




Dijagrami razreda

Oblikovanje programske potpore

Ak. god. 2012 / 2013

Pripremio: mr.sc. Marko Horvat

Auditorne raspored

1. Dokumentacija . Danko Ivozevi
2. ArgoUML, use case i sekvencijski dijagrami . Alan Jovi
3. Dijagrami razreda . Marko Horvat  tu smo
4. Ostali dijagrami . Marko Horvat

Karakteristike dijagrama razreda (1)

- “ **Dijagram razreda** opisuje vrste objekata unutar nekog sustava i njihove međusobne odnose.
 - . Class diagram opisuje razrede (klase) i njihove međusobne veze.
- “ Strukturalni UML diagram, opisuje statične 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.

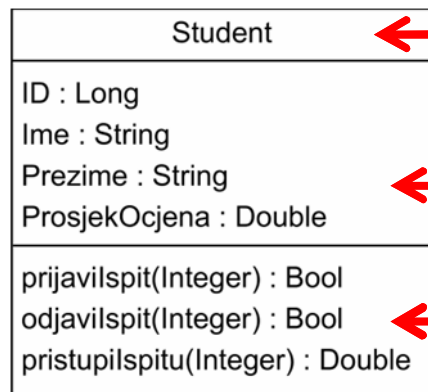
Karakteristike dijagrama razreda (2)

“ Naj ez i termin koji koristimo je:

• **Class diagram**

“ Hrvatski termini:

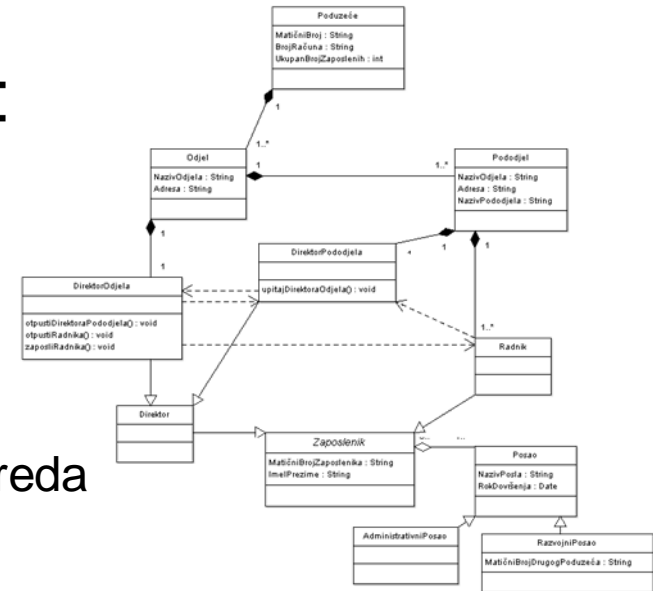
• **Dijagram razreda ili dijagram klasa**



