



TEHNIČKO VELEUČILIŠTE U ZAGREBU
POLYTECHNICUM ZAGRABIENSE

Objektno orijentirani razvoj programa

ISVU: 130938/19881

Dr. sc. Danko Ivošević, dipl. ing.
Predavač

Akademska godina 2021./2022.
Ljetni semestar

OBJEKTNO ORIJENTIRANI RAZVOJ PROGRAMA

UML DIJAGRAMI RAZREDA

Napomena

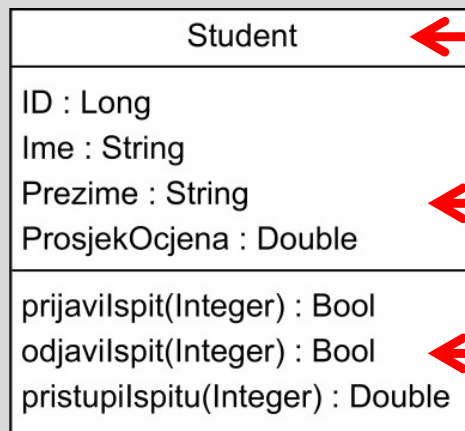
- Prikazani primjeri su rađeni u alatu ArgoUML i dijelom u alatu Enterprise Architect 15
- Svrha primjera je prikaz koncepata neovisno o upotrijebljenom alatu

Karakteristike dijagrama razreda (1)

- **Dijagram razreda** (engl. *class diagram*) opisuje vrste objekata unutar nekog sustava i njihove međusobne odnose.
 - opisuje razrede (klase) i njihove međusobne veze.
- Strukturni UML dijagram, opisuje statičke odnose.
- **Dva osnovna tipa odnosa između razreda:**
 1. Pridruživanje (veza ili asocijacija)
 2. Podtip
- **Pridruživanje dijelimo na:**
 - Jednosmjerno, dvosmjerno, refleksivno i agregacija.
 - Kompozicija je podvrsta agregacije.

Karakteristike dijagrama razreda (2)

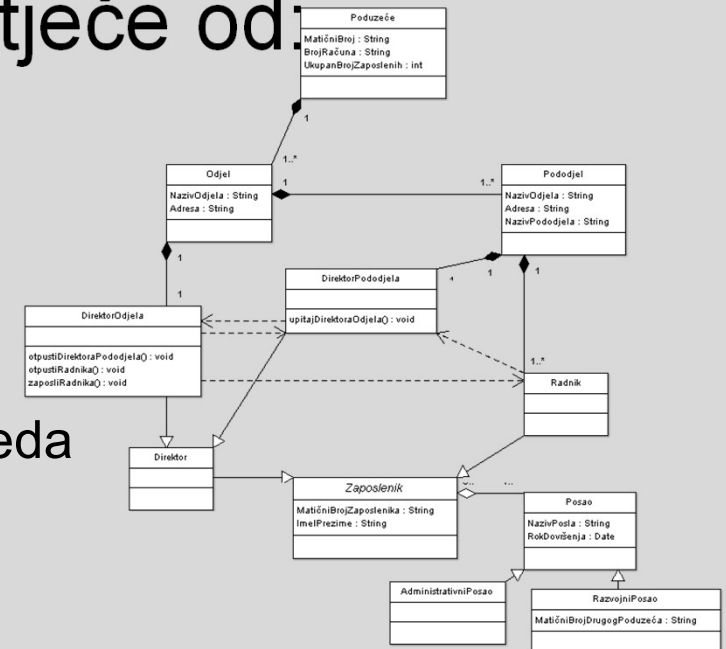
- Najčešći termin koji koristimo potječe od:
 - **Class diagram**
- Hrvatski termini:
 - **Dijagram razreda ili dijagram klasa**



← Jedinestveni naziv razreda

← Skup atributa

← Skup operacija



- Prikazuju razrede, attribute i operacije razreda, njihova svojstva i ograničenja, sučelja, pridruživanja, vlastite tipove podataka, enumeracije, pakete i komentare.

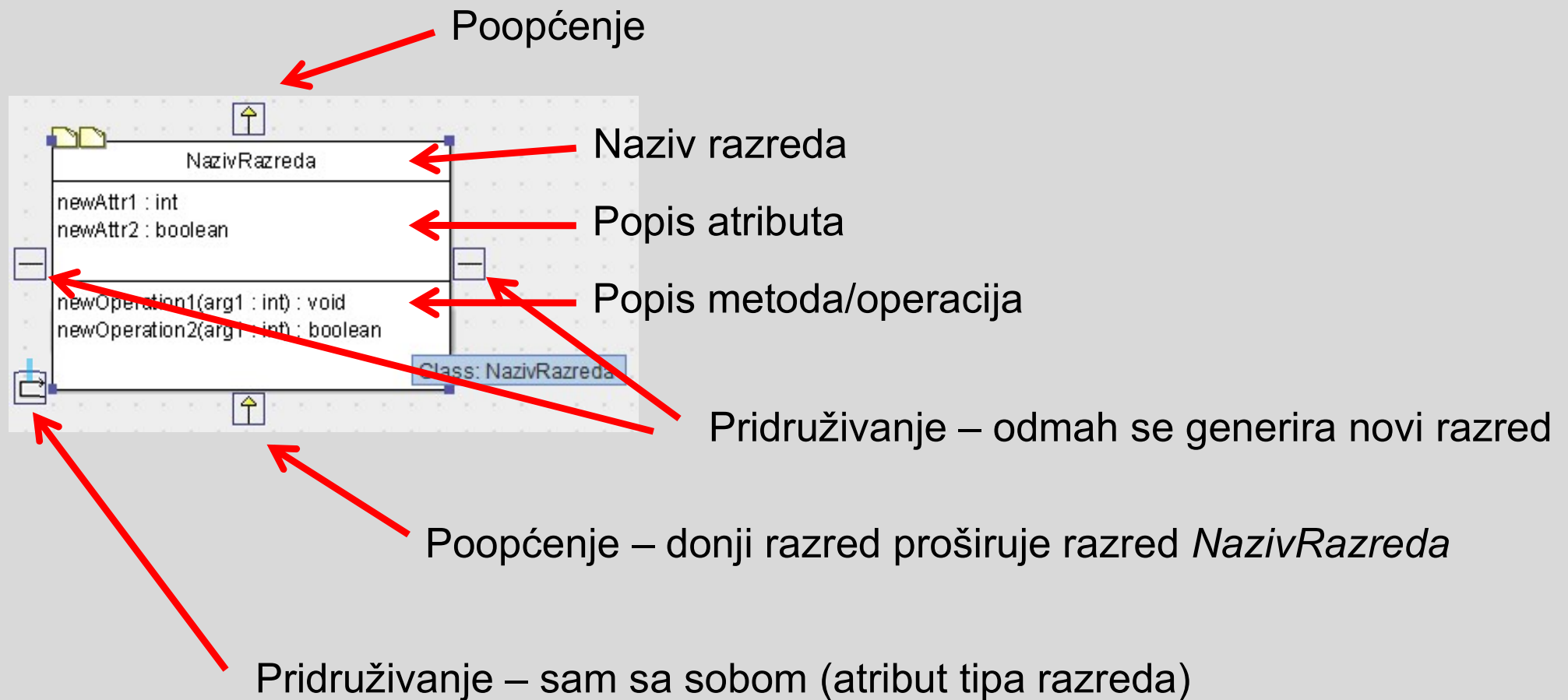
Načini uporabe (1)

- Općenito, dijagrami razreda mogu se koristiti na **tri različita načina** (pogleda, perspektive ili razine):
 - 1. Koncept**
 - Razmatranje neke domene. Dijagram ne mora biti povezan s programskom potporom. Neovisan o programskom jeziku.
 - 2. Specifikacija**
 - Sučelja sustava.
 - 3. Implementacija**
 - Izrada programske potpore. U primjeni najčešća perspektiva.
- Vrijedi za sve UML dijagrame, ali najočitije je s dijagramima razreda

Načini uporabe (2)

- Razine uporabe nisu formalni dio UML specifikacije, ali su jako korisne u postupku izrade modela.
- Granice između razina nisu strogo definirane. Nema velikih razlika između koncepta i specifikacije. Između specifikacije i implementacije postoji veća razlika.
- Možemo se poslužiti stereotipovima <<>> u označivanju razreda, npr. <<implementation>> ili <<type>> za implementaciju i koncept.

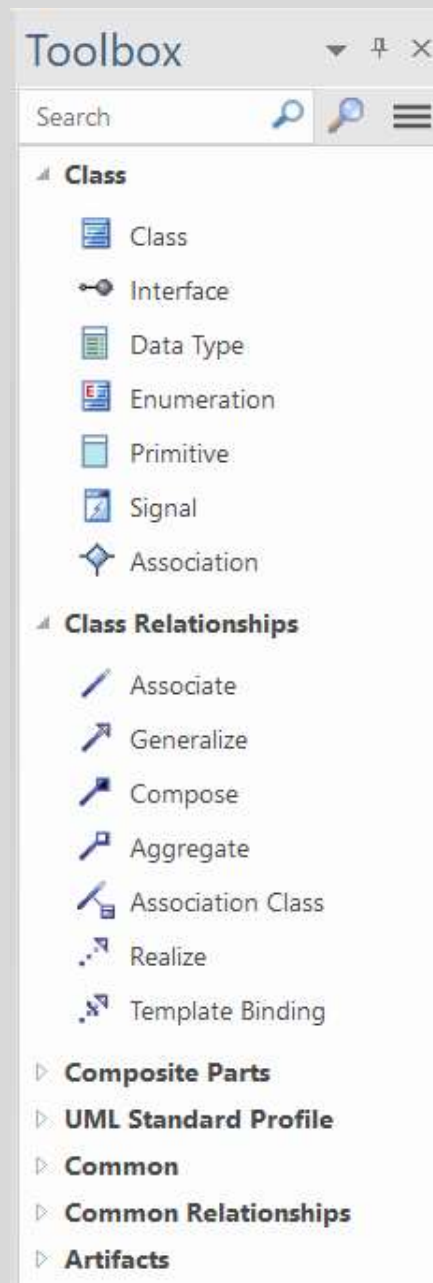
Simbol razreda u uređivaču ArgoUML



Napomena: Izvorni kôd u ArgoUML dobiven je iz kartice Source Code. Kôd je koristan za ilustraciju dijagrama, ali potrebno ga je proširiti i provjeriti njegovu ispravnost u nekom od jezičnih procesora.

