

Specifikacija softverskih sistema

Predavanje br. 4 – Dijagram klasa, 1. deo
Osnovni pojmovi, asocijacije

Gordana Milosavljević

Katedra za informatiku, FTN, Novi Sad
2022.

UML Diagrams

Structure Diagrams

Package
Diagram

Class
Diagram

Component
Diagram

Deployment
Diagram

Object
Diagram

Composite
Structure
Diagram

Behavior Diagrams

Use Case
Diagram

Activity
Diagram

State
Machine
Diagram

Interaction Diagrams

Sequence
Diagram

Communication
Diagram

Interaction
Overview
Diagram

Timing
Diagram

Osnovni elementi dijagrama klasa

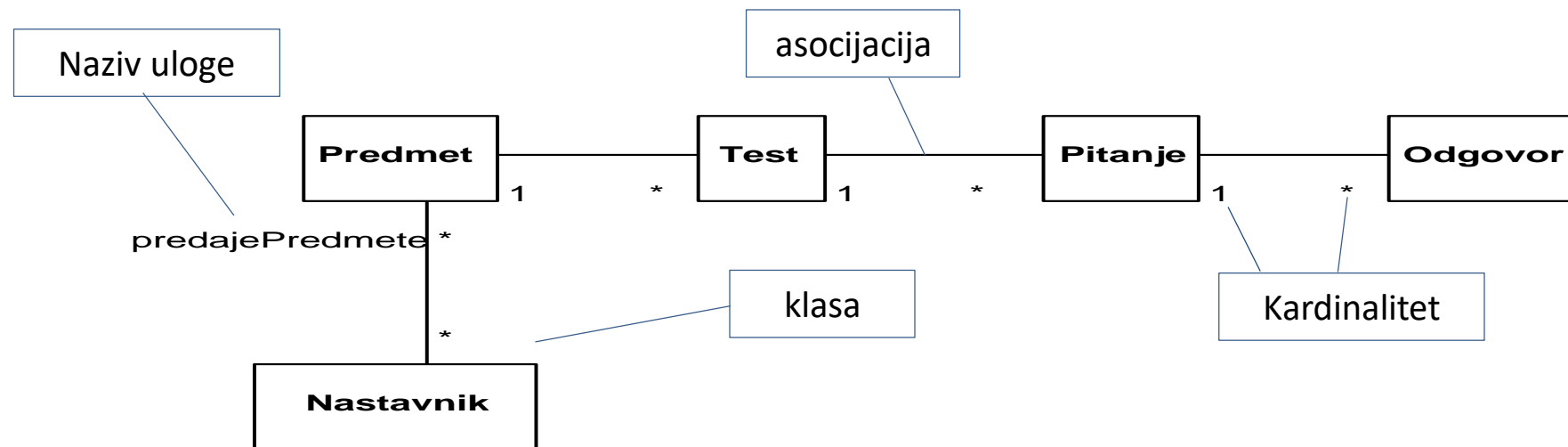
- Klase
 - Obeležja
 - Metode
- Interfejsi
- Veze
 - Asocijacija
 - Generalizacija
 - Veza zavisnosti
 - Implementacija interfejsa

Korišćenje dijagrama klasa

- Tokom analize zahteva
 - za skiciranje strukture realnog sistema za koji se softver implementira
 - za konceptualno (domensko) modelovanje
- Tokom narednih faza
 - za istraživanje različitih projektantskih odluka
 - specifikaciju implementacije
 - generisanje koda (ako se koristi neka od MDSE – *Model Driven Software Engineering* tehnika)
 - dokumentovanje rešenja.
- U zavisnosti od namene, dijagram može biti više ili manje precizan, sa više ili manje detalja.

Primer 1: Konceptualni dijagram klasa na visokom nivou apstrakcije

Izvod iz specifikacije zahteva sistema za elektronsko ocenjivanje studenata: *Nastavnici kreiraju testove za predmete koje predaju. Testovi se sastoje od pitanja, gde svako pitanje treba da ima više ponuđenih odgovora. Jedno pitanje mora imati minimalno jedan tačan odgovor.*



Ovaj dijagram prikazuje uočene koncepte sistema i veze između njih

Kreiranje testova

Predmet: Osnove programiranja Nastavnik: Jovan Jovanović

Testovi

Dodaj... Izmeni... Obriši

Rbr.	Naziv testa	Ukupno poena
1	Dinamičke strukture	100
2	Rukovanje događajima	100

Pitanja

Dodaj... Izmeni... Obriši Pretraga...

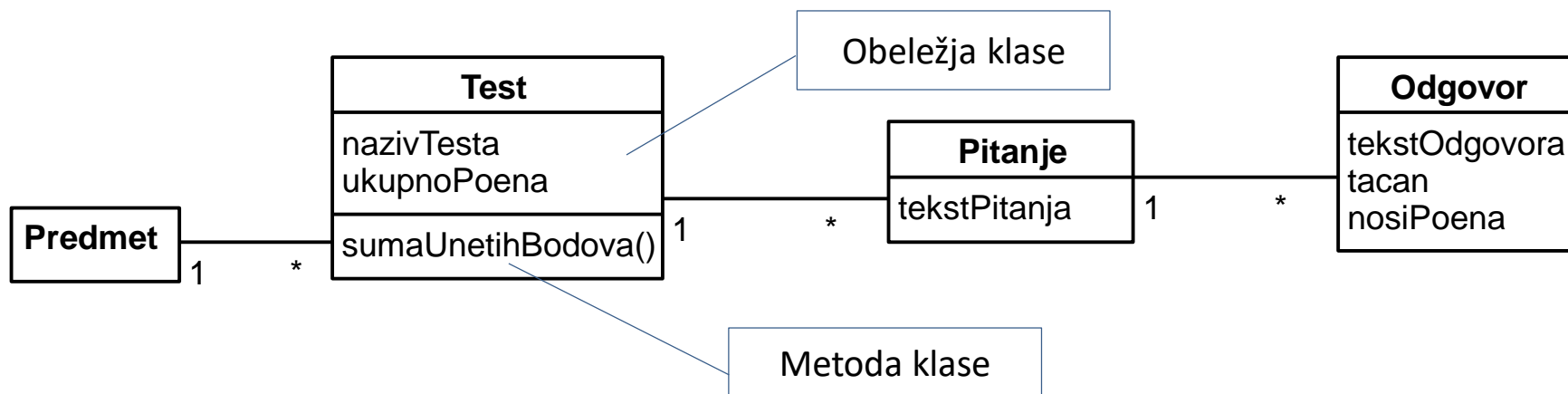
Rbr.	Pitanje
1	Pitanje 1
2	Pitanje 2
3	Pitanje 3
4	Pitanje 4

