SLUČAJEVI KORIŠĆENJA

OSNOVNI KONCEPTI

- Nijedan sistem ne postoji izolovano bez interakcije sa ljudskim ili automatizovanim izvođačima, koji taj sistem koriste iz određenih razloga
- Izvođači koji koriste određeni sistem očekuju od njega da se ponaša na predvidive načine
- Korisničke funkcije se primenjuju da se prikaže željeno ponašanje sistema koji se razvija, a da se pri tome ne morra definisati kako se to ponašanje realizuje
- Korisničke funkcije obezbeđuju način da se projektanti sporazumevaju sa krajnjim korisnicima sistema i ekspertima iz posmatrane oblasti

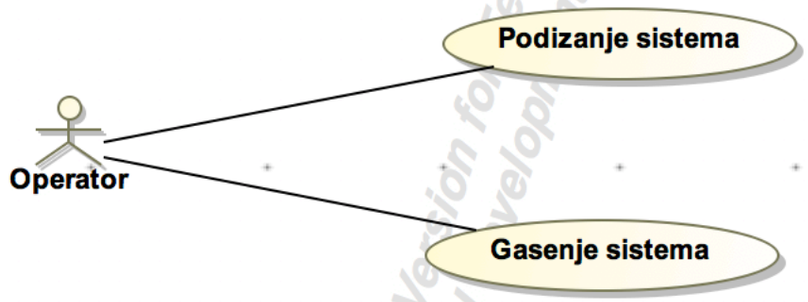
OSNOVNI KONCEPTI

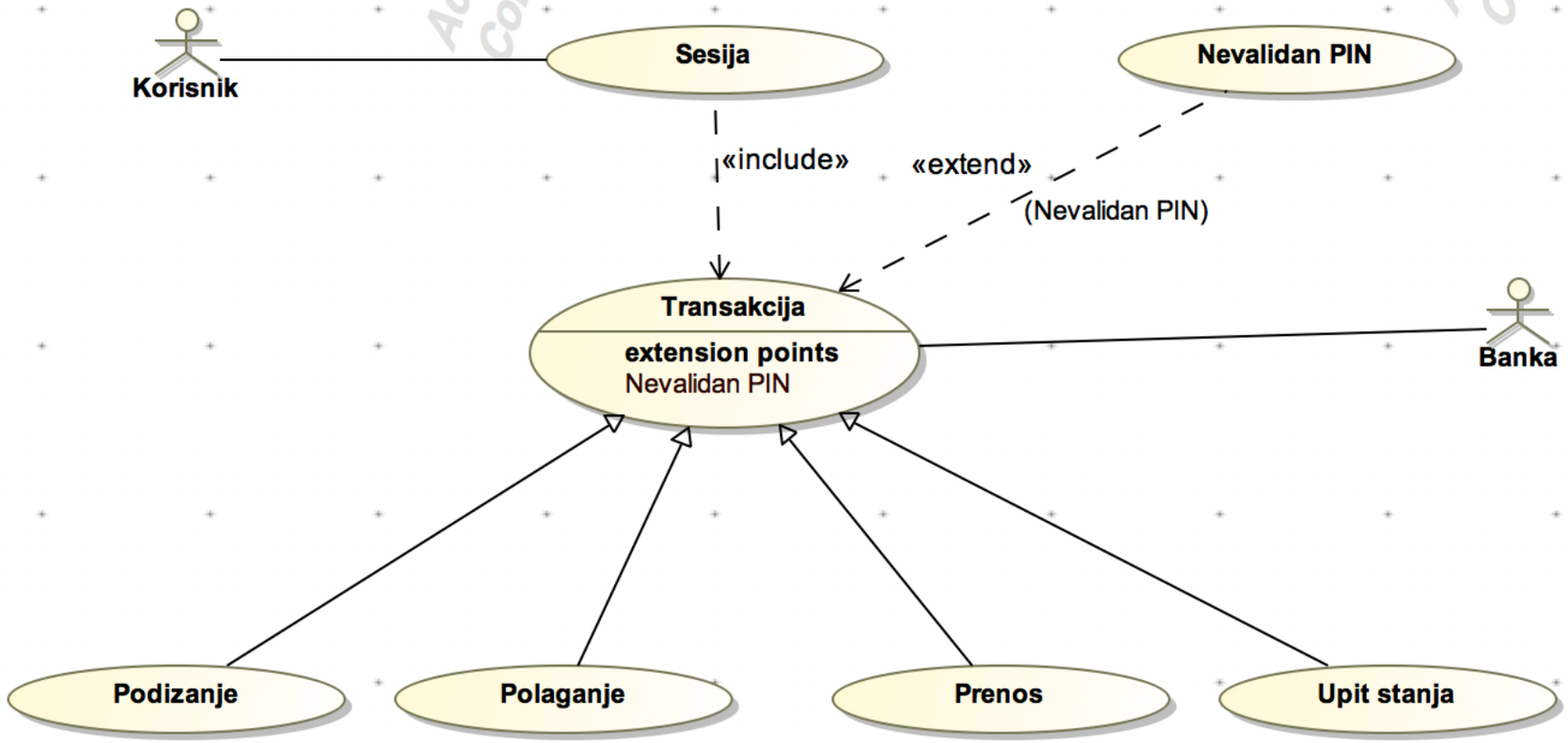
- U UML-u se dijagrami korisničkih funkcija koriste za vizualizaciju sistema, tako da korisnici mogu shvatiti kako da koriste sistem, odnosno da projektanti mogu realizovati posmatrani sistem
- Dobro struktuirane korisničke funkcije prikazuju samo bitne pojedinosti u ponašanju sistema i nisu ni preteano uopštene, ni preterano detaljne