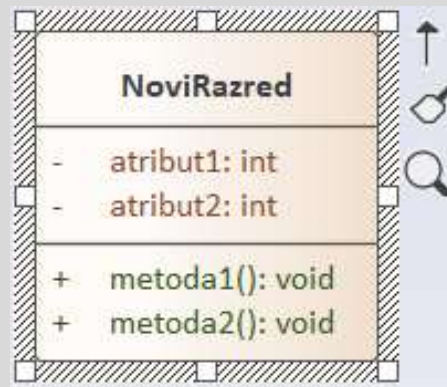
Uporaba razreda u alatu EA 15

- Razredi
- Odnosi



Uporaba razreda u alatu EA 15

- Razred



- Svojstva

Features					
Attributes Operations Receptions Parts / Properties Interaction Points					
Name	Type	Scope	Stereotype	Alias	Initial Value
atribut1	int	Private			
atribut2	int	Private			
New Attribute...					

Razred

- **Razred ili klasa** (engl. *class*) je osnovni tvorbeni element UML class dijagrama.

1. Objekt

- Predstavlja entitet iz stvarnog svijeta ili neki koncept.
- Apstrakcija nečega što ima dobro definirane granice i smisao u sustavu.

2. Razred

- Opis skupine objekata sa sličnim svojstvima.
- Svaki objekt je pojedinac (instanca) jednog razreda.

Atributi

- **Atributi** (engl. *attributes*) razreda imaju sljedeća svojstva:
 - **Razina vidljivosti** (engl. *visibility*)
 - **Naziv** (engl. *name*)
 - **Vrsta ili tip** (engl. *type*)
 - **Početnu vrijednost** (engl. *initial value*)
- Dodatno:
 - **Promjenjivost** (engl. *changeability*)
 - **Modifikator** (engl. *modifier*)

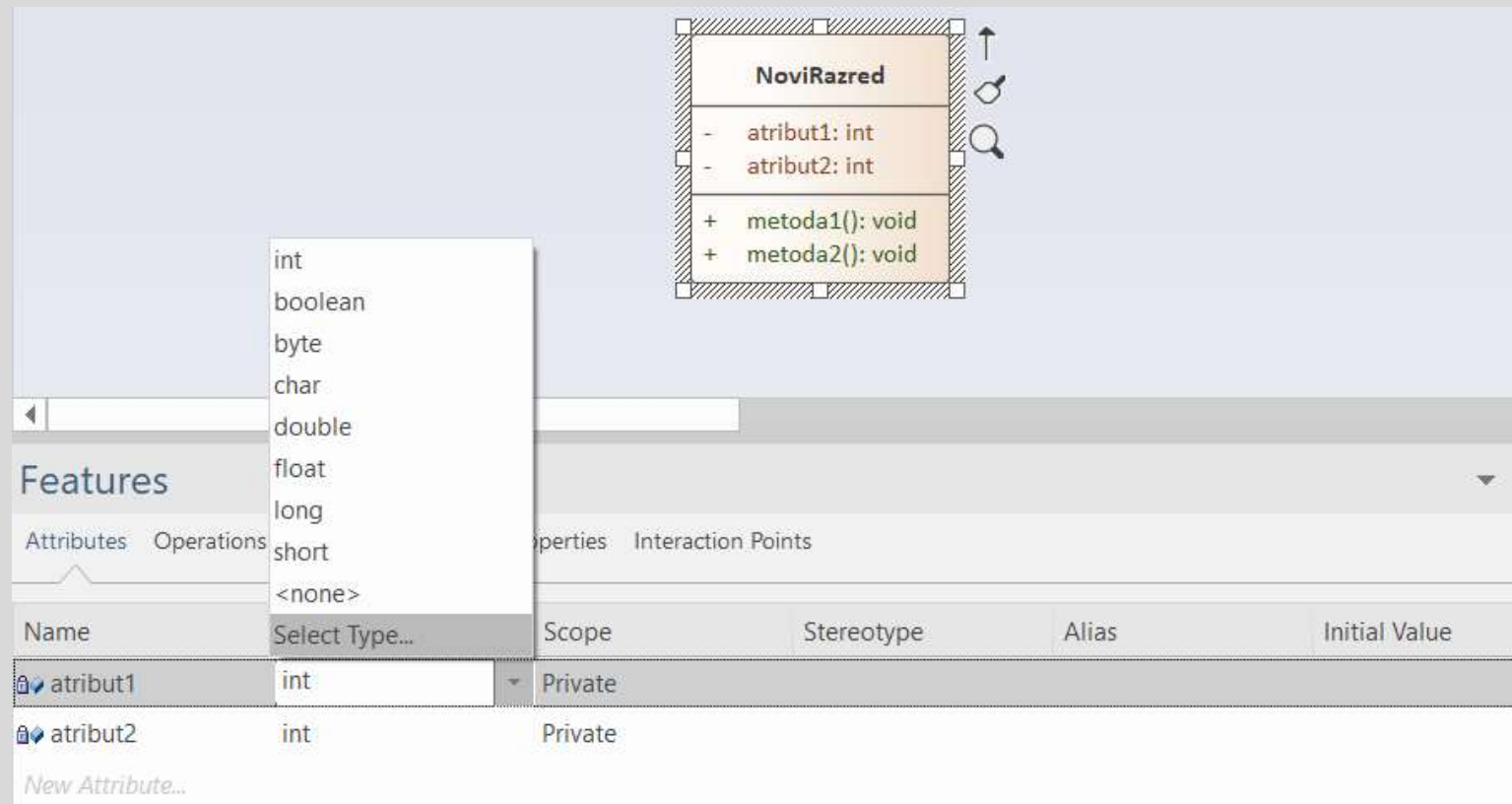
Stupanj vidljivosti atributa

- Moguće su **četiri vrijednosti**:
 - **Public** (simbol: **+**)
 - Atribut je dostupan svim razredima i paketima.
 - **Private** (simbol: **-**)
 - Atribut je dostupan samo unutar istog razreda.
 - **Protected** (simbol: **#**)
 - Atribut je dostupan unutar istog razreda i izvedenih razreda.
 - **Package** (simbol: **~**)
 - Atribut je dostupan svim razredima istog paketa.
- Mogu se primijeniti i na operacije razreda

Vrsta ili tip atributa

- Ugrađeni

- Vlastiti



Promjenjivost atributa (primjer ArgoUML)

- **addOnly**
 - Vrijednost atributa može se samo povećavati.
- **changeable**
 - Vrijednost atributa može se nesmetano mijenjati. Podrazumijevana (*default*) postavka.
- **static**
 - Modifikator, vrijednost atributa ne mijenja se (konstanta) i ne ovisi o životu objekta.
 - Ključne riječi u programskim jezicima *static* ili *const*.

Promjenjivost atributa (primjer ArgoUML)

- **frozen**
 - Vrijednost atributa (ili asocijacije) može se promijeniti samo jednom tijekom života (engl. *lifetime*) pripadajućeg objekta.
- **read-only**
 - Vrijednost atributa ne može se mijenjati izvan objekta kojemu pripada.
 - Nije isto što i *frozen*.

Operacije

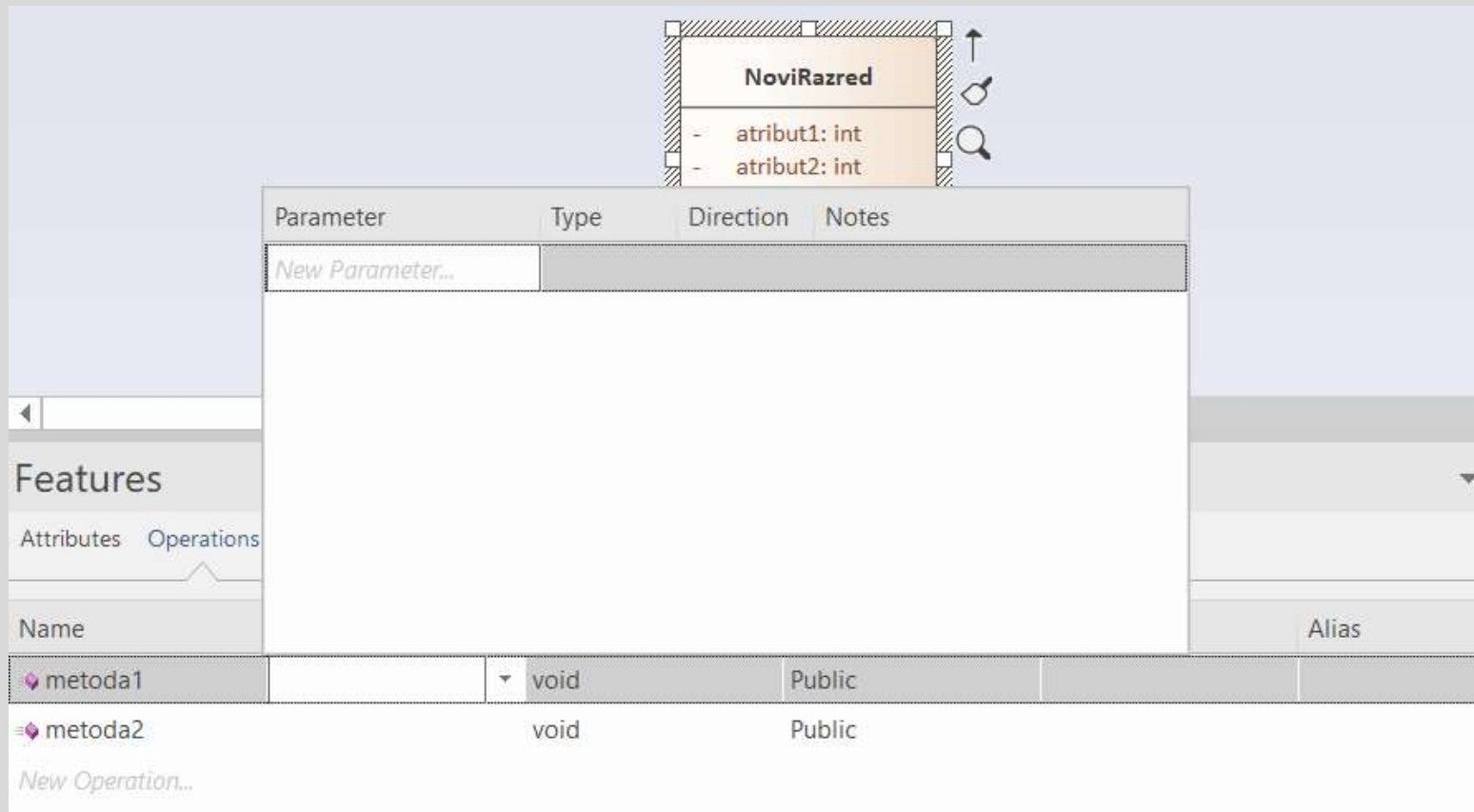
- **Operacije** (engl. *operations*) su procesi koje razred može izvršiti.
 - Drugim riječima, to su vlastite metode i funkcije razreda.
- Za njih možemo odrediti (primjer ArgoUML):
 - **Vidljivost** (*public, package, protected, private*)
 - Modifikatore (***static, abstract***, *leaf, root, query*)
 - Istodobnost (*sequential, guarded, concurrent*)
 - **Parametre ili argumente**