Odgovori

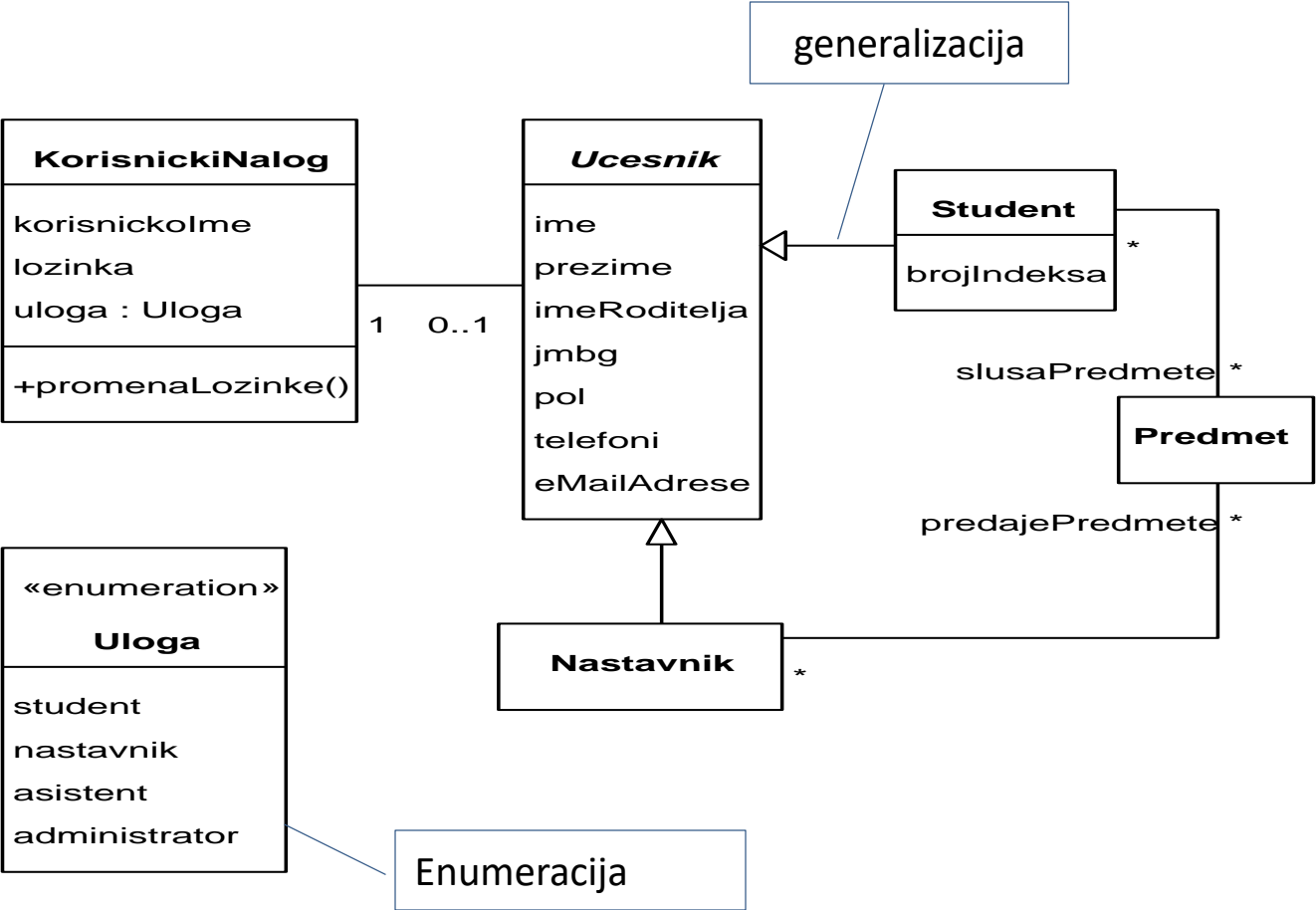
Dodaj... Izmeni... Obriši

Rbr.	Odgovor	Tačan?	Bodovi
1	Odgovor 1	da	3
2	Odgovor 2	ne	-1
3	Odgovor 3	ne	-1

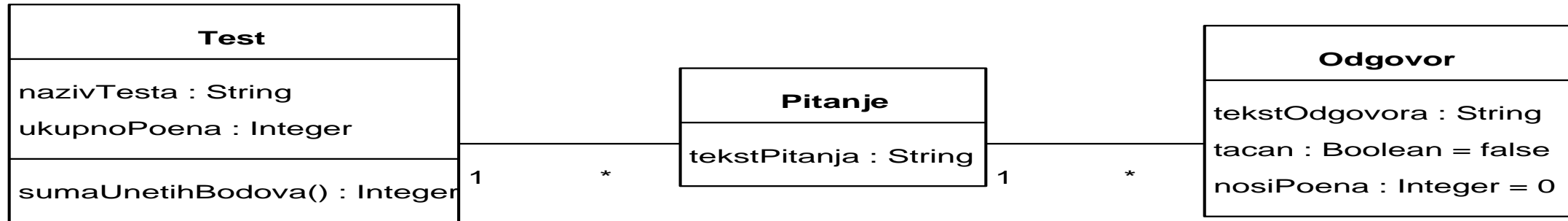
Primer 2: Konceptualni dijagrama klasa, sa malo više detalja



Primer 2b: Konceptualni model koji opisuje učesnike u sistemu elektronskog ocenjivanja

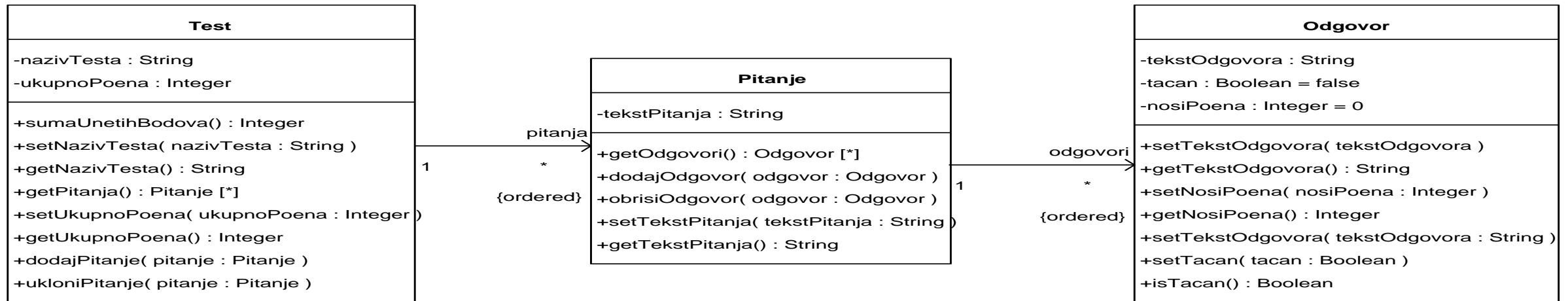


Primer 3: Detaljan konceptualni model



- Obično se ne navode *get* i *set* metode i konstruktori
- Cilj
 - brže modelovanje
 - čitljiviji modeli

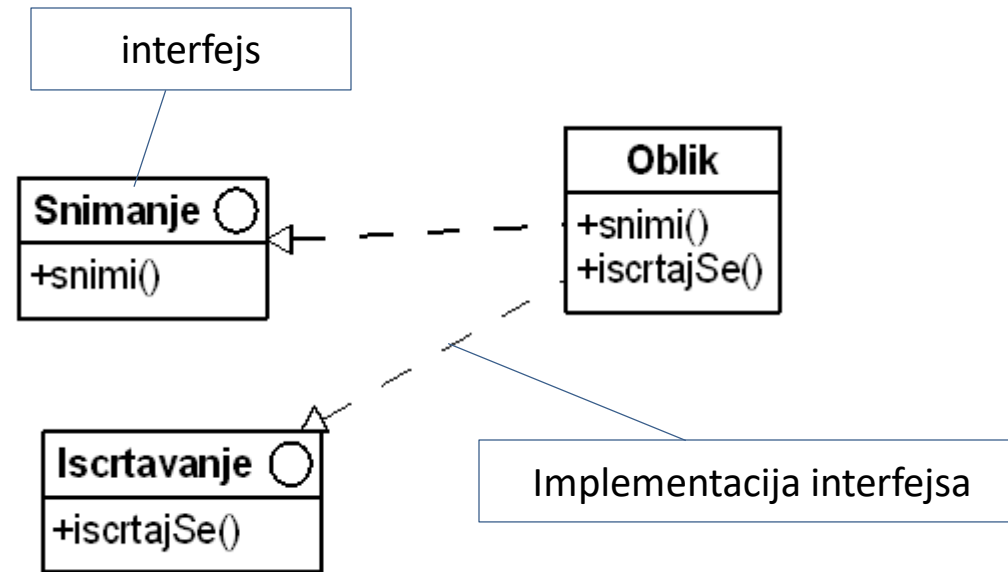
Primer 4: Veoma detaljan implementacioni dijagram klasa



Moguća namena:

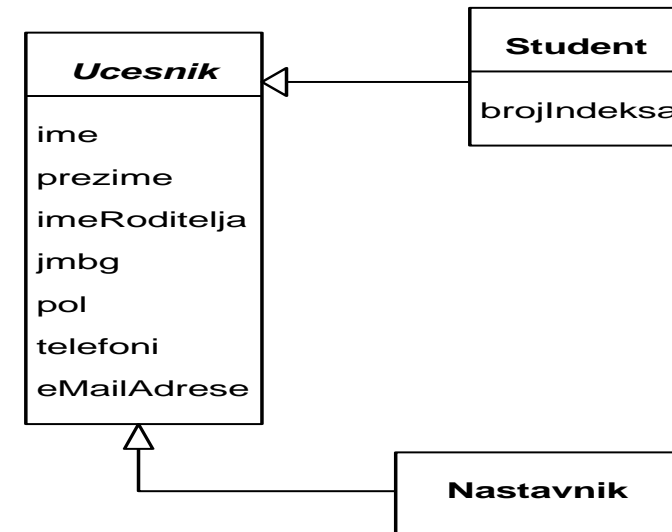
- Specifikacija implementacije od strane softverskog arhitekta za programere-početnike
- Generisanje koda

Primer 5: Implementacija interfejsa



Klase - imenovanje

- Imena klasa se pišu prema Upper Camel Case konvenciji
- Imena treba tako davati da se odmah vidi namena klase
- Primeri imena klasa: PozajmicaKnjige, GrafToka, ReseniTest, DeoLukaFigure
- Apstraktne klase
 - Ime se piše zakošenim slovima
 - Neki alati dodaju {abstract} ispod imena



Obeležja klase

- Nazivi se pišu u Camel Case notaciji
- Primer imena obeležja: imeOsobe, datumDokumenta, broj
- Format za specifikaciju obeležja:

vidljivost naziv: tip-podataka kardinalitet = inicijalna-vrednost {dodatne-opcije}

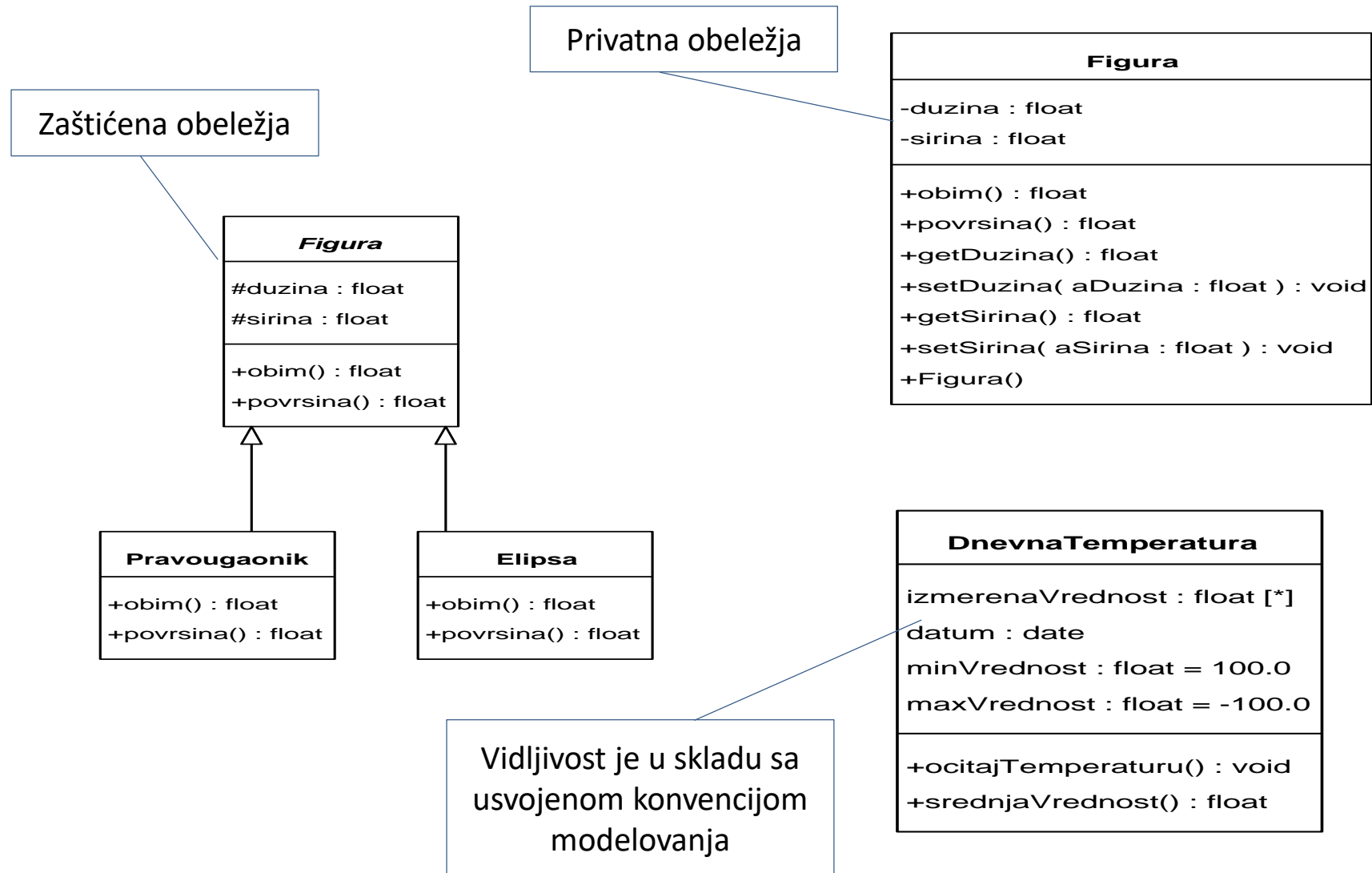
- Primer:

+ brojTelefona: String[0..1] = ""

Vidljivost obeležja

Simbol	Vidljivost obeležja	Značenje
-	private	Privatno obeležje – može mu se pristupiti samo u okviru date klase.
+	public	Javno obeležje – može mu se pristupati bez ograničenja.
#	protected	Zaštićeno obeležje – pristup ima klasa u kojoj je definisano i svi njeni naslednici
~	package	Obeležje je vidljivo na nivou paketa

Vidljivost obeležja - primeri



Tipovi podataka obeležja

- Unapred definisani tipovi
 - Celobrojni,
 - Realni
 - Datumski
 - Logički
 - Znakovni
- Nabrojani tip (enumeration)
- Klasa
- Interfejs

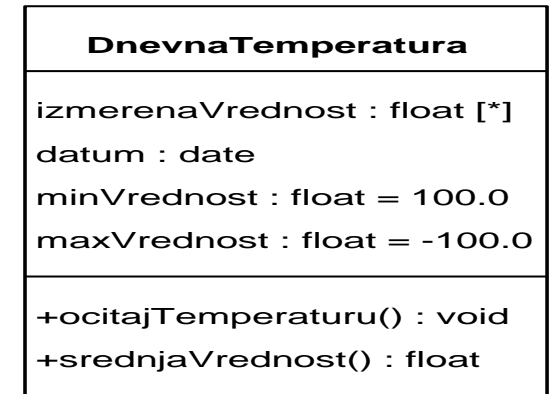
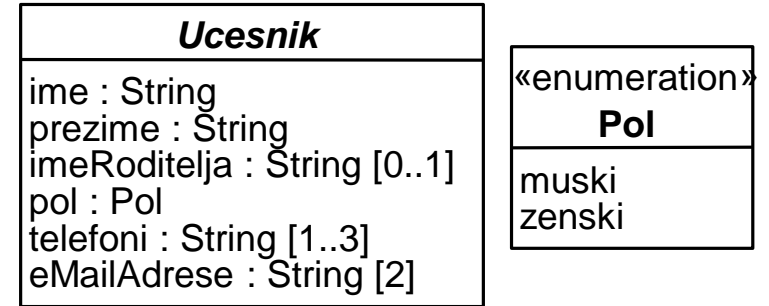
<i>Ucesnik</i>
ime : String prezime : String imeRoditelja : String [0..1] pol : Pol telefoni : String [1..3] eMailAdrese : String [2]

«enumeration» Pol
muski zenski

DnevnaTemperatura
izmerenaVrednost : float [*] datum : date minVrednost : float = 100.0 maxVrednost : float = -100.0
+ocitajTemperaturu() : void +srednjaVrednost() : float

Kardinalitet

- Definiše broj vrednosti određenog obeležja koji može ili mora da postoji u okviru instance klase
- Može da se zadaje kao **interval** ili kao **jedan broj**
- Za interval **m..n**, gde **n** mora biti **>= m**, u okviru klase mora da postoji najmanje **m** vrednosti datog obeležja, a može da postoji najviše **n** vrednosti.
- Ako je kao kardinalitet zadat kao jedan broj, on definiše tačan broj vrednosti koje obeležje mora da ima
 - npr. 5 je ekvivalentno intervalu 5..5.



Inicijalna – podrazumevana vrednost

- Vrednost koju obeležje treba da dobije prilikom instanciranja klase.

Odgovor
tekstOdgovora : String tacan : Boolean = false nosiPoena : Integer = 0

DnevnaTemperatura
izmerenaVrednost : float [*] datum : date minVrednost : float = 100.0 maxVrednost : float = -100.0
+ocitajTemperaturu() : void +srednjaVrednost() : float

```
...  
if (ocitanaTemp < minVrednost)  
    minVrednost = ocitanaTemp;  
if (ocitanaTemp > maxVrednost)  
    maxVrednost = ocitanaTemp;  
...
```

Inicijalna – podrazumevana vrednost

- Vrednost koju obeležje treba da dobije prilikom instanciranja klase.

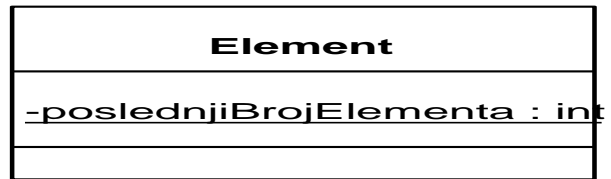
Odgovor
tekstOdgovora : String tacan : Boolean = false nosiPoena : Integer = 0

DnevnaTemperatura
izmerenaVrednost : float [*] datum : date minVrednost : float = 100.0 maxVrednost : float = -100.0
+ocitajTemperaturu() : void +srednjaVrednost() : float

```
...  
if (ocitanaTemp < minVrednost)  
    minVrednost = ocitanaTemp;  
if (ocitanaTemp > maxVrednost)  
    maxVrednost = ocitanaTemp;  
...
```

Statička obeležja

- obeležja kojima se pristupa na nivou klase, ne instance klase



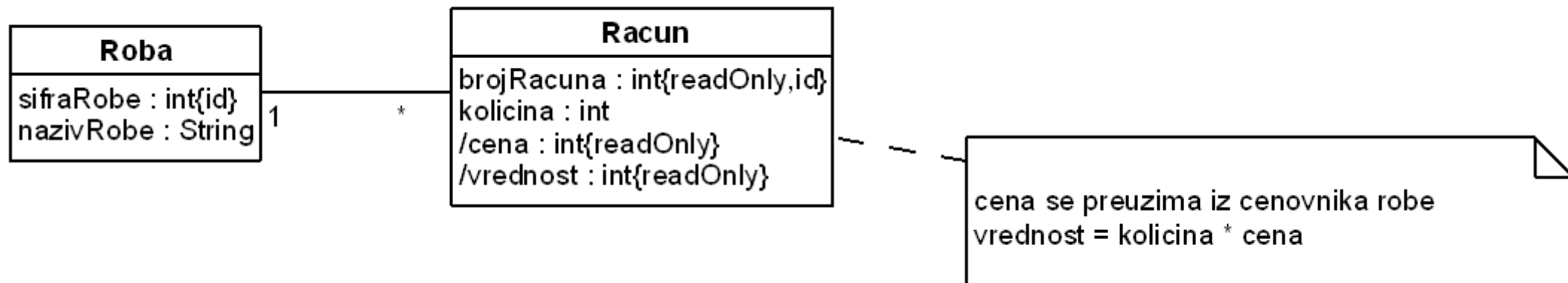
Dodatne opcije obeležja

Opcija	Značenje
Id	Obeležje je identifikator ili deo identifikatora
readOnly	Obeležje ne može da se menja posle inicijalizacije
derived	Obeležje je izvedeno
ordered	Kolekcija je uređena (bitan je redosled elemenata)
unique	Ne može postojati više elemenata sa istom vrednošću u kolekciji
nonunique	Može postojati više elemenata sa istom vrednošću u kolekciji

DnevnaTemperatura
izmerenaVrednost : float [*]{ordered,nonunique} datum : date{readOnly} minVrednost : float = 100.0{readOnly} maxVrednost : float = -100.0{readOnly}
+ocitajTemperaturu() : void +srednjaVrednost() : float

Odnosi se obeležja čiji je kardinalitet veći od 1 (niz ili kolekcija).

Primer izvedenih obeležja



Get i set metode

```
public abstract class Ucesnik {  
    private String ime;  
    private String imeRoditelja;  
    ...  
  
    public String getIme() {  
        return ime;  
    }  
  
    public void setIme(String ime) {  
        if (ime == null)  
            throw new java.lang.IllegalArgumentException();  
        this.ime = ime;  
    }  
  
    public String getImeRoditelja() {  
        return imeRoditelja;  
    }  
  
    public void setImeRoditelja(String imeRoditelja) {  
        this.imeRoditelja = imeRoditelja;  
    }  
    ...  
}
```

<i>Ucesnik</i>	«enumeration» Pol
ime : String prezime : String imeRoditelja : String [0..1] pol : Pol telefoni : String [1..3] eMailAdrese : String [2]	muski zenski

Metode za pristup obeležjima čiji je kardinalitet *

- Obeležja sa kardinalitetom * se u programskom kodu implementiraju kao kolekcija: `List`, `Set`,...
- Obično za njih ne treba da se implementira `set` metoda
- `Get` metoda treba da vrati readonly kopiju kolekcije
- Obično su potrebne metode za dodavanje i brisanje elemenata iz kolekcije, ako nekom spolja treba omogućiti da doda ili obriše element

```
public class DnevnaTemperatura {
```

```
    private java.util.List<Double> izmereneVrednosti = new  
    java.util.ArrayList<Double>();
```

```
    public java.util.Collection<Double> getIzmereneVrednosti() {  
        return Collections.unmodifiableCollection(izmereneVrednosti);  
    }
```

```
    ...  
}
```

DnevnaTemperatura
izmerenaVrednost : float [*] datum : date minVrednost : float = 100.0 maxVrednost : float = -100.0
+ocitajTemperaturu() : void +srednjaVrednost() : float

Metode klase za pristup obeležjima čiji je kardinalitet *

- `Collections.unmodifiableCollection`
- `Collections.unmodifiableList`
- `Collections.unmodifiableSet`
- `Collections.unmodifiableMap`
- `Collections.unmodifiableSortedMap`
- `Collections.unmodifiableSortedSet`

Klase i obeležja – kako ih otkriti?

