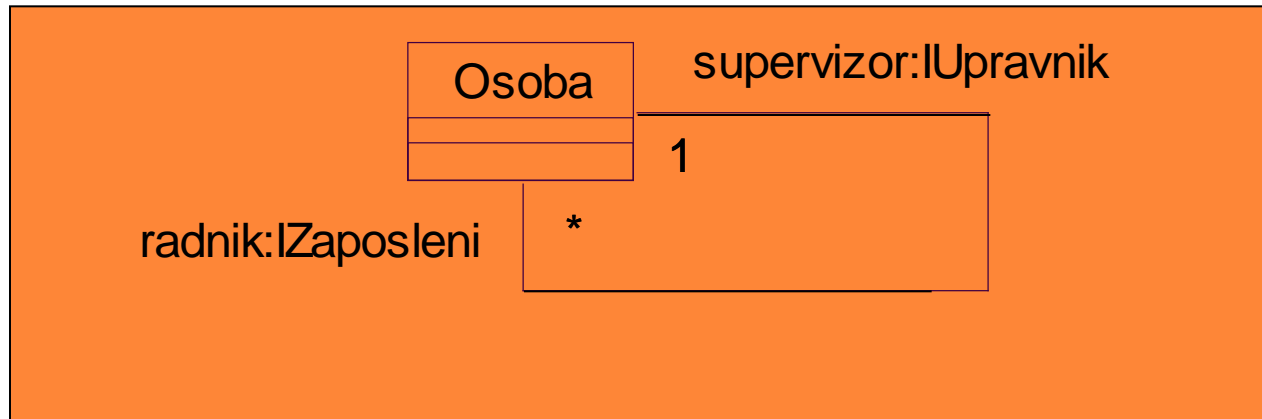
- Koristi se da označi ključ (**kvalifikator**) koji se koristi za selekciju objekta iz neke strukture
- Objekat strukture za datu vrednost ključa selektuje jedan ili grupu elemenata strukture
- Primeri struktura kojima se pristupa pomoću ključa su heš tabela, B-stablo
- Kvalifikator može imati više atributa



Specifikator interfejsa

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

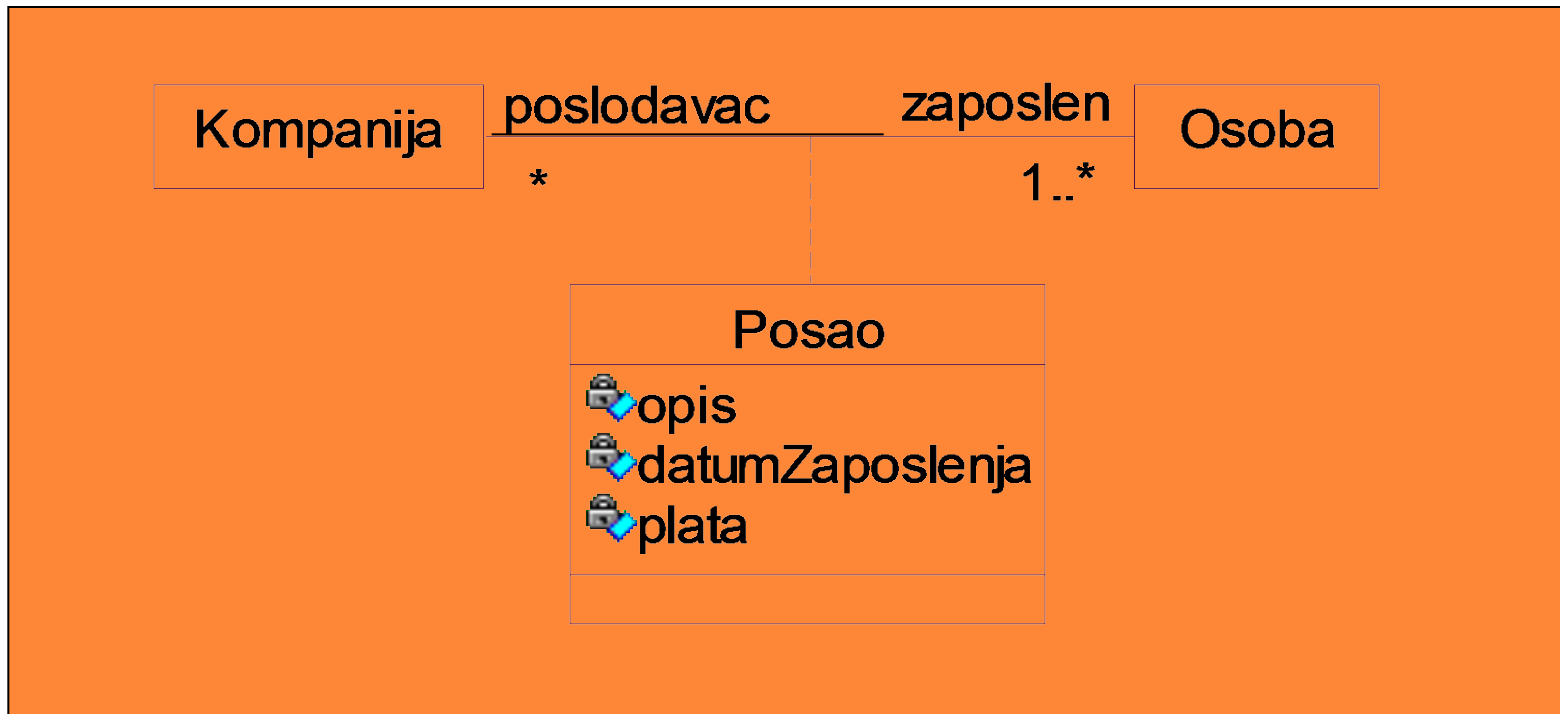
Primer:



Klasa asocijacija

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

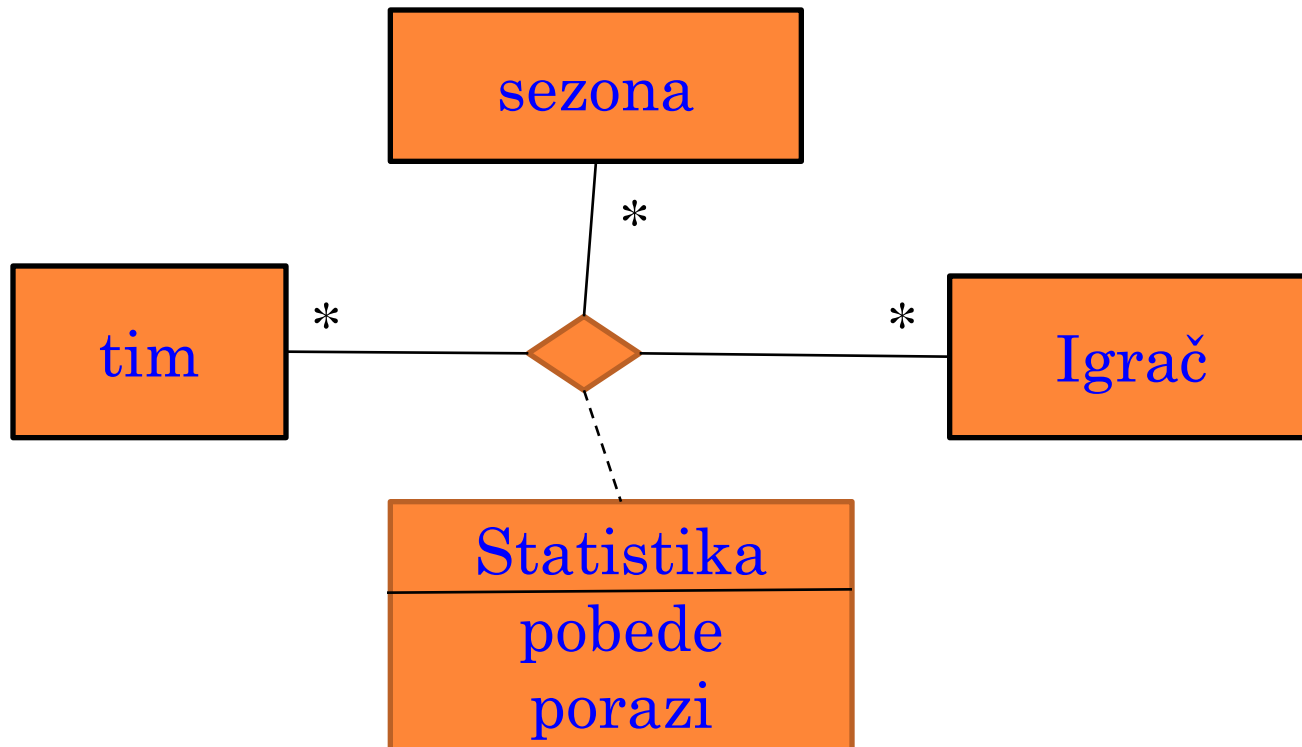
Primer:



N-arna asocijacija

- Sama asocijacija može povezivati više klasa i zove se n-arna asocijacija
- Instanca n-arne asocijacije je n-torka vrednosti klasa