← Jedinstveni naziv razreda

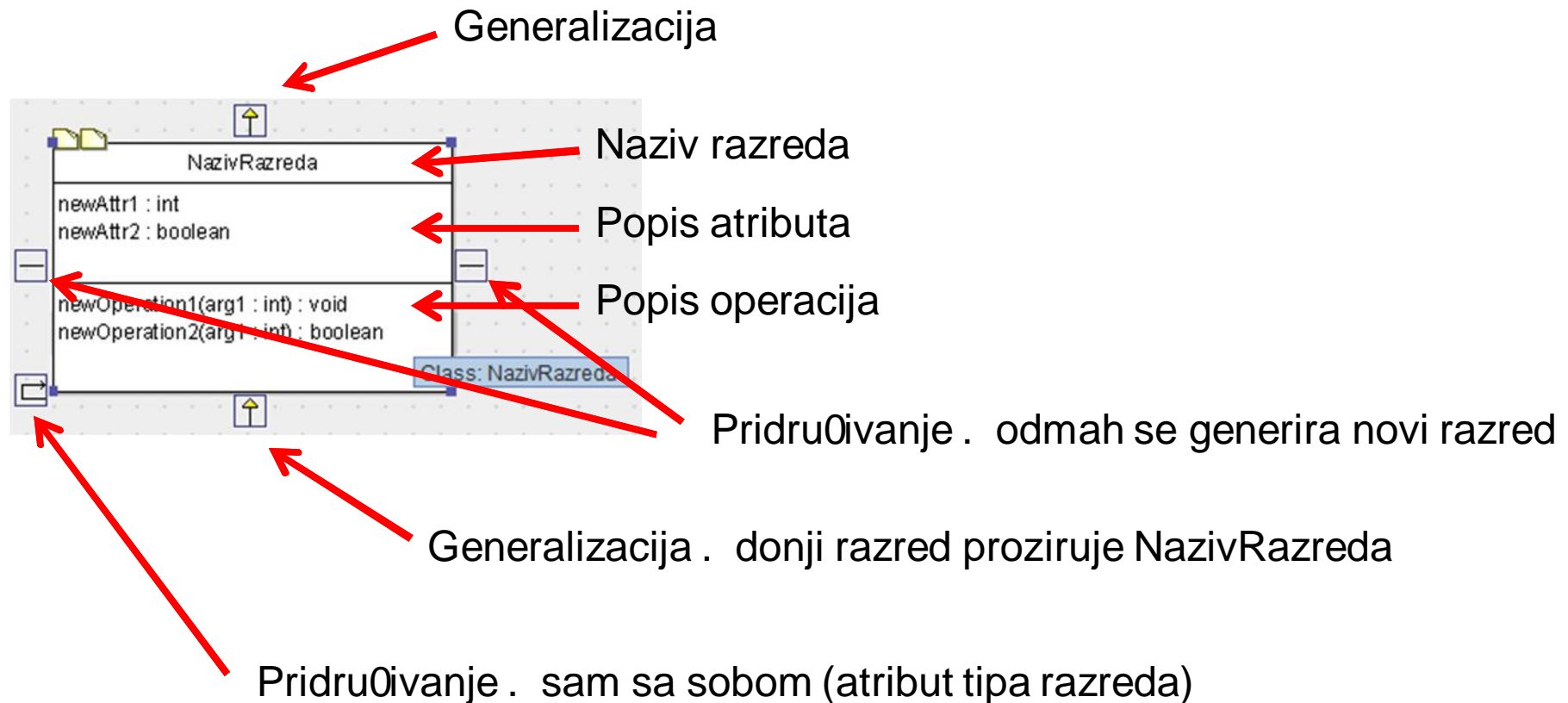
← Skup atributa

← Skup operacija



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

Simbol razreda u uređivaču u ArgoUML



Napomena: Svi dijagrami izrađeni su u uređivačima ArgoUML i Microsoft Visio. Izvorni kod dobiven je iz ArgoUML-a (kartica Source Code). Kod je koristan za ilustraciju dijagrama, ali potrebno ga je proširiti i provjeriti njegovu ispravnost u nekom od jezičnih procesora.

Razred

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

1. Objekt

- . Predstavlja entitet iz stvarnog svijeta ili neki koncept.
- . Apstrakcija ne čega zto ima dobro definirane granice i smisao u sustavu.

2. Razred

- . Opis grupe objekata sa sličinim svojstvima.
- . Svaki objekt je pojedinac (instanca) jedne klase.

Atributi

” **Atributi** (engl. *attributes*) razreda imaju sljedeća svojstva:

- . **Stupanj 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**

” Atribut je dostupan svim razredima i paketima.

- **Package**

” Atribut je dostupan svim razredima istog paketa.

- **Protected**

” Atribut je dostupan unutar istog razreda i izvedenih razreda.

- **Private**

” Atribut je dostupan samo unutar istog razreda.

Vrsta ili tip atributa

- “ Dozvoljeni su sljede i UML tipovi:
 - . Boolean, Integer, String, UnlimitedInteger
- “ Dozvoljeni su i dodatni Java tipovi podataka:
 - . byte, char, double, float, int
 - . I razredi iz java.util (Collection, Date, List, Set, SortedSet, Time, Vector ō), java.lang, java.lang.math i java.net (URL) paketa
 - . I vlastiti tipovi razreda i podataka **u istom dijagramu**

Promjenjivost atributa (2)

“ **addOnly**

- . Vrijednost atributa može se samo povećavati.

“ **changeable**

- . Vrijednost atributa može se nesmetano mijenjati. Podrazumijevana (*default*) postavka.

“ **frozen**

- . Vrijednost atributa (ili asocijacije) ne smije se promijeniti tijekom života (engl. *lifetime*) pripadajućeg objekta.

Promjenjivost atributa (2)

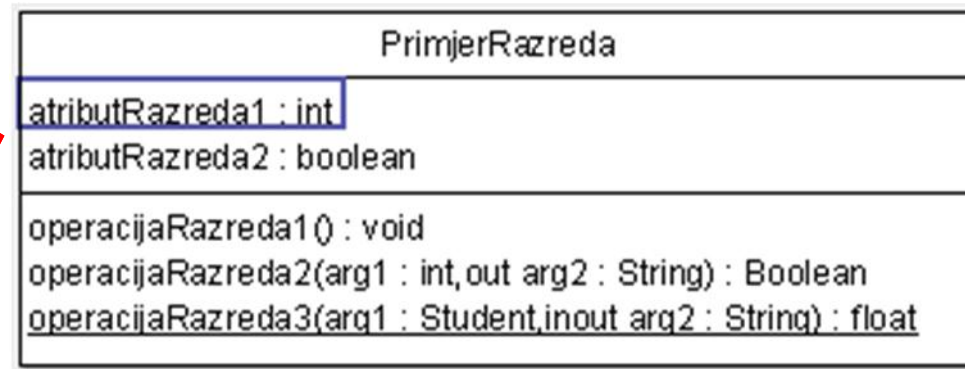
“ **static**

- . Modifikator, vrijednost atributa ne mijenja se (konstanta) i ne ovisi o Oivotu objekta.

“ **read-only**

- . Vrijednost atributa ne može se mijenjati izvan objekta kojemu pripada.
 - “ Nije isto zto i frozen.
- . **ArgoUML ne podržava direktno read-only,** ali va0no ga je spomenuti.

