Objektno orijentirana analiza i dizajn

Primjer stupnjevitog projektiranja aplikacije

Postavljanje zahtjeva

- **Korisnički zahtjevi**
  - Krajnji korisnik ili njegov predstavnik pišu specifikaciju sustava.
- **Primjer aplikacije za vođenje knjižnice**
  - Trebamo sustav za vođenje knjižnice.
  - Knjižnica posuđuje knjige i časopise članovima koji su evidentirani u sustavu, kao i knjige i časopisi.
  - Knjižnica naručuje nove naslove. Popularni naslovi kupuju se u više primjeraka. Stare knjige i časopisi uklanjaju se kada su zastarjeli ili su u lošem stanju.
  - Knjižničar je zaposlenik knjižnice koji je na usluzi posuđivačima, a sustav mora podržavati njegov rad.
  - Posuđivač može rezervirati knjigu ili časopis koji trenutno nije raspoloživ tako da ga sustav obavijesti kad primjerak bude vraćen. Rezervacija se otkazuje kada se knjiga ili časopis podignu ili kroz proceduru za otkazivanje.
  - Knjižničar može jednostavno u sustavu dodavati, mijenjati ili brisati informacije o naslovima, posuđivačima, posudbama i rezervacijama.
  - Aplikacija treba biti raspoloživa na svim popularnim operacijskim sustavima, kao što su Linux, Unix, Windows, itd., te imati moderno grafičko sučelje (GUI).
  - Sustav se jednostavno može nadograđivati novim funkcionalnostima.
  - Prva inačica sustava ne mora slati obavijesti kada rezervirani naslovi postanu dostupni, niti mora provjeravati kada je naslov predugo posuđen.

Analiza zahtjeva

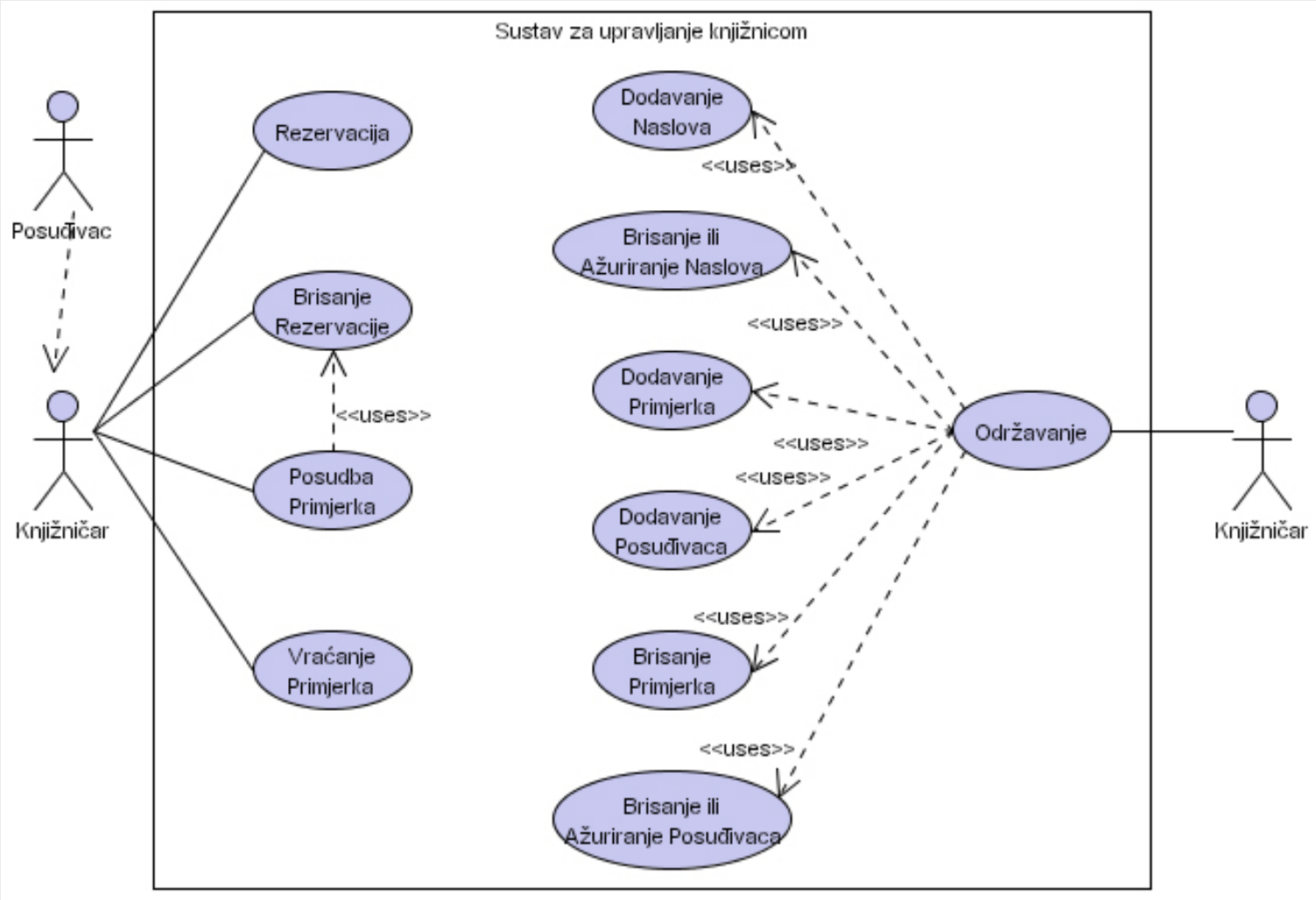
- **Analiza zahtjeva**
  - Analiza je namijenjena uočavanju i opisivanju svih zahtjeva vezanih uz sustav i stvaranju modela koji definira ključne domenske razrede u sustavu (što je to čime sustav upravlja).
  - U analizi najprije treba otkriti za što će se koristiti sustav i tko će ga koristiti. To se opisuje slučajevima korištenja i sudionicima (Actors).
  - Slučajevi korištenja opisuju funkcionalne zahtjeve na sustav.
  - Analiza slučajeva korištenja uključuje čitanje i analizu specifikacija, kao i raspravu o sustavu s potencijalnim korisnicima ili naručiteljem sustava.
  - Odluke o načinu ugradnje biti će načinjene tijekom oblikovanja sustava.
- **Slučaj korištenja (*Use Case*)**
  - Slučaj korištenja je skup povezanih scenarija za ostvarivanje neke korisničke akcije.
  - Osim opisa primarnog scenarija u obliku numeriranih koraka, može sadržavati i opise alternativnih scenarija.
  - Predstavlja vanjski pogled na sustav pa ne korelira s razredima u sustavu.
  - U manjem (jednogodišnjem projektu) s do 10ak članova može biti 10ak osnovnih dijagrama slučajeva korištenja. Svaki osnovni slučaj može imati više scenarija i varijanti pa ukupno može biti i preko 100 dijagrama.

Analiza zahtjeva

- **Sudionici u knjižnici identificirani su kao knjižničar i posuđivač.**
  - Knjižničari su izravni korisnici sustava.
  - Posuđivači posredno posuđuju, rezerviraju i vraćaju knjige i časopise,
  - Knjižničari ili druga knjižnica također mogu biti posuđivači.
  - Sudionici su ljudi ili drugi sustavi izvan sustava kojeg razvijamo.
    - Npr. pisač ili baza podataka mogu biti sudionici.
- **Slučajevi korištenja u knjižnici**
  - posudba, vraćanje, rezervacija, otkazivanje rezervacije, dodavanje naslova
  - izmjena ili brisanje naslova, dodavanje knjige ili časopisa (primjerka), brisanje primjerka, dodavanje posuđivača, izmjena ili brisanje posuđivača.
- **Budući da knjižnica često ima više kopija popularnih naslova, sustav mora odvojiti koncept naslova od koncepta primjerka.**
- **Analiza sustava za upravljanje knjižnicom je dokumentirana u dijagramu slučaja korištenja**
- **Svaki slučaj korištenja je dokumentiran i opisno.**

Dijagram slučajeva korištenja

- Prvi korak - utvrditi tko i kako koristi sustav
- Svi UC moraju počinjati sa sudionicima a neki i završavaju s njima.