- Kandidati za klase se u rečenicama prepoznaju kao imenice
 - departman, student, predmet, osoba, test
- Međutim, obeležja klasa su takođe imenice
 - ime, prezime, datum rođenja, naziv predmeta
- Ugrubo, u pitanju je klasa, a ne obeležje, ako pojam koji razmatramo ima svoja obeležja koja su bitna za sistem koji se modeluje

Metode klasa

- Nazivi metoda klasa i njihovih parametara treba da se pišu po istoj konvenciji kao i nazivi obeležja klasa (Camel Case konvencija)
- Primer davanja imena metodama:
- `ocitajVrednost`, `stampamPlatnihListica`, `unosPodataka`

Format za specifikaciju metoda i parametara

- Specifikacija metode
vidljivost naziv (lista-parametara): tip-povratne-vrednosti {dodatna-podešavanja}
- Specifikacija parametra
smer naziv-parametra: tip kardinalitet = podrazumevana-vrednost
- Smer parametra može biti:
in, out, inout

+stampaPozajmica(odDatuma: date, doDatuma: date): void

+suma(): int

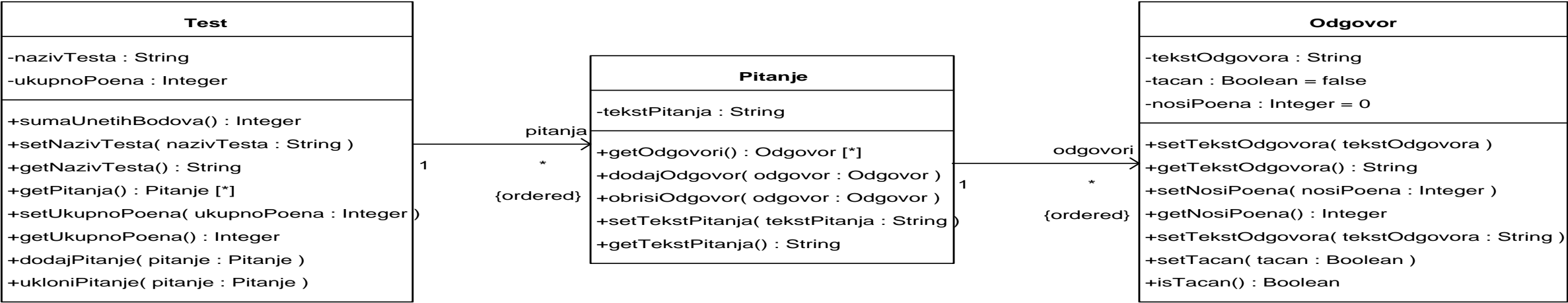
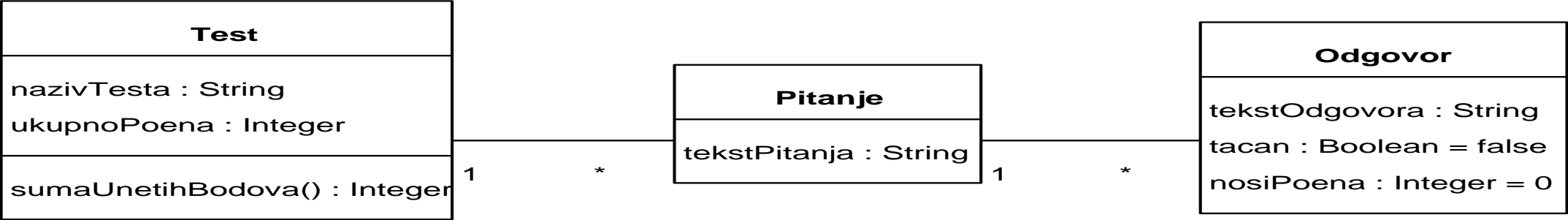
+statistika(out min: float, out max: float, out srednjaVrednost: float): void

+dodajVrednosti(vrednosti int[0..10]): void

Pravila dobre prakse

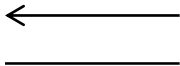
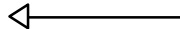
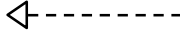
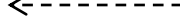
- Metodu staviti u onu klasu koja ima obeležja nad kojima metoda radi
- Ako metoda ima povratnu vrednost („funkcija“), ne treba da menja stanje sistema (tj. vrednosti obeležja svoje ili neke druge klase).
- Ako metoda nema povratnu vrednost (**void** metoda, „procedura“), očekuje se da može menjati stanje sistema.

Rezime



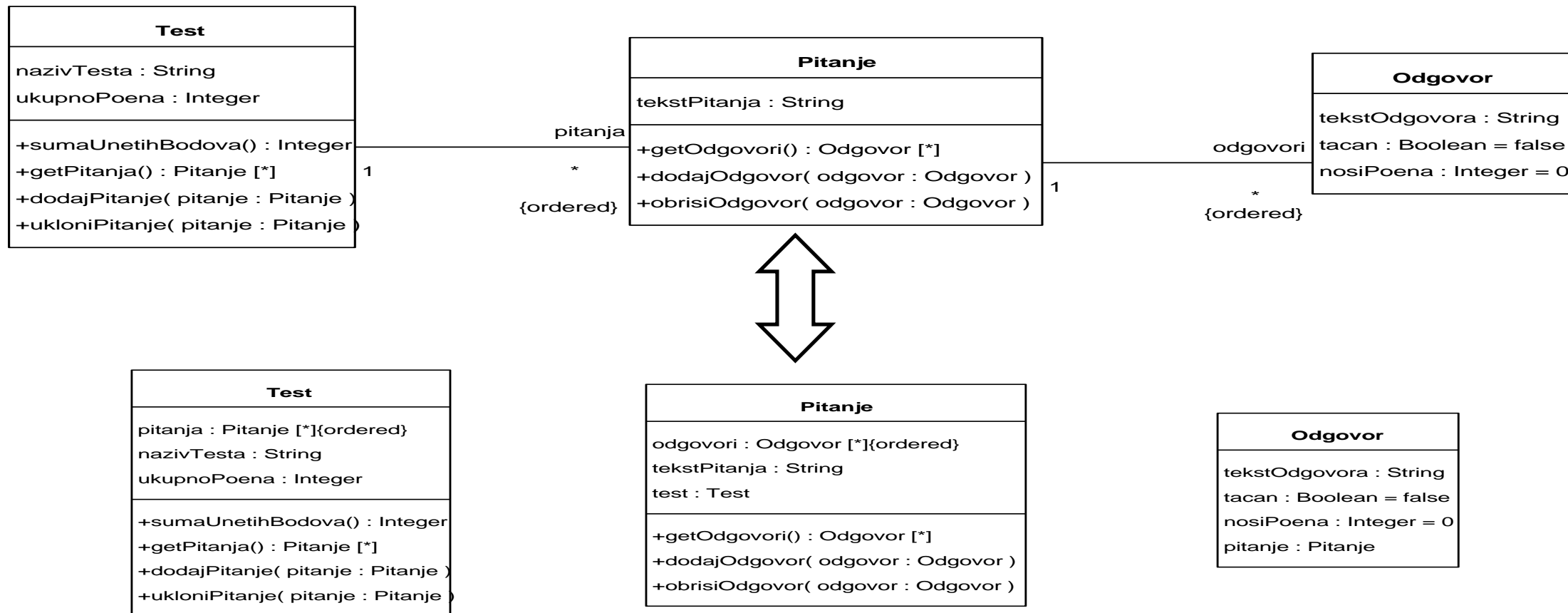
Veze u dijagramima klasa

- Specificiraju saradnju klasa

Vrsta veze	Ilustracija
Asocijacija	
Generalizacija	
Implementacija interfejsa	
Veza zavisnosti	

Asocijacija

Specificira da su instance jedne klase povezane sa instancama druge klase



- Ako postoji asocijacija, obeležja čiji je tip klasa na suprotnom kraju se ne navode!
- Osobine tih obeležja se specificiraju na osnovu osobina suprotnih krajeva asocijacije
- Krajevi asocijacije se zovu uloge

Primer implementacije na programskom jeziku java

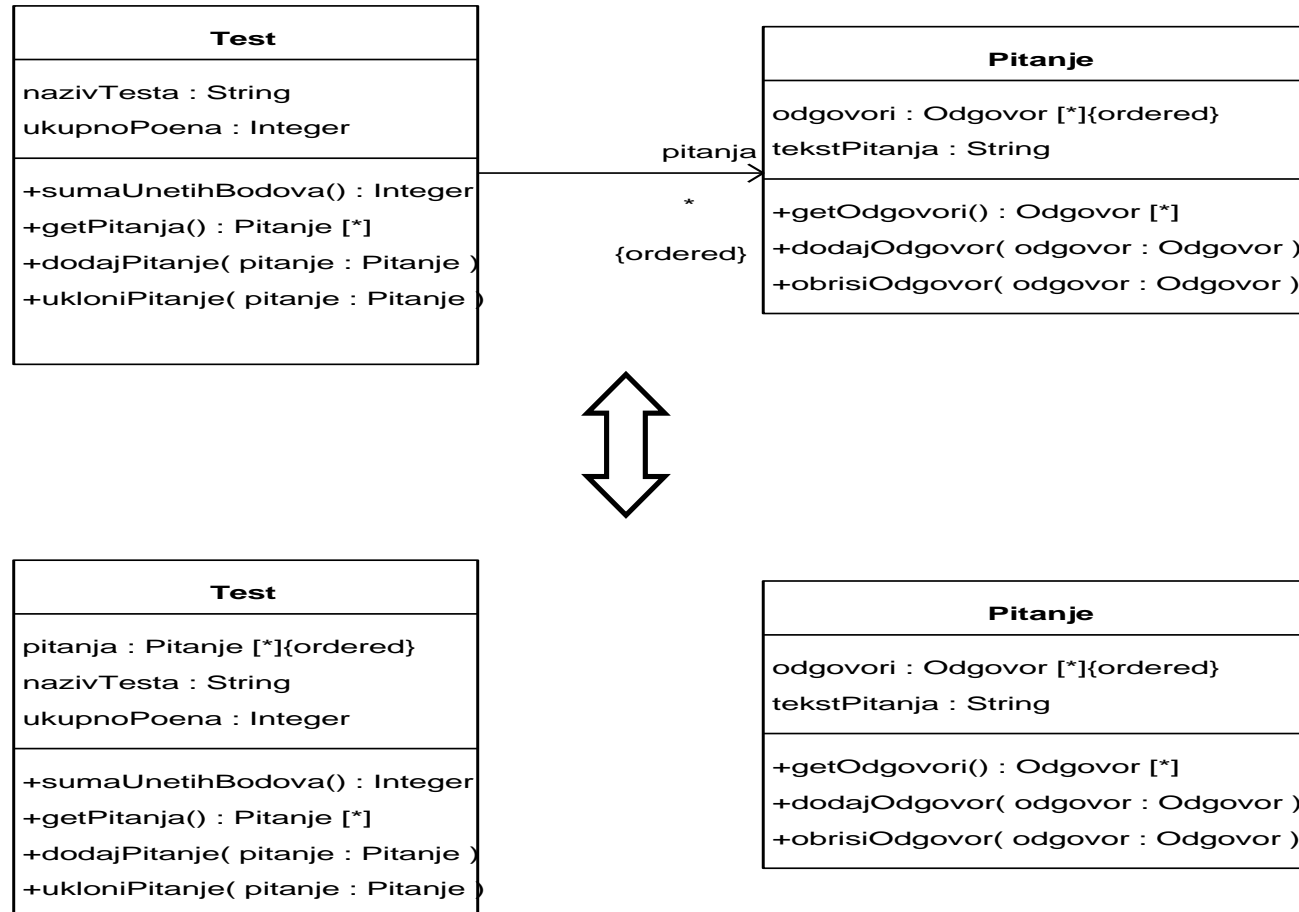
```
public class Test {
    private String nazivTesta;
    private int ukupnoPoena;
    private java.util.List<Pitanje> pitanja = new ArrayList<Pitanje>;
    ...
}

public class Pitanje {
    private String tekstPitanja;
    private java.util.List<Odgovor> odgovori = new ArrayList<Odgovor>;
    private Test test;
    ...
}

public class Odgovor {
    private String tekstOdgovora;
    private Boolean tacan = false;
    private int nosiPoena = 0;
    private Pitanje pitanje;
    ...
}
```

