Istorija UML – a

➔ Sredinom 70-ih godina prošlog veka pojavljuju se jezici za OO modelovanje zbog povećane kompleksnosti softverskih sistema.
➔ U periodu 1989-1994.godine drastično se povećao broj metoda (sa 10 na više od 50). Najveći uticaj su imale OOAD(*OO Analysis and Design*, Booch), OMT (*Object Modeling Technique*, Rambaugh), OOSE (*Object Oriented Software Engineering*, Jacobson), Shlaer-Mellor metoda i dr.
➔ **1994. godine** počinje rad na UML-u kada se Rumbaugh pridružio Boochu u firmi Rational (sada je to deo IBM-a).
➔ U oktobru 1995.godine pojavila se verzija 0.8 dokumenta UM (*Unified Method*)
➔ U jesen 1995.godine Jacobson se pridružuje Rational-u i otpočinje rad na objedinjenju UM i OOSE.
➔ U junu 1996.godine pojavila se verzija 0.9 UML-a. Uključuju se važni partneri: DEC, HP, IBM, Microsoft, Oracle i dr.
➔ U januaru 1997.godine grupi OMG (*Object Management Group*) podnet je predlog za standardizaciju UML 1.0. Lista partnera se proširuje: Object Time, Ericsson i dr.
➔ U novembru 1997.godine UML 1.1 postaje standard prihvaćen od OMG.
➔ U junu 2003.godine izlazi verzija UML 2.

UML-dijagrami

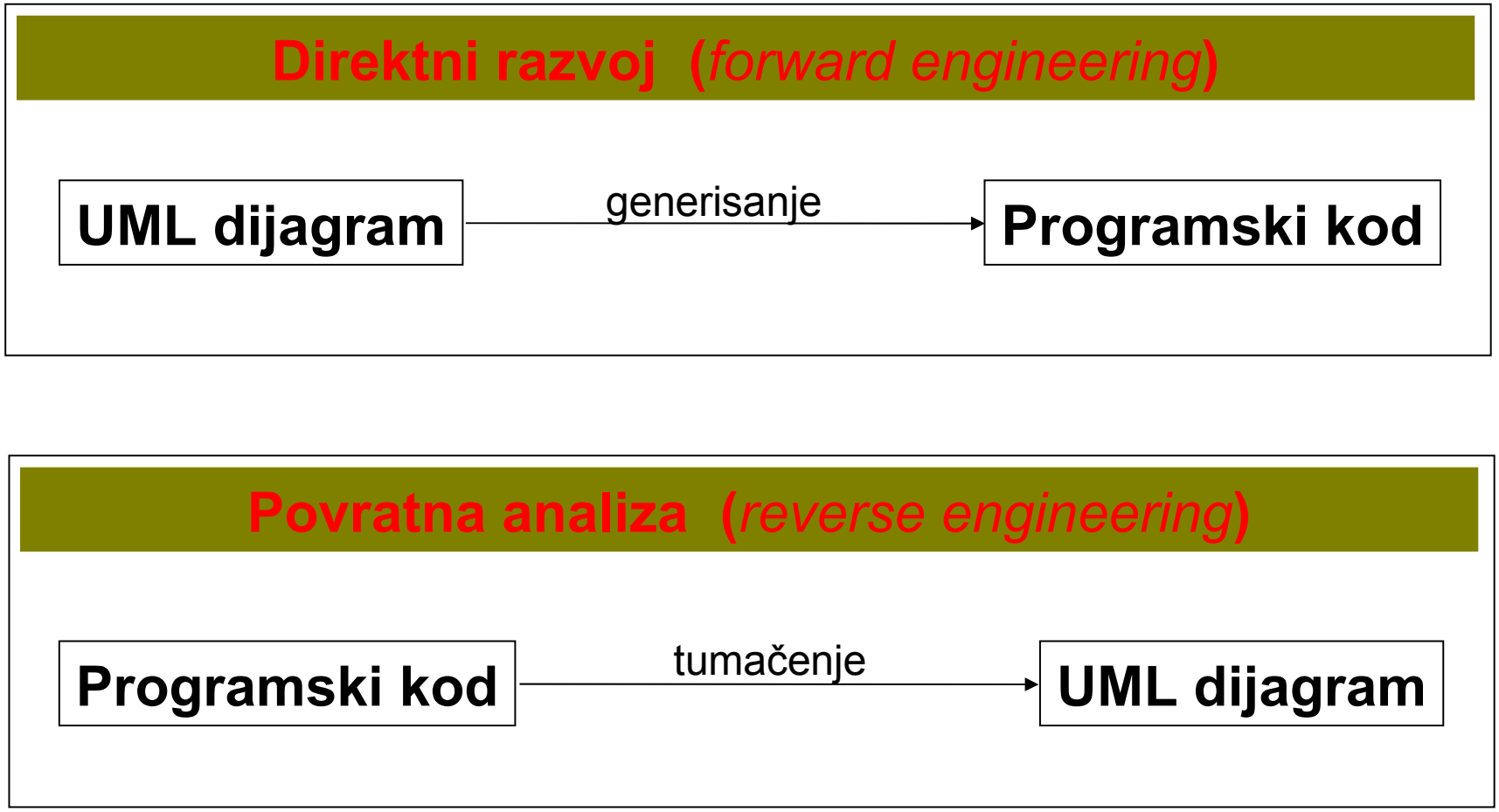
U UML-u je osnovni pojam dijagram, jer je UML jezik za vizuelno predstavljanje informacija.

UML predstavlja skup specifikacija OMG-a (http://www.omg.org).

UML 2.0 se distribuira preko 4 specifikacije:

- UML 2.0 Diagram Interchange,
- UML 2.0 Infrastructure,
- Superstructure specification,
- UML 2.0 Object Constraint Language (OCL).

Postupci razvoja



Vrste dijagrama u UML-u

Da bi se razumeo UML i negovi dijagrami zahteva se poznavanje:

- osnovnih blokova za izgradnju UML-a;
- pravila za spajanje blokova u celine;
- opših mehanizama koji se primenjuju u UML-u.

Koje vrste dijagrama postoje u UML-u?