OSNOVNI KONCEPTI

- Na prrimer, da bi se odredilo ponašanje bankomata navodeći u korisničkim funkcijama kakva uzajamna dejstva sa korisnicima treba da postoje, nije potrebno ništa znati o unutrašnjosti sistema
- Korisničke funkcije definišu željeno ponašanje, a ne diktiraju kako to ponašanje treba realizovati
- Na taj način omogućena je komunikacija između krajnjih korisnika i eksperta iz raznih domena sa projektantima sistema (koji treba da naprave sistem koji ispunjava zahteve bez preteranog ulaženja u detalje)

OSNOVNI KONCEPTI





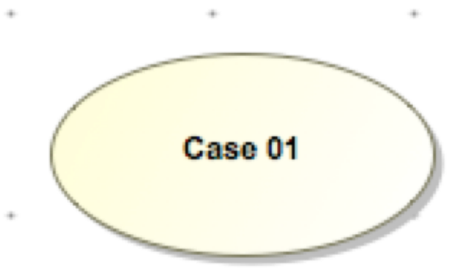
SADRŽAJI DIJAGRAMA KORISNIČKIH FUNKCIJA

- Dijagrami korisničkih funkcija obično sadrže:

1. Korisničke funkcije (Use case)
2. Uloge, izvođače (Actor)
3. Relacije generalizacije, asocijacije i zavisnosti
4. Komentare (Note)
5. Pakete (Package)
6. Granice sistema

KORISNIČKE FUNKCIJE (USE CASE)

- Korisnička funkcija predstavlja funkcionalni zahtev posmatranog sistema koji izvršava određenu, sagledivu količinu posla
- Iz perspektive datog izvođača, korisnička funkcija radi nešto što je izvođaču korisno
- **Korisnička funkcija opisuje šta sistem radi, ali ne i kako to radi**
- UML dijagramom se predstavlja kao oblačić (elipsa) sa svojim imenom koje mora biti jedinstveno

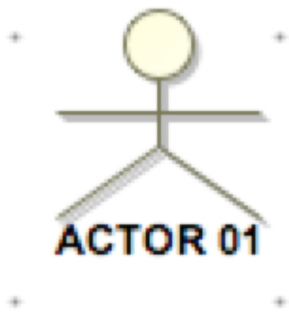


KORISNIČKE FUNKCIJE (USE CASE)

- Korisnička funkcija može imati svoje varijatete
- Jedna korisnička funkcija može biti:
    A. Specijalizovana verzija neke druge korisničke funkcije (ostvaruje se relacijom **generalizacije**)
    B. Sadržana u nekim drugim korisničkim funkcijama (ostvaruje se relacijom **uključivanja**)
    C. Proširenje nekih drugih korisničkih funkcija (ostvaruje se relacijom **proširivanja**)
- Korisnička funkcija predstavlja određenu funkcionalnost sistema iz perspektive korisnika tog sistema i kao takva mora imati ime, koje obično predstavlja glagol

ULOGE, IZVOĐAČI (ACTOR)

- Izvođač predstavlja ulogu koju korisnici korisničkih funkcija izvode kada su u interakciji sa tim korisničkim funkcijama
- Uobičajeno je da izvođač predstavlja ulogu koju čovek, hardverski uređaj ili čak neki drugi sistem igra sa posmatranim sistemom
- Uloga se u UML-u prestavlja kao ljudska figurica



- Izvođači mogu biti povezani sa korisničkom ulogom samo pomoću veze **asocijacije**

ULOGE, IZVOĐAČI (ACTOR)

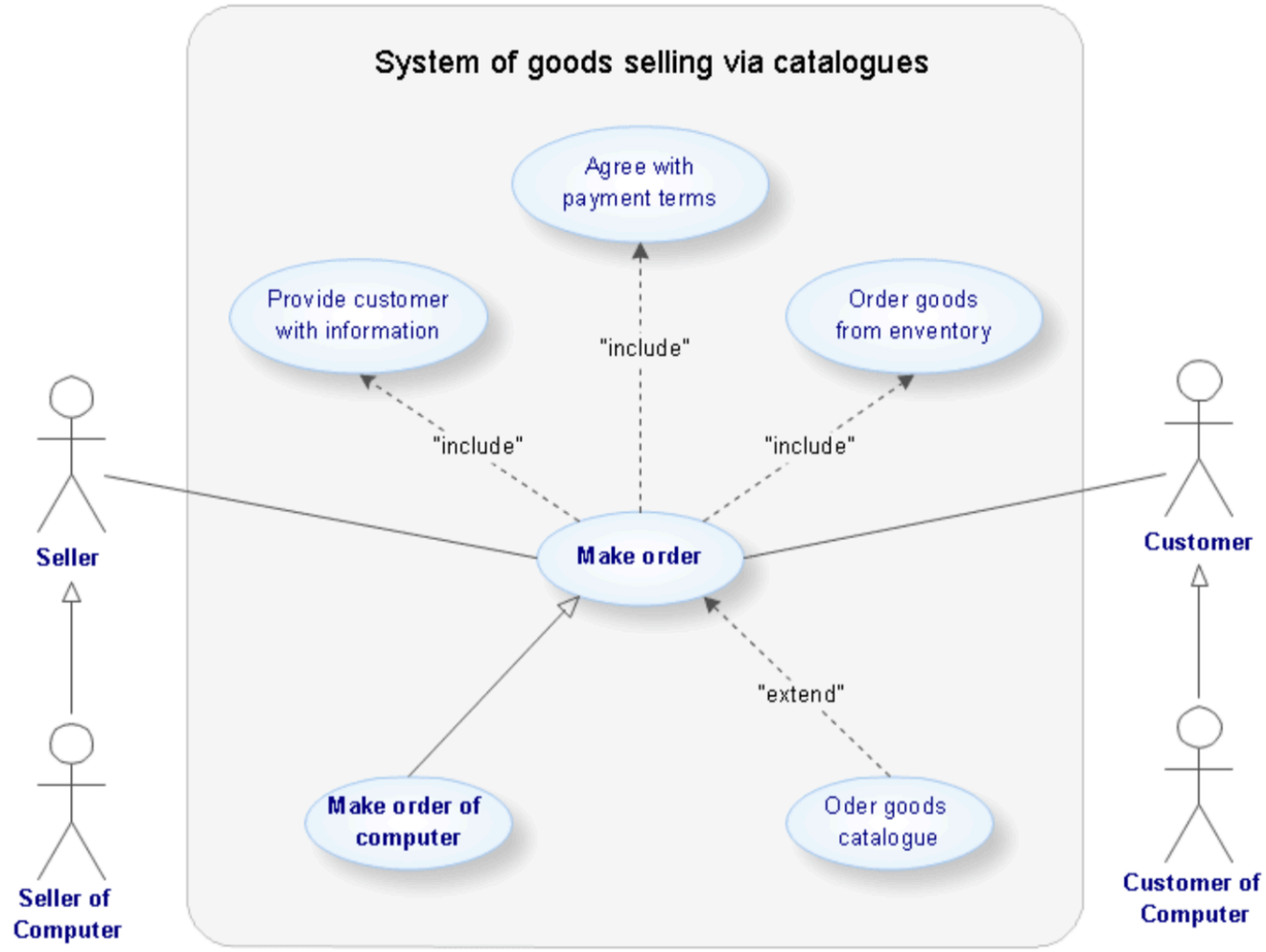
- Da bi se detektovale uloge u posmatranom sistemu postavljaju se sledeća pitanja:
  - Ko su glavni korisnici sistema?
  - Kome su potrebne informacije od sistema?
  - Ko obezbeđuje infomacije potrebne za rad sistema?
  - Ko vrši CRUD podataka?
  - Ko pokeće/gasi sistem?
  - Da li postoje sistemi sa kojima je naš sistem u interakciji?
  - Da li se nešto dešava u tačno određeno vreme?