Operacije

- **Operacije** (engl. *operations*) su procesi koje razred može izvršiti.
 - Drugim riječima, to su vlastite metode i funkcije razreda.
- Za njih možemo odrediti (primjer ArgoUML):
 - **Vidljivost** (*public, package, protected, private*)
 - Modifikatore (*static, abstract*)
 - **Parametre ili argumente**

Operacije

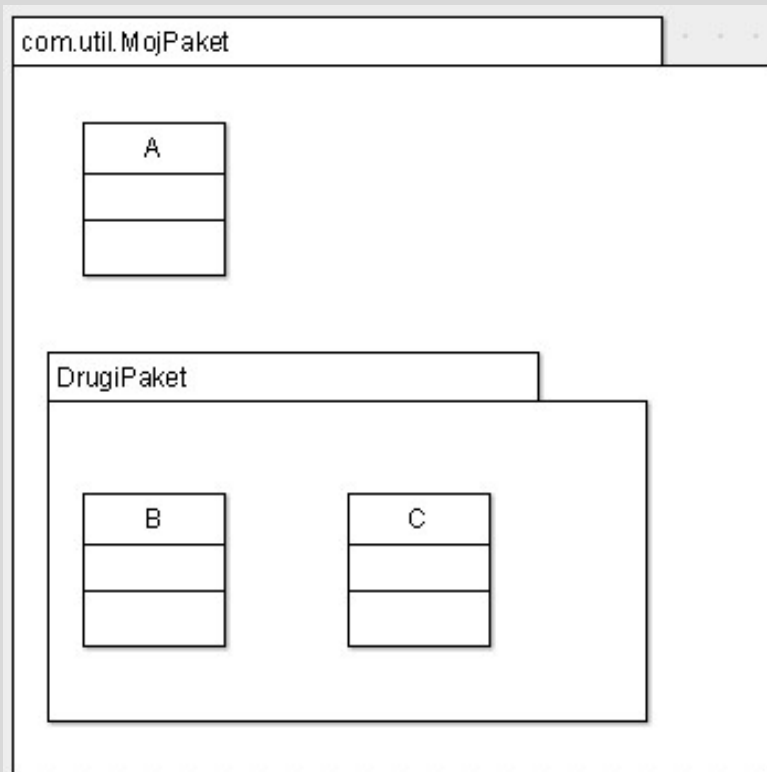
- Primjer EA 15:



Paket

- **UML paket** (engl. *package*) je skup različitih objekata.
- Svrha paketa je omogućiti hijerarhijsku organizaciju elemenata u UML dijagramu.
- Mogu sadržavati druge pakete, objekte, razrede, komponente, UC-ove, itd.
- U programskom kodu interpretira se kao namespace u C++ i C#, package u Javi, ...

Primjer paketa



Java

```
package com.util.MojPaket;
```

```
public class A {}
```

C++

```
namespace com {
namespace util {
namespace MojPaket {
namespace DrugiPaket {
```

```
class B {};
```

```
} /* End of namespace
```

```
com.util.MojPaket::DrugiPaket */
```

```
} /* End of namespace com.util.MojPaket */
```

```
} /* End of namespace com.util */
```

```
} /* End of namespace com */
```

Java

```
package com.util.MojPaket.Drugipaket;
```

```
public class B {}
```

Pridruživanje


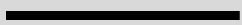
- **Pridruživanje** (engl. *association*) ili **veza** opisuje odnose između pojedinaca (instanci) razreda.

- “Student pristupa ispitu”

Student	pristupa	Ispit
	*	*

- Svaka veza ima dva vrha. Svaki vrh je pridružen (dodiruje) jedan od razreda u vezi.
- Vrh može imati vlastito ime:
 - **Naziv uloge** (engl. *role name*).
 - Vrhovi se još nazivaju **uloge** ili **role** (engl. *association role*).

Smjer pridruživanja

- Ako je vrh neke asocijacije označen strelicom onda je definiran njezin **smjer** (engl. *navigability*)
- Podjela veza po smjeru:
 - Jednosmjerna (unidirekcionalna)
 - Smjer je definiran na samo jednom vrhu. 
 - Dvosmjerna (bidirekcionalna)
 - Smjer je definiran na oba vrha. 
- Ako smjer nije definiran smatra se da je veza nepoznata (nedefinirana) ili bidirekcionalna.

Primjeri pridruživanja

C++

Razred Sveučilište

```
class Fakultet;  
class Sveučilište {  
public:  
    Fakultet *Pripadnost sveučilištu;  
};
```

Razred Fakultet

```
class Student;  
class Fakultet {  
public:  
    Student *Pripadnost fakultetu;  
};
```

Razred Student

```
class Fakultet;  
class Student {  
public:  
    Fakultet *Pripadnost fakultetu;  
};
```

Java

Razred Sveučilište

```
public class Sveučilište {  
    public Fakultet Pripadnost sveučilištu;  
}
```

Razred Fakultet

```
public class Fakultet {  
    public Student Pripadnost fakultetu;  
}
```

Razred Student

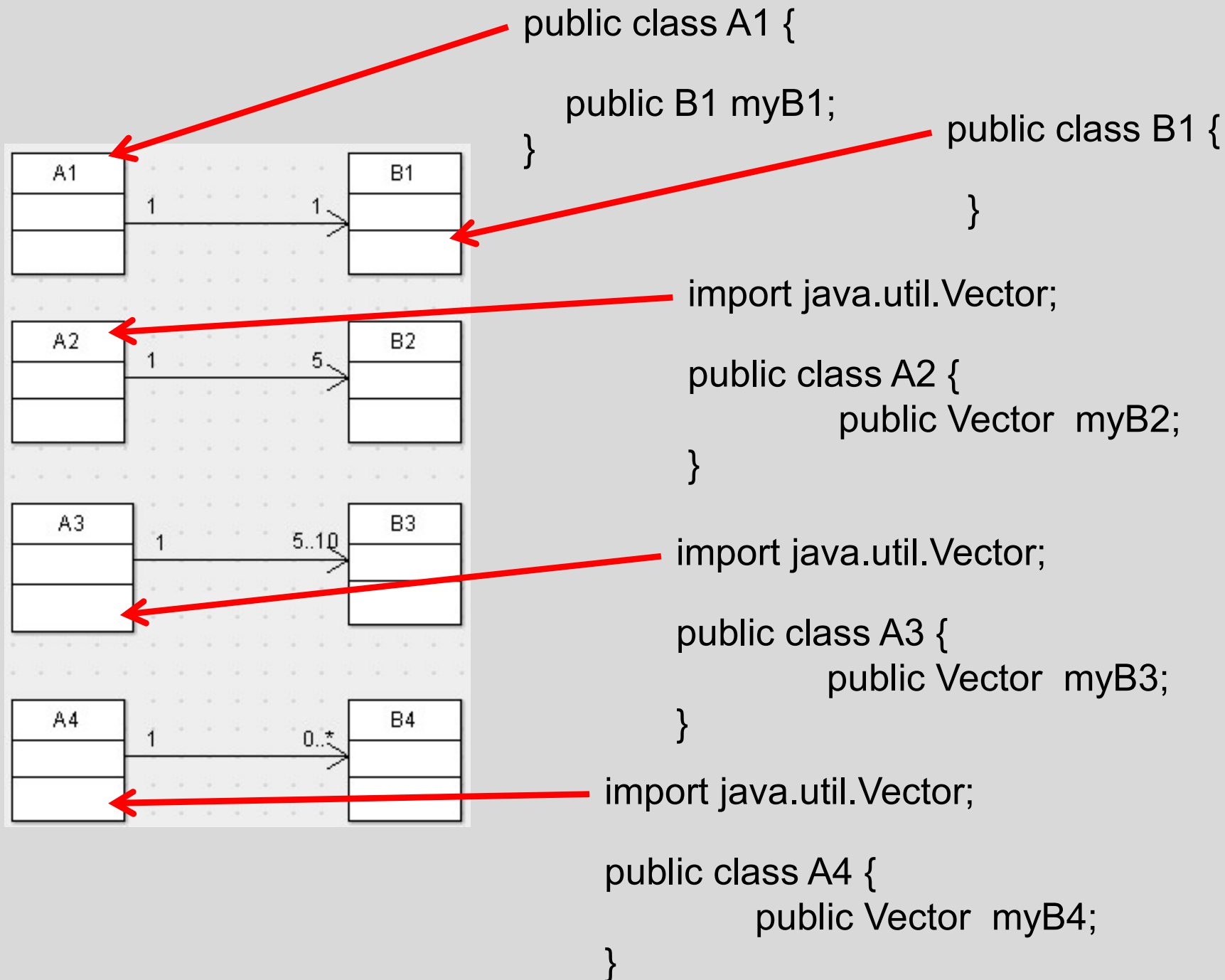
```
public class Student {  
    public Fakultet Pripadnost fakultetu;  
}
```



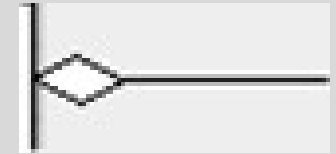
Višestrukost pridruživanja

- Vrh može određivati **višestrukost veze** (engl. *multiplicity*).
 - Govori nam koliko objekata može sudjelovati u određenom odnosu.
- Dozvoljene vrijednosti **na bilo kojoj strani pridruživanja**:
 - 1 = točno 1 pojedinac (podrazumijevana vrijednost)
 - n_1 = bilo koji točno određen broj, npr. 0, 1, 3, 5, 15
 - $n_1..n_2$ = između n_1 i n_2
 - $n_1..*$ ili $n_1..n$ = između n_1 i više pojedinaca, neograničeno
 - $0..*$ ili $*$ ili n = više pojedinaca, neograničeno

Primjeri višestrukosti asocijacija



Agregacija



- **Vrsta pridruživanja** koja pokazuje da jedan razred sadrži druge, tj. da je dio drugog razreda.
 - Razred je agregiran (sadržan) u drugom razredu → oblik odnosa cjelina-dio (tj. podskup-nadskup) → veza DIO-OD (engl. PART-OF).