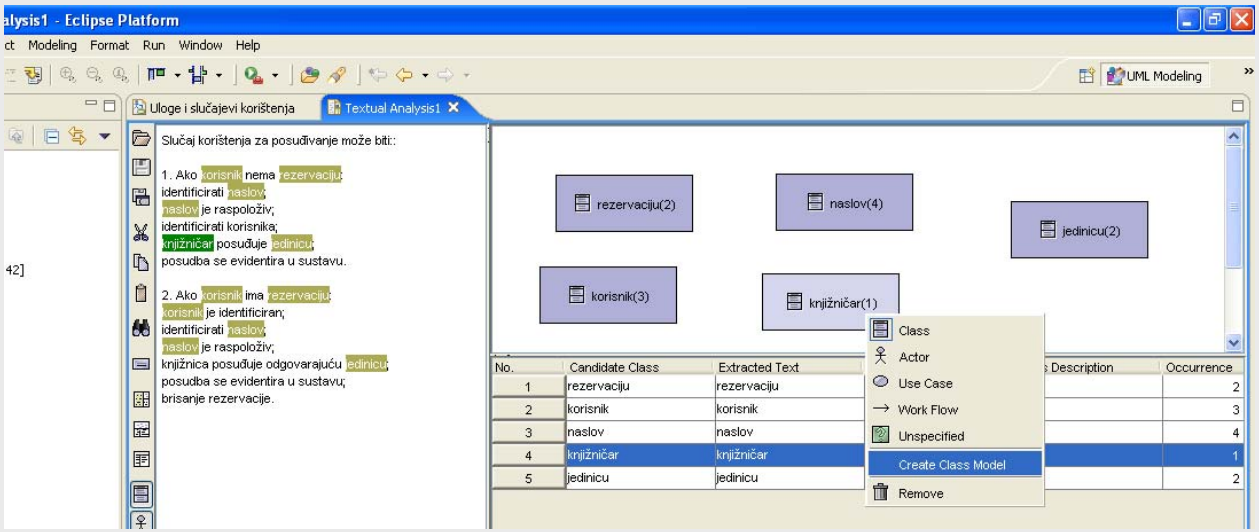


Opisivanje scenarija

- **Slučaj korištenja za posuđivanje:**
  - 1. Ako korisnik nema rezervaciju:
    - identificirati naslov;
    - naslov je raspoloživ;
    - identificirati posuđivača;
    - knjižničar posuđuje jedinicu;
    - posudba se evidentira u sustavu.
  - 2. Ako korisnik ima rezervaciju:
    - identificirati posuđivača;
    - identificirati naslov;
    - naslov je raspoloživ;
    - knjižnica posuđuje odgovarajuću jedinicu;
    - posudba se evidentira u sustavu;
    - brisanje rezervacije.

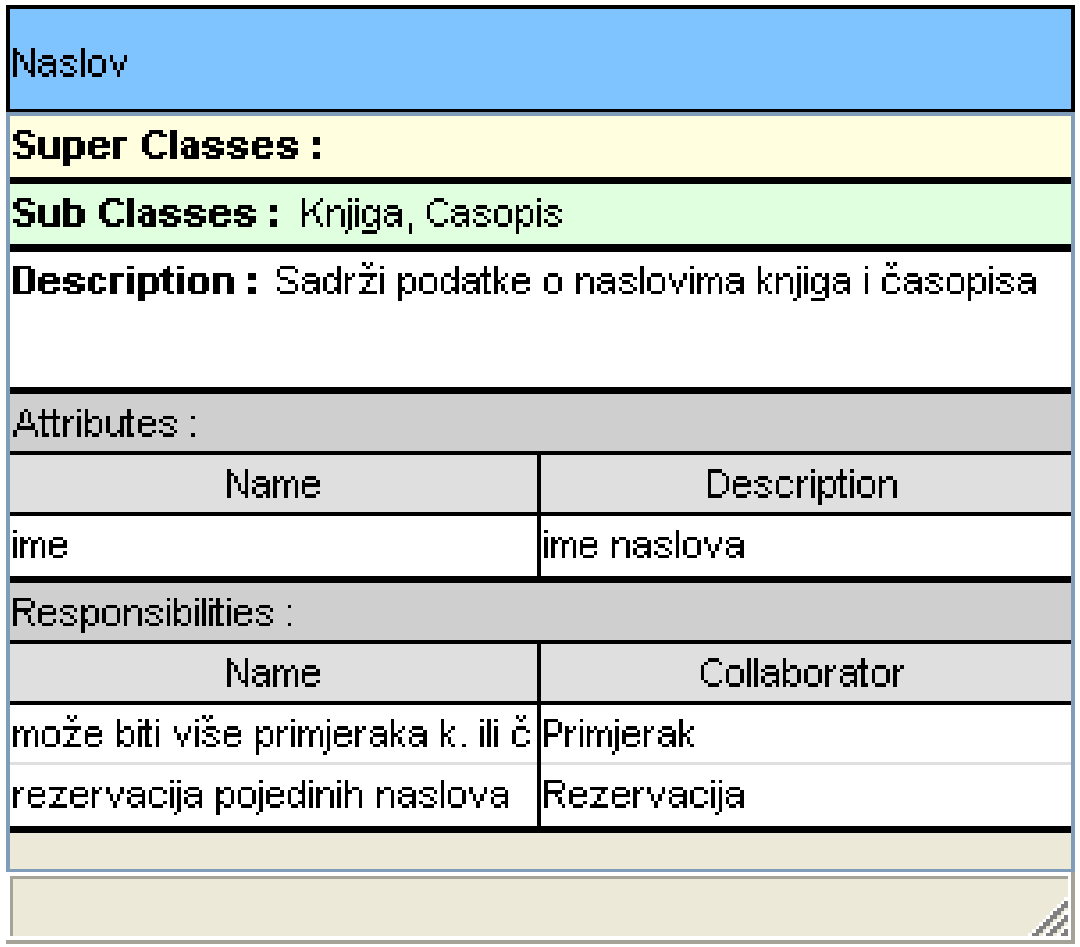
Analiza teksta CASE pomagalom

- **Dijagram slučaja korištenja može se dopuniti tekstovnom analizom (*Textual Analysis*).**
- **Primjer: upotreba tekstovne analize na slučaju posudbe primjerka.**
  - Dodavanje započinje desnim klikom na element *Posudba Primjerka*, zatim redom *Sub Diagrams – Textual Analysis – Create Textual Analysis*.
  - Slučaj se opisuje u nekoliko rečenica iz kojih se odabiru kandidati za razrede u sustavu, dovlačenjem kandidata za razrede na desni panel.
  - Desnim klikom na kandidata otvara se izbornik u kojem se može odabrati opcija *Create Class Model* koja će kandidata smjestiti u UML model iz kojega će se razred dalje moći koristiti u dijagramima razreda.



CRC kartice

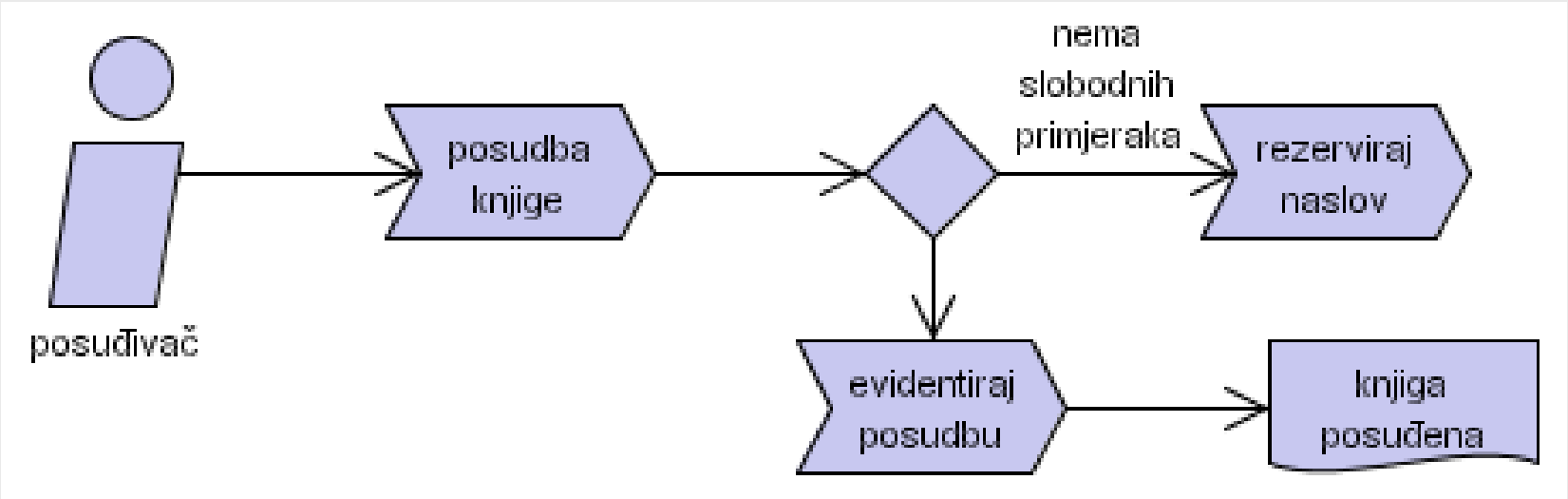
- CRC (*Class-Responsibility-Collaboration*) kartice mogu se koristiti umjesto dijagrama
- Umjesto atributa i metoda, na kartici se mogu prikazati odgovornosti razreda.
- Suradnja se odnosi na druge razrede s kojima razred opisan CRC karticom surađuje čime se modelira interakcija među razredima.
- Preporučuje se da kartica sadrži najviše tri odgovornosti visoke razine. U protivnom treba razmisliti o razdvajanju razreda u nekoliko manjih razreda koji se mogu efikasno opisati.