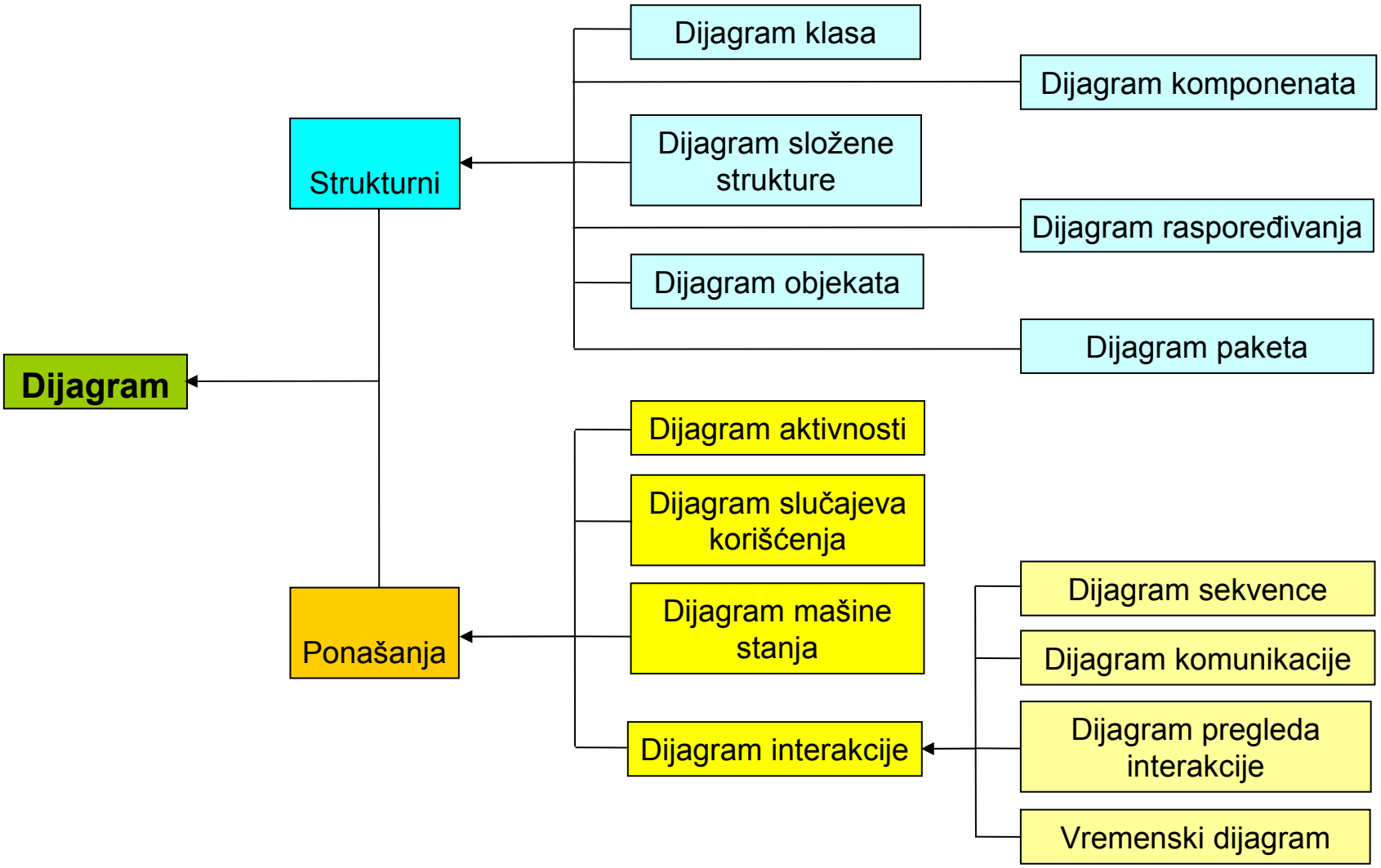
UML 2.0 definiše 13 tipova dijagrama: klasni, komponentni, kompozitne strukture, razvojni, paketni, objektni, dijagrami aktivnosti, komunikacioni, interakcioni, sekvencijalni, dijagrami stanja mašine, vremenski i korisničke funkcije.

Oni se mogu podeliti u 2 kategorije:

strukturni dijagrami (*structural diagrams*) i

dijagrami ponašanja (*behavioral diagrams*).

Klasifikacija dijagrama



Strukturni dijagrami (6 vrsta dijagrama)

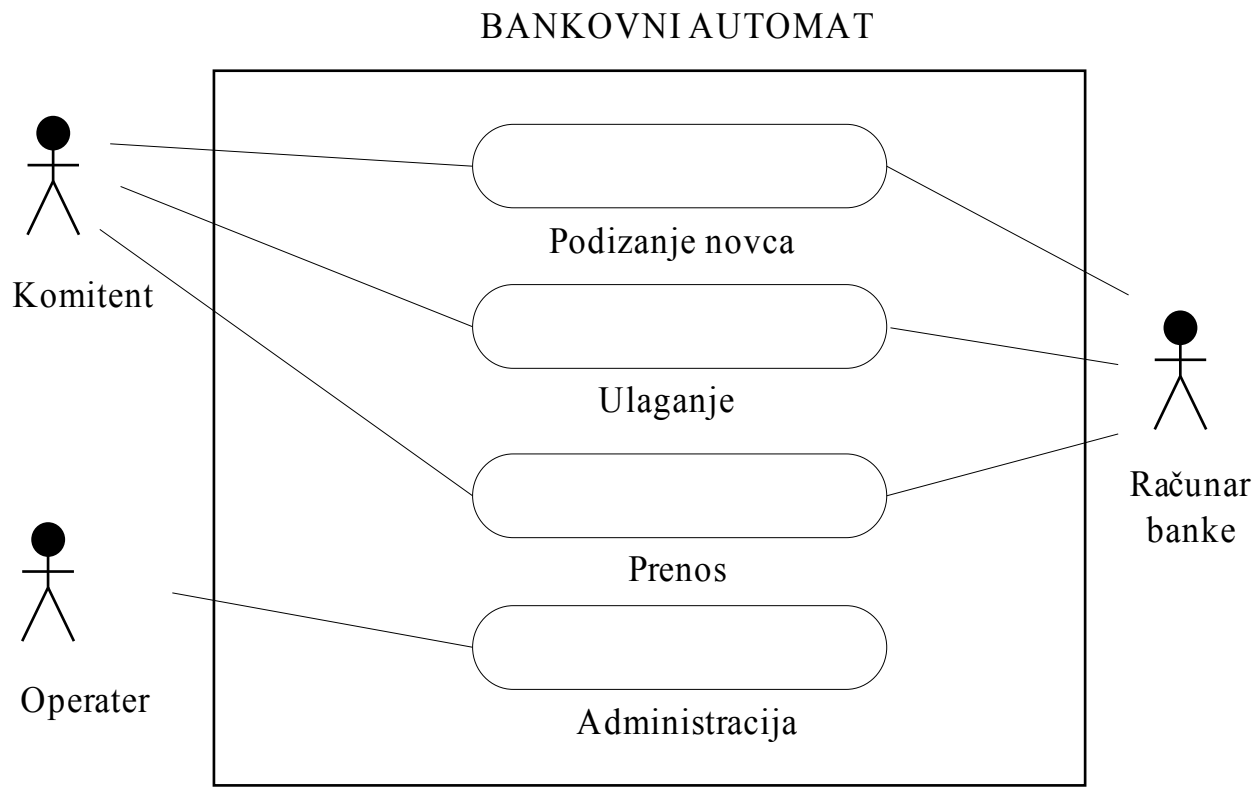
Class Diagram, Object Diagram, Composite structure, Package diagrams, Component Diagram, and Deployment Diagram