ULOGE, IZVOĐAČI (ACTOR)

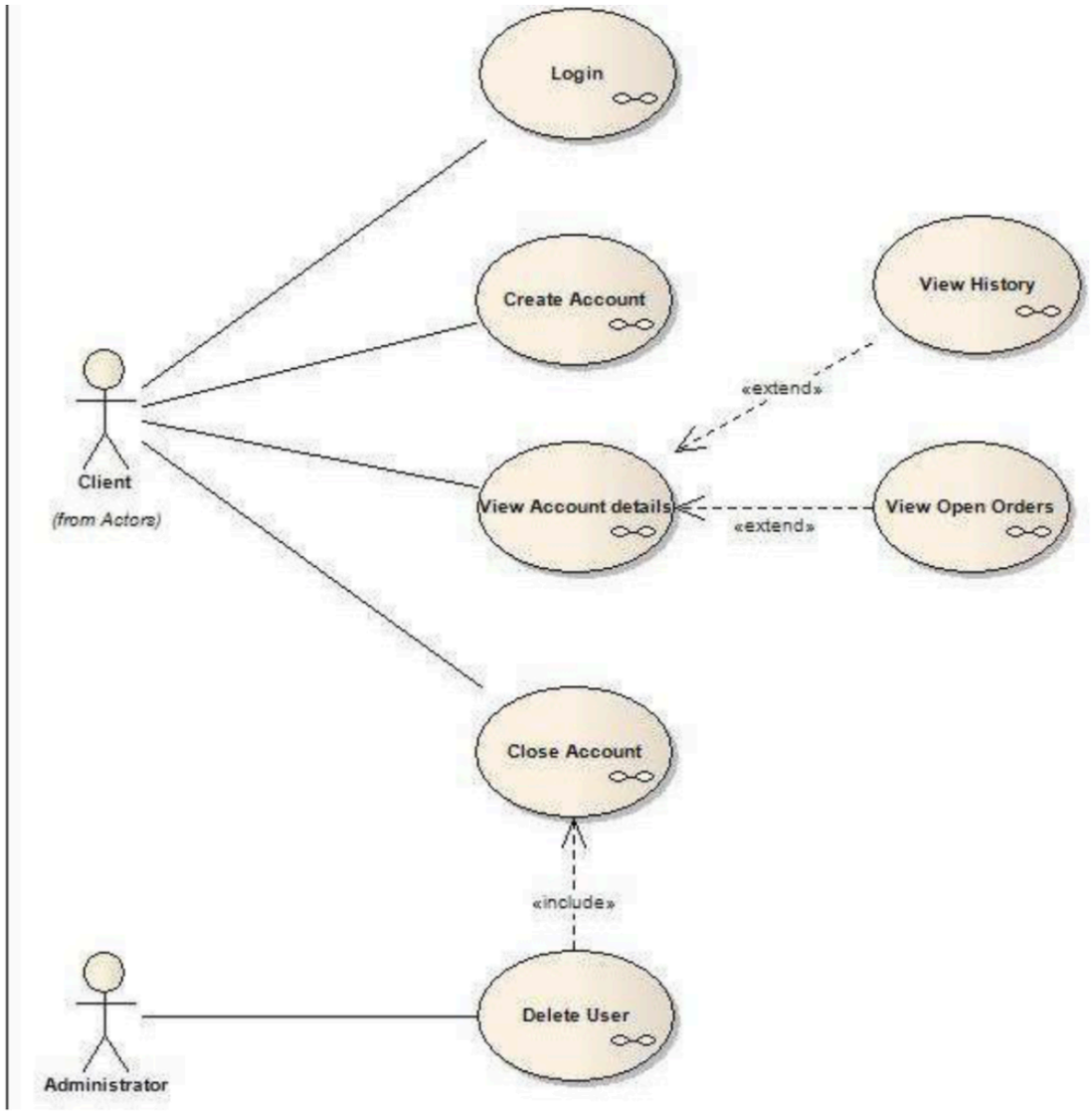
- Da bi se otkili slučajevi korišćenja nekog učesnika postavljaju se sledeća pitanja:
  - Koji su osnovni zadaci ovog učesnika?
  - Šta mu je potrebno da bi mogao da radi svoj posao?
  - Nad kojim podacima treba da vrši CRUD?
  - Da li sistem treba da šalje neke podatke ili obaveštava o ovom učesniku?