Naslov	
Super Classes :	
Sub Classes : Knjiga, Casopis	
Description : Sadrži podatke o naslovima knjiga i časopisa	
Attributes :	
Name	Description
ime	ime naslova
Responsibilities :	
Name	Collaborator
može biti više primjeraka k. ili č	Primjerak
rezervacija pojedinih naslova	Rezervacija

Dijagrami radnih tokova

- **Dijagrami poslovnih procesa - Poslovni dijagrami toka (*Business Workflow*) prikazuju poslovne procese na visokoj razini.**



Generiranje dijagrama slijeda

- *npr. Posudba – UC Details ... - Description.*
- *Flow Of Events*
- *Generate Sequence Diagram*
- dijagram nadopuniti objektima i događajima

PosudbaPrimjerka Details

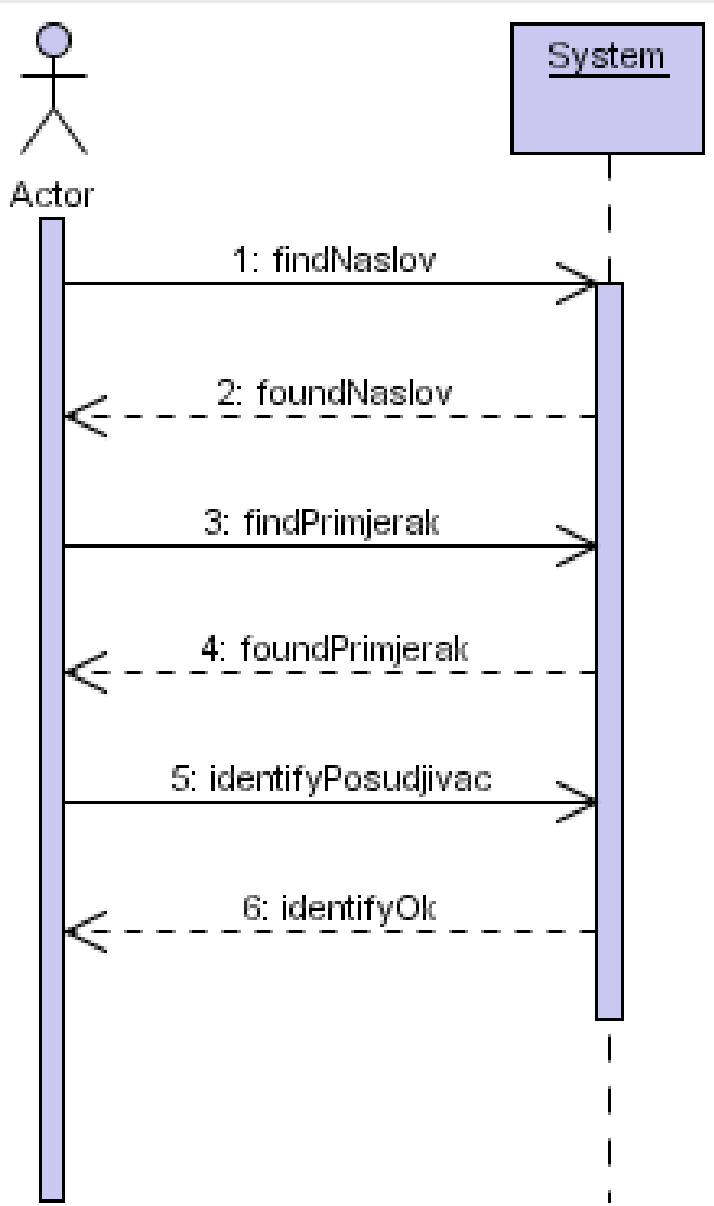
Name: Posudba¶Primjerka

Info | Description | Diagrams

Name	
Super Use Case	
Author	
Date	
Brief Description	
Pre-conditions	
Post-conditions	

Flow of Events

	Actor Input	System Respons
1	findNaslov	
2		foundNaslov
3	findPrimjerak	
4		foundPrimjerak
5	identifyPosudjivac	
6		identifyOk