Dijagrami ponašanja (7 vrsta dijagrama)

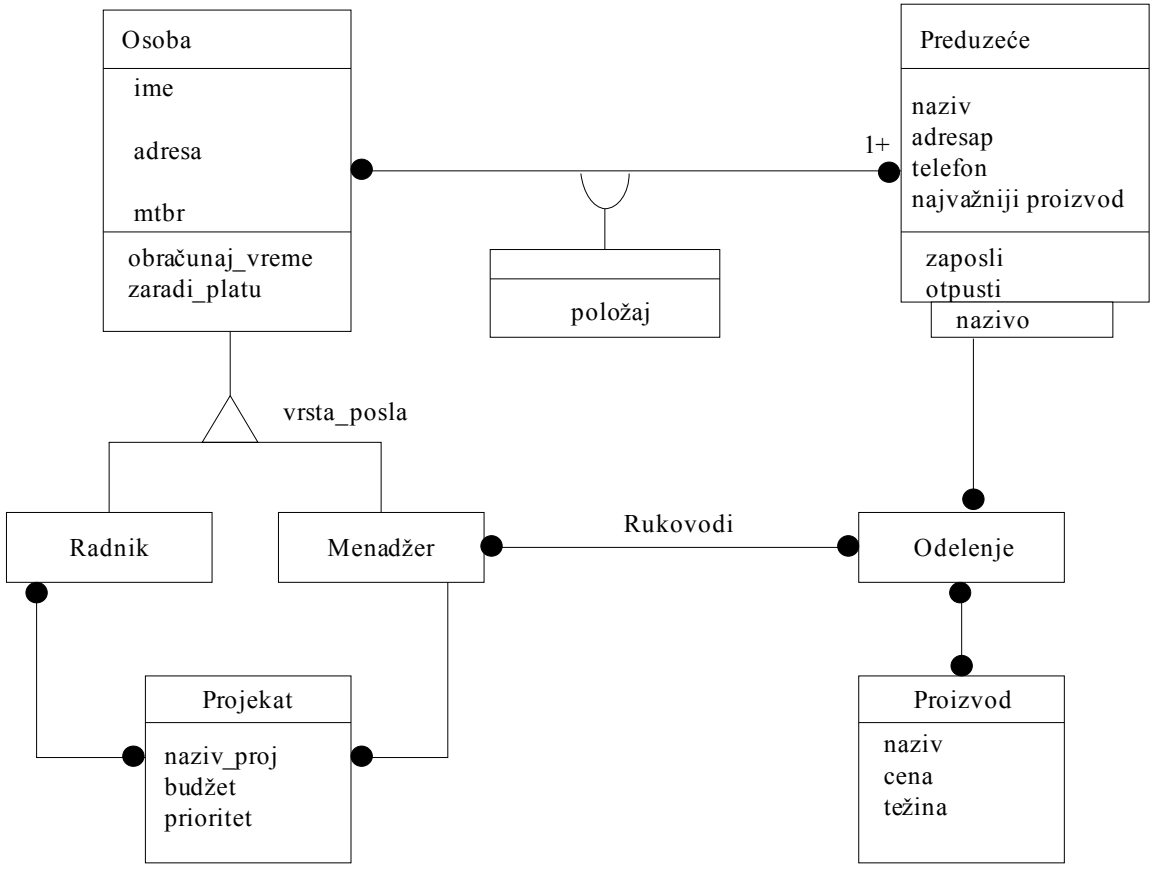
Use Case Diagram, Sequence Diagram, Activity Diagram, Interaction overview diagrams, Collaboration Diagram, State machine Diagram, Timing diagrams

Klasifikacija dijagrama može se izvršiti i drugačije. (Neki autori razlikuju: statičke, dinamičke i fizičke grupe dijagrama)