Java

```
import java.util.Vector;

public class Fakultet {

    public Vector myStudent;
}
```

C++

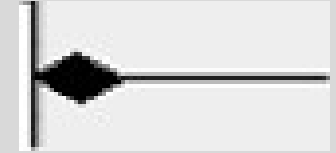
```
#ifndef Fakultet_h
#define Fakultet_h

#include <vector>
#include "Student.h"

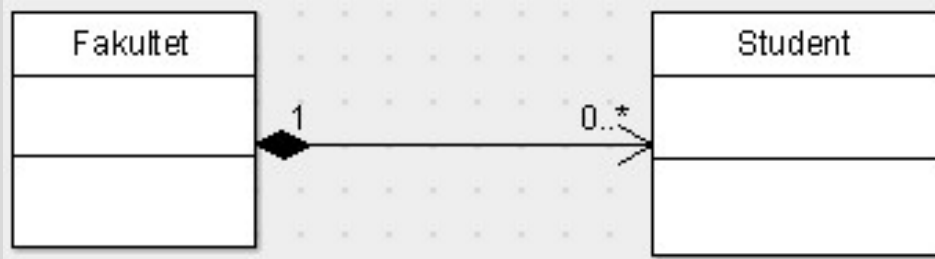
class Fakultet {
public:
    std::vector< Student* > myStudent;
};

#endif // Fakultet_h
```

Kompozicija



- **Vrsta pridruživanja** slična agregaciji, ali kod uništavanja objekta (tj. pojedinca) uništavaju se i pojedinci razreda koji su dio tog objekta.



Java

```
import java.util.Vector;

public class Fakultet {

    public Vector myStudent;
}
```

C++

```
#ifndef Fakultet_h
#define Fakultet_h

#include <vector>
#include "Student.h"

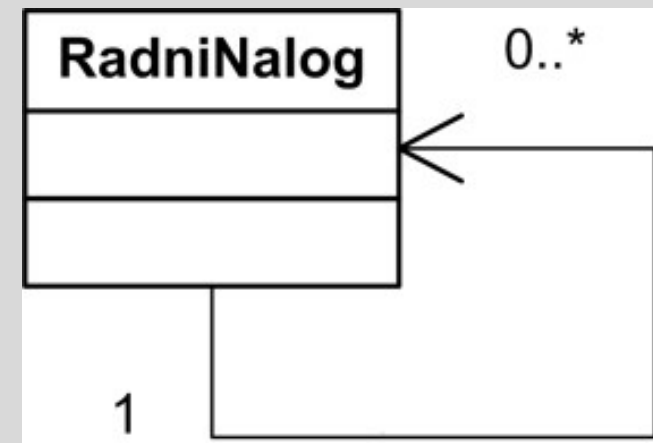
class Fakultet {
public:
    std::vector< Student > myStudent;
};

#endif // Fakultet_h
```

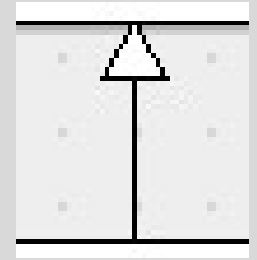
Refleksivno pridruživanje

- Više pojedinaca istog razreda ponekad ovise jedan o drugome ili međusobno komuniciraju.
- Ova vrsta ovisnosti realizira se pomoću refleksivnog pridruživanja, agregacije ili kompozicije.

Moguća je jednosmjerna i dvosmjerna veza, agregacija, kompozicija, ovisnost, ali nije moguća veza poopćenja.



Nasljeđivanje (1)



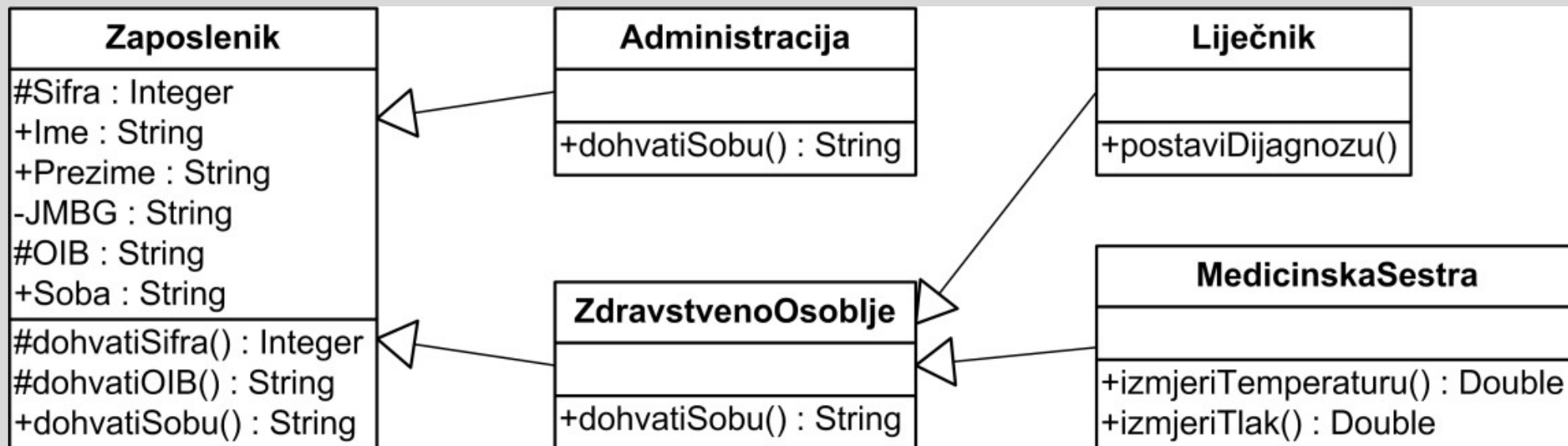
- **Nasljeđivanje** (engl. *inheritance*) je koncept UML-a u kojemu objekt koji se nasljeđuje je proširen u objektu koji ga nasljeđuje.
- Oblik UML odnosa - podtip.
 - Nasljedna veza između razreda.
 - Jedan razred je *roditelj* (nadrazred) drugome razredu (*dijete* ili podrazred).
 - Odnos roditelj-dijete → pravilo JE, podskup-skup, (engl. IS-A).

Nasljeđivanje (2)

- **Poopćenje** – omogućuje stvaranje nadrazreda koja objedinjuje strukturu i ponašanje zajedničko za nekoliko razreda.
- **Specijalizacija** – omogućuje stvaranje podrazreda koja predstavlja dodavanje novih elemenata.
 - Podrazred uvijek ima više ili jednak broj svojstava u odnosu na nadrazred
 - Podrazred nasljeđuje od nadrazreda attribute, relacije i operacije.
 - Podrazred može biti proširen atributima, operacijama ili relacijama.
 - Podrazred može imati svoju implementaciju operacija koje je naslijedio.

Primjer nasljeđivanja

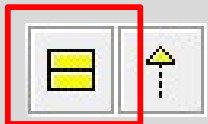
poopćenje



specijalizacija

Sučelje

- **Sučelje** (engl. *interface*) je skup operacija koja specificira usluge nekog razreda.
 - Sučelje definira skup operacijskih specifikacija (tj. njihovih potpisa), ali nikada skup njihovih implementacija.
 - **Sučelje je razred, ali bez atributa i sve operacije imaju samo tijelo, bez implementacije.**