Napomena: Izabrana je kolekcija ArrayList zato što je u modelu naznačeno da je redosled elemenata bitan - opcija {ordered}

Navigabilna asocijacija u jednom smeru



- Test “vidi” pitanje
- Pitanje “ne vidi” test

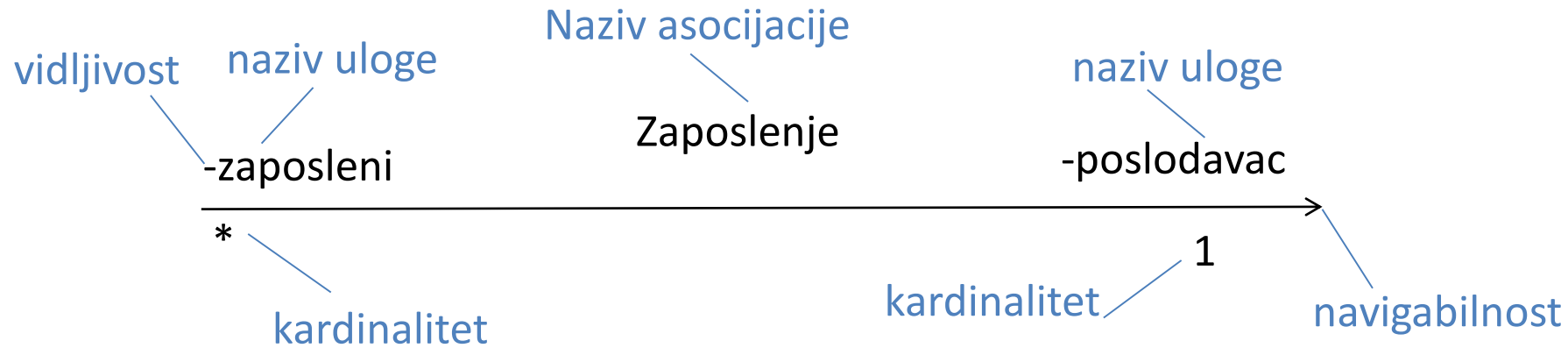
Primer implementacije na programskom jeziku java

```
public class Test {  
    private String nazivTesta;  
    private int ukupnoPoena;  
    private java.util.List<Pitanje> pitanja = new ArrayList<Pitanje>;  
    ...  
}
```

```
public class Pitanje {  
    private String tekstPitanja;  
    private java.util.List<Odgovor> odgovori = new ArrayList<Odgovor>;  
    ...  
}
```

- Test “vidi” pitanje
- Pitanje “ne vidi” test

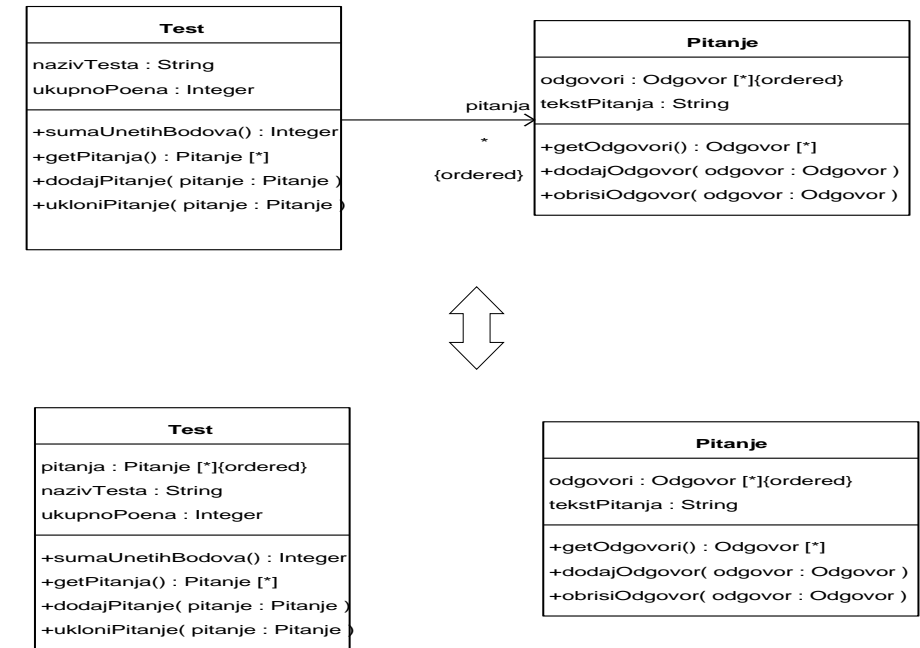
Osobine asocijacije



- Sve osobine koje može da ima obeležje se mogu specificirati i pri modelovanju krajeva asocijacije
- Ako se ništa ne navede, primenjuju se konvencije

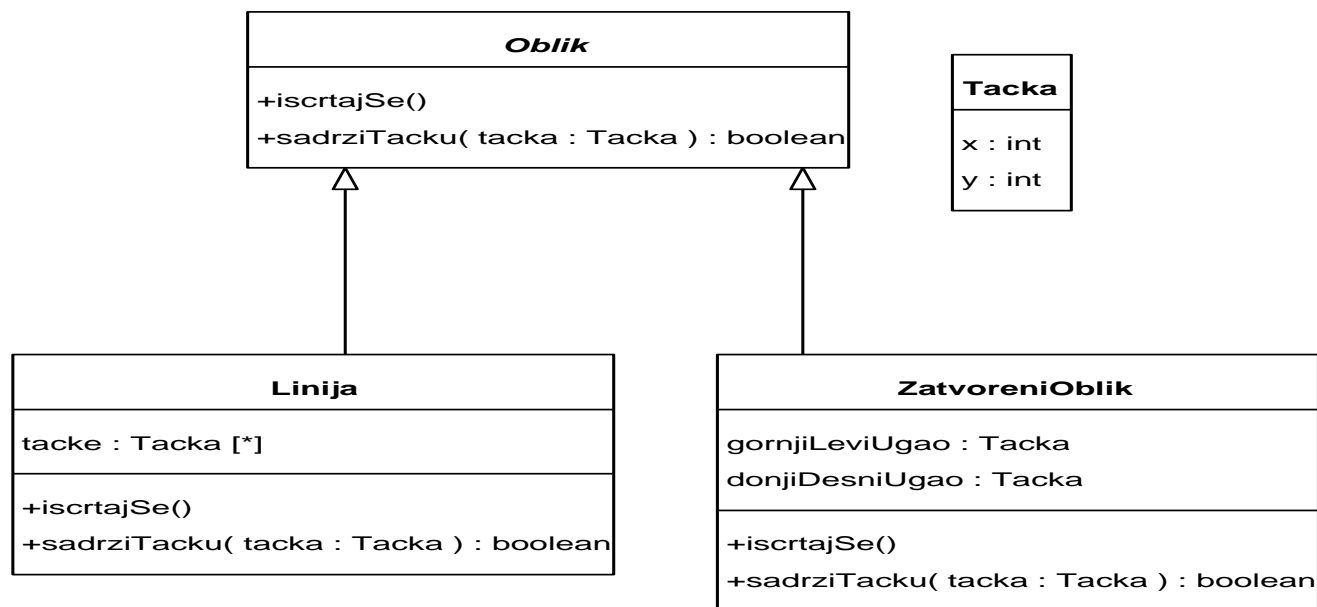
Asocijacije ili direktan unos obeležja čiji je tip klasa? 1/2

- Po UML sintaksi, obe verzije su korektne
- Asocijacije bolje pokazuju strukturu sistema i saradnju klasa i treba ih koristiti!
- Modeli bez asocijacija su teži za čitanje i razumevanje