PRIMER DIJAGRAMA SLUČAJEVA KORIŠĆENJA

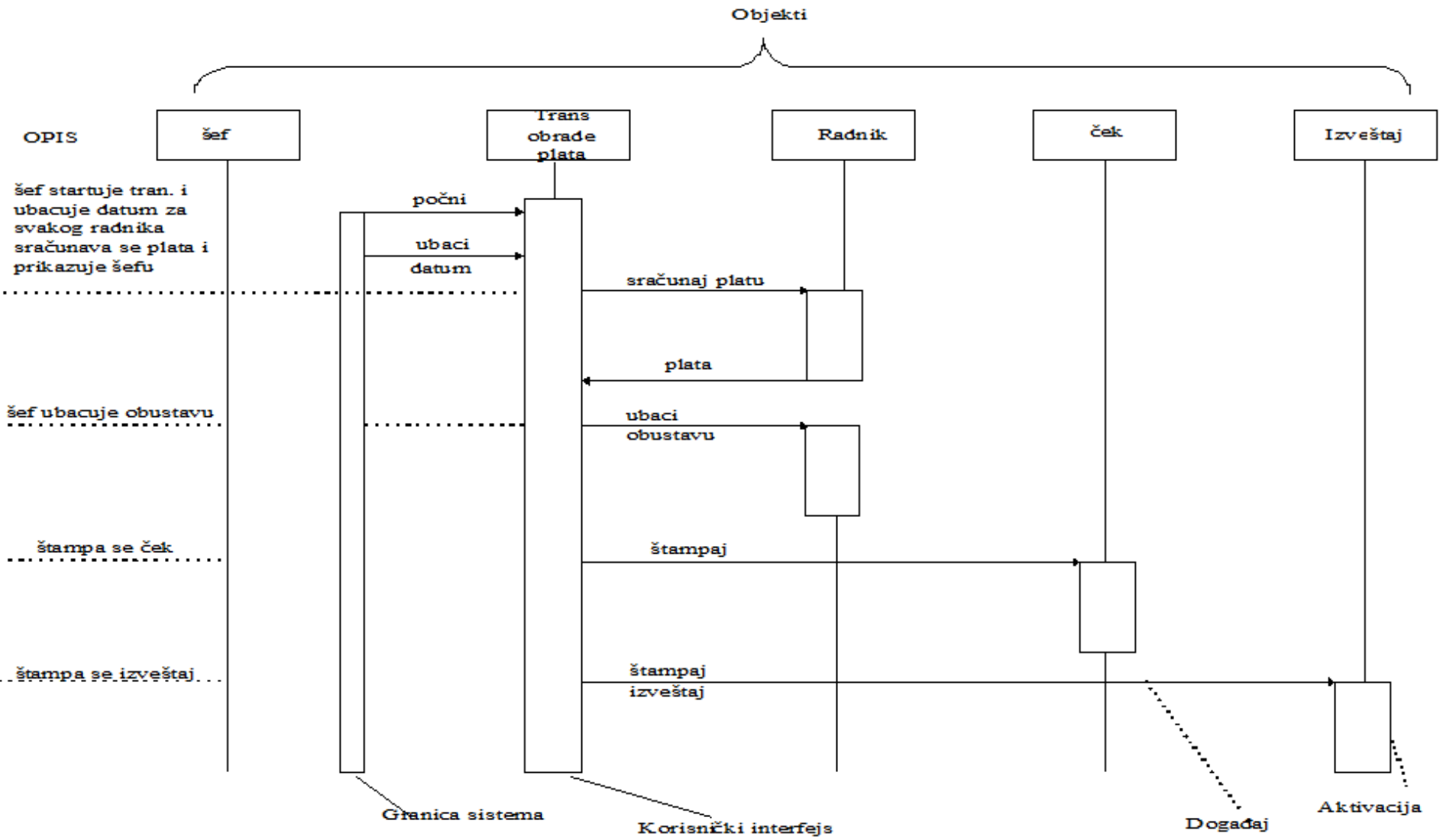


PRIMER DIJAGRAMA KLASA

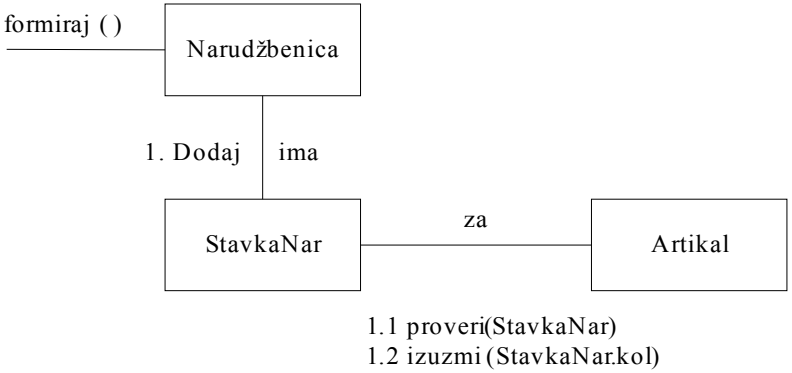


{najvažnijii_proizvod=Proizvod.naziv
za max (proizvod.cena)}

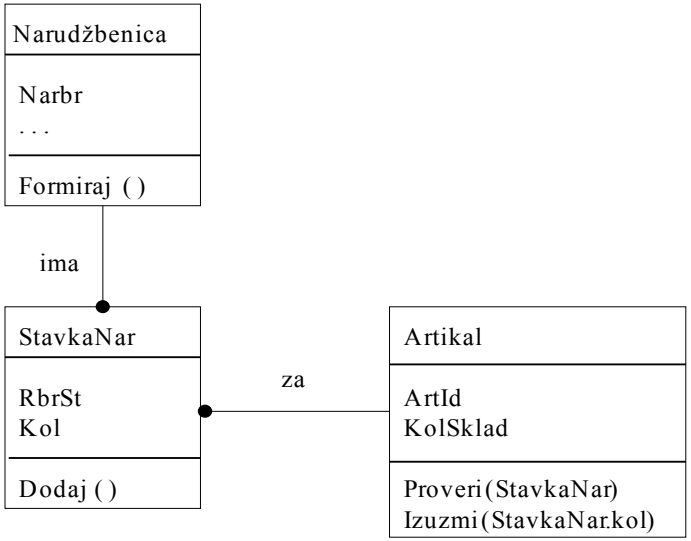
PRIMER DIJAGRAMA SEKVENCI



PRIMER DIJAGRAMA KOMUNIKACIJE (KOLABORACIJE)