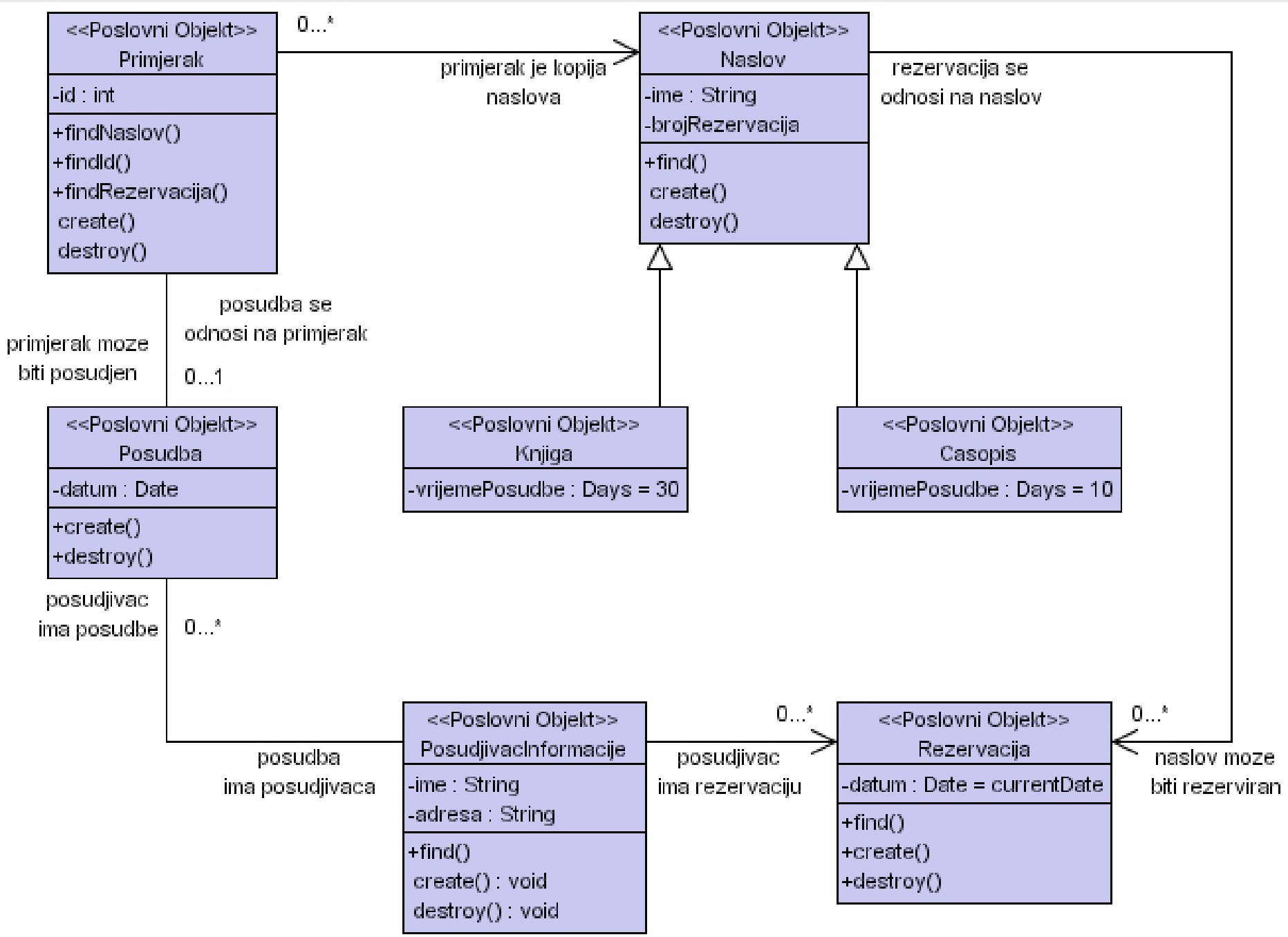


Domenska analiza

- **Analiza definira problemsko područje to jest ključne razrede**

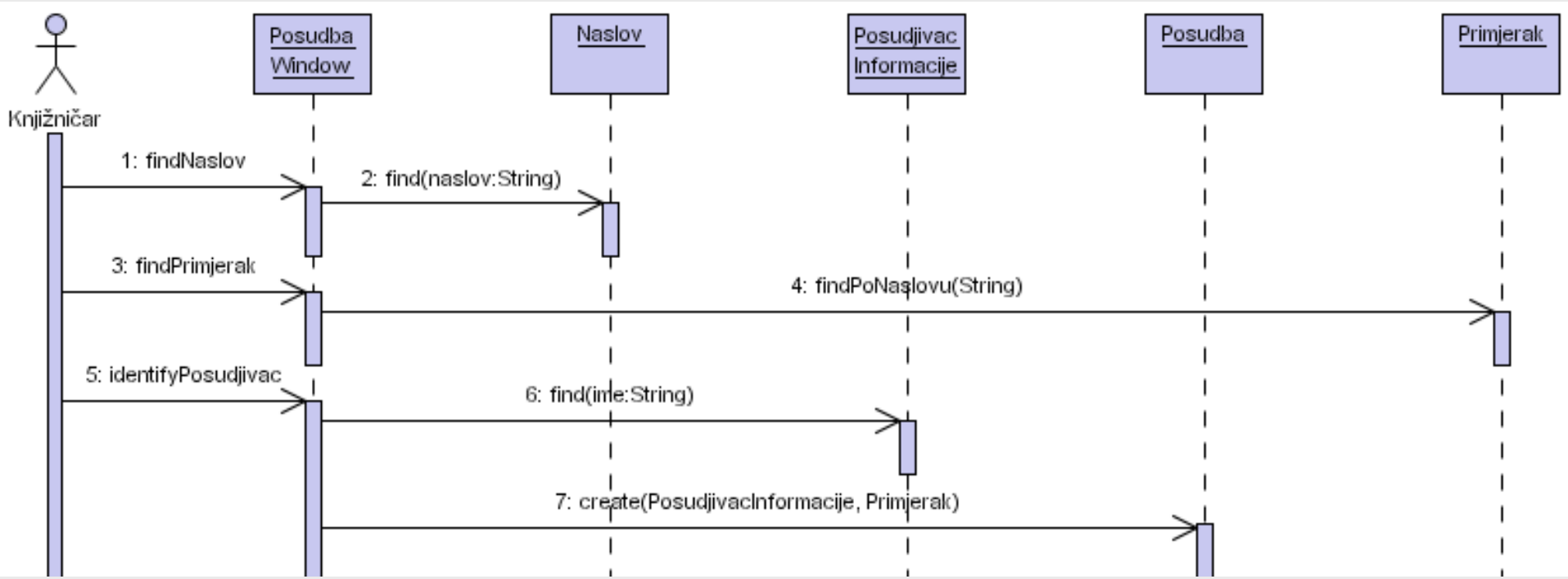
- **Primjer**
  - Domenski razredi (*Domain Class*) u sustavu za upravljanje knjižnicom mogu biti npr.: *PosudjivacInformacije* (treba razlikovati od *Posudjivac* kao sudionika u slučajevima korištenja), *Naslov*, *Knjiga*, *Casopis*, *Primjerak*, *Rezervacija* i *Posudba*.
  - Domenski razredi su definirani upotrebom korisnički definiranog stereotipa "Poslovni Objekt", koji određuje da su objekti razreda dijelovi ključne domene te da stoga moraju biti trajno pohranjeni u sustavu.

- **Temeljem analize može se proširiti osnovni dijagram slijeda**
  - npr. *Posudba Primjerka* - otvoriti izbornik te odabrati *Sub Diagrams – Sequence Diagram – Create Sequence Diagram*
  - istom slučaju korištenja može biti pridruženo više dijagrama


<<Poslovni Objekt>>
Primjerak
-id : int
+findNaslov()
+findId()
+findRezervacija()
create()
destroy()
0...*
primjerak je kopija naslova
<<Poslovni Objekt>>
Naslov
-ime : String
-brojRezervacija
+find()
create()
destroy()
rezervacija se odnosi na naslov
posudba se odnosi na primjerak
primjerak moze biti posudjen
0...1
<<Poslovni Objekt>>
Posudba
-datum : Date
+create()
+destroy()
<<Poslovni Objekt>>
Knjiga
-vrijemePosudbe : Days = 30
<<Poslovni Objekt>>
Casopis
-vrijemePosudbe : Days = 10
posudjivac ima posudbe
0...*
<<Poslovni Objekt>>
PosudjivacInformacije
-ime : String
-adresa : String
+find()
create() : void
destroy() : void
posudba ima posudjivaca
0...*
posudjivac ima rezervaciju
<<Poslovni Objekt>>
Rezervacija
-datum : Date = currentDate
+find()
+create()
+destroy()
0...*
naslov moze biti rezerviran

Proširenje dijagrama slijeda

- **Primjer,**
  - Dijagram slijeda za slučaj korištenja *Posudba* kada korisnik nema rezervaciju.
- **Interakciju uvijek potiče sudionik putem korisničkog sučelja**
  - Detalji korisničkog sučelja mogu biti razrađeni naknadno.



Proširenje dijagrama slijeda

- **Pri modeliranju dijagrama slijeda postaje očito da je potrebno sučelje prema sudionicima.**
  - U ovoj analizi dovoljno je biti svjestan da su dijelovi sučelja (prozori ili dijalozi) potrebni za rezervaciju, posuđivanje i vraćanje jedinica.
  - Detalji korisničkog sučelja mogu biti razrađeni naknadno.

- **Za razdvajanje GUI razreda u analizi od domenskih razreda koristi se razdvajanje i grupiranje GUI razreda u paket *GUI Package*, a domenskih razreda u paket *Business Package*.**

Dizajn sustava

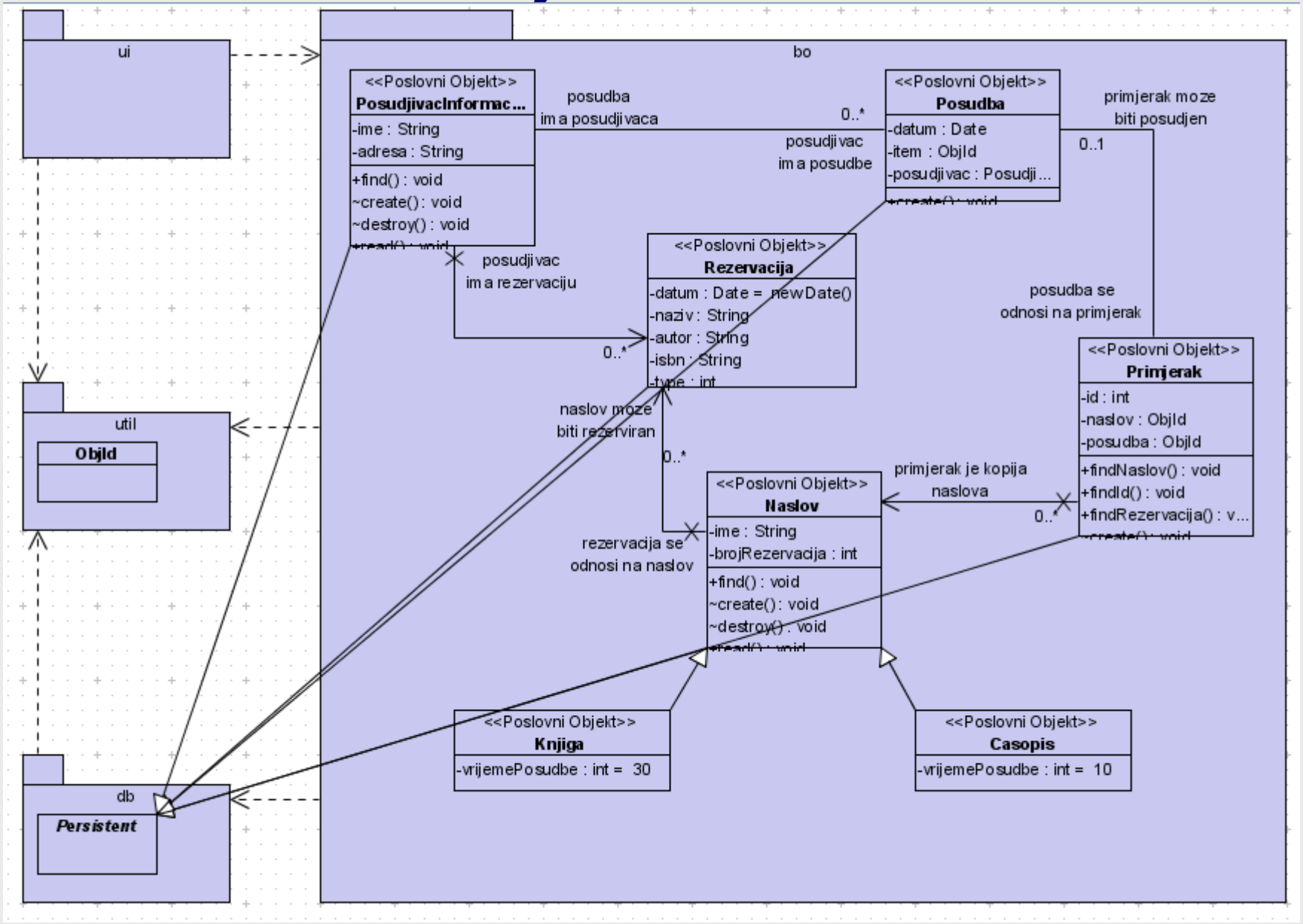
- **Faza dizajna proširuje i detaljizira analitički model uzimajući u obzir sve tehničke implikacije i restrikcije.**
  - Svrha dizajna je specificiranje operativnog rješenja koje se može ugraditi u programski kôd.
  - Tijekom projektiranja može doći do razdvajanja i grupiranja pojedinih razreda radi uslojavanja (npr. razredi sučelja, razredi poslovne logike, razredi pristupa podacima).

- **Razina dizajna**
  - Dizajn arhitekture je dizajn na visokoj razini gdje se definiraju paketi (podsustavi) uključujući ovisnosti i osnovne komunikacijske mehanizme između paketa.
  - Detaljni dizajn opisuje sve razrede koji predstavljaju jasnu specifikaciju za ugradnju. Dinamički UML modeli koriste se za demonstraciju kako se objekti razreda ponašaju u specifičnim situacijama.