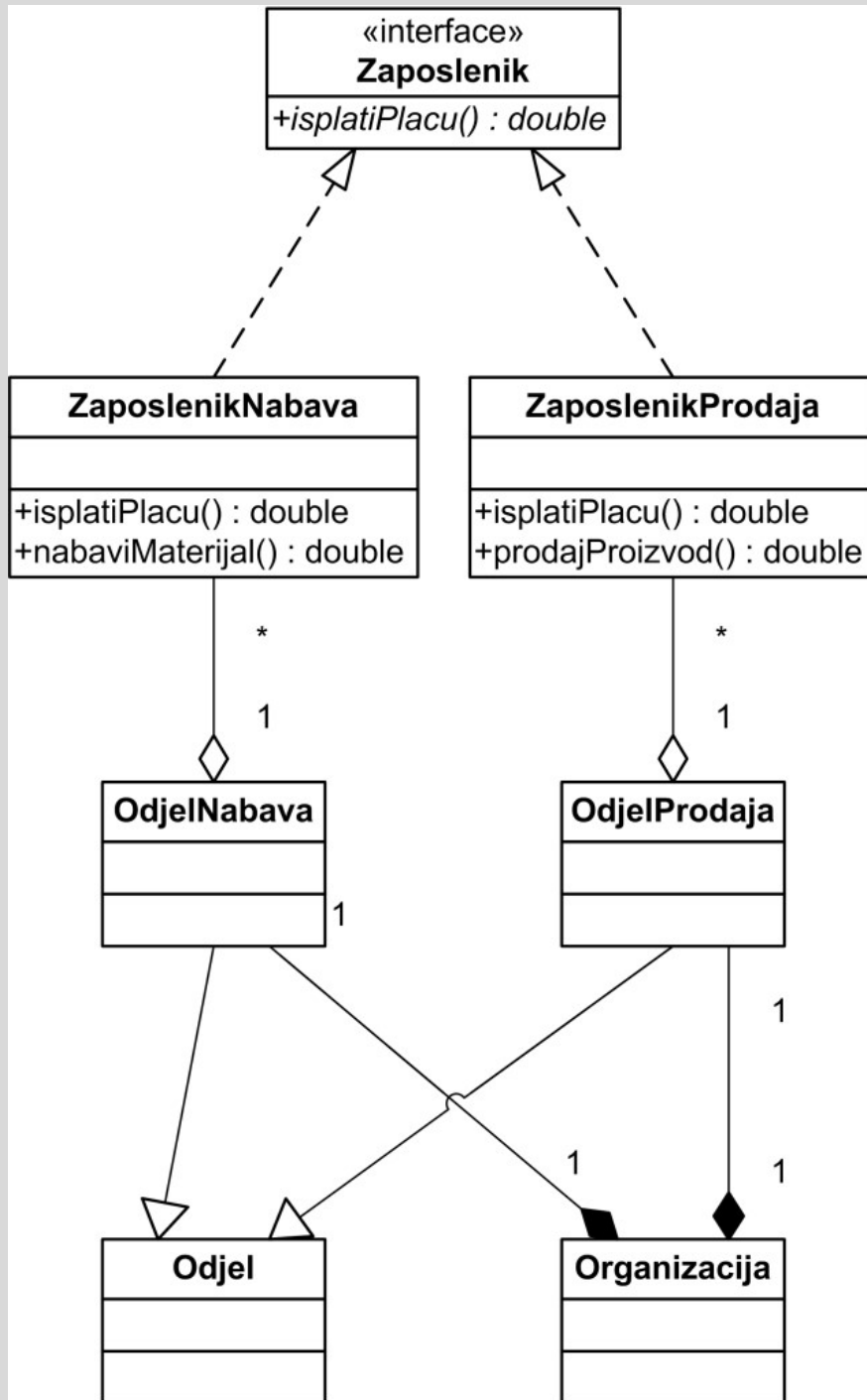
```
public interface MojeSučelje {  
  
    public boolean nekaMetoda(int arg1);  
  
    public void josJednaMetoda(int arg1);  
}
```

Realizacija sučelja



- **Realizacija** (engl. *realization*) je veza UML-a koja označava ostvarenje sučelja.
- **Razred realizira ili ostvaruje sučelje.**
 - **Veza realizacije** (strelica) je usmjerena **od razreda prema sučelju**.
- Realizacija je slična nasljeđivanju, ali u realizaciji nasljeđuju se samo definicije operacija, bez njihove **implementacije**. S nasljeđivanjem specifičniji razred dobiva sve atribute i operacije općenitijeg ili nadređenog razreda.

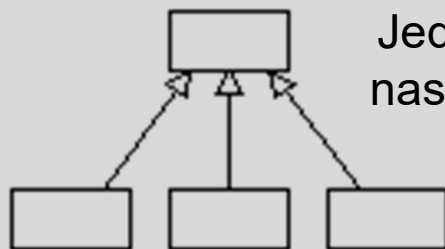
Primjer sučelja i realizacije



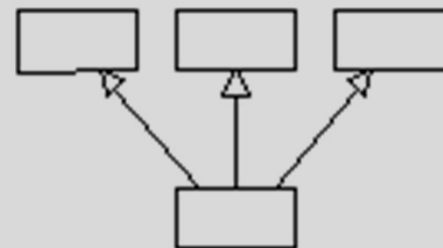
U nekoj organizaciji postoje odjeli Nabave i Prodaje. Odjeli imaju vlastite zaposlenike, koji rade samo u tom odjelu. Svi zaposlenici dobivaju isplate mjesečnih plaća, ali zaposlenici nabave i prodaje imaju drugačiji algoritam izračuna iznosa plaće. Zaposlenici prodaje mogu prodati proizvode organizacije, a zaposlenici Nabave kupiti ulazni materijal potreban za proizvodnju. Operacije nabave i prodaje vraćaju *double* vrijednosti. Zaposlenike je potrebno definirati koristeći zajedničko sučelje.

Višestrukost nasljeđivanja i realizacije

- Višestruko nasljeđivanje (engl. *multiple inheritance*) je zabranjeno u nekim OO programskim jezicima (npr. Java i C#). U C++ je dozvoljeno.
- Višestruka realizacija je uvijek dopuštena.
 - Omogućuje opće višestruko nasljeđivanje (engl. *general multiple inheritance*) i u Javi tako da je moguće implementirati više razreda bez mijenjanja njihove definicije, što je u konačnici slično učinku višestrukog nasljeđivanja.



Jednostruko
nasljeđivanje



Višestruko
nasljeđivanje

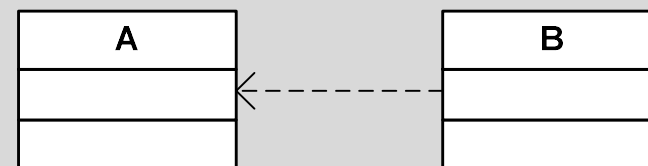
Identificiranje vrste odnosa

- Kako odrediti koja vrsta odnosa između dva razreda je ispravna: **pridruživanje, agregacija, kompozicija ili nasljeđivanje?**
- **Agregacija:** ako jedan razred obuhvaća ili sadržava drugog (povezani su odnosom cjelina-dio).
- **Kompozicija:** ako su razredi u odnosu cjelina-dio, ali nakon uništavanja pojedinaca cjeline moraju se uništiti i pojedinci koji čine dio cjeline.
- **Nasljeđivanje:** ako su razredi u odnosu roditelj-dijete.
- **Pridruživanje:** ako razredi nisu u odnosima cjelina-dio i roditelj-dijete.

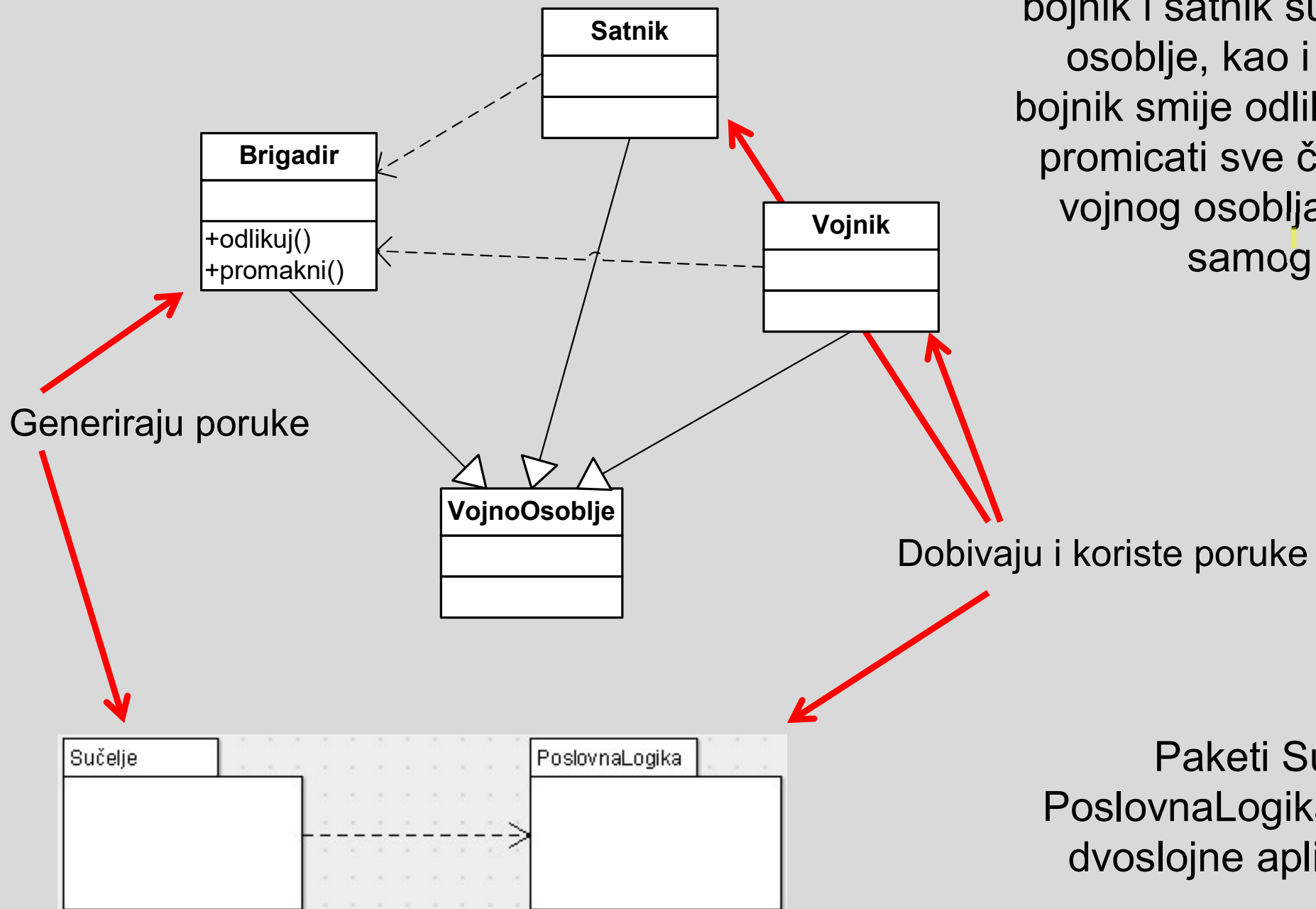
Ovisnost



- **Ovisnost** (engl. *dependency*) pokazuje da jedan razred ili paket dijagrama ovisi o drugome.
 - Semantička relacija između dvije stvari u kojoj promjena u jednoj (neovisnoj stvari) može utjecati na semantiku druge (ovisne stvari).
- Ovisnost je uvijek jednosmjerna.
- Čitamo: “**B ovisi o A**” u smjeru strelice.
 - **A** se naziva isporučitelj (engl. *supplier*) i **B** se naziva klijent (engl. *client*).
- ArgoUML ne preslikava svojstvo ovisnosti u programski kôd.