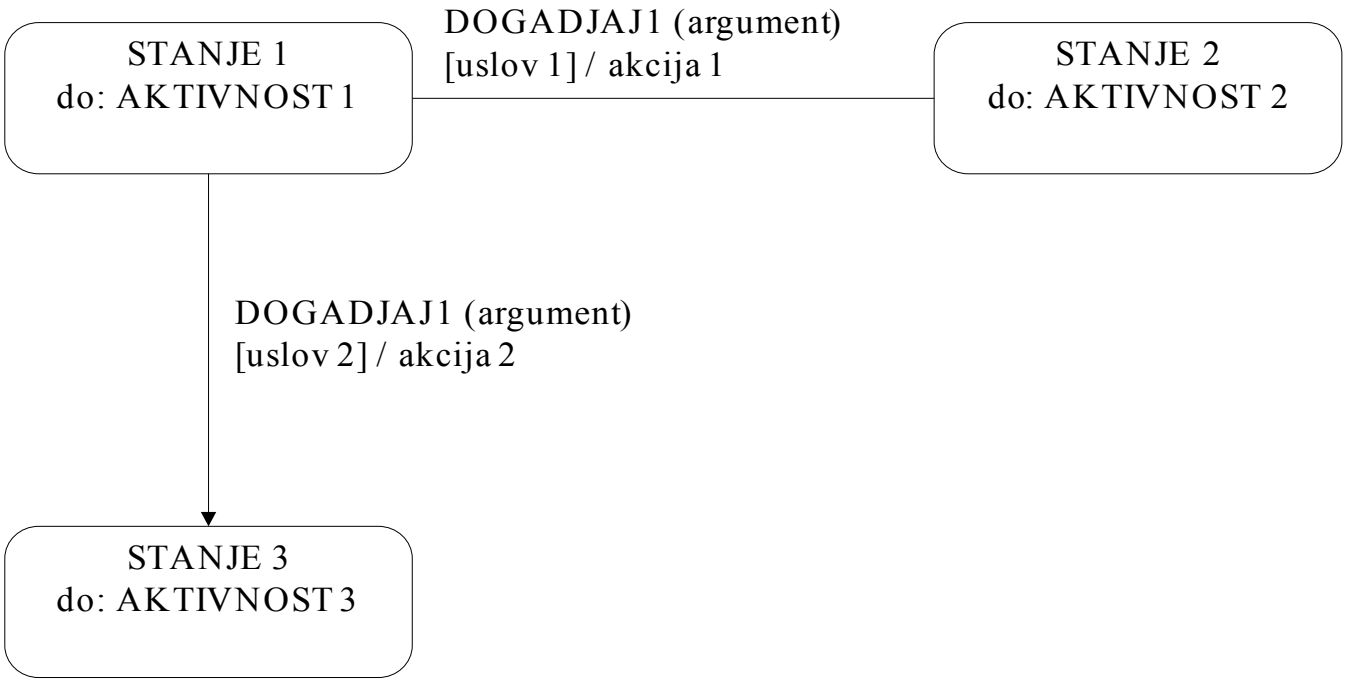


(a) Dijagram kolaboracije

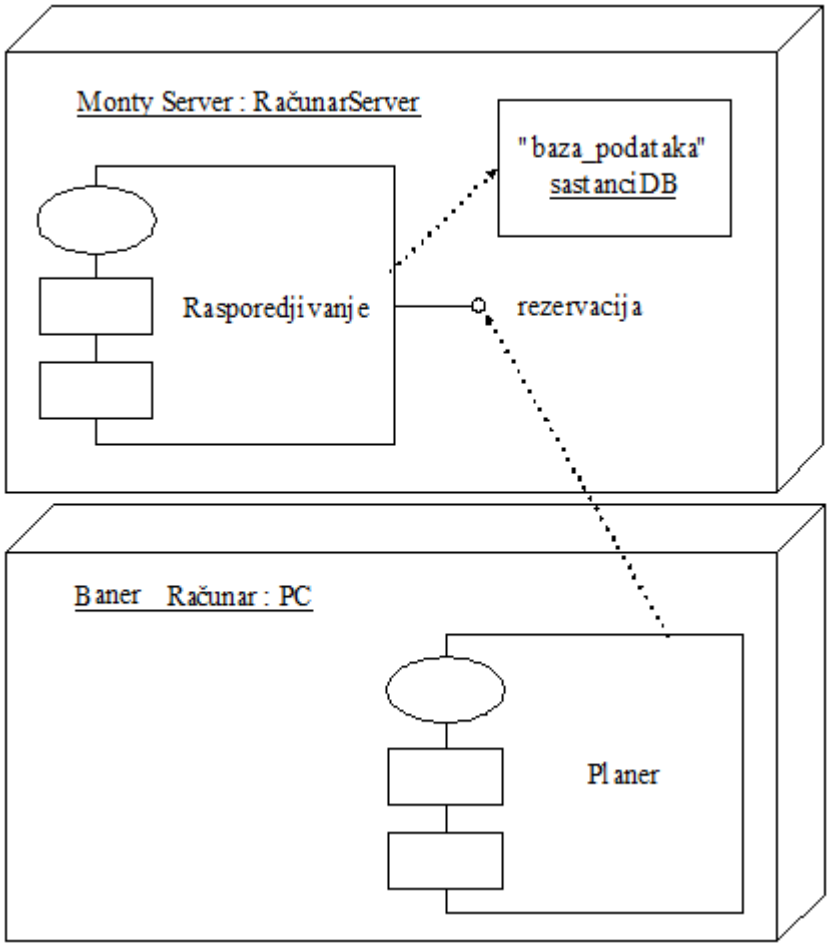


(b) Dijagram klasa

PRIMER DIJAGRAMA MAŠINE STANJA



PRIMER DIJAGRAMA RASPOREĐIVANJA



Dijagrami UML-a

Dijagram	Namena	Uvedeno u verziji
Aktivnosti	proceduralno i paralelno ponašanje	UML 1
Klasa	klase, odlike i veze	UML 1
Komponenata	struktura i veze komponenata	UML 1
Komunikacije	interakcija između objekata-naglasak na vezama	UML 1
Mašine stanja	kako događaji menjaju objekat tokom njegovog postojanja	UML 1
Objekata	primeri konfiguracije instanci	Nezvanično u UML 1
Paketa	hijerarhijska struktura tokom prevođenja	Nezvanično u UML 1
Pregleda interakcije	kombinacija dijagrama sekvence i aktivnosti	Novo u UML 2
Raspoređivanja	raspoređivanje artefakata po čvorovima	UML 1
Sekvence	interakcija između objekata-naglasak na sekvenci	UML 1
Složene strukture	razlaganje klasa tokom izvršavanja	Novo u UML 2
Slučajeva korišćenja	Interakcija korisnika i sistema	UML 1
Vremenski	interakcija između objekata-naglasak na vremenskoj promeni	Novo u UML 2

Način korišćenja UML-dijagrama

Pogodno je početi sa klasnim dijagramima (najčešće korišćeni) i sekvencnim dijagramima. Kasnije se može širiti skup dijagrama.

Neki autori razlikuju 4+1 pogleda na sistem u procesu pravljenja modela (za svaki od njih se vezuje neki tip dijagrama):

- Dizajnerski (klasni, paketni, ...)
- Razvojni (komponetni, razvojni, interakcioni, ...)
- Implementacioni (slož. strukture, raspoređivanja ...)
- Procesni (interakcioni, dijagrami aktivnosti, ...) i
- Korisnički (korisnički slučajevi, interakcioni, ...)

Gde se najčešće koristi?

- **za dizajn softvera;**
- **opis komunikacije kod softverskih i poslovnih procesa;**
- **navođenje detalja softverskih sistema u toku analize i opisa zahteva;**
- **dokumentovanje postojećeg sistema, procesa ili organizacije.**

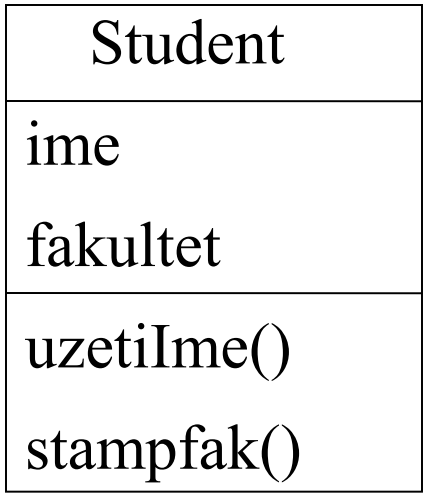
Pravila primene:

- **skoro sve u u UML-u je opciono**
- **UML modeli su retko kada kompletni**
- **UML je dizajniran tako da bude otvoren za interpretaciju**
- **UML je dizajniran tako da bude proširiv.**

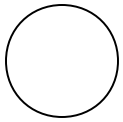
Osnovni gradivni elementi dijagrama

Svi dijagrami u UML-u sastoje se od konačnog skupa grafičkih simbola koji se povezuju međusobno. Pored svakog od njih može stajati neko tekstualno objašnjenje. Ovde ćemo opisati najčešće korišćene grafičke elemente u dijagramima.