RELACIJA GENERALIZACIJE, ASOCIJACIJE I ZAVISNOSTI

- Korisničke funkcije u uloge mogu se organizovati definišući relacije generalizacije, zavisnosti (uključivanja - include i proširivanja - extend) i asocijacije između njih

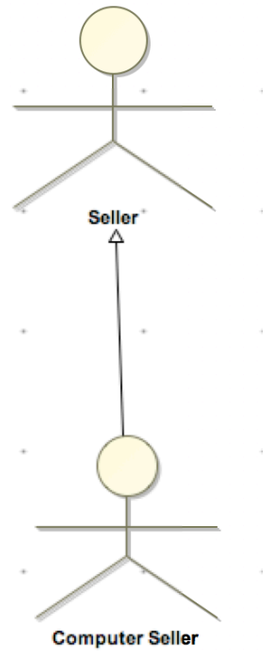


RELACIJA GENERALIZACIJE, ASOCIJACIJE I ZAVISNOSTI



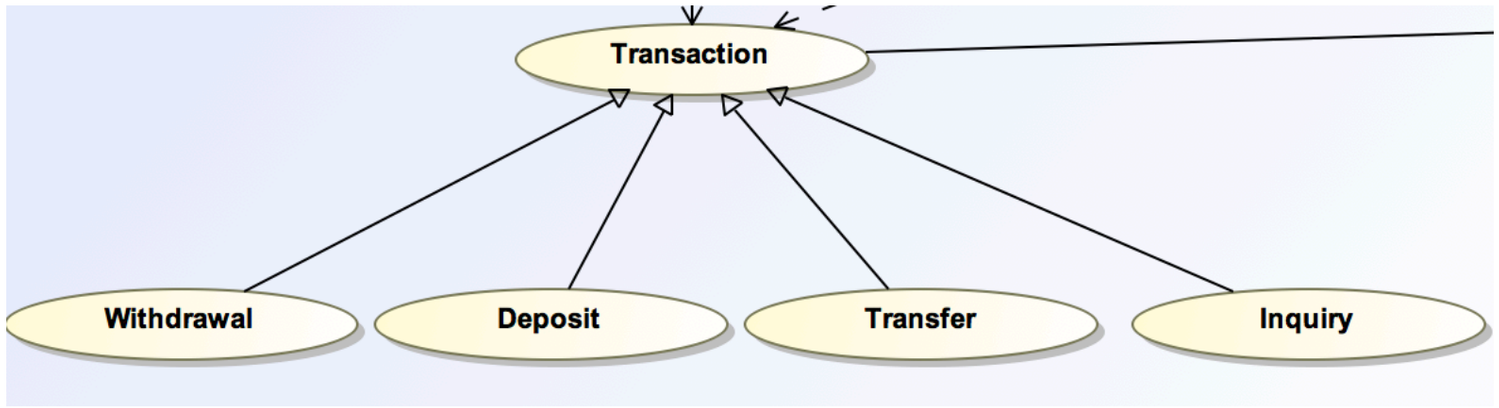
RELACIJA GENERALIZACIJE

- Generalizacija (**nasleđivanje**) između korisničkih funkcija ili između uloga je ista kao i veza generalizacije između klasa
- Generalizacija mora imati svoj naziv koji opisuje prirodu veze
- Generalizacija između uloga znači da potomak neke uloge nasleđuje ponašanje i smisao roditeljske uloge: naslednik (dete) može da dopuni ili nadjača ponašanje roditelja: dete može da zameni roditelje na svakom mestu na kome se roditelj pojavljuje
- Sve korisničke funkcije sa kojima je roditelj vezan vezom asocijacije, postaju dostupne i nasledniku (iako nije direktno vezan vezom asocijacije sa tom korisničkom funkcijom)
- Naslednik može redefinisati bilo koji element pretka: preduslove, posledice, opise, cele tokove, kao i ppojedinačne korake
- PRIMER:
  - PRRODAVAC_RAČUNARA_NASLEĐUJE PRODAVCA
  - Sve korisničke funkcije koje može da izvrši prodavac, može da izvrši i prodavac računara, iako nije direktno povezan sa tim korisničkim funkcijama



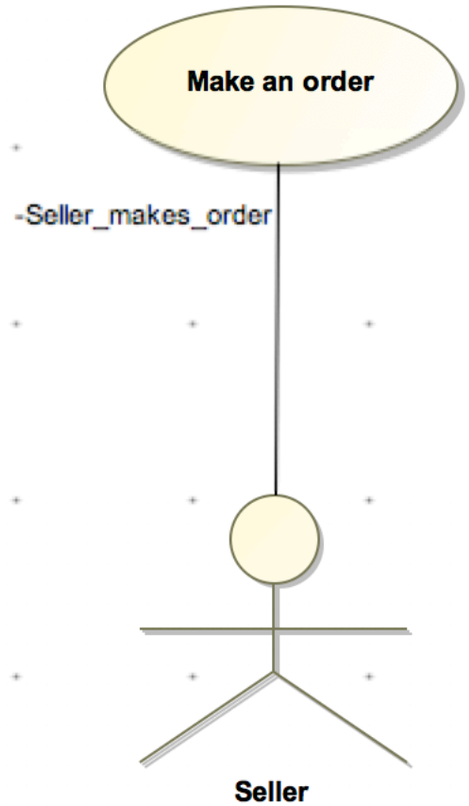
RELACIJA GENERALIZACIJE

- Generalizacija između korisničkih funkcija znači da potomak neke korisničke funkcije nasleđuje ponašanje i smisao roditeljske korisničke funkcije
- Naslednik (dete) može da dopuni ili nadjača ponašanje roditelja
- Dete može da zameni roditelja na svakom mestu na kom se roditelj pojavljuje
- Ukoliko je neka uloga asocijacijom vezana za korisničku funkciju koja ima naslednike, uloga može da izvrši i korisničke funkcije koje nasleđuju datu funkciju, iako uloga i naslednici korisničkih funkcija nisu direktno vezani