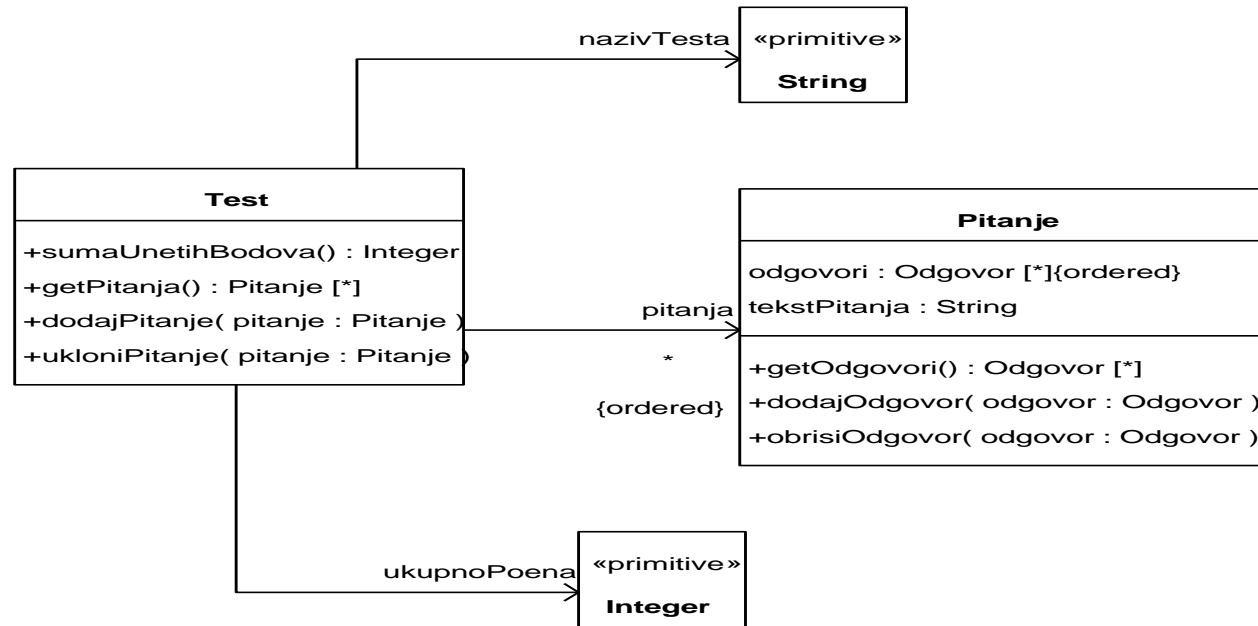
Asocijacije ili direktan unos obeležja čiji je tip klasa? 2/2

- Izuzetak: male pomoćne klase koje se koriste na puno mesta
- U ovakvim slučajevima, povlačenje asocijacija bi dovelo do nečitkog modela



Ovo svakako nemojte praktikovati!

...iako je ispravno po UML sintaksi



Pravila „lepog ponašanja“ pri modelovanju

- Obeležja koja su podrazumevanih ili nabrojanih tipova navode se direktno, u okviru klase
- Obeležja čiji su tipovi klase se navode korišćenjem asocijacija, radi lakšeg uvida u strukturu
 - sem ako su u pitanju obeležja čiji je tip mala, pomoćna klasa koja se često koristi

Kako otkriti asocijacije?

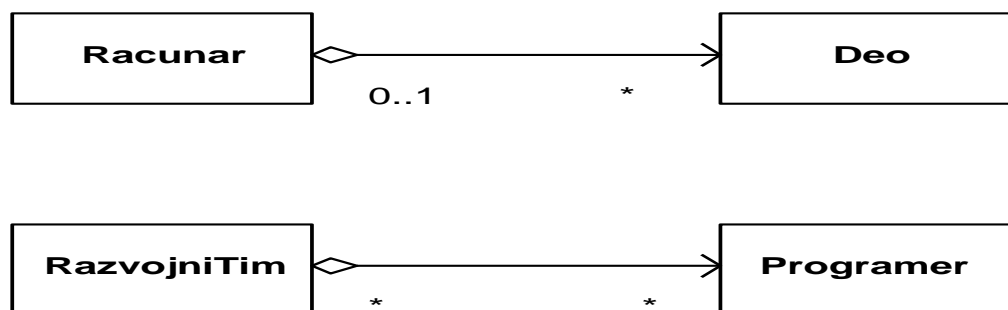
1. Posmatramo glagole u rečenicama
 - Fakultet **se sastoji** od departmana.
 - Student **bira** predmete.
 - Član biblioteke **iznajmljuje** primerke knjige.
 - Test **se sastoji** od pitanja.
2. Imenice između kojih se nalazi glagol treba da budu klase, a ne klasa i njeno obeležje koje je predefinisano tipa. Ovde nećemo povlačiti asocijacije:
 - Osoba **ima** ime, prezime i datum rođenja.
3. Odrediti kardinalitet oba kraja asocijacije

Određivanje kardinaliteta

- Bitno je razmotriti donju i gornju granicu oba kraja asocijacije
 - Koliko predmeta može da bira student?
 - Da li jedan predmet može da bude izabran od strane više studenata?
 - Koliko minimalno predmeta student mora da izabere?
 - Koliko minimalno studenata mora da sluša neki predmet?

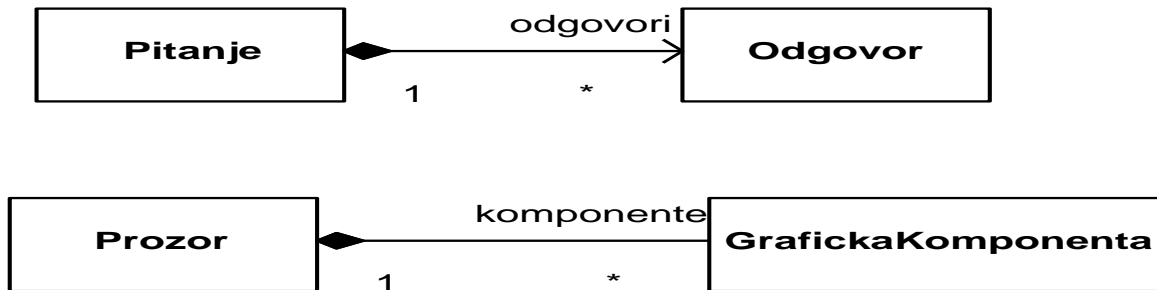
Agregacija

- Agregacija je vrsta asocijacije kojom se modeluje odnos „celina-deo“ između dve klase
- Romb se stavlja na kraj agregacije gde se nalazi klasa koja predstavlja celinu



Kompozicija

- Kompozicija je vrsta agregacije kod koje klase nisu „ravnopravne“
 - delovi ne mogu da postoje bez celine, a njihov životni ciklus je čvrsto povezan
- Klasa koja modeluje celinu kreira svoje delove i njihov je jedini vlasnik
- Kada se celina briše, brišu se i svi njeni delovi

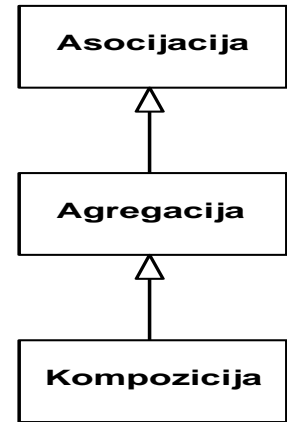


Agregacija i kompozicija – greške

- Kada se agregacija ili kompozicija koriste u situacijama koje nisu „celina-deo“
- Ovde bi bilo **pogrešno** koristiti agregaciju ili kompoziciju:
 - Departman ima studente.
 - Preduzeće ima zaposlene i robu.
- Ovde se mogu koristiti agregacija ili kompozicija:
 - Departman se sastoji od katedri (ima katedre).
 - Preduzeće se sastoji od poslovnica (ima poslovnice).

Preporuka

- Agregacija je vrsta (specijalizacija) asocijacije
- Kompozicija je vrsta (specijalizacija) agregacije
- Ako niste sigurni koju vrstu asocijacije treba da upotrebite, bolje je koristiti osnovnu asocijaciju, nego pogrešno upotrebiti njene specijalizacije!

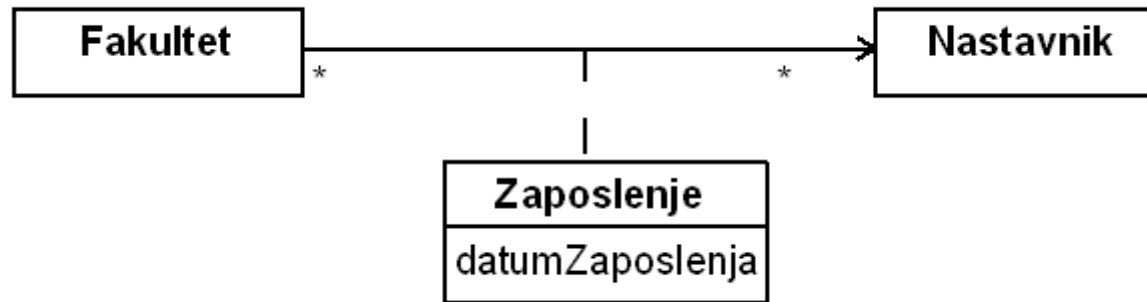


Agregacija ili kompozicija?