Primjeri ovisnosti



bojnik i satnik su vojno osoblje, kao i vojnik. bojnik smije odlikovati i promovirati sve članove vojnog osoblja (osim samog sebe).

Odbrojčavanje

- **Odbrojčavanje** (engl. *enumeration*) je oblik tipa podatka koji sadržava uređene parove imenovanih identifikatora i njima pridruženih vrijednosti.
 - Te vrijednosti nazivaju se odbrojčani literali.

• **Koriste se za opis diskretnih vrijednosti.**

«enumeration» KomisijaOcjena
+JednoglasnoPoložio = 1
+Položio = 2
+NijePoložio = 0

Komentari

- Unatoč formalnoj izražajnosti UML dijagrama razreda ponekada su potrebni i **komentari**.
 - Ne upotrebljavaju se uvijek. Koriste se za dodatni opis svrhe nekog razreda, atributa, veza, operacija i drugih elemenata dijagrama.
 - U komentarima je poželjno biti **jasan i sažet**, te **obuhvatiti sve bitne** aspekte UML elementa koji se opisuje, a koji nisu nedvosmisleno jasni iz samog dijagrama.
 - Komentari mogu biti povezani s konkretnim elementom dijagrama, ili se mogu odnositi na cijeli dijagram. Specifični komentari povezani su s elementom neoznačenom vezom, a komentari o cijelom dijagramu nemaju takvih veze i nalaze se na rubu crteža obično u jednom od uglova dijagrama.

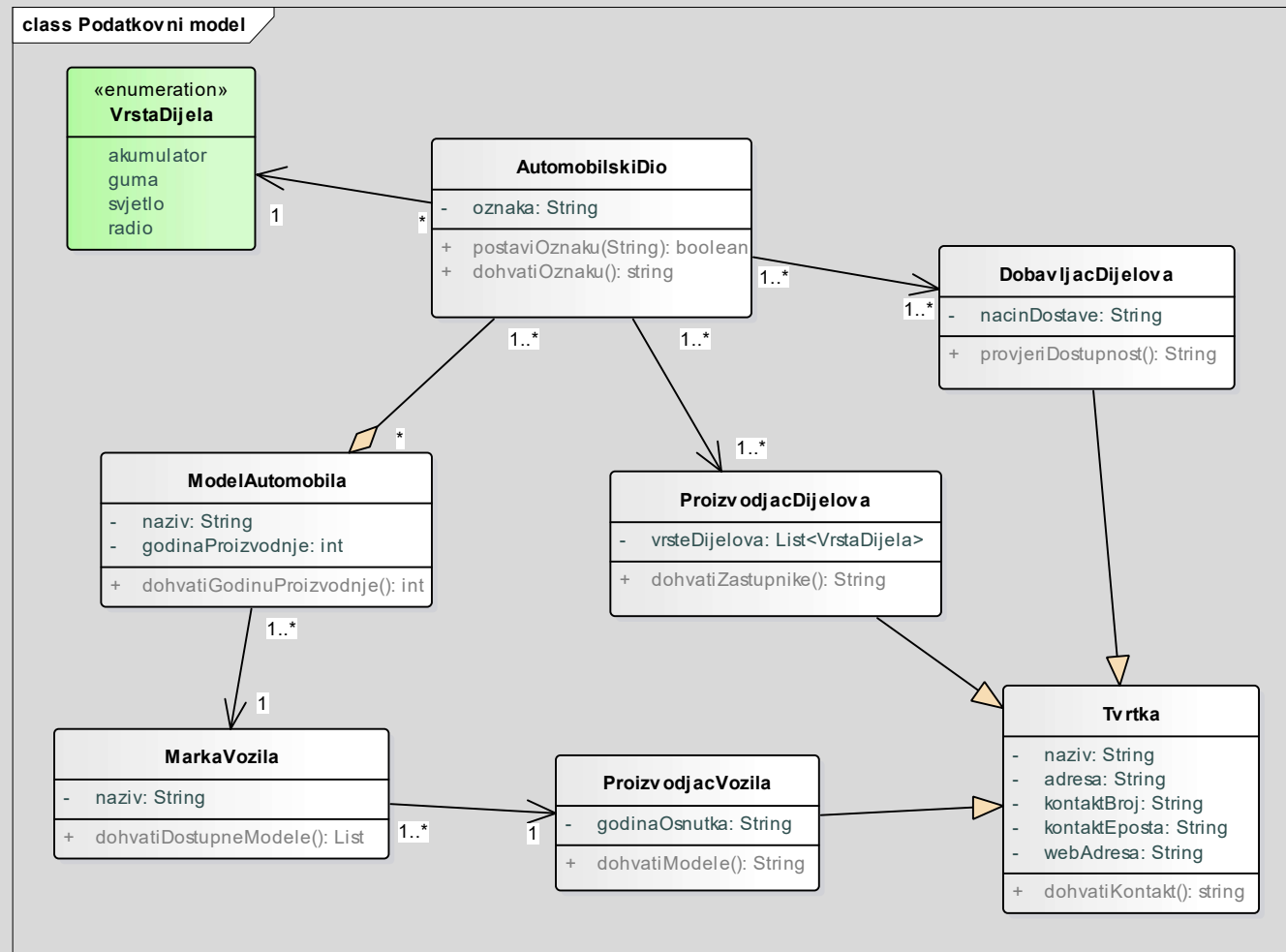
Laboratorijske vježbe

UML DIJAGRAMI RAZREDA U DOKUMENTU ZAHTJEVA

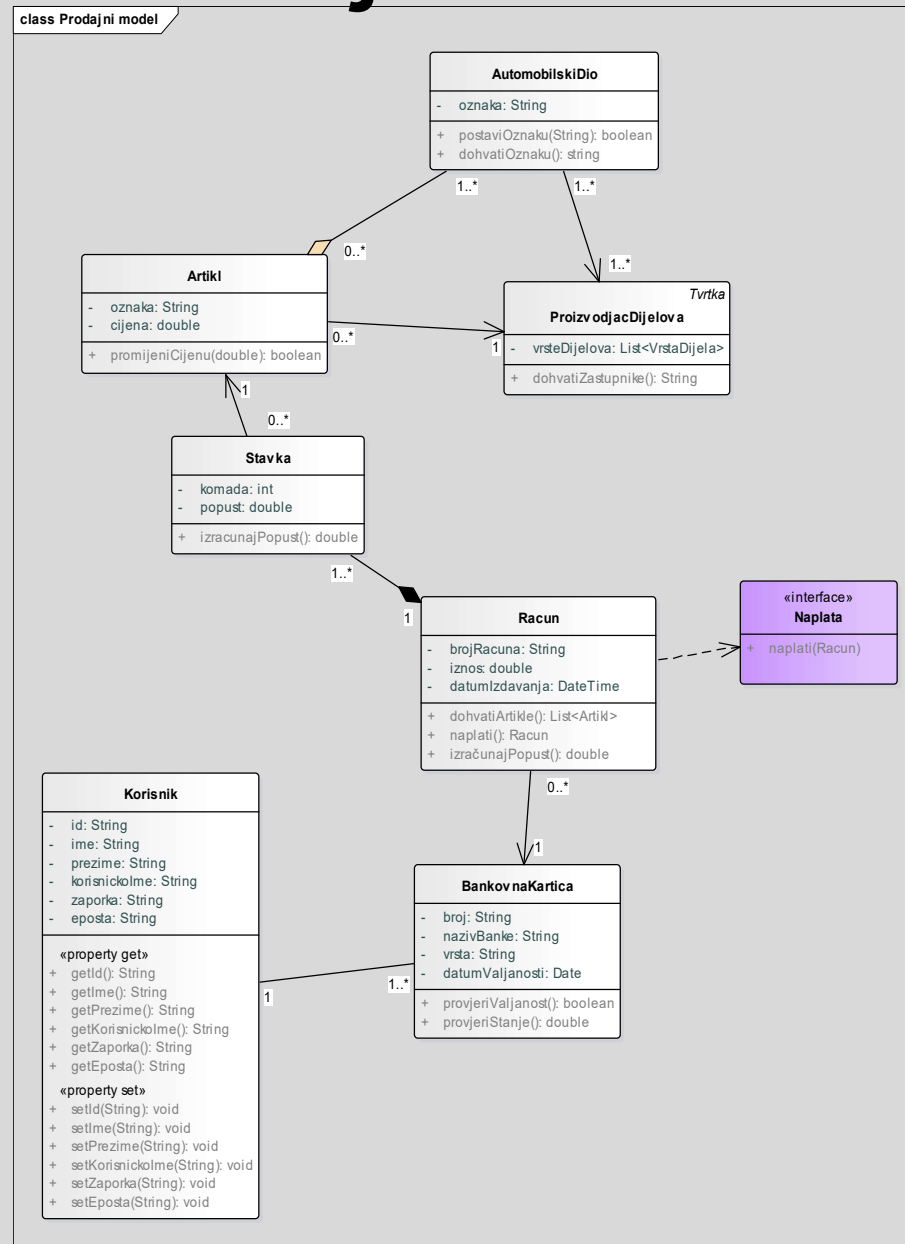
Dijagram razreda

- Opis arhitekture:
 - Opis domene
 - Opis podatkovnog modela
- Poslije, u dokumentu ostvarenja:
 - opis ostvarenja