Dizajn arhitekture

- **Paket korisničko sučelje (*User-Interface Package, UI*)**
  - Paket surađuje s poslovnim paketom koji sadrži razrede u kojima se zapravo pohranjuju podaci. UI paket poziva operacije poslovnih objekata kako bi dohvatio ili upisao podatke u njih.
- **Poslovni paket (*Business-Objects Package, BO*)**
  - Uključuje domenske razrede iz analitičkog modela kao što su *PosudjivacInformacije*, *Naslov*, *Primjerak*, *Posudba*, itd.
  - Dizajn potpuno definira njihove operacije i dodaje podršku za trajnu pohranu (*Persistance*).
  - U našem primjeru, poslovni objekti surađuju s paketom za pohranu tako što svi poslovni razredi nasljeđuju razred *Persistent* iz tog paketa.
- **Paket za rad s bazom podataka (*Database Package, DB*)**
  - Služi kao servis ostalim razredima iz poslovnog paketa.
- **Uslužni paket (*Utility Package, UTIL*).**
  - Sadrži usluge koje se koriste u ostalim paketima u sustavu.
  - Npr. *ObjId* razred koristi se za dohvat trajno pohranjenih objekata u sustavu uključujući UI i DB pakete te poslovni paket.

Dizajn arhitekture

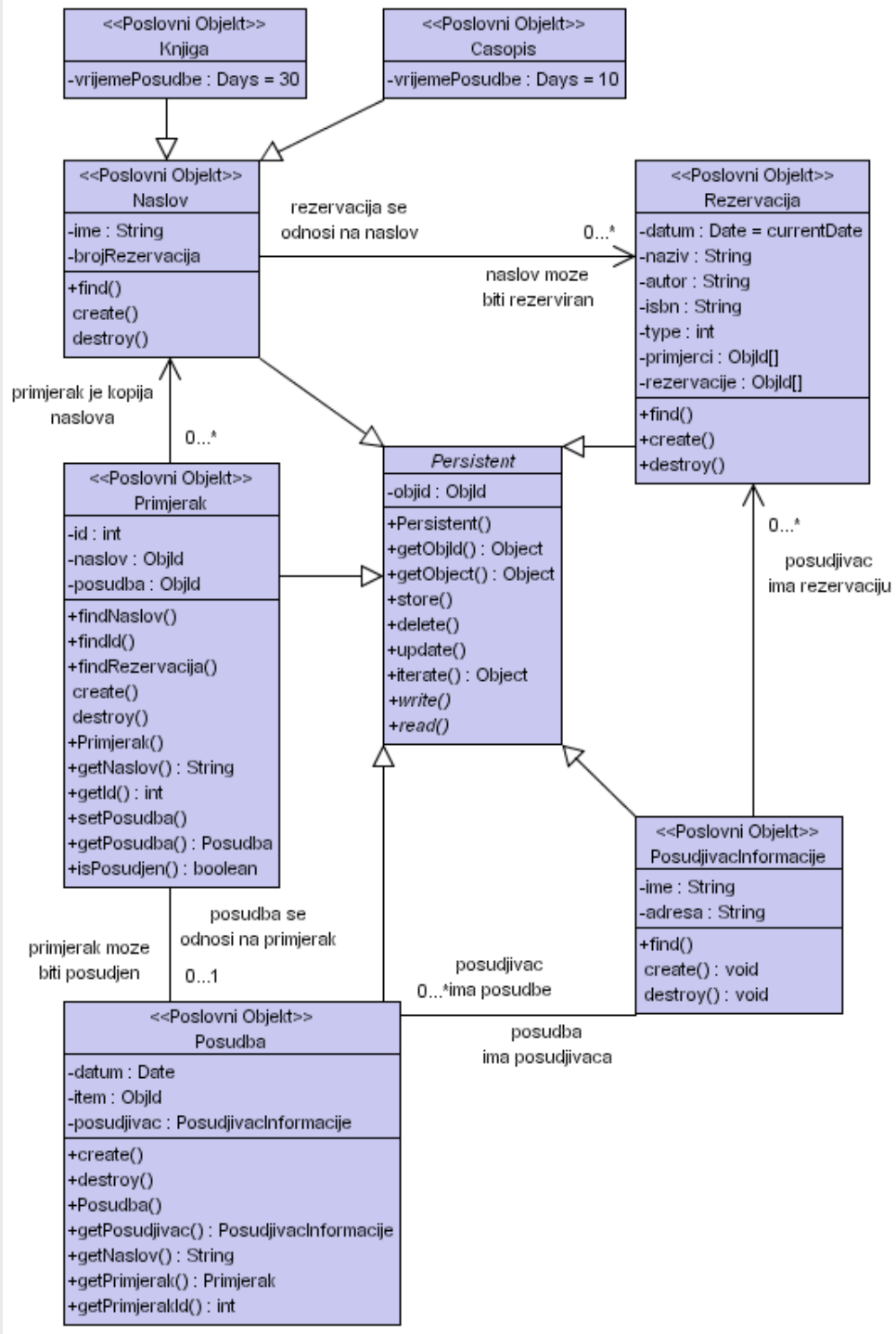


Detaljni dizajn

- Paket za rad s bazom podataka.
  - Aplikacija mora imati mogućnost trajne pohrane objekata.
  - Detalji o smještaju su skriveni od aplikacije koja samo poziva zajedničke operacije kao što su *store()*, *update()*, *delete()* ili *find()*.
  - Ovo su članovi osnovnog razreda *Persistent*, kojeg nasljeđuju drugi razredi koji zahtijevaju pohranu.
  - Kao identifikator podataka koristi se *ObjId* (*Object Identity*), razred čiji se objekti koriste za referenciranje bilo kojeg perzistentnog objekta u sustavu (bez obzira je li objekt na disku ili je učitan u aplikaciju)
  - *ObjId* je općeniti razred koji koriste svi paketi u sustavu pa je smješten u uslužni paket, a ne u DB paket.
- Poslovni paket
  - Zasnovan je na odgovarajućem paketu iz analize, tj. domenskim razredima. Razredi, njihovi odnosi i ponašanja su očuvani, ali su razredi detaljnije opisani, što uključuje i opis implementacije odnosa i ponašanja.

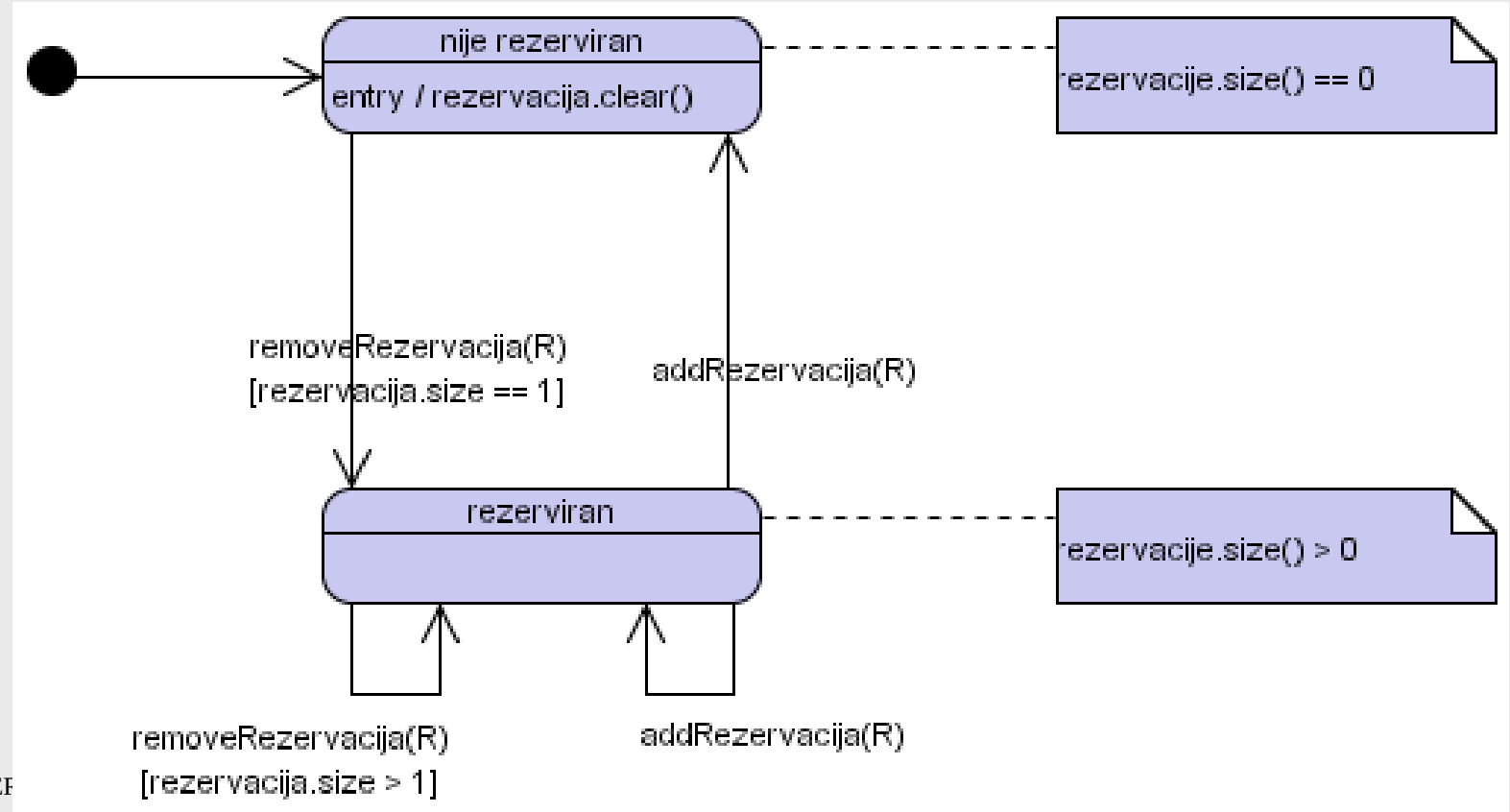
Detaljni dizajn

- Primjer: Dizajn poslovnih objekata
- Dizajn počinje određivanjem detalja modela. Sučelja su potpunije specificirana, odabrani su tipovi atributa, itd.
- Na slici nisu prikazani paketi kojima razredi pripadaju: svi razredi poslovnih objekata pripadaju *bo* paketu, osim razreda *Persistent* koji pripada *db* paketu.
- Neke operacije su razrađene u nekoliko operacija, a neke su preimenovane u odnosu na prethodne modele