- Ako smo sigurni da je u pitanju odnos „celina-deo“, tada:
 - Da li deo može postojati bez celine?
 - da – agregacija
 - ne - kompozicija
 - Šta se dešava sa delovima kada se briše celina?
 - brišu se i delovi – kompozicija
 - delovi nastavljaju da postoje – agregacija

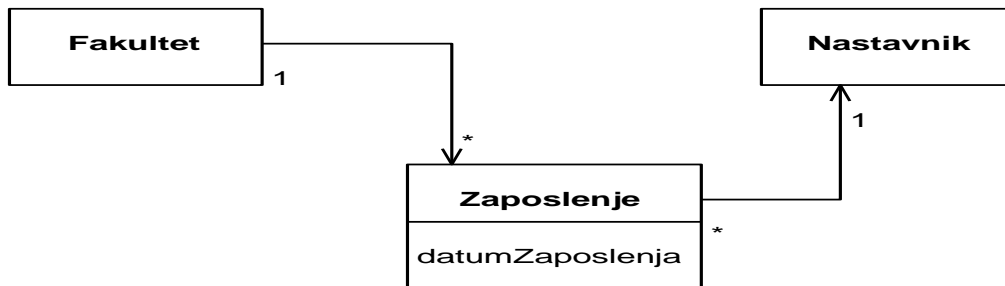
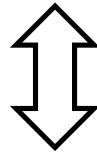
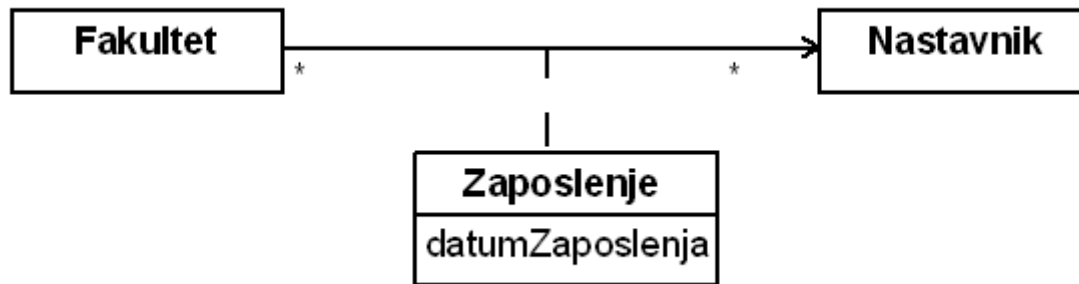
Asocijativne klase

- Na fakultetu su zaposleni nastavnici. Jedan nastavnik može da radi na više fakulteta. Bitno je znati datum zaposlenja, za svakog nastavnika i svaki fakultet na kojem radi.*



Modelovanje bez asocijativnih klasa

1/2



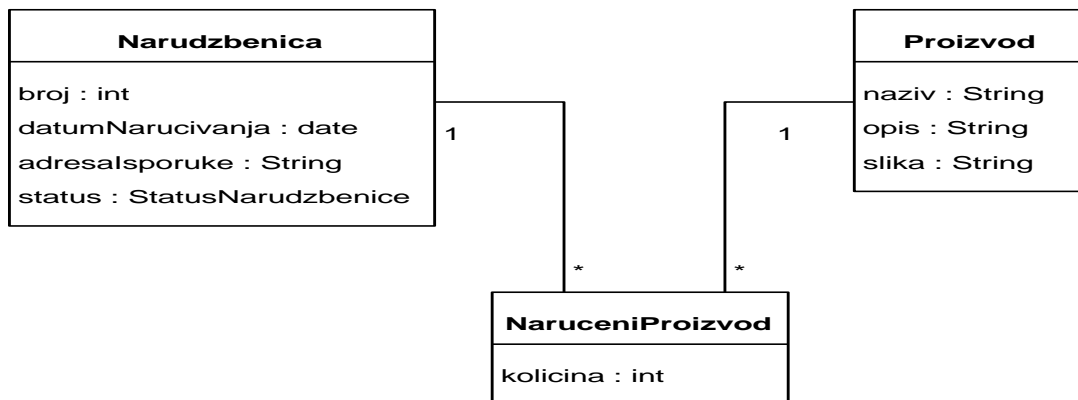
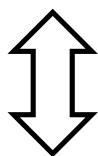
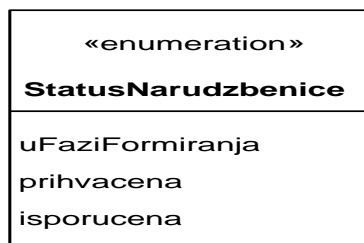
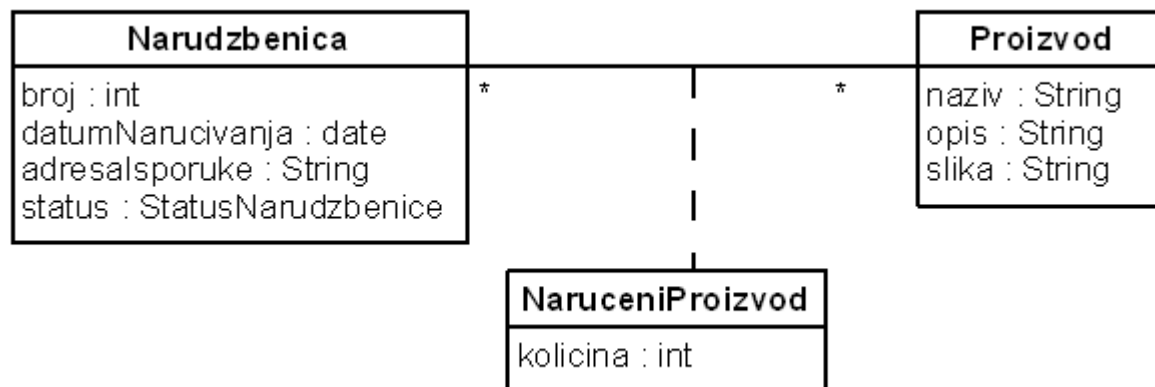
```
public class Fakultet {
    private String nazivFakulteta;
    private java.util.Set<Zaposlenje> zaposlenje =
        new java.util.HashSet<Zaposlenje>();
    ...
}
```

```
public class Zaposlenje {
    private Date datumZaposlenja;
    private Nastavnik nastavnik;
    ...
}
```

```
public class Nastavnik {
    ...
}
```


Modelovanje bez asocijativnih klasa

2/2



```
public enum StatusNarudzbenice {
    U_FAZI_FORMIRANJA,
    PRIHVACENA,
    ISPORUCENA;
}
```

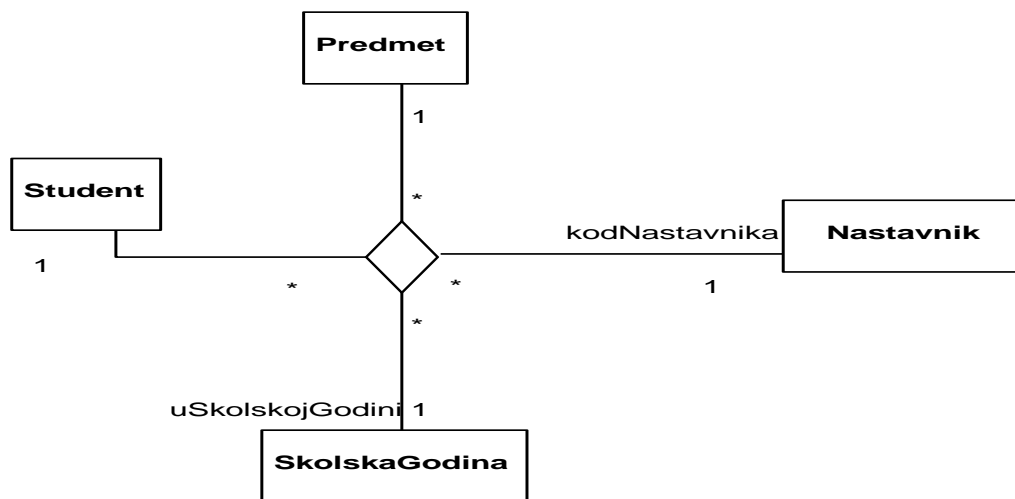
```
public class Narudzbenica {
    private int broj;
    private Date datumNarucivanja;
    private String adresaIsporuke;
    private StatusNarudzbenice status;
    private java.util.Set<NaruceniProizvod> naruceniProizvod = new
    java.util.HashSet<NaruceniProizvod>();
    ...
}
```

```
public class NaruceniProizvod {
    private int kolicina;
    private Proizvod proizvod;
    private Narudzbenica narudzbenica;
    ...
}
```

```
public class Proizvod {
    private String naziv;
    private String opis;
    private String slika;
    private java.util.Set<NaruceniProizvod> naruceniProizvod = new
    java.util.HashSet<NaruceniProizvod>();
    ...
}
```

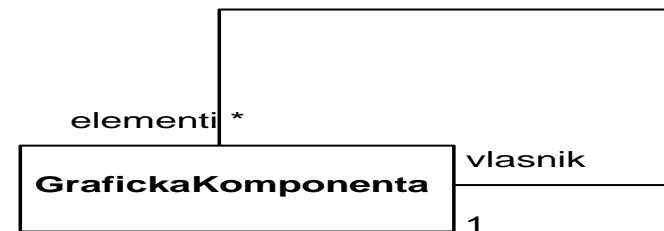
N-arna asocijacija

- Postoji od verzije UML 2.0



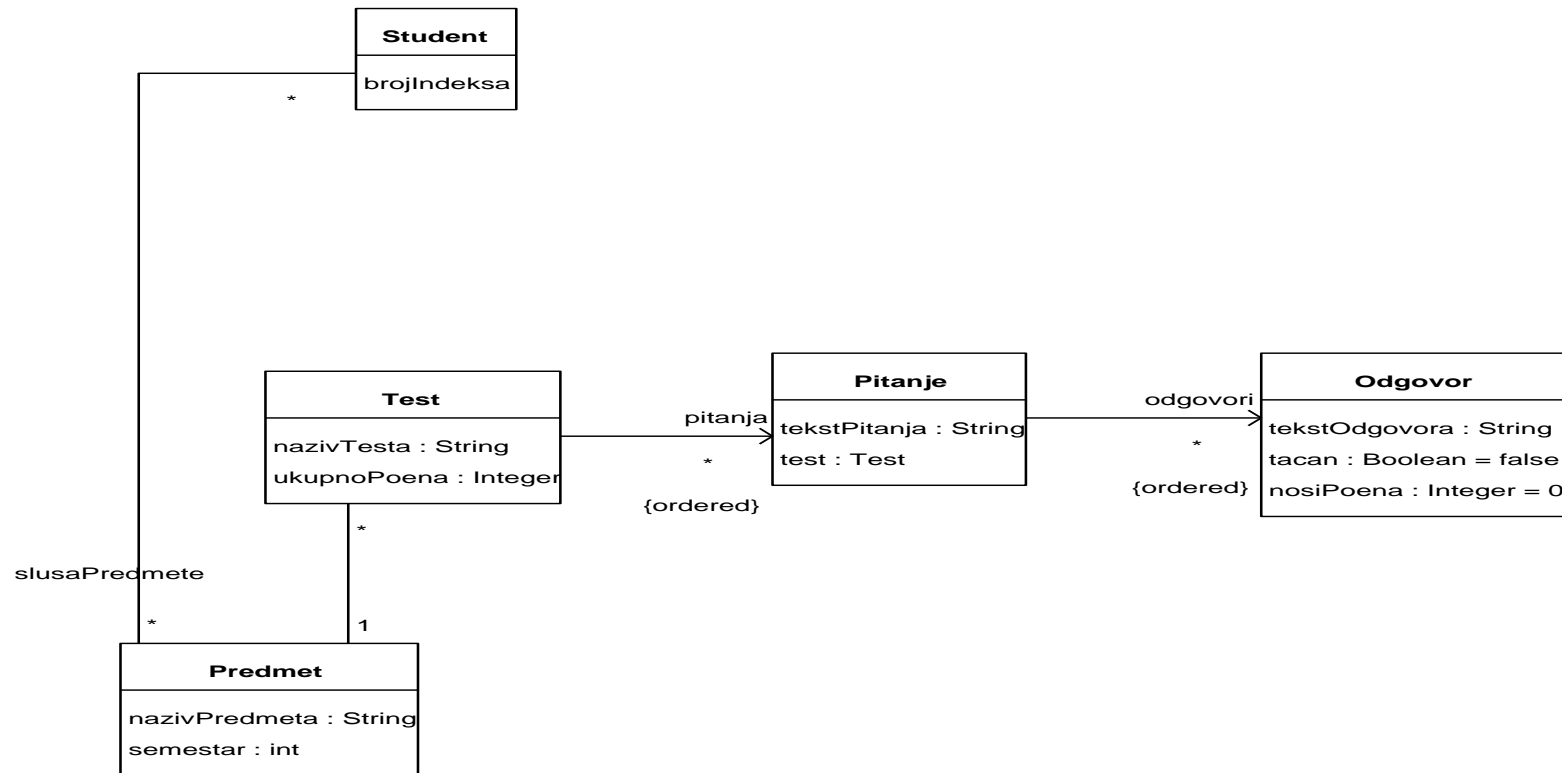
```
// Klasa koja nastaje kao posledica n-arne asocijacije:  
public class StudentSlusaPredmet {  
    private Student student;  
    private Predmet predmet;  
    private SkolskaGodina uSkolskojGodini;  
    private Nastavnik kodNastavnika;  
    ...  
}
```