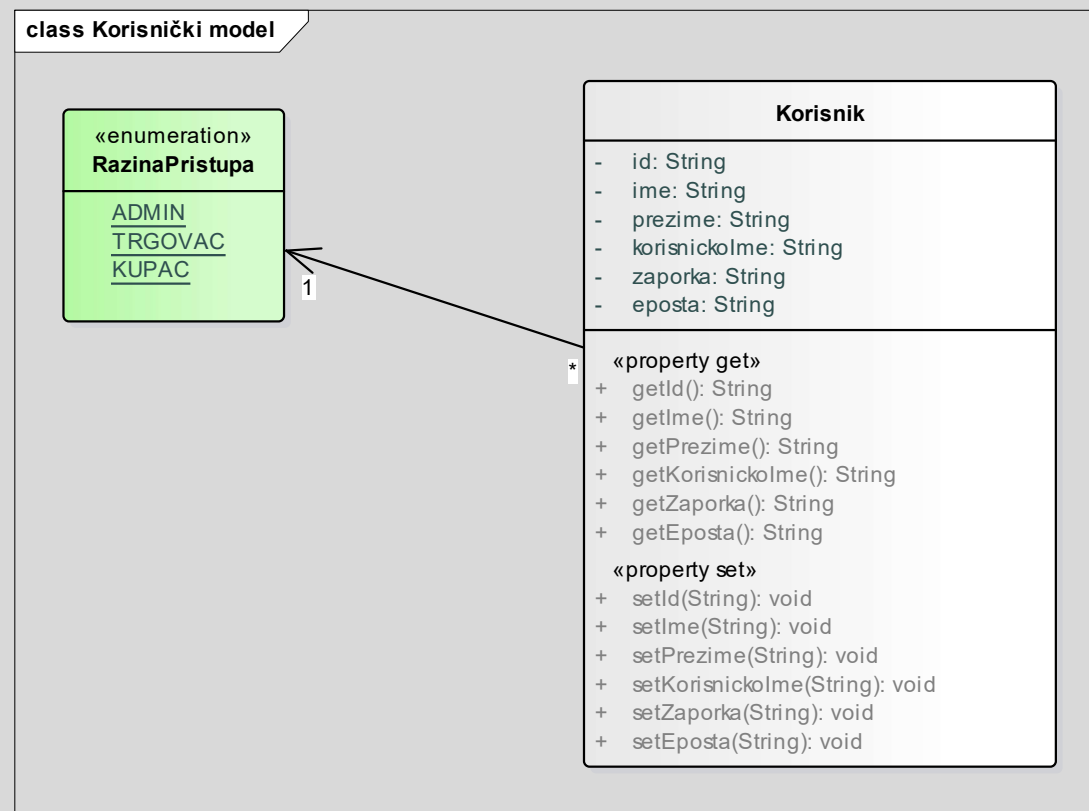
Podatkovni model prodaje auto-dijelova



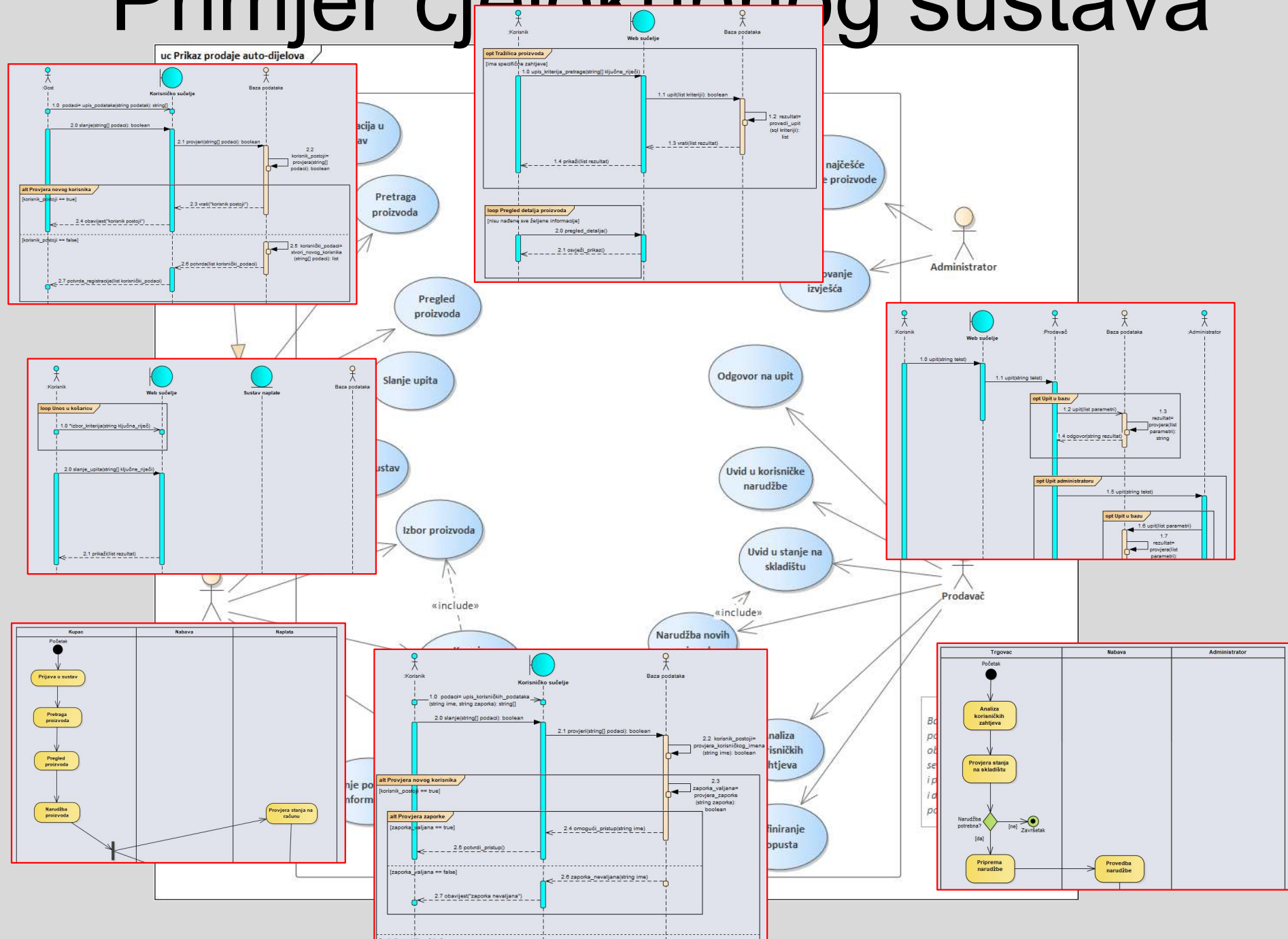
Podatkovni model prodaje auto-dijelova



Podatkovni model prodaje auto-dijelova



Primjer cjelokupnog sustava



Projektna dokumentacija

- moraju biti zastupljeni svi UML dijagrami o kojima se govorilo na predavanjima:
 - jedan ili više dijagrama obrazaca uporabe kojima se prikazuje cijeli sustav
 - najmanje dva dijagrama aktivnosti (ili jedan dijagram stanja) i koliko god je potrebno dijagrama slijeda za opis svih obrazaca uporabe
 - jedan ili više dijagrama razreda
 - jedan ili više dijagrama komponenti
 - dijagram razmještaja

Nekoliko savjeta

- Dijagrami razreda su dio gotovo svih objektno-orijentiranih (OO) paradigmi i **koriste se veoma često**.
- **Mogu biti** bogati informacijama i stoga **teško čitljivi**, stoga nekoliko savjeta:
 - Ne koristite odmah sve moguće notacije. Počnite s jednostavnim dijagramom i postepeno dodajte detalje.
 - Razlikujte različite poglede na sustav.
 - U praksi ne morate modelirati baš sve. Koncentrirajte se na bitne segmente. Bolje je imati nekoliko kvalitetnih dijagrama po dijelovima sustava, nego zastarjeli dijagram cijelog sustava.

Različita gledišta na dijagram

- Prema razini detalja → u različitim razvojnim fazama projekta:
 - konceptualna
 - specifikacijska
 - implementacijska

Različita gledišta na dijagram

- **konceptualna:**
 - opis stvari iz stvarnog svijeta
 - prikaz koncepta iz domene sustava → odgovaraju razredima
 - **jezično neovisni prikaz**

Različita gledišta na dijagram

- **specifikacijska:**
 - apstrakcije i komponente sa specifikacijom i sučeljima
 - nema čvrste obveze prema stvarnoj implementaciji
 - **pogled na razini sučelja**, ne implementacije