Primjer rada s atributima



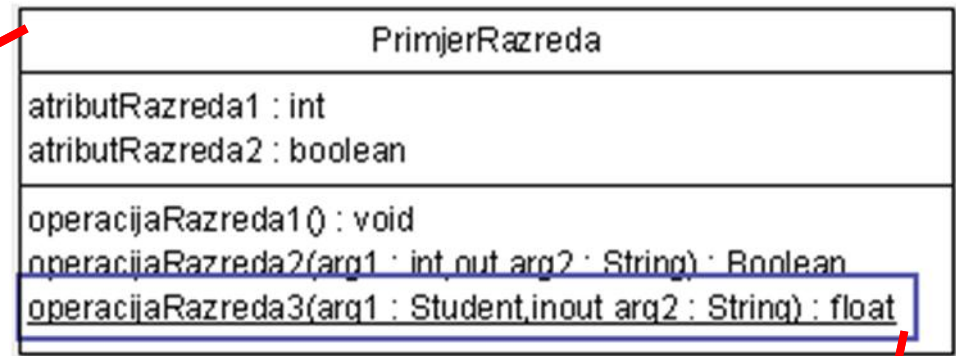
The software interface displays the **Properties** tab for the attribute `atributRazreda1`. The interface includes the following fields and options:

- Name:** `atributRazreda1`
- Type:** `int [UML Profile for Java]`
- Multiplicity:** ☒ 1
- Owner:** `PrimjerRazreda [untitledModel]`
- Visibility:** ☒ public, ☐ package, ☐ protected, ☐ private
- Changeability:** ☒ addOnly, ☐ changeable, ☐ frozen
- Modifiers:** ☐ static
- Initial Value:** (Two empty text boxes)

Operacije

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

Primjer rada s operacijama



UML Editor Properties Panel for Operation:

- Operation:**
 - Name: operacijaRazreda3
 - Owner: PrimjerRazreda [untitledModel]
 - Parameters:
 - return: float
 - arg1: Student
 - arg2: String
- Visibility:**
 - ☐ public
 - ☐ package
 - ☐ protected
 - ☒ private
- Modifiers:**
 - ☐ abstr...
 - ☐ leaf
 - ☐ root
 - ☐ query
 - ☒ static
- Concurrency:**
 - ☒ sequential
 - ☐ guarded
 - ☐ concurrent
- Raised Signals:**
- Methods:**
- Specification:**

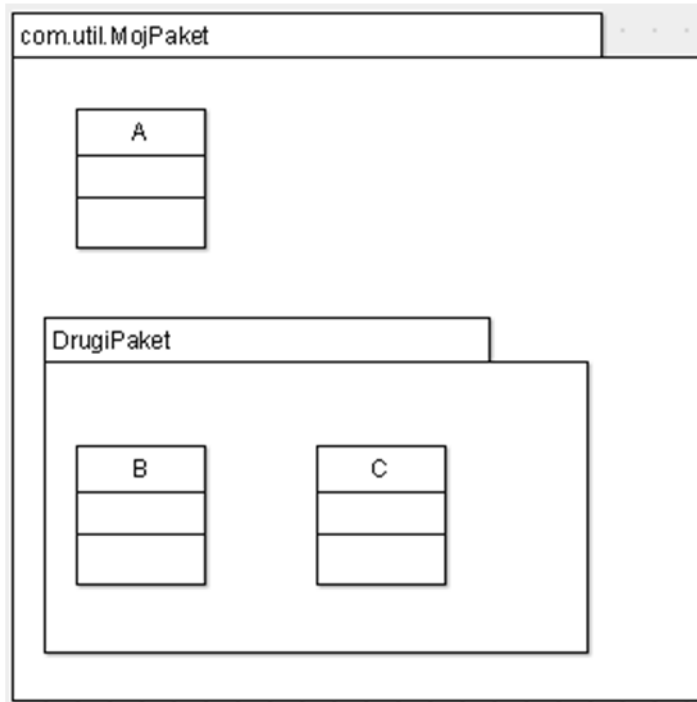
UML Editor Parameter Properties Panel:

- Parameter:**
 - Name: arg2
 - Owner: operacijaRazreda3 [untitledModel::PrimjerRazreda]
- Type:** String [UML 1.4 Standard Elements]
- Default Value:**
- Direction Kind:**
 - ☐ in
 - ☐ out
 - ☒ in/out
 - ☐ return

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++, package u Javi, ò

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

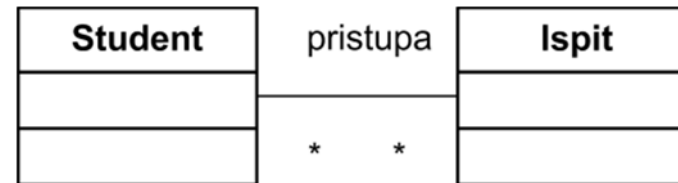
```
package com.util.MojPaket.Drugipaket;
```

```
public class B {}
```