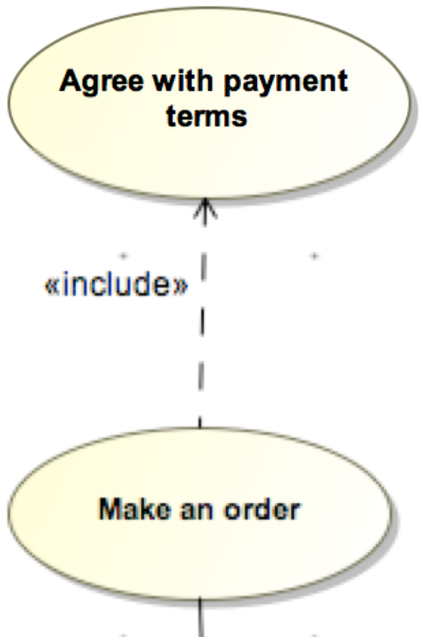
RELACIJA ASOCIJACIJE

- Relacija asocijacije označava interakciju uloga (izvođača) sa korisničkom funkcijom
- Relacija asocijacije predstavlja jedini način povezivanja uloge i korisničke funkcije
- Asocijacija treba da ima naziv koji označava funkciju koju izvodi uloga
  - KUPAC_PRAVI_PORUDŽBINU
  - PRODAVAC_PRIMA_PORUDŽBINU
  - OPERATOR_STARTUJE_SISTEM

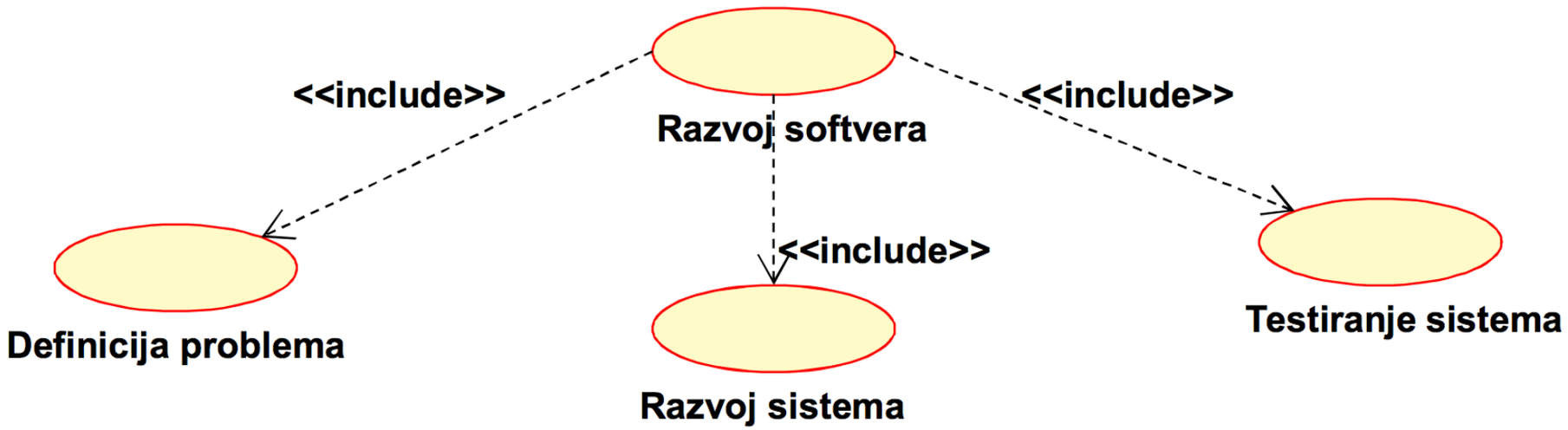


RELACIJE ZAVISNOSTI – RELACIJA UKLJUČIVANJA

- Relacija uključivanja (**include**) se može primeniti samo između korisničkih funkcija
- Relacija uključivanja između korisničkih funkcija znači da osnovna korisnička funkcija eksplicitno uključuje ponašanje druge korisničke funkcije. Slučaj korišćenja koji uključuje drugi mora uvek izvršiti i svoje korake i korake uključenog
- Uključena korisnička funkcija nikada ne može da stoji sama za sebe, već se javlja kao deo neke veće korisničke funkcije koja je uključuje
- Koristi se da bismo izbegli da se isti tok događaja opisuje više puta, stavljajući zajedničko ponašanje u posebnu korisničku funkciju
- Relacija uključivanja označava se kao zavisnost koristeći stereotip **include**
- **Ukoliko jedan slučaj korišćenja koristi drugi, mora uvek izvršiti i svoje korake i korake uključenog slučaja**
- **Koristimo ga kada želimo da više slučajeva korišćenja deli neko zajedničko ponašanje**