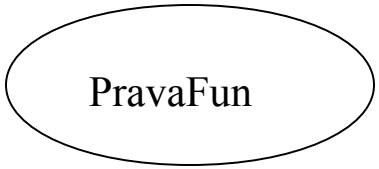
Većina ovih gradivnih blokova je povezana sa nekim bitnim pojmovima iz softverskog inženjerstva, a pre svega OO programiranja.



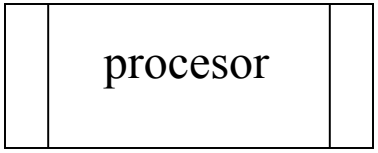
Blok za predstavljanje klasa i elemenata klase.



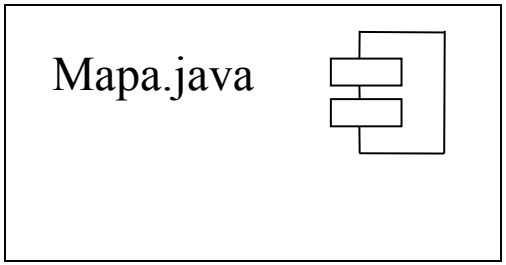
Grafički simbol za predstavljanje interfejsa



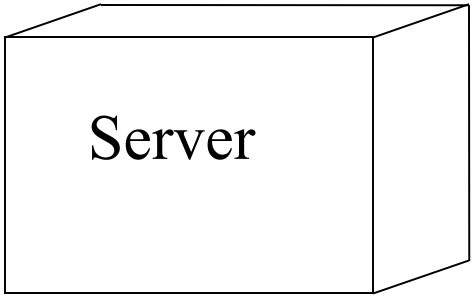
Blok za predstavljanje korisničkih funkcija (use case) pomoću kojih se opisuju ponašanja u modelu.



Grafički simbol za predstavljanje aktivne klase



Grafički simbol za predstavljanje komponenti.



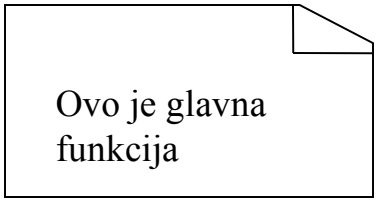
Grafički simbol za predstavljanje čvora – fizičkog elementa koji predstavljaju neki računarski resurs.

Nacrtaj

Grafički simbol za prikaz interakcije između objekata. Interakcija se zasniva na razmeni poruka. Za opis niza stanja objekata ili interakcija koriste se konačni automati.

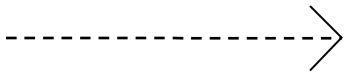
Grafički simbol za prikaz nekog stanja

Grafički simbol za prikaz grupisanja nekih elemenata



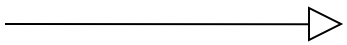
Grafički simbol za prikaz komentara

Opis relacija u UML-u se vrši se raznim vrstama linija

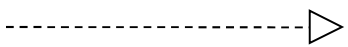


Opis zavisnosti između nekih elemenata

Opis asocijacija, tj skupa veza između objekata



Grafički simbol za opis generalizacije



Opis realizacije

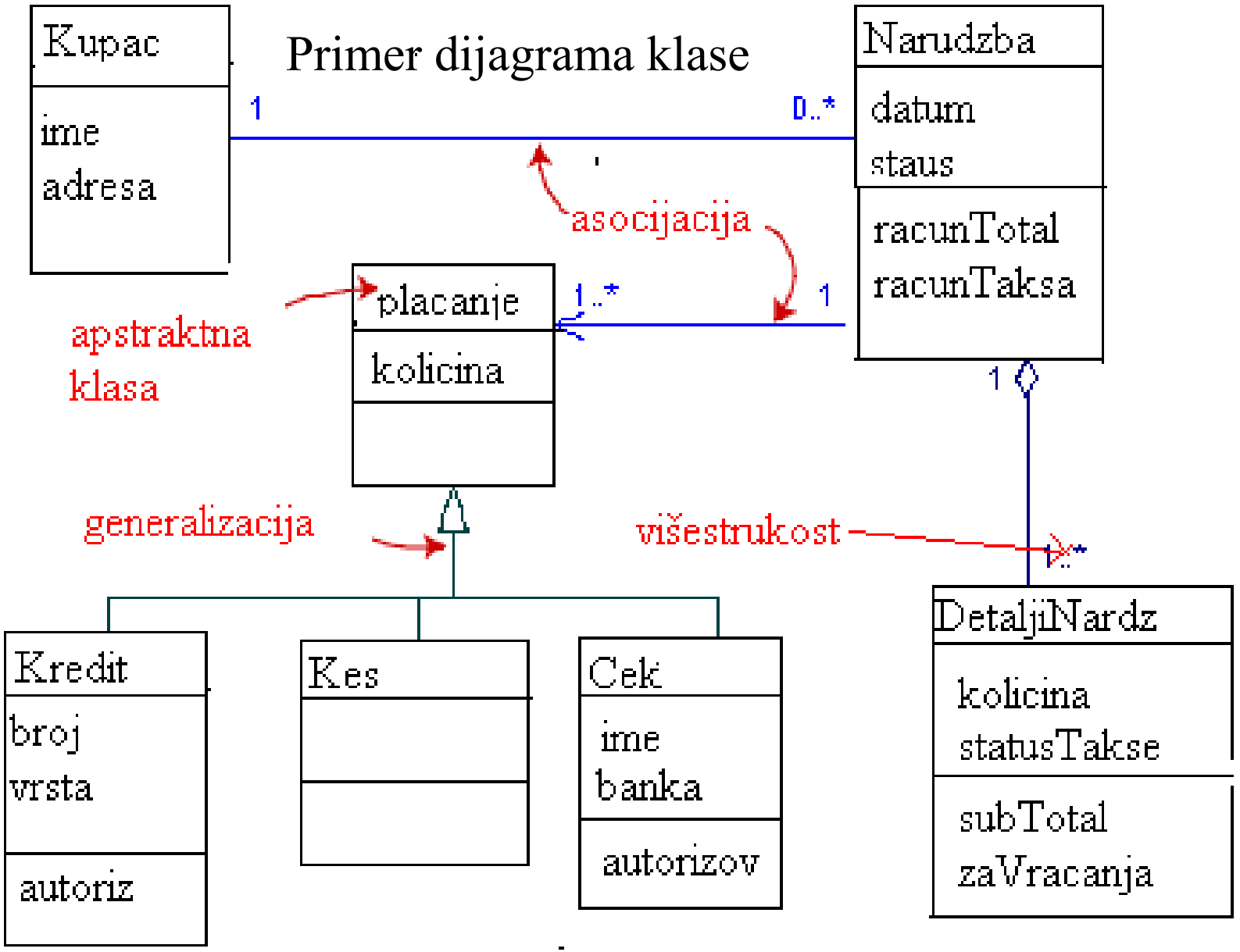
Dijagrami klasa

Najčešće korišćeni dijagrami!