Pridruživanje

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

- Student pristupa ispitu





” 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 pridru0ivanja

- “ Ako je vrh neke asocijacije ozna en 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 pridru0ivanja

C++

Razred Sveučilište

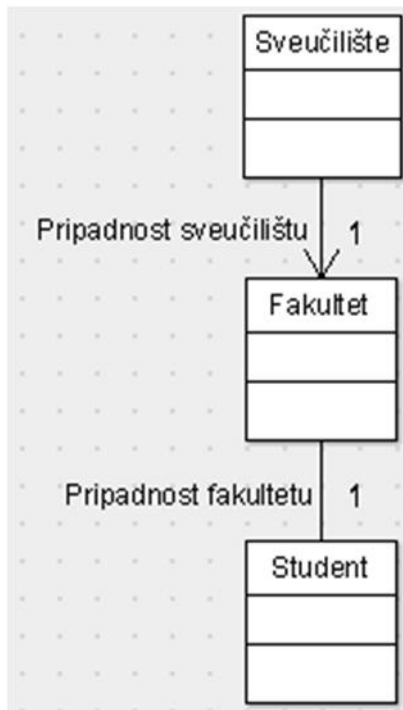
```
class Fakultet;  
class Sveu ilizte {  
public:  
    Fakultet *Pripadnost sveu iliztu;  
};
```

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 Sveu ilizte {  
    public Fakultet Pripadnost sveu iliztu;  
}
```

Razred Fakultet

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

Razred Student

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

Vizestruktost pridru0ivanja

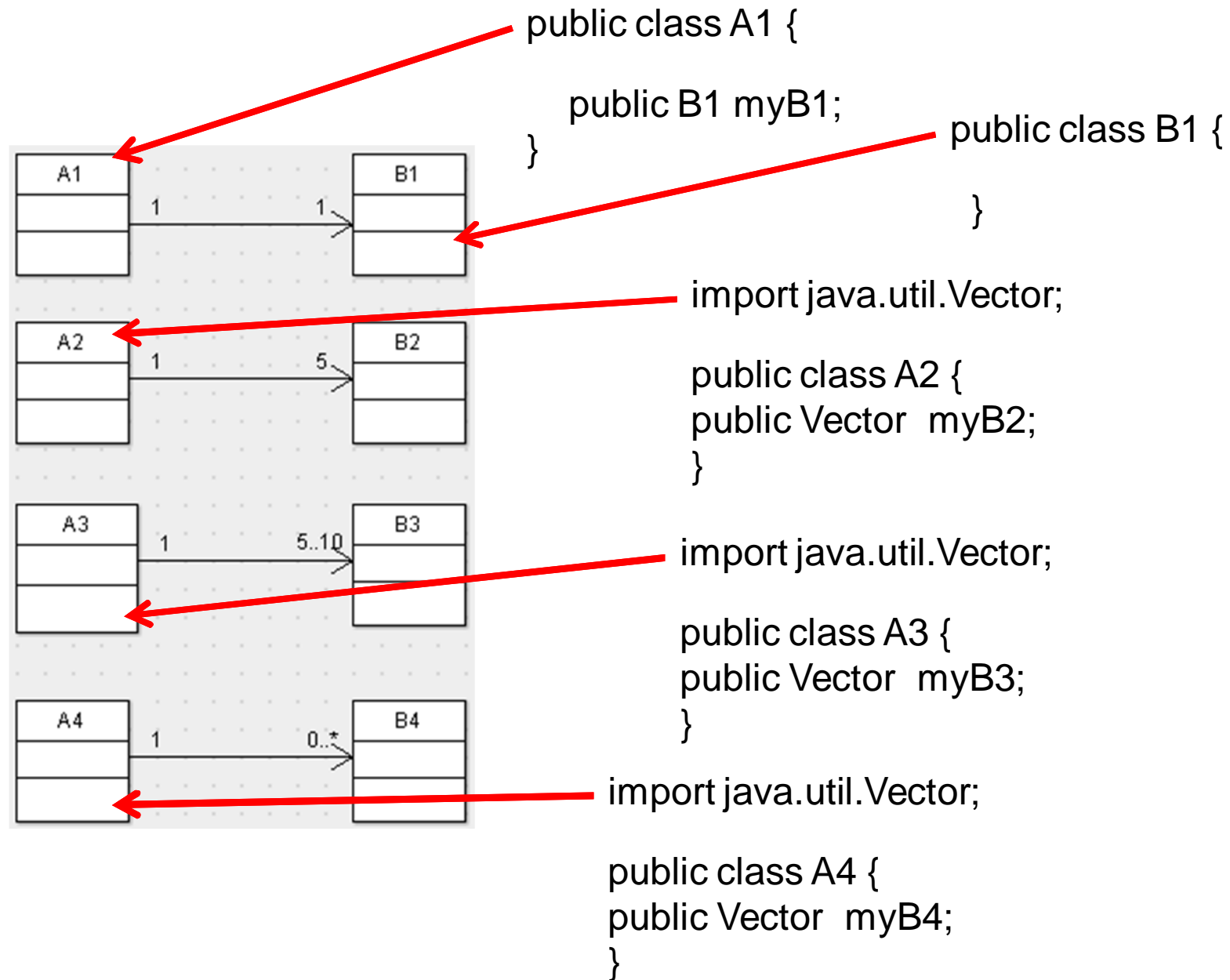
” Vrh mo0e odre ivati **višestrukost veze** (engl. *multiplicity*).

- . Govori nam koliko objekata mo0e sudjelovati u odre enom odnosu.

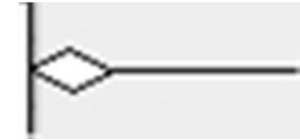
” Dozvoljene vrijednosti **na bilo kojoj strani** pridru0ivanja:

- . 1 = to no 1 pojedinac (podrazumijevana vrijednost)
- . n_1 = bilo koji to no odre en broj, npr. 0, 1, 3, 5, 15
- . $n_1.. n_2$ = izme u n_1 i n_2
- . $n_1..*$ ili $n_1..n$ = izme u n_1 i vize pojedinaca, neograni eno
- . $0..*$ ili $* ili n$ = vize pojedinaca, neograni eno

Primjeri vizestrukosti 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 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"

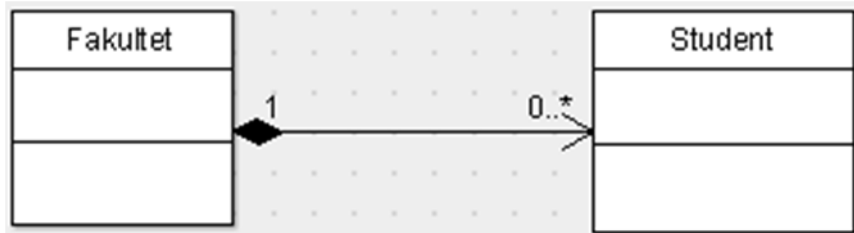
class Fakultet {
public:
    std::vector< Student* > myStudent;
};

#endif // Fakultet_h
```