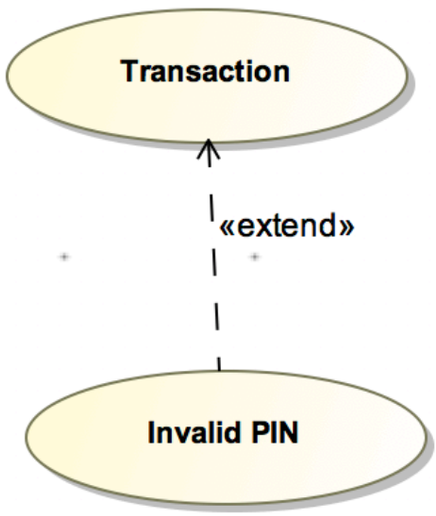


RELACIJE ZAVISNOSTI – RELACIJA UKLJUČIVANJA



RELACIJE ZAVISNOSTI – RELACIJA PROŠIRIVANJA

- Relacija proširivanja znači da osnovna korisnička funkcija indirektno ugrađuje ponašanje druge korisničke funkcije na mestu koje je indirektno definisano proširujućom korisničkom funkcijom
- Osnovna korisnička funkcija može stajati sama, ali pod određenim okolnostima, njeno ponašanje može biti prošireno ponašanjem druge korisničke funkcije
- Osnovna korisnička funkcija može biti proširena samo na određenim mestima koja se nazivaju **tačke proširenja**. Proširivanje se može shvatiti kao da proširujuća korisnička funkcija ubacuje ponašanje u osnovnu KF ukoliko su zadovoljeni određeni uslovi
- Relacija proširivanje koristi se za modelovanje dela korisničke funkcije koje korisnik može videti kao opciono ponašanje. Na taj način se razdvaja opciono od obaveznog ponašanja.
- Ukoliko su uslovi zadovoljeni izvršavaju se koraci osnovnog slučaja korišćenja, ali i koraci slučaja koji ga proširuje. Ukoliko slučaj nije zadovoljen, izvršavaju se samo koraci osnovnog slučaja
- Relacija proširivanja označava se kao zavisnot koristeći stereotip **extend**

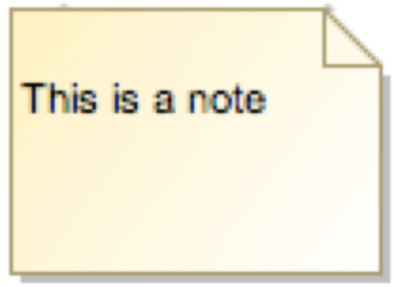


Ako slučaj InvalidPIN proširuje slučaj Transaction:

- i slučaj InvalidPIN i slučaj Transaction mogu da postoje sami
- slučaj Transaction može (a ne mora) da bude proširen slučajem InvalidPin

KOMENTAR

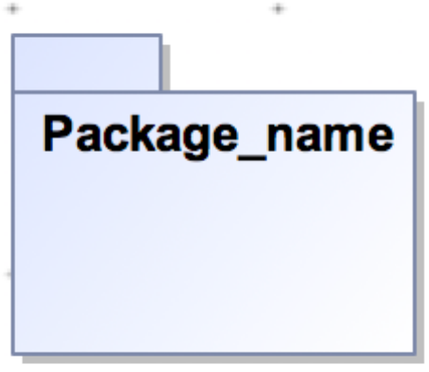
- Komentar (note) predstavlja osnovno anataciono sredstvo koje se može uvrstiti u UML dijagrame



- Komentari se obično koriste za dopunjavanje dijagrama sa ograničenjima ili napomenama koje se najbolje izražavaju neformalnim tekstom

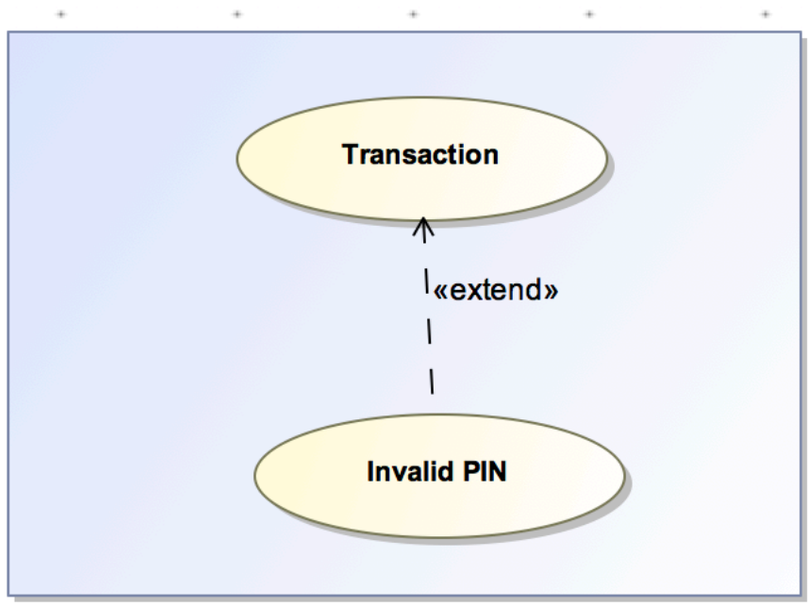
PAKET

- Elementi na dijagramima korisničkih funkcija mogu se orrganizovati gupisanjem u pakete (package), na isti način na koji se organizuju i klase
- Paket je mehanizam opšte namene za organizovanjee elemenata u grupe



GRANICE SISTEMA

- Pravougaonik oko korisničkih funkcija označava opseg sistema
- Korisničke priče unutar pravougaonika predstavljaju funkcionalnost koju sistem implementira



USE CASE MODELOVANJE

- Da bi se modelovalo ponašanje posmatranog sistema:

1. Identifikujte uloge (aktore) koji su u interakciji sa posmatranim sistemom. Kandidati za uloge su grupe koje zahtevaju određeno ponašanje radi izvođenja svojih zadataka ili koji su neophodni direktno ili indirektno radi izvršavanja funkcija tog sistema

2. Za svakog izvođača razmotiti ponašanje koje on očekuje ili zahteva da sistem obezbedi i ta ponašanja potrebno je prikazati kao korisničke funkcije

3. Organizujte uloge indetifikujući opšte i. Specijalne uloge (generalizacija uloga)

4. Za svaku ulogu ponaosob odrediti načine na koje on stupa u interakciju sa sistemom koje menjaju stanje sistema ili njegove okolone ili koje obuhvataju slanje odgovora na neki događaj

5. Organizujte korisničke funkcije primenjujući relacije proširivanja i uključivanja radi razlaganja na zajedničke delove ponašanja i razlikovanja izuzetnih ponašanja

6. Sve korisničke funkcije, uloge, generalizacije, asocijacije i veze zavisnosti treba da imaju imena koja govore o njihovoj nameni