Razrada stanja

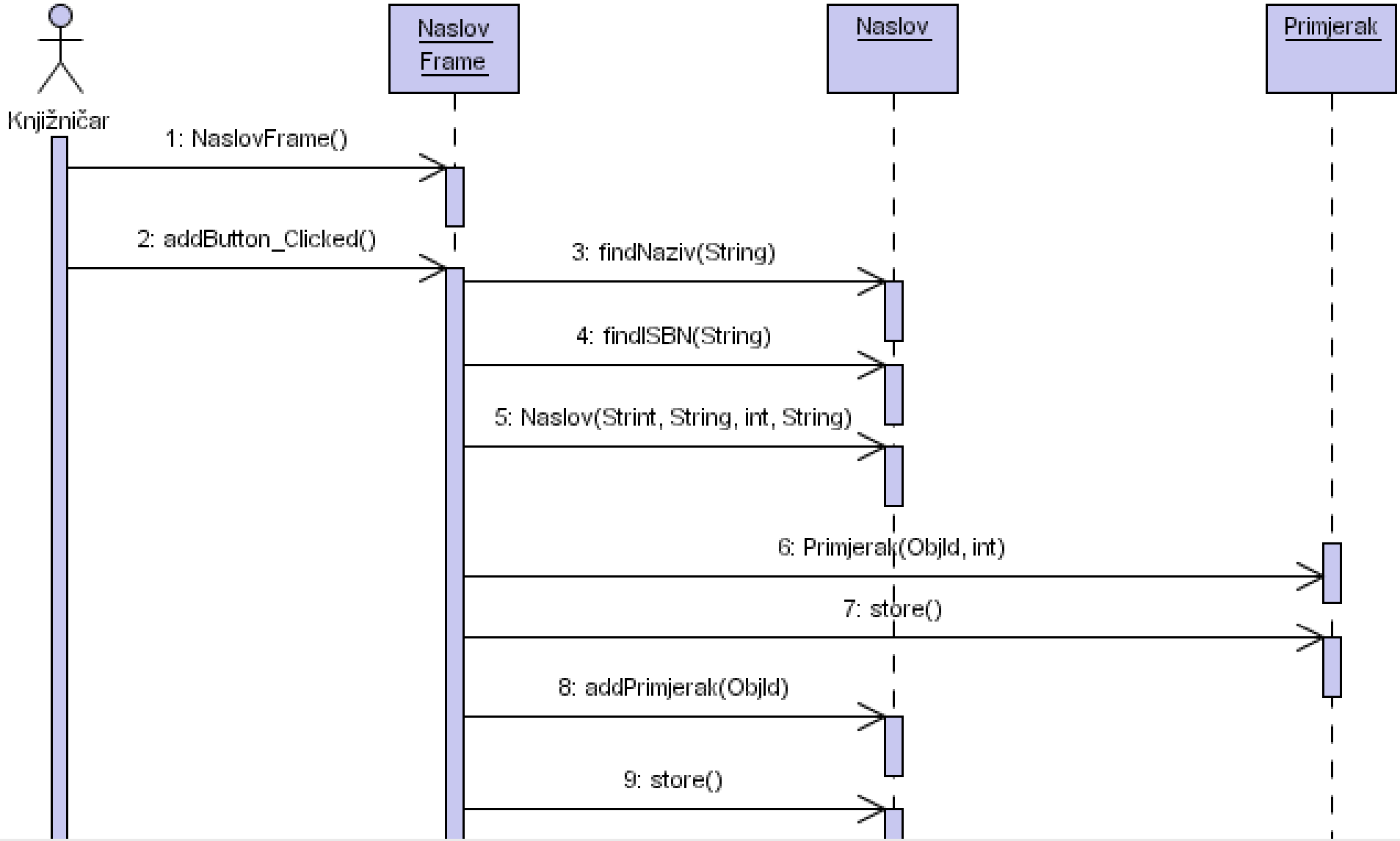
- Dijagrami stanja iz analize su također detaljnije razrađeni u dizajnu, prikazujući kako su stanja prezentirana i vođena u pravom sustavu.
- Primjer: Stanja za naslov (rezervirano i slobodno) implementirana su u vektorom *rezervacije*. Promjene se obavljaju pozivom metoda *addRezervacija()* i *removeRezervacija()*.



Paket korisničkog sučelja

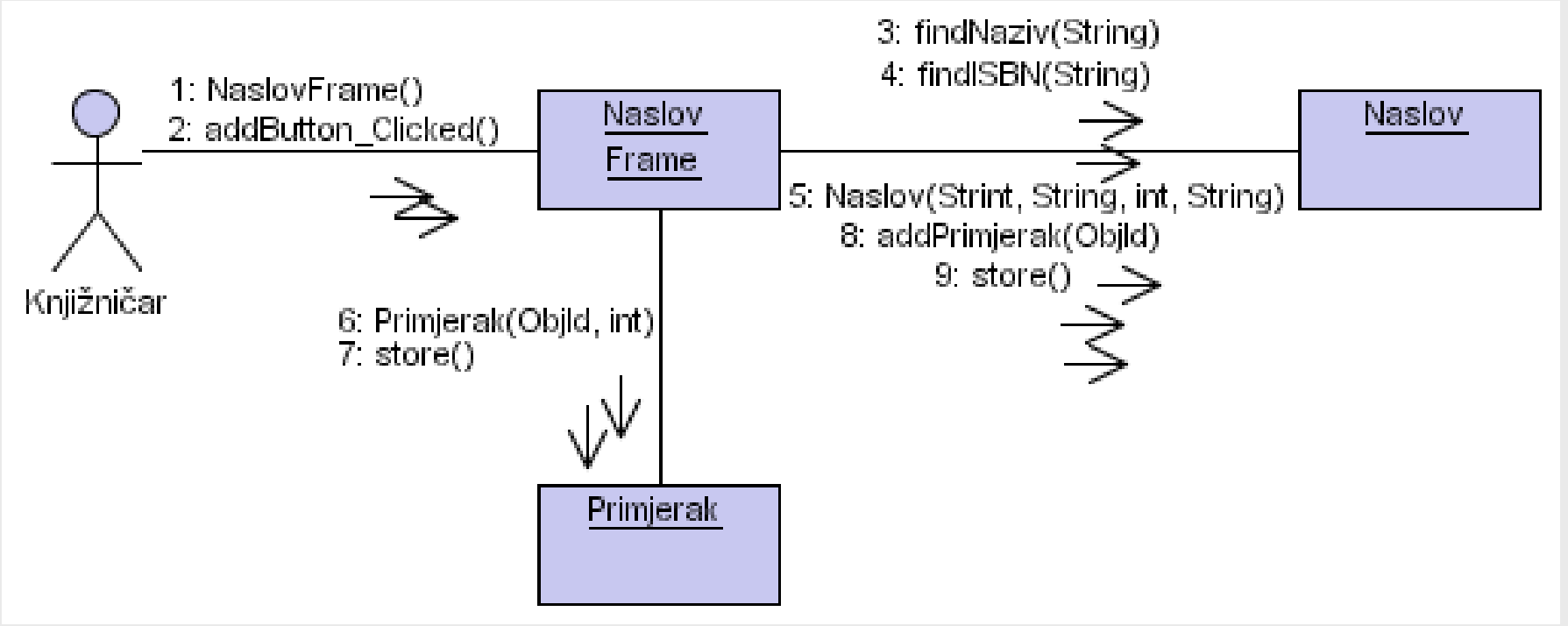
- **UI paket**
  - Prezentira usluge i informacije korisniku sustava.
  - Sadrži dinamičke modele, jer se sva interakcija s korisnikom ostvaruje putem korisničkog sučelja.
- **Realizacija slučajeva korištenja u modelu dizajna prikazana je detaljno, uključujući stvarne operacije razreda.**
- **Dijagrami slijeda razrađuju se iterativno, čak i tijekom kodiranja.**
  - Metode na slici detaljnog dizajna sučelja mogu biti stvarno kodirane.