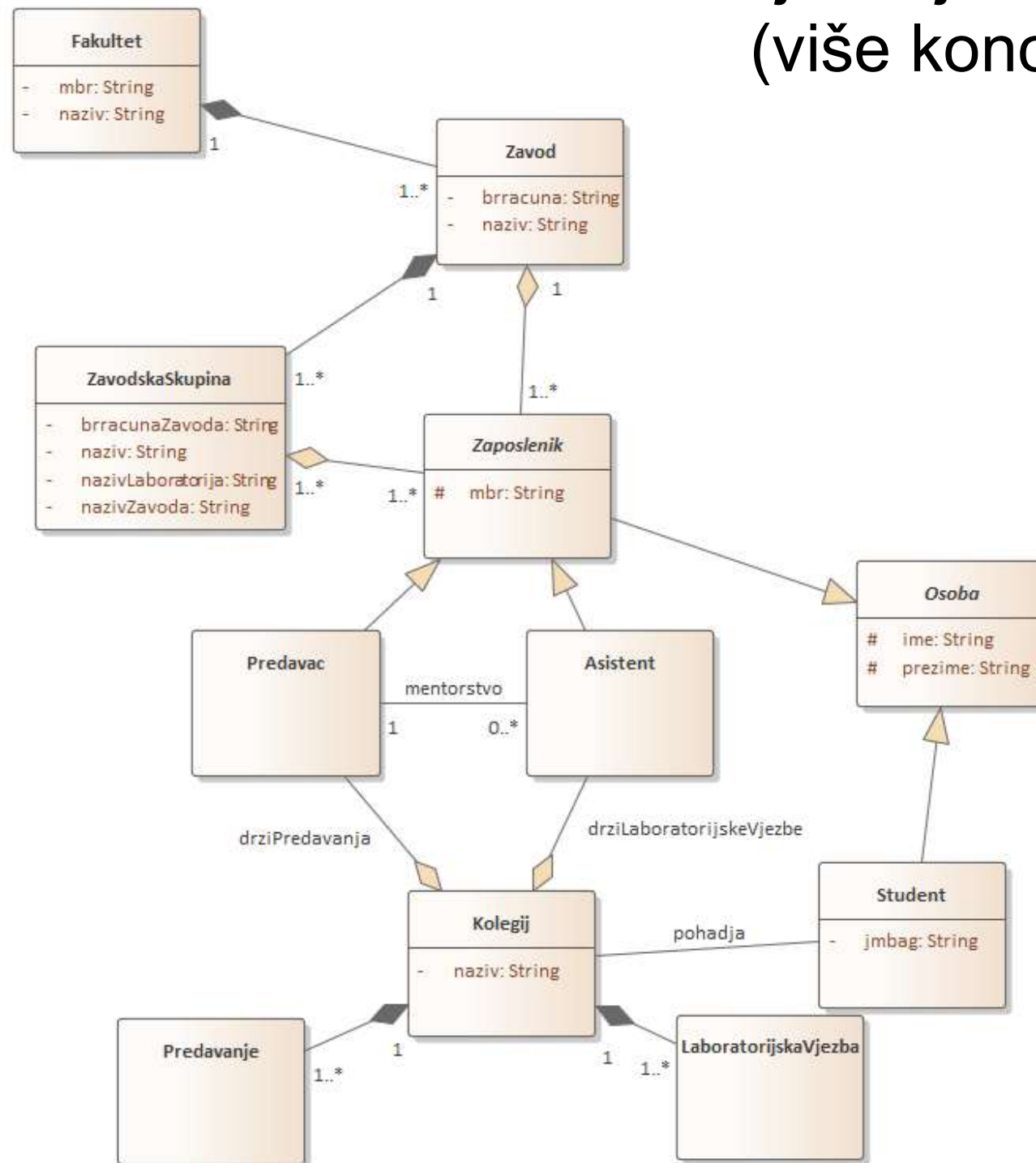
Različita gledišta na dijagram

- **implementacijska:**
 - opis na razini konkretne tehnologije i jezika
 - **pogled na razini implementacije**

Primjer 1: Modeliranje organizacije fakulteta

Neki fakultet sastoji se od jednog ili više zavoda, a svaki zavod od jedne ili više zavodskih skupina. Zavodsku skupinu čine zaposlenici. Zaposlenici mogu raditi i u nekoliko zavodskih skupina, ali ne mogu raditi na više zavoda. Postoje dva konkretna tipa zaposlenika: predavači i asistenti. Svaki predavač ima barem jedan kolegij koji predaje, a svaki asistent drži laboratorijske vježbe iz barem jednog kolegija. Svaki kolegij može imati jednog ili više predavača i asistenata. Asistent ima jednog predavača u funkciji mentora, a predavač može imati više asistenata. Svaki kolegij se sastoji od više predavanja i više laboratorijskih vježbi i ima svoj naziv (String). Ukidanjem kolegija ukidaju se predavanja i laboratorijske vježbe, ali naravno, ne otpuštaju se zaposlenici koji kolegij drže. Student je zasebna kategorija u organizaciji fakulteta i u ovom modelu pretpostavite samo da sluša jedan ili više kolegija. I student i zaposlenik su osobe. Svaka osoba ima svoje ime i prezime. Dodatno, svaki zaposlenik ima svoj matični broj zaposlenika (String), a svaki student svoj JMBAG (String). Fakultet ima svoj matični broj (String) i naziv (String). Zavod ima svoj naziv (String) i broj računa (String). Naziv i broj računa zavoda nasljeđuju i zavodske skupine, s tim da one osim toga imaju i svoj naziv skupine te dodatno, naziv glavnog laboratorija (String).

Rješenje Primjera #1 (više konceptualno)

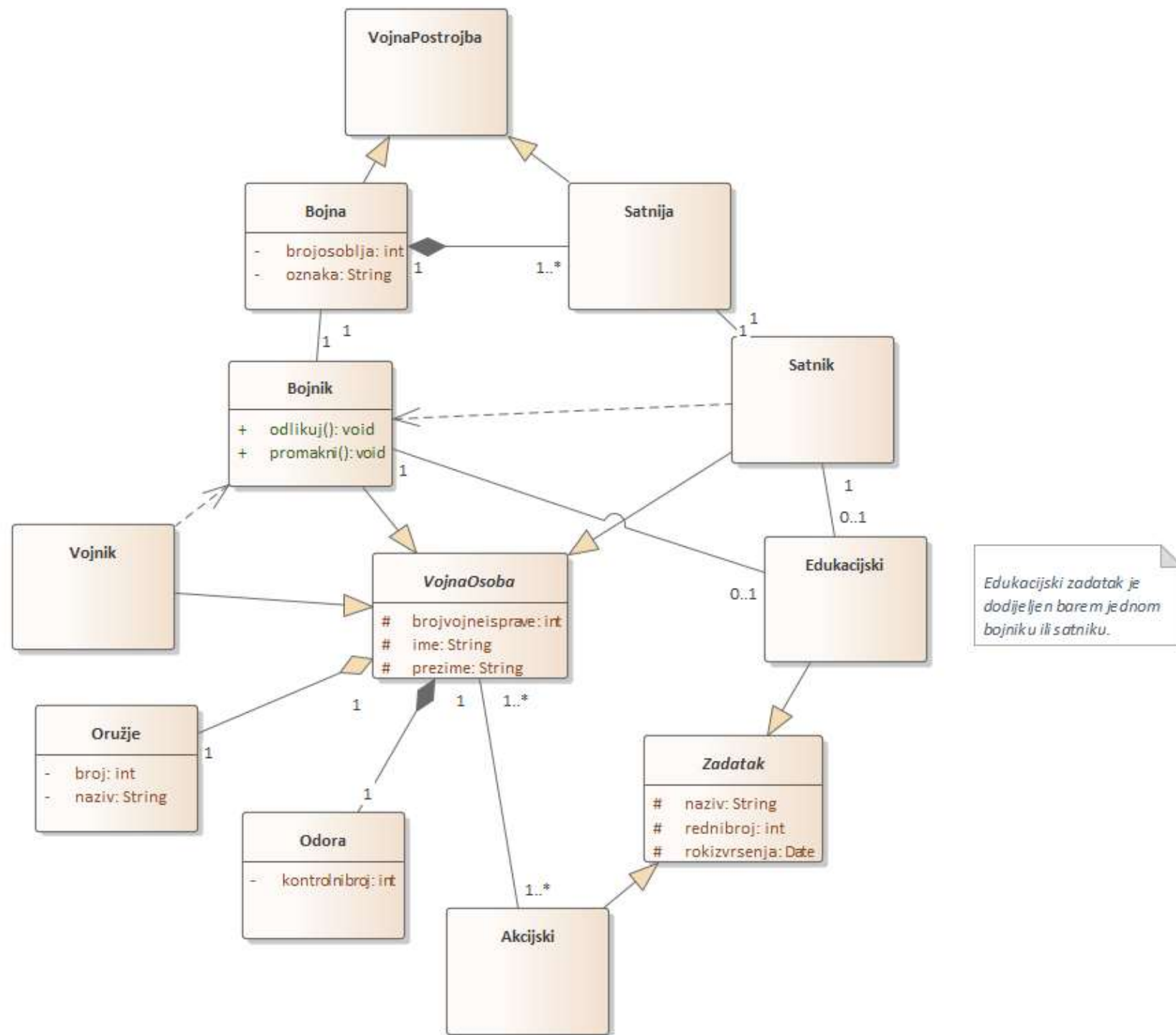


Komentar rješenja Primjera #1

- Veza *pohadja* može biti agregacija između razreda *Student* i *Kolegij*.
- Veza *drziLaboratorijskeVježbe* može biti jednosmjerna umjesto agregacije, kao i veza *drziPredavanja*
- Nije potrebno napraviti dodatnu vezu između razreda *Asistent* i *LaboratorijskeVježbe* jer ovakav sustav omogućuje povezivanje ta dva entiteta.

Primjer 2: Modeliranje organizacije vojne postrojbe

Prepostavite da neka vojna postrojba može biti bojna ili satnija. Svaka bojna sadrži jednu ili više satnija. Na čelu bojne nalazi se bojniki, a na čelu satnije satnik. Bojnik i satnik su vojno osoblje, kao i vojnik. Bojnik smije odlikovati i promicati sve članove vojnog osoblja (osim samog sebe). Svaki član vojnog osoblja ima svoje zadatke. Zadatak ima svoj redni broj (int), naziv (String) i rok izvršenja (Date). Postoje dva tipa zadataka: edukacijski i akcijski. Edukacijske zadatke smiju obavljati samo bojniki i satnici. Oni mogu imati najviše jedan edukacijski zadatak. Svaki edukacijski zadatak drži samo jedan bojniki ili satnik, ali jedan edukacijski zadatak može istodobno imati bojnika i satnika. Svaki član vojnog osoblja može imati jedan ili više akcijskih poslova, a jedan akcijski posao može obavljati više različitih članova vojnog osoblja. Svaki član vojnog osoblja nosi po jedan komad oružja i vojnu odoru. Vojna odora je prilagođena svakom pojedinom članu vojnog osoblja i ako iz bilo kojeg razloga član vojnog osoblja napusti vojnu postrojbu, vojna odora se uništava. Oružje nosi svaki član vojnog osoblja, ali ono ostaje na raspolaganju čak i ako pojedinac napusti vojnu postrojbu. Svaki član vojnog osoblja ima svoje ime i prezime (String) i broj vojne isprave (int). Svaka bojna ima svoju oznaku (String) i broj vojnog osoblja (int). Svaki komad oružja ima svoj broj (int) i naziv (String). Svaka odora ima svoj kontrolni broj (int).



Komentar rješenja Primjera #2

- Veza razreda Bojnik i Zadatak, te Satnik i Zadatak?

REFERENCE I LITERATURA

- Sveučilišna zbirka zadataka iz UML-a - A. Jović, M. Horvat, I. Grudenić, “UML-dijagrami, zbirka primjera i riješenih zadataka”, 2014
- Allen Holub's UML Quick Reference:
<http://www.holub.com/goodies/uml>
- Booch G., Jacobson I., Rumbaugh J. “UML Distilled”

Hvala na pažnji!

Pitanja?