7. Rasporrediti elemente tako da se broj linija koje se seku svede na minimum

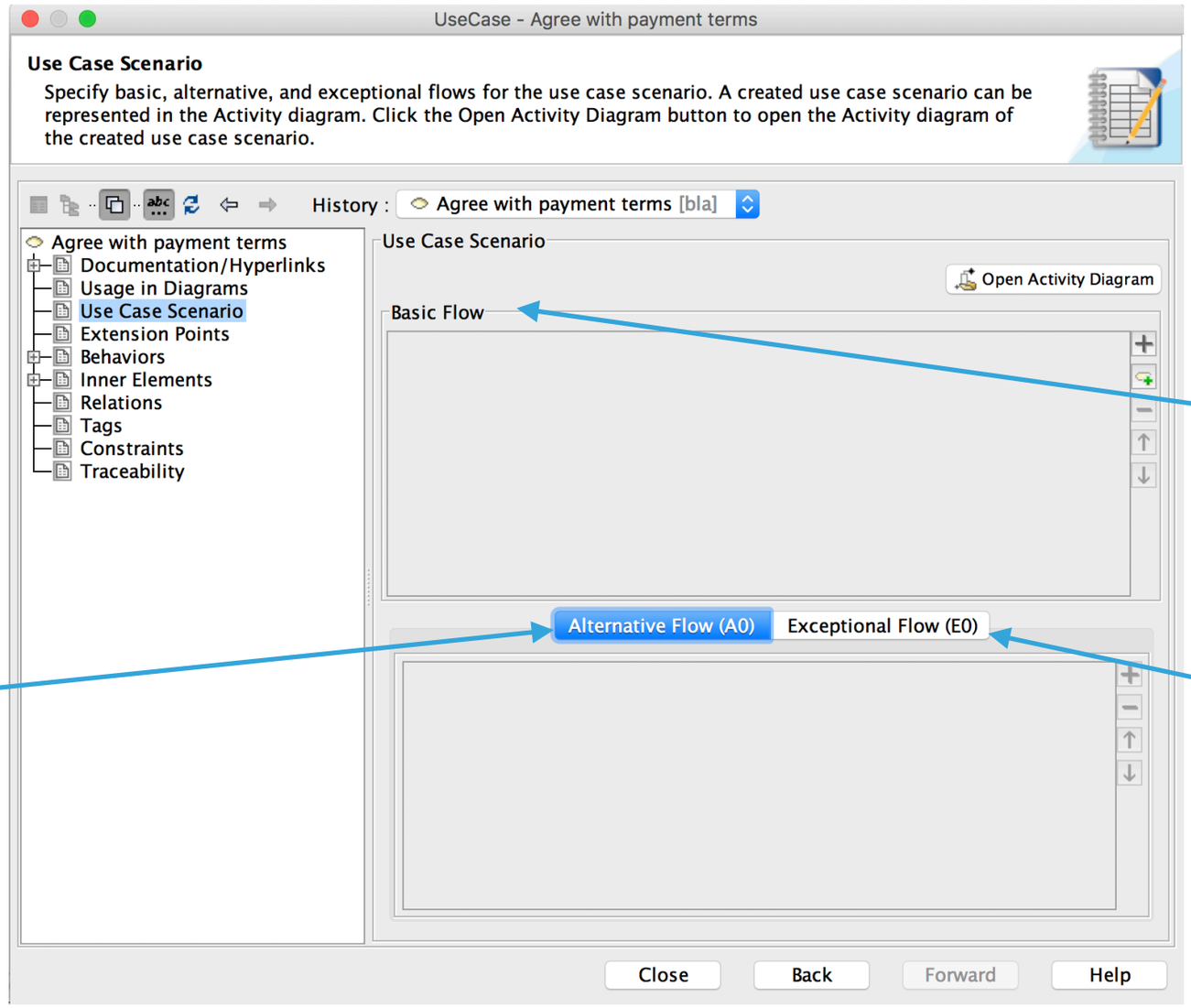
8. Ukoliko nije pomoće prostorno orrganizovati elemente tako da uloge i ponašanja koja su semantički slični budu i fizički blisko raspoređeni, uvesti pakete

SPECIFIKACIJA KORISNIČKE FUNKCIJE

- Korisnička funkcija opisuje šta sistem radi, ali ne definiše i kako to radi
- Moguće je definisati ponašanje korisničke funkcije tekstualno opisujući tok događaja dovoljno jasno da to lako razume i onaj ko je neupućen u sistem
- Kada se piše tok događaja, potrebno je naznačiti:
  - kada i kako korisnička funkcija započinje i završava,
  - kada stupa u interakciju sa izvođačima,
  - koji se objekti razmenjuju,
  - potrebno je opisati osnovni i alternativne tokove ponašanja

SPECIFIKACIJA KORISNIČKE FUNKCIJE

- Specifikaciju korisničke funkcije moguće je odraditi u kartici “Use case scenario” koji se otvara nakon što izvršite desni klik na korisničku funkciju, a zatim “Specification”