Kompozicija



” **Vrsta pridruživanja** sli na agregaciji, ali kod uniztavanja objekta (tj. pojedinca) uniztavaju 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

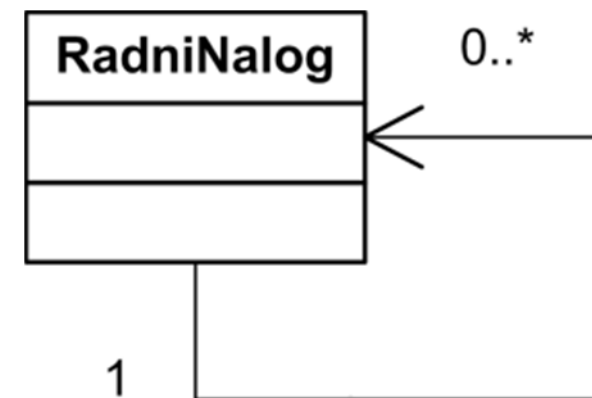
#include <vector>
#include "Student.h"

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

#endif // Fakultet_h
```

Refleksivno pridru0ivanje

- ” Vize pojedinaca istog razreda ponekada ovise jedan o drugome ili me usobno komuniciraju.
- ” Ova vrsta ovisnosti realizira se pomo u refleksivnog pridru0ivanja, agregacije ili kompozicije.
- ” Mogu a je jednosmjerna i dvosmjerna veza.



Nasljeđivanje (1)

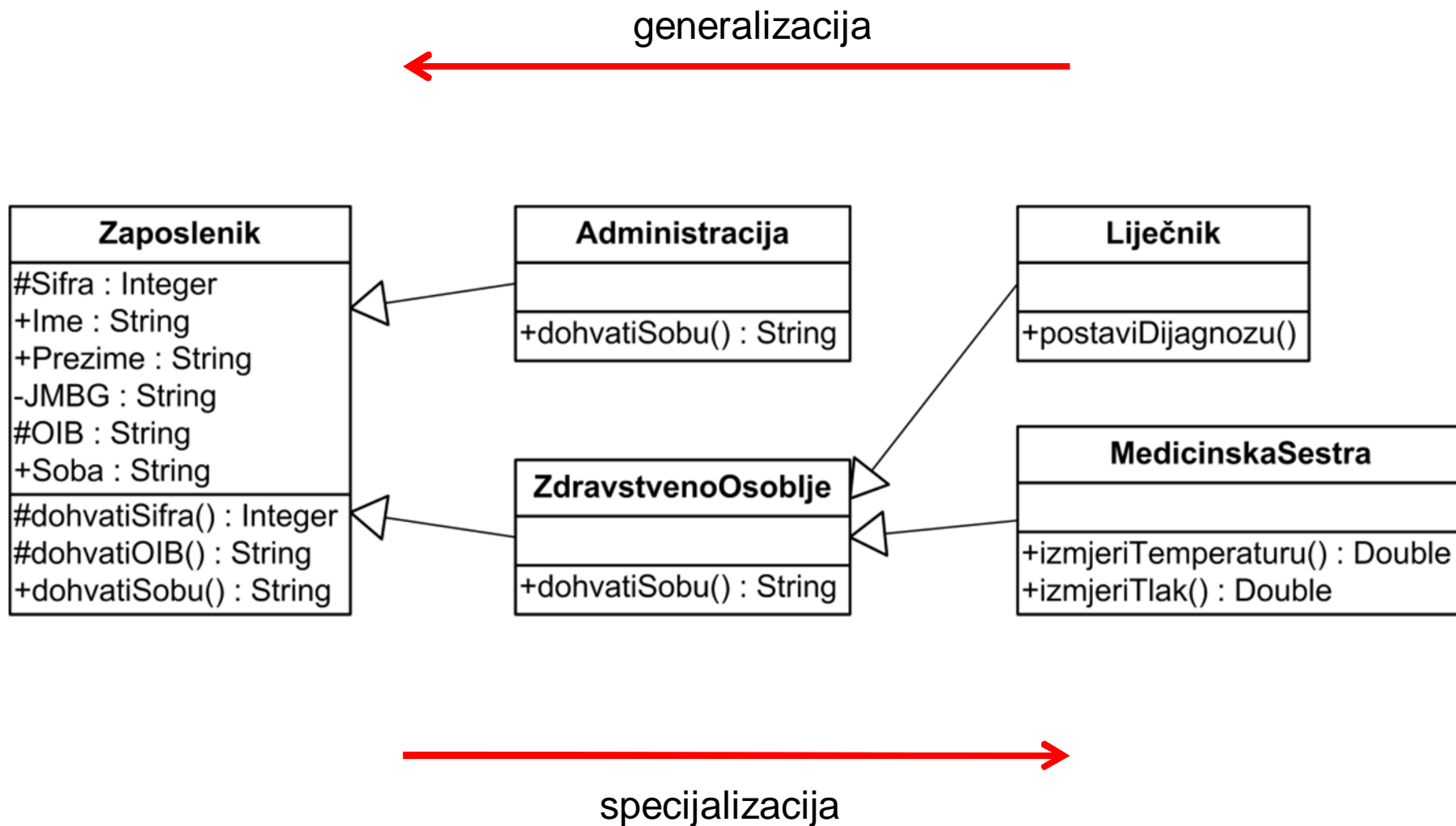


- “ **Nasljeđivanje** (engl. *inheritance*) je koncept UML-a u kojemu objekt koji se nasljeđuje je proziren u objektu koji ga nasljeđuje.
- “ Oblik UML odnosa - podtip.
 - . Nasljedna veza između razreda.
 - . Jedan razred je *roditelj* (nadklasa) drugome razredu (*dijete* ili podklasa).
 - . Odnos roditelj-dijete je veza IS-A.