Dijagram klase opisuje tipove objekata u sistemu i razne vrste statičkih relacija koje postoje među njima. Pokazuju svojstva i operacije klase, kao i ograničenja koja se nameću na način povezivanja objekata. U UML-u svojstva i operacije se nazivaju osobine klase.

Da bi se razumeli ovi dijagrami, treba znati osnovne pojmove OO-programiranja: klasa, objekat, instanca, ...

Primer nekih osnovnih dijagrama klasa (ne i jednostavnih!) dat je na sledećoj slici.


Primer dijagrama klase
Kupac
ime
adresa
1
0..*
Narudzba
datum
staus
racunTotal
racunTaksa
asocijacija
placanje
kolicina
1..*
1
apstraktna klasa
1
generalizacija
višestrukost
0..*
Kredit
broj
vrsta
autoriz
Kes
Cek
ime
banka
autorizov
DetaljiNardz
kolicina
statusTakse
subTotal
zaVracanja

U UML-u važe sledeće sugestije za dijagrame klasa:

Ime klase treba da počne velikim slovom,

da bude centrirano na gornjem delu dijagrama,

ispisano bold-fontom,

ispisano italikom ako je klasa apstraktna.

Svojstva klasa

Svojstva – strukturne osobine klasa (polja - instancne promenljive klase). Pojavljuju se kao atributi i asocijacije.

Atribut se pojavljuje u boksu za klasu kao linija teksta u kojoj se opisuju njegova svojstva: vidljivost, ime, tip, višestrukost, podrazumevana vrednost i stringovno svojstvo. Odnosno:

vidljivost **ime** tip višestrukost podraz-vrednost {string-svojstvo, ograničenja}

Primer: - ime: String[1] = ‘’Nesto’’ {readOnly}

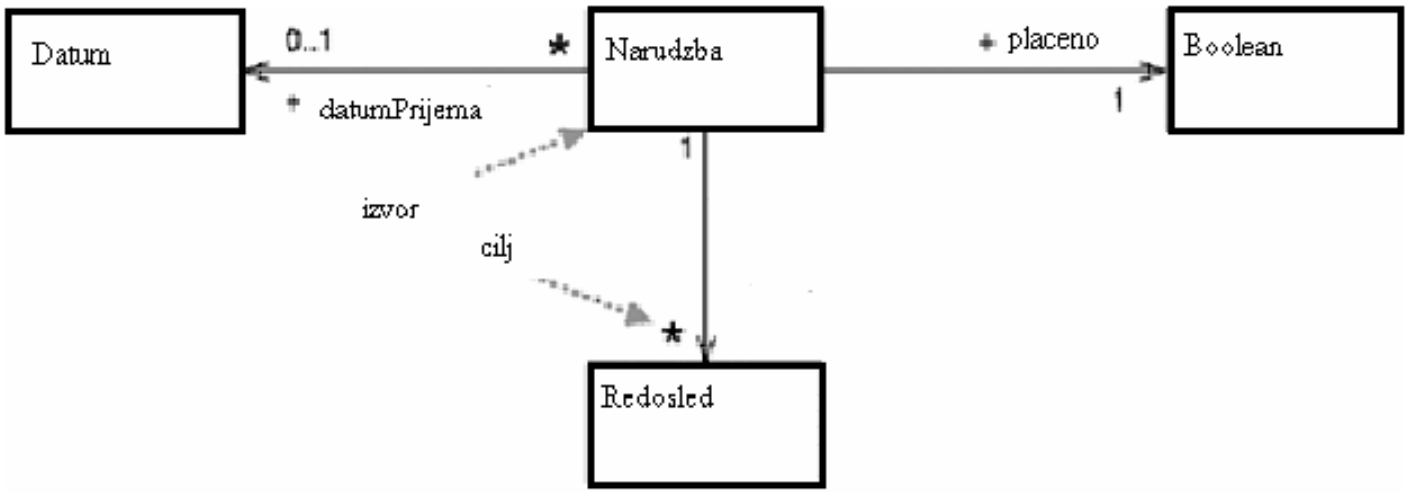
Vidljivost: public (+) ili private (-), protected (#) ili package (~)

Ime: obavezno (odgovara imenu instance)

Ostala svojstva atributa nisu obavezna.

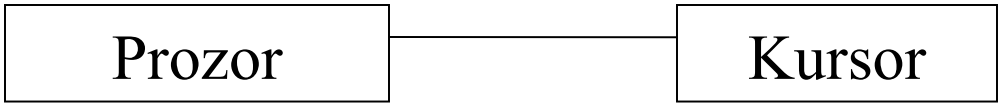
Narudzbenica
+datumPrijema: Datum [0..1] +placeno: Boolean [1]

Asocijacije predstavljaju drugi način da se zabeleže svojstva.

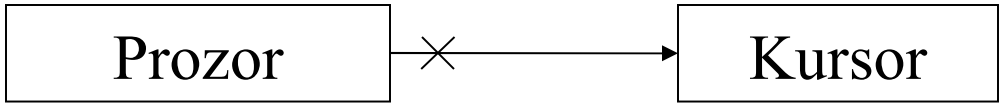


Asocijacija je puna linija između 2 klase usmerena od izvorne ka odredišnoj klasi. Ime svojstva je usmereno ka odredišnoj klasi.

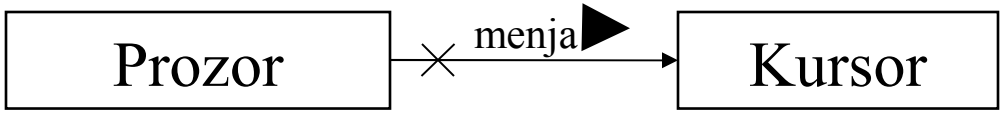
Višestrukost se bolje vidi iz asocijacija. Višestrukost pokazuje kako više objekata može biti povezano asocijacijom. Najčešće: 1 (jedan kupac jedna narudžba), 0..1 –nula ili jedan i * - nula ili više. Generalno *min..max*



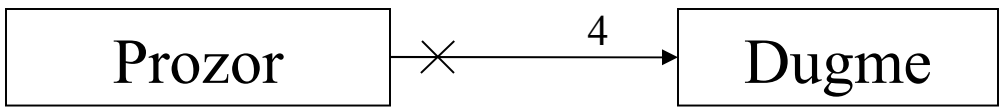
Obično se tumači ovako:
Prozor ima kursor.