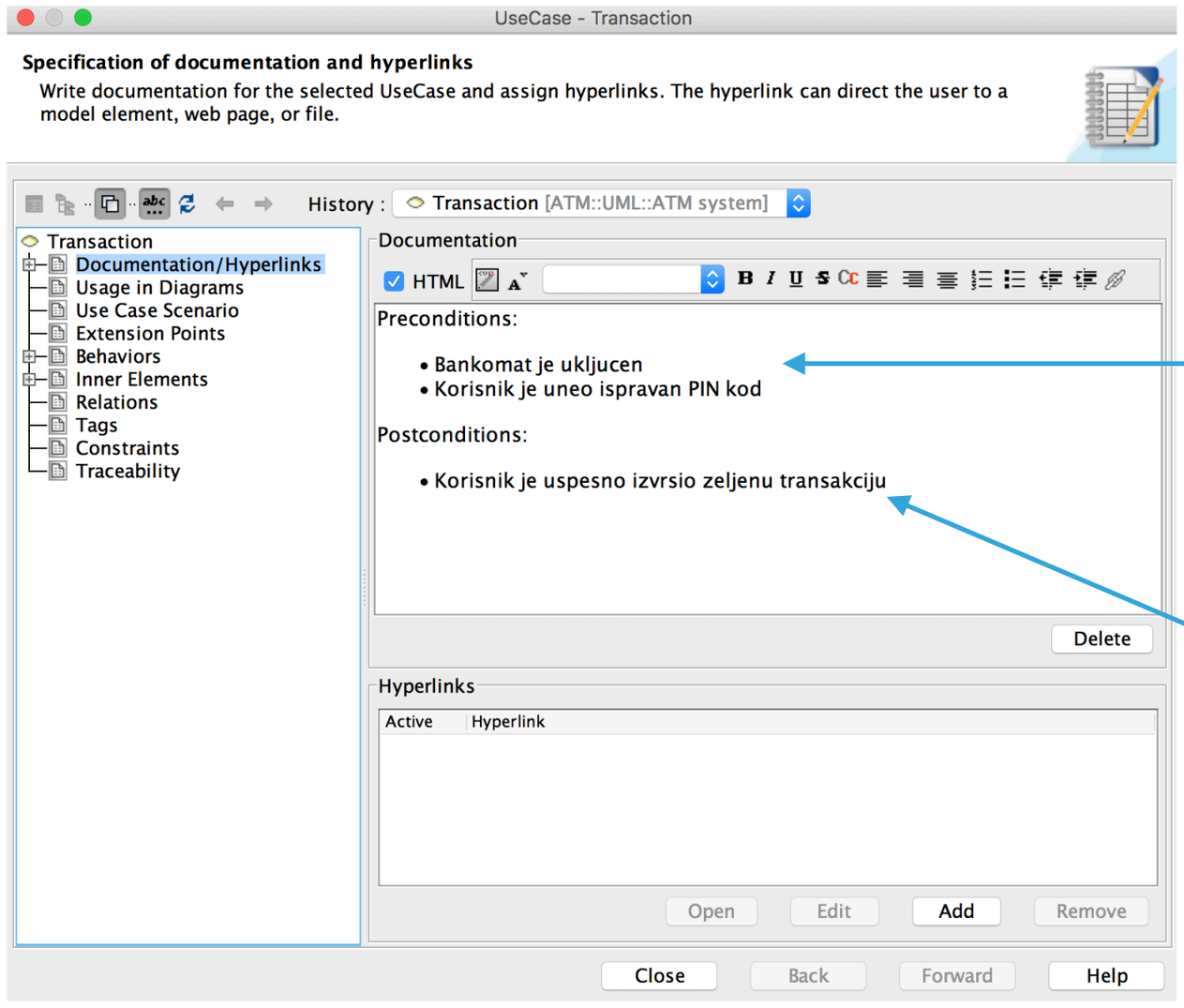
BASIC FLOW - Opisuje se osnovni (uspešni) tok događaja. Pod osnovnim tokom navodi se kada korisnička funkcija počinje svoje izvršavanje, ko je inicira. Ukoliko korisnička funkcija uključuje i neke drige funkcije (postoji include), navodi se i kada se uključena funkcija izvšava

ALTERNATIVE FLOW - Opisuje alternativne tokove događaja. Ukoliko funkcija ima proširenja ona se ovde

EXCEPTIONAL FLOW - Opisuje se neuspešni tok događaja

SPECIFIKACIJA KORISNIČKE FUNKCIJE

- Definisanje pre i post conditions-a moguće je odraditi u kartici “Documentations” koji se otvara nakon što izvršite desni klik na korisničku funkciju, a zatim “Specification”



PRECONDITIONS - navođenje potrebnih preduslova da bi se korisnička funkcija uopšte izvršila. Navode se samo uslovi koji su dovoljno značajni. Očigledni uslovi se ne navode.

POSTCONDITIONS - rezultat uspešnog i/ili neuspešnog izvršavanja. Unose se samo netrivijalne posledice

ZADATAK 1

- Modelirati sistem za praćenje rada klinike
  - Sistem omogućuje pacijentu da zakazuje pregled, otkazuje pregled, zahteva lečenje i da plaća račun
  - Zakazivanje i otkazivanje pregleda vrši medicinska sestra
  - Prilikom zakazivanja pregleda i podnošenja zahteva za lečenje, medicinska sestra provera dokumentaciju pacijenta
  - Zahtev za lečenje pacijent podnosi doktoru
  - Plaćanje može uključivati plaćanje karticom
  - Plaćanje može biti preko osiguranja

ZADATAK 2

- Realizovati aplikaciju za podršku rada kadrovske službe nekog preduzeća, koja funkcioniše na sledeći način.
  - Referent kadrovske službe obavlja sledeće poslove:
    - Unos i izmenu podataka o novim zaposlenima (ime, prezime, adresa, pol, datum rođenja)
    - Raspored zasposlenog na radno mesto (naziv radnog mesta i broj bodova). Prilikom zasnivanja radnog odnosa, referent prvo unosi podatke o novom radniku i zatim ga raspoređuje na neko od postojećih radnih mesta.
    - Premeštanje zaposlenog na novo radno mesto.
    - Prekid radnog odnosa iz bilo kog razloga (odlazak u penziju, sporazumni raskid, otkaz).
    - Prikaz izveštaja o aktivnim radnicima.
    - Prikaz izveštaja o penzionerima.
    - Prikaz izveštaja o svim radnim mestima radnika – tekućem i nekadašnjim, sa datumom raspoređivanja na dato radno mesto.
- Administrator obavlja kreiranje korisničkih naloga za zaposlene. Promenu svoje lozinke mogu da obavljaju svi koji imaju korisnički nalog.
- Na početku svakog meseca aplikacija treba da proveri sve aktivne zaposlene i šefu kadrovske evidencije pošalje spisak onih koji u narednih mesec dana pune 65 godina (spisak kandidata za penziju).
- Šef kadrovske službe treba da ima mogućnost da radi sve poslove kao i referent, a dodatno i da održava podatke o raspoloživim radnim mestima.

ZADATAK 3

- Modelovati speficikaciju Restorana sa prošlog časa