Nasljeđivanje (2)

- “ **Generalizacija** . omogućuje stvaranje nadklase koja objedinjuje strukturu i ponašanje zajedničko za nekoliko klasa.
- “ **Specijalizacija** . omogućuje stvaranje podklase koja predstavlja dodavanje novih elemenata.
 - . Podklasa uvijek ima vize ili jednak broj svojstava u odnosu na nadklasu.
 - . Podklasa nasljeđuje od nadklase atribute, relacije i operacije.
 - . Podklasa može biti prozirena atributima, operacijama ili relacijama.
 - . Podklasa može imati svoju implementaciju operacija koje je naslijedila.

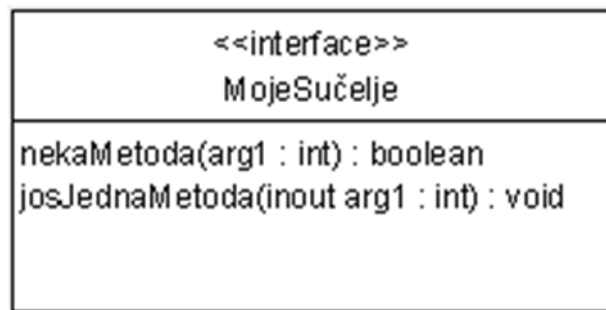
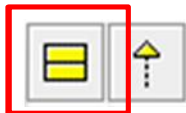
Primjer nasljeđivanja



Su elje

” **Sučelje** (engl. *interface*) je skup operacija koja specificira usluge nekog razreda.

- Su elje 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 MojeSu elje {

    public boolean nekaMetoda(int arg1);

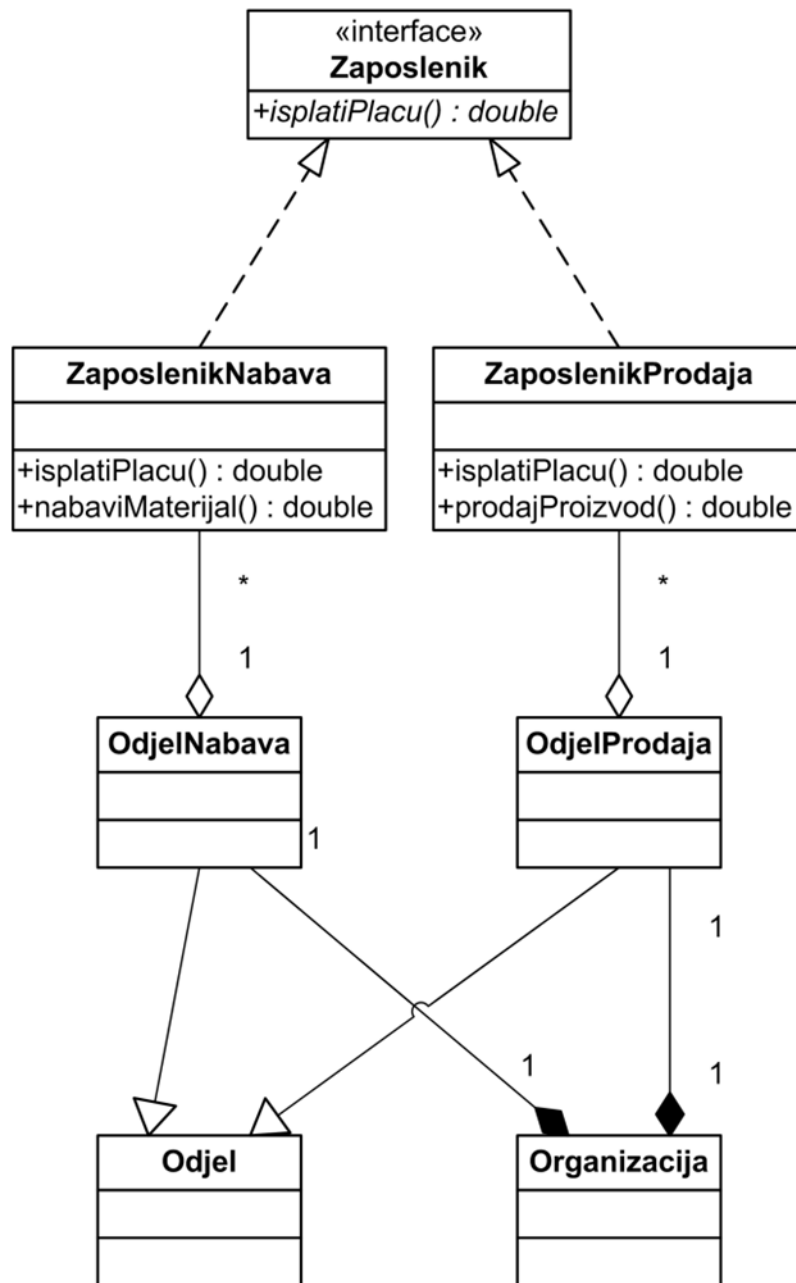
    public void josJednaMetoda(int arg1);
}
```