Primer ternarne asocijacije:

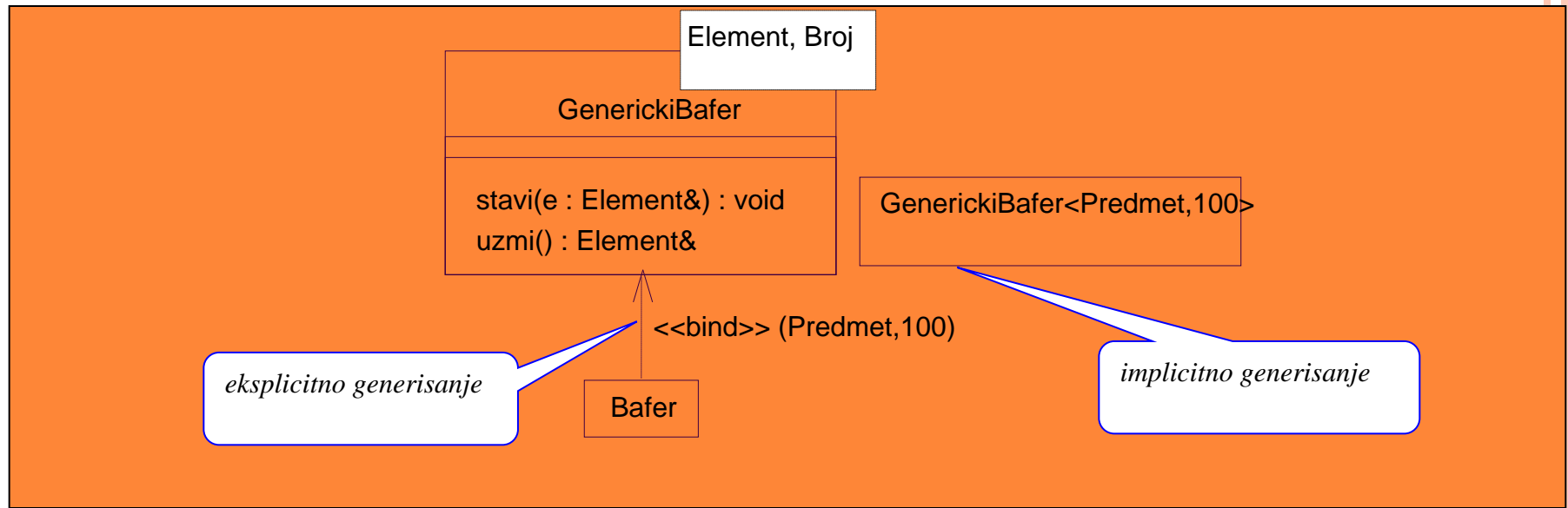


Šabloni

- Šablon (*template*) je parametrizovani element

Primer na C++:

```
template <class Element, int Broj> class GenerickiBafer{  
    public:  
        virtual void stavi(const Element&);  
        virtual Element& uzmi()const;  
        ...  
};  
typedef GenerickiBafer<Predmet,100> Bafer;  
GenerickiBafer<Predmet,100> bafer;
```



Standardni stereotipovi klasifikatora

- **utility** – klasa čiji su atributi i operacije zajednički za sve instance klase
- **stereotype** – klasifikator je stereotip koji se može primeniti na druge elemente
- **metaclass** – klasifikator čije su instance klase
- **power type** – klasifikator čije su instance sva deca datog roditelja

Standardni stereotip relacije generalizacije

- **implementation** – dete nasleđuje implementaciju roditelja, ali je čini privatnom i ne podržava interfejs roditelja, pa ne može zamenjivati roditelja



PITANJA



Standardni stereotipovi relacije zavisnosti

1) Između klasa i/ili objekata na dijagramu klasa:

use – semantika izvornog elementa zavisi od semantike javnog dela odredišta

bind – izvor instancira ciljni šablon pomoću zadatih parametara

friend – izvoru se daju posebna prava pristupa odredištu

instanceOf – izvor je objekat koji je instanca odredišnog klasifikatora

instantiate – izvor kreira instance odredišta

derive – izvor se može izračunati na osnovu odredišta; relacija između dva atributa ili dve asocijacije: jedan je konkretan, a drugi konceptualan

refine – izvor je finiji stepen apstrakcije od odredišta

powertype – odredište je *powertype* izvora

2) Između paketa:

access – izvornom paketu se garantuje pravo pristupa elementima odredišnog

import – javni sadržaj odredišnog paketa ulazi u prostor imena izvornog paketa (kao da su imena odredišnog paketa deklarirana u izvornom paketu)