Dizajn UI paketa



Konverzija u dijagram suradnje

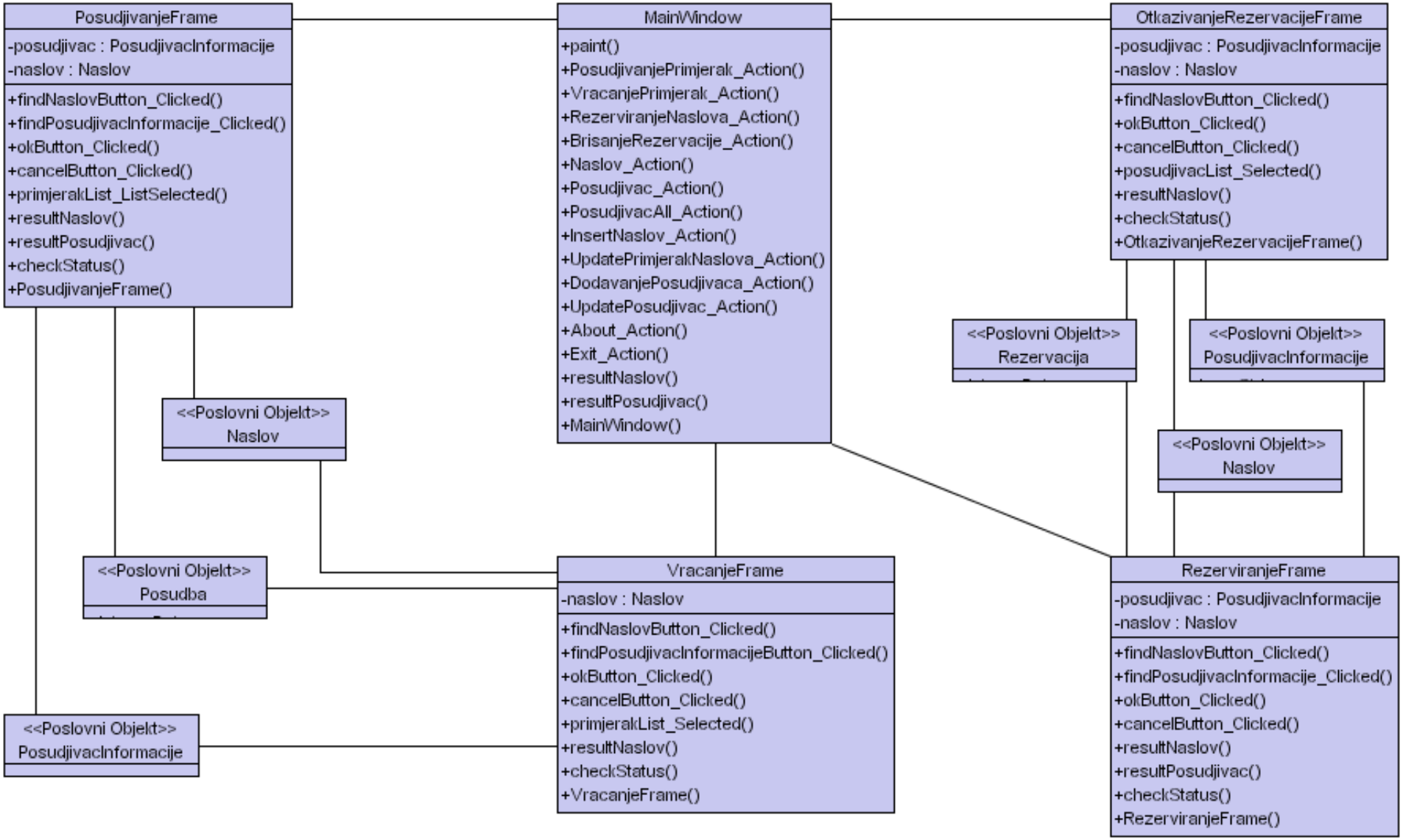
- **Dijagrami suradnje (*Collaboration Diagram*)**
  - Suradnja obuhvaća više od jednog razreda.
  - Dijagram suradnje koristan je kada se želi opisati interakcija.
  - Mogu se koristiti kao alternativa dijagramima slijeda
  - mogu se generirati iz dijagrama slijeda i obrnuto (VP-UML *Synchronize To Colaboration Diagram, To Sequence*).



Dizajn korisničkog sučelja

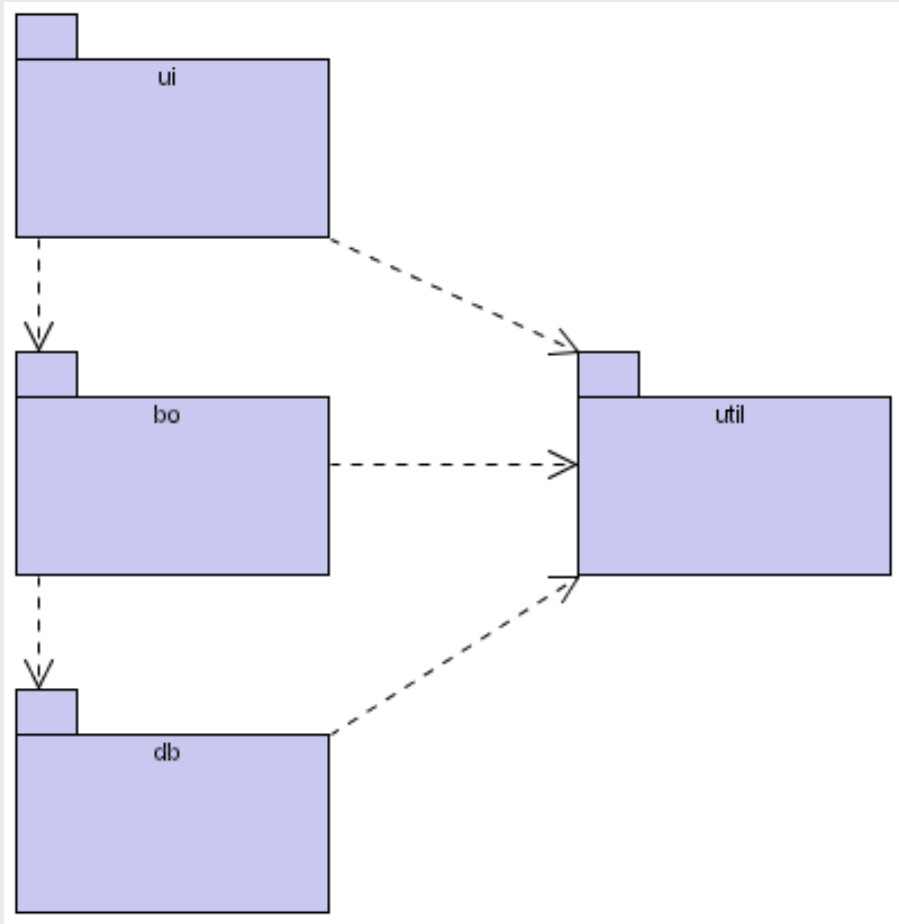
- **Stvaranje korisničkog sučelja posebna je aktivnost tijekom faze dizajna.**
  - Korisničko sučelje u aplikaciji za upravljanje knjižnicom zasnovano je na slučajevima korištenja te je podijeljeno na sljedeće cjeline, od kojih svaka ima svoj posebni izbornik u glavnom izborniku:
- **Funkcije.**
  - Prozori za primarne funkcije u sustavu; posuđivanje i vraćanje jedinica te rezervaciju naslova.
- **Informacije.**
  - Prozori za pregled informacija u sustavu o naslovima i korisnicima.
- **Održavanje.**
  - Prozori za održavanje sustava; dodavanje, izmjena i brisanje naslova, korisnika ili jedinica
- **Sljedeća slika prikazuje primjer dijagrama razreda u UI paketu koji sadrži tipične rukovatelje događajima (*Event Handlers*).**
  - Atributi za gumbe, natpise i polja za unos nisu prikazani.

Razredi korisničkog sučelja



Ugradnja (implementacija)

- **Programiranje tijekom faze konstrukcije ili implementacije**
  - Organizacija koda može biti prikazana dijagramom komponenti koji u modelu dizajna sadrži (u ovom slučaju) jednostavno preslikavanje (mapiranje) logičkih paketa u komponente.
  - Razredi su također mapirani prema paketima
- **Dijagram komponenti (*Component Diagram)***
  - Dijagram komponenti prikazuje različite komponente sustava i njihove međuzavisnosti.
  - Komponenta je fizički programski modul.
  - Ovisnosti među komponentama prikazuju kako promjena u jednoj komponenti može utjecati na ostale, a mogu biti komunikacijske ovisnosti, ovisnosti tijekom prevođenja, itd.