Realizacija sučelja



- “ **Realizacija** (engl. *realisation*) 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 specifični razred dobiva sve atribute i operacije općenitijeg ili nadređenog razreda.

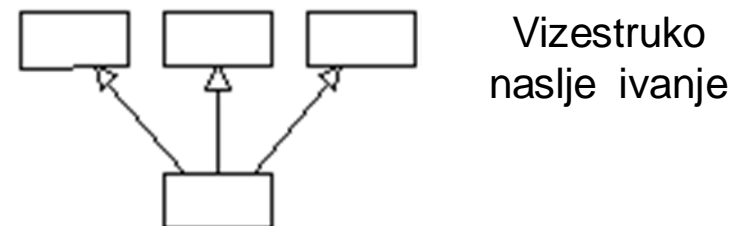
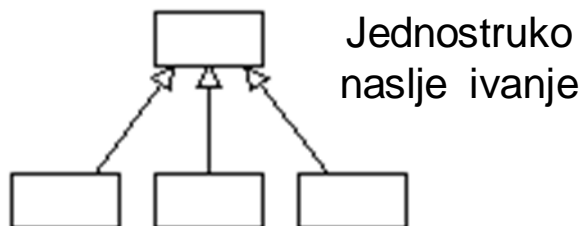
Primjer su elja i realizacije



U nekoj organizaciji postoje odjeli Nabave i Prodaje. Odjeli imaju vlastite zaposlenike, koji rade samo u tom odjelu. Svi zaposlenici dobivaju isplate mjese njih 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 zajedničku su elju.

Vizestrukost nasljeđivanja i realizacije

- “ Vizestruko nasljeđivanje (engl. *multiple inheritance*) je zabranjeno u nekim OO programskim jezicima (npr. Java i C#). U C++ je dozvoljeno.
- “ Vizestruka realizacija je uvijek dopuztena.
 - . Omogućuje općenito vizestruko nasljeđivanje (engl. *general multiple inheritance*) i u Javi tako da je moguće implementirati vize razreda bez mijenjanja njihove definicije, što je u konačnici slično u slučaju vizestrukog nasljeđivanja.