Ne možemo se kretati od instace kursor ka instanci prozor.



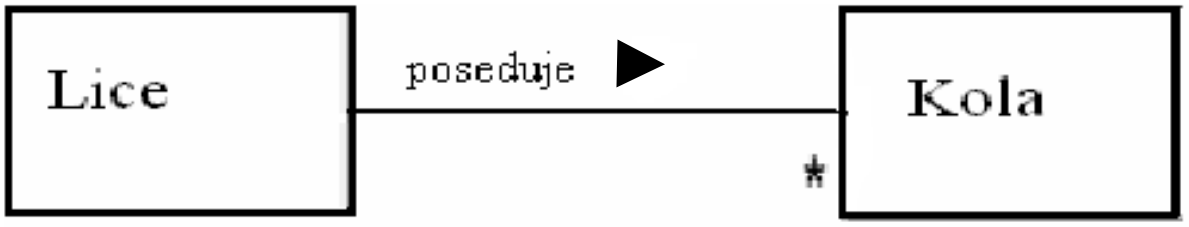
Iz prozora se može menjati kursor



Višestrukost u asocijaciji

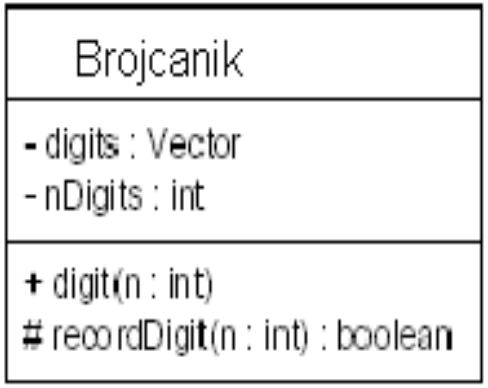
Asocijacije mogu biti dvostrane:

Prethodna asocijacija može se opisati i ovako:



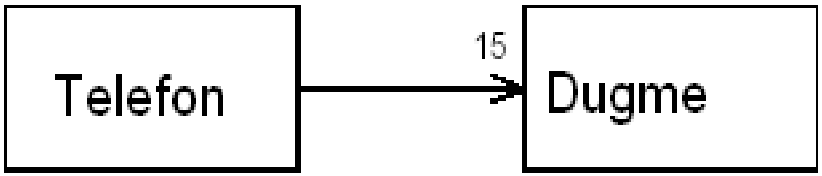
Primeri klasnih dijagrama i asocijacija sa odgovarajućim Java-kôdom

Primer UML dijagrama klase i odgovarajućeg Java-kôda:




```
public class Brojcanik
{
   private Vector digits;
    private int nDigits;
   public void digit(int n);
   protected boolean recordDigit(int n);
}
```

Primer asocijacije i odgovarajućeg Java-koda




```
public class Telefon
{
   private Dugme nDugmad[15];
}
```

Primer višestrukosti


```
public class Imenik
{
   private Vector brojevi;
}
```

Operacije

Operacije su akcije koje klasa zna da izvrši. Operacije moraju da odgovaraju metodima u klasi.

Puna UML sintaksa za operaciju je:

vidljivost ime (parametar-lista): povratni tip {stringovno svojstvo}

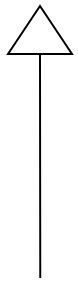
Primer: + balanceOn (date: Date) : Money

Generalizacija

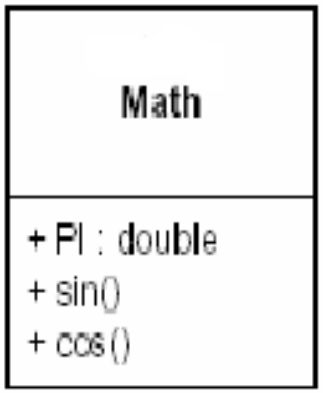
Generalizacija je drugo ime za nasleđivanje.

Generalizacija se prikazuje strelicom od potklase ka superklasi.

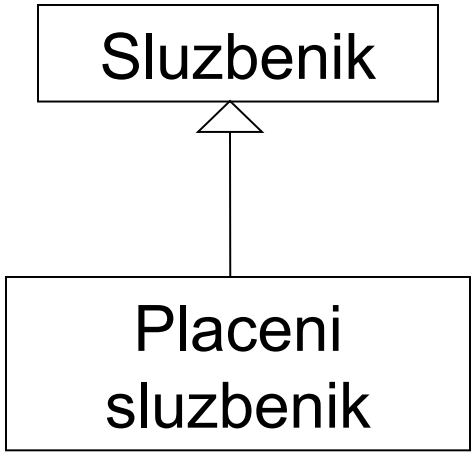
Strelica ima oblik:



Primeri sa Java-kôdom



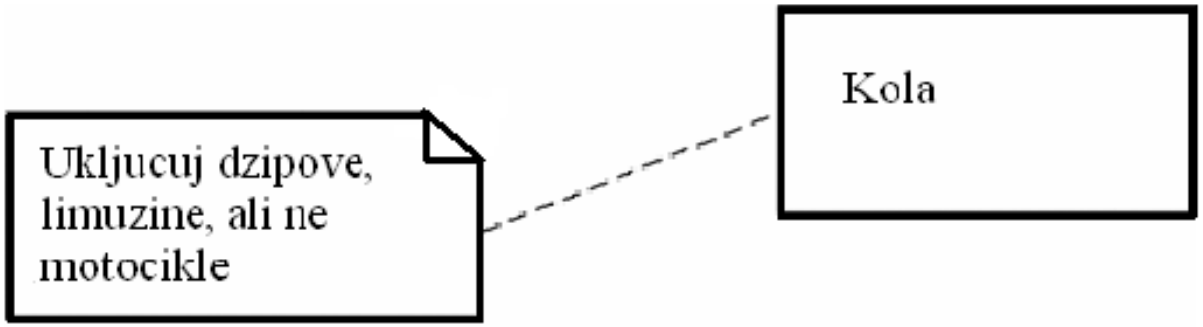

```
public class Math
{
  public static final double PI =
                                3.14159265358979323;

  public static double sin(double theta){...}
  public static double cos(double theta){...}
}
```




```
public class Sluzbenik
{
  …
}
Public class PlaceniSluzbenik extends Sluzbenik
{
…
}
```

Napomene i komentari

Napomene i kometari mogu stajati samostalno ili biti povezani isprekidanom linijom sa elementima na koje se odnose.



Napomene i komentari imaju istu ulogu kao i u drugim programskim jezicima. Upotrebljavaju se da detaljnije objasne pojedine delove modela.

Ovde imaju čak i veću ulogu, jer intenzivno koriste kod kreiranja dokumentacije.