Ugradnja

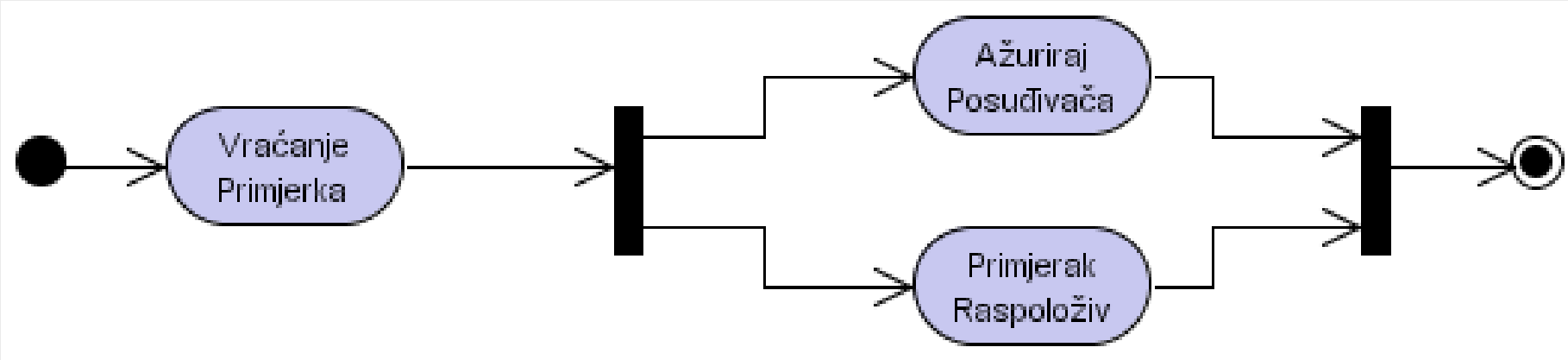
- **Ručno kodiranje ili generiranje izvornog koda**
  - specifikacije se dohvaćaju iz sljedećih dijagrama
- **specifikacija razreda:**
  - specifikacija razreda detaljno prikazuje potrebne atribute i operacije
- **dijagram razreda:**
  - statička struktura i ovisnosti o drugim razredima
- **dijagram stanja:**
  - dijagram stanja za razred koji prikazuje moguća stanja i prijelaze koje treba obrađivati (s operacijama koje okidaju prijelaze)
- **dinamički dijagrami (slijeda, suradnje i aktivnosti) u kojima su uključeni objekti razreda:**
  - prikaz ugradnje metoda i uporabe objekata razreda
- **dijagrami slučajeva korištenja i specifikacije:**
  - prikazuju rezultat sustava
  - koriste se kada programer treba više informacija o tome kako će se sustav koristiti (kada se ne vidi kontekst).

Aktivnosti

- **Aktivnost (*Activity*)**
  - bilo stvarni proces ili izvršavanje neke procedure, npr. metode u razredu.
- **Dijagrami aktivnosti (*Activity Diagrams*)**
  - opis ponašanja, naročito kod paralelne obrade
  - Dijagrami aktivnosti opisuju redoslijed aktivnosti s mogućnostima opisa uvjetnih i paralelnih ponašanja. Oni su varijanta dijagrama stanja.
- **Uvjetna ponašanja prikazana grananjem i spajanjem (*branch*, *merge*)**
  - Grananje ima jedan ulaz i više izlaza od kojih je moguće odabrati samo jedan (međusobno isključivanje).
  - "*else*" smjer grananja odabire se ako su svi drugi uvjeti nezadovoljeni.
  - Spajanje ima više ulaza i jedan izlaz. Predstavlja kraj uvjetnog grananja.
- **Paralelna ponašanja prikazana dijeljenjem i ujedinjavanjem (*fork, join*).**
  - Dijeljenje ima jedan ulaz i više izlaza.
  - Svi izlazni procesi odvijaju se paralelno pri čemu nije važan redoslijed ili međusobno preplitanje koraka pojedinih procesa.
  - Ujedinjavanje služi i za sinkronizaciju, što znači da se radnja nastavlja tek kada svi paralelni procesi završe.

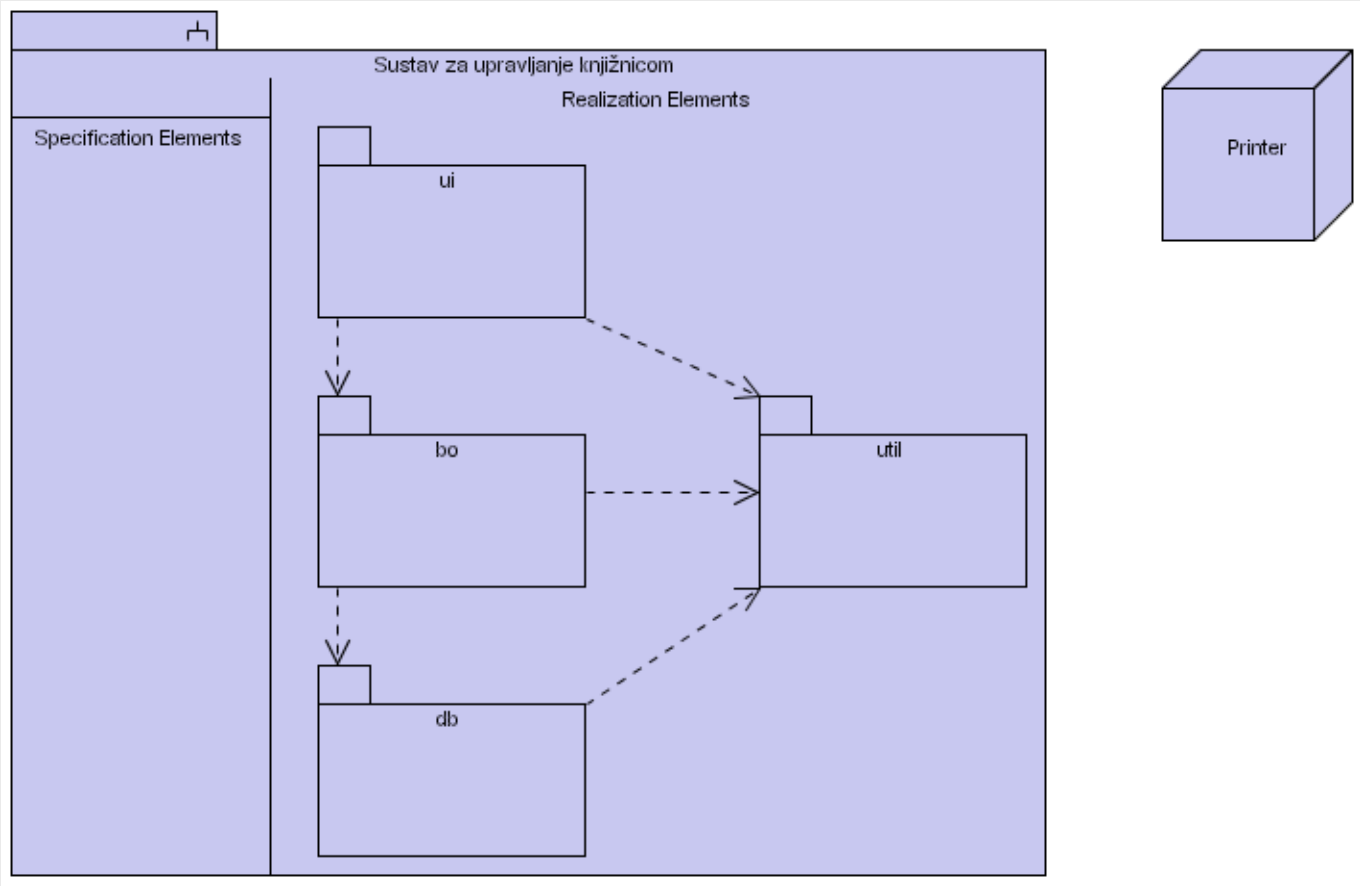
Dijagram aktivnosti

- **Plivaće staze (*Swimlanes*)**
  - koriste se kada se želi navesti tko obavlja određenu operaciju.
  - Dijagram se prikazuje tako da se vertikalno podijeli na zone zaduženja.
  - U svaku zonu ucrtavaju se aktivnosti za koje je zona zadužena (osoba, odjel i sl.).
- **Dijagrami aktivnosti korisni su pri opisivanju paralelnih aktivnosti (npr. dijagram toka ili višenitnost).**
  - Nedostatak je što ne prikazuju izravnu vezu između aktivnosti i objekata.
  - Plivaće staze donekle ublažavaju ovaj nedostatak tako što prikazuju i nositelje aktivnosti.



Dijagrami ugradnje

- **Dijagrami isporuke (*Deployment Diagrams*)**
  - prikazuju fizički odnos softverskih i hardverskih komponenti u sustavu
  - Svaki čvor u dijagramu predstavlja fizičku cjelinu – vrlo često komad hardvera, npr. osobno računalo, pisač, poslužitelj.



Reference

- **Design Java Apps with UML; Hans-Erik Erikkson, Magnus Penker**
  - http://www.ecestudents.ul.ie/Course_Pages/MEng_CS/Modules/EE6421/Examples/UML/Design Java Apps with UML.htm

- **UML Distilled, Second Edition, A Brief Guide to the Standard Object Modeling Language; Martin Fowler, Kendall Scott, Addison-Wesley, 2000.**

- **http://www.patterndepot.com/put/8/JavaPatterns.htm**
- **http://www.eclipse.org**
- **http://www.visual-paradigm.com**