Primeri korišćenja asocijacija

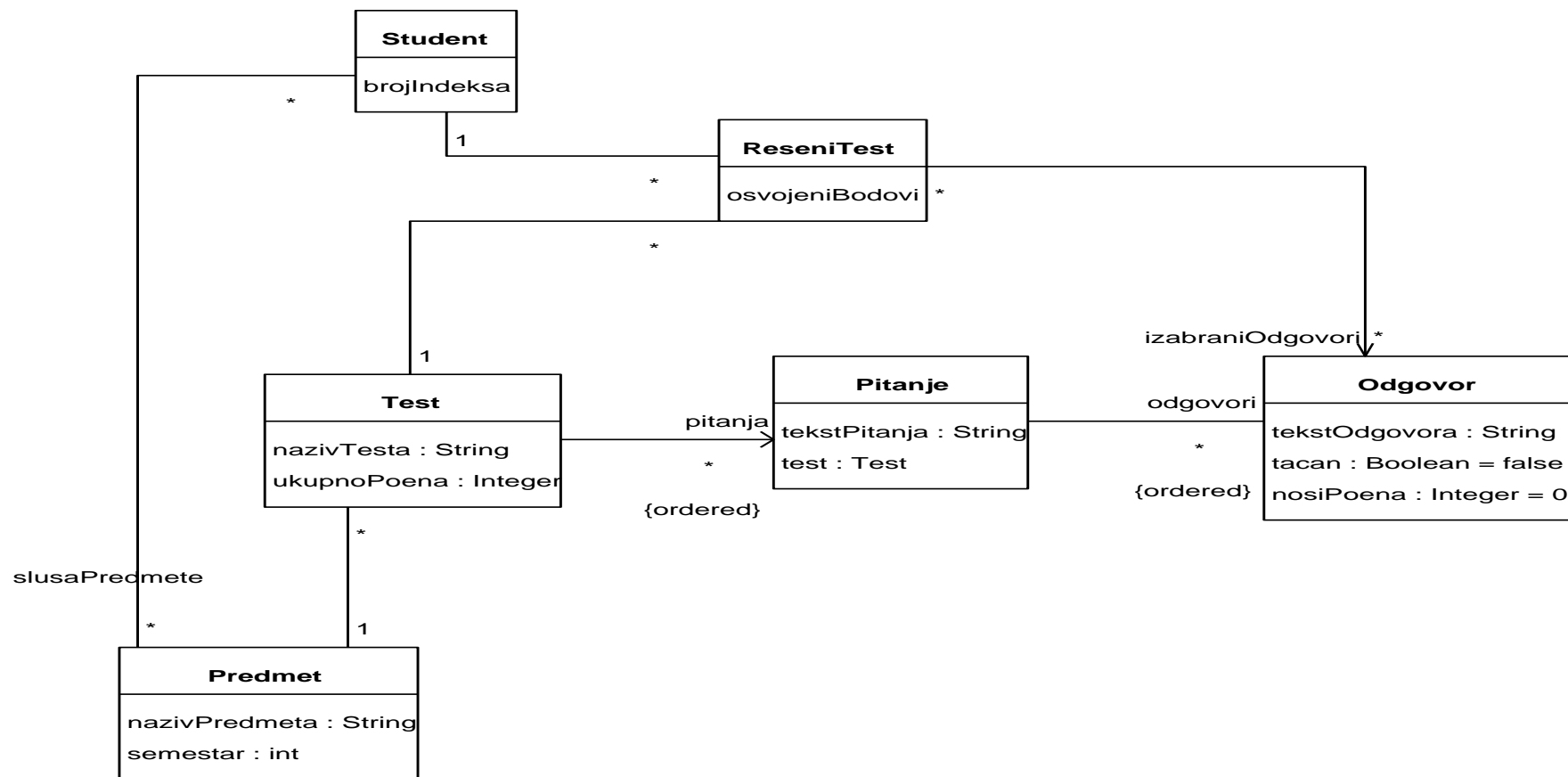


Vežba

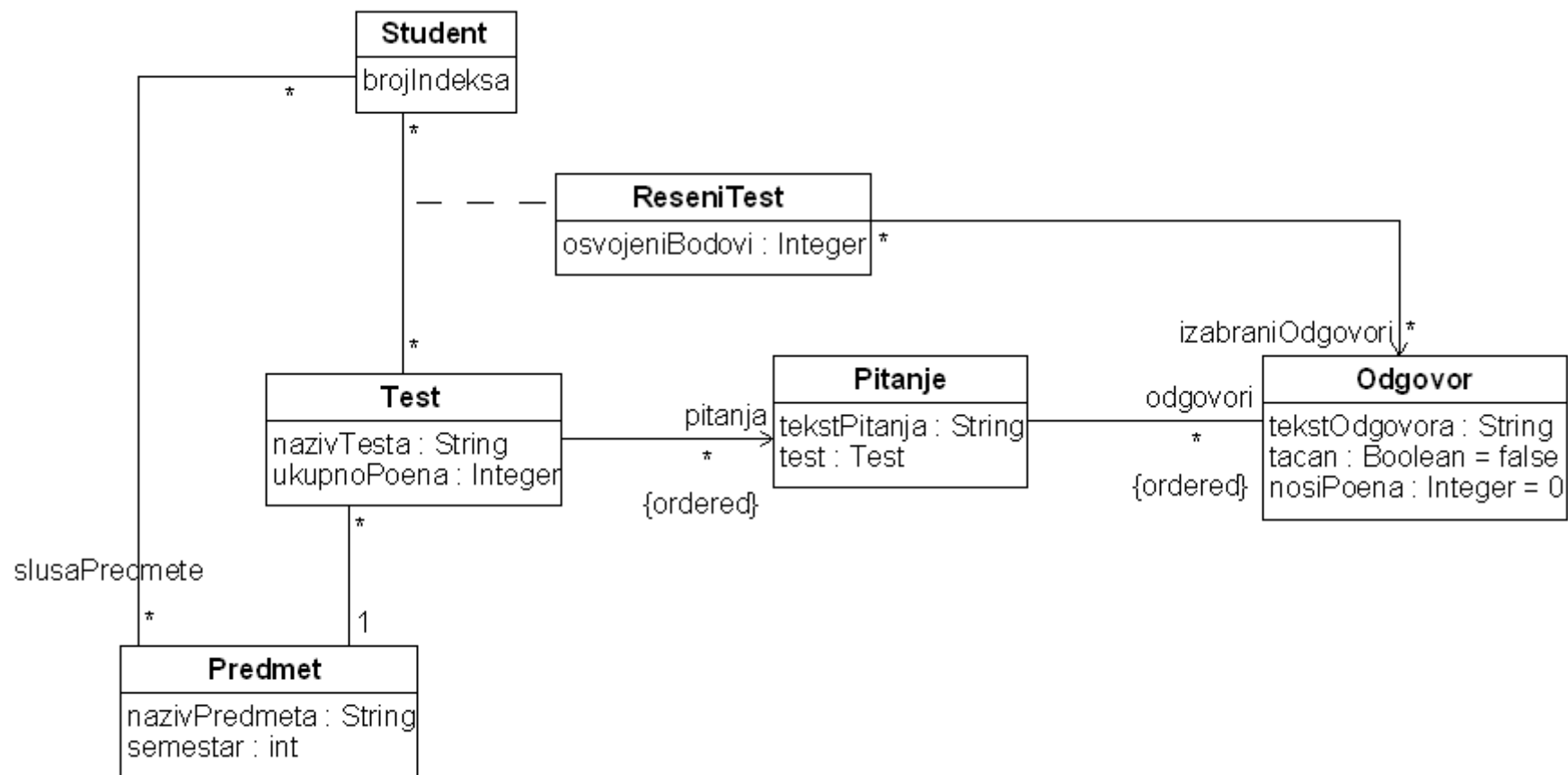
- Gde staviti bodove koje je student osvojio rešavanjem testa?
- Kako znati koje je odgovore dao?



Rešenje 1



Rešenje 2



Literatura

1. James Rumbaugh, Ivar Jacobson, Grady Booch, The Unified Modeling Language Reference Manual, Second Edition, Addison-Wesley, 2004
2. Scott W. Ambler, The Object Primer: Agile Model-Driven Development with UML 2.0, Cambridge University Press, 2004
3. M. Fowler, UML Distilled - A Brief Guide to the Standard Object Modeling Language, Third Edition, Addison Wesley, Boston, 2004.