Identificiranje vrste odnosa

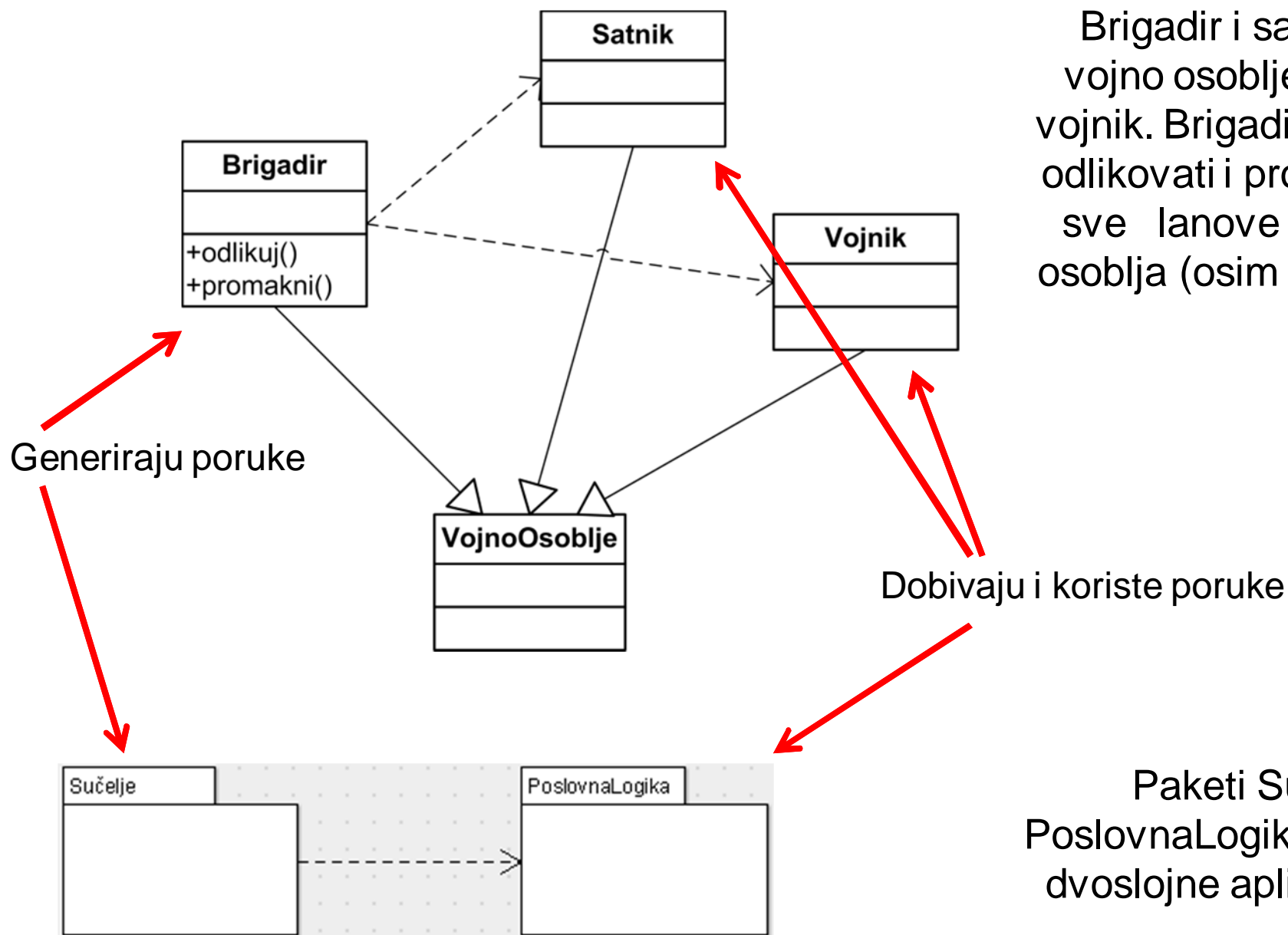
- “ Kako odrediti koja vrsta odnosa izme u dva razreda je ispravna: **pridruživanje**, **agregacija**, **kompozicija** ili **nasljeđivanje**?
- “ **Aggregacija**: ako jedan razred obuhva a ili sadr0ava drugog (povezani su odnosom cjelina-dio).
- “ **Kompozicija**: ako su razredi u odnosu cjelina-dio, ali nakon uniztavanja pojedinaca cjeline moraju se uniztiti i pojedinci dio.
- “ **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.
 - . Semanti ka relacija izme u dvije stvari u kojoj promjena u jednoj (neovisnoj stvari) mo0e utjecati na semantiku druge (ovisne stvari).
- ” Ovisnost je uvijek jednosmjerna: **A ovisi o B**+u smjeru strelice.
 - . **A** se naziva isporu itelj (engl. *supplier*) i **B** klijent (engl. *client*).
- ” ArgoUML ne preslikava svojstvo ovisnosti u programski kod.

Primjeri ovisnosti



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

Dobivaju i koriste poruke

Paketi Sučelje i PoslovnaLogika neke dvoslojne aplikacije.

Enumeracija

” **Enumeracija** (engl. *enumeration*) je oblik tipa podatka koji sadr0ava ure ene parove imenovanih identifikatora i njima pridru0enih vrijednosti.

. Te vrijednosti nazivaju se enumerirani literal.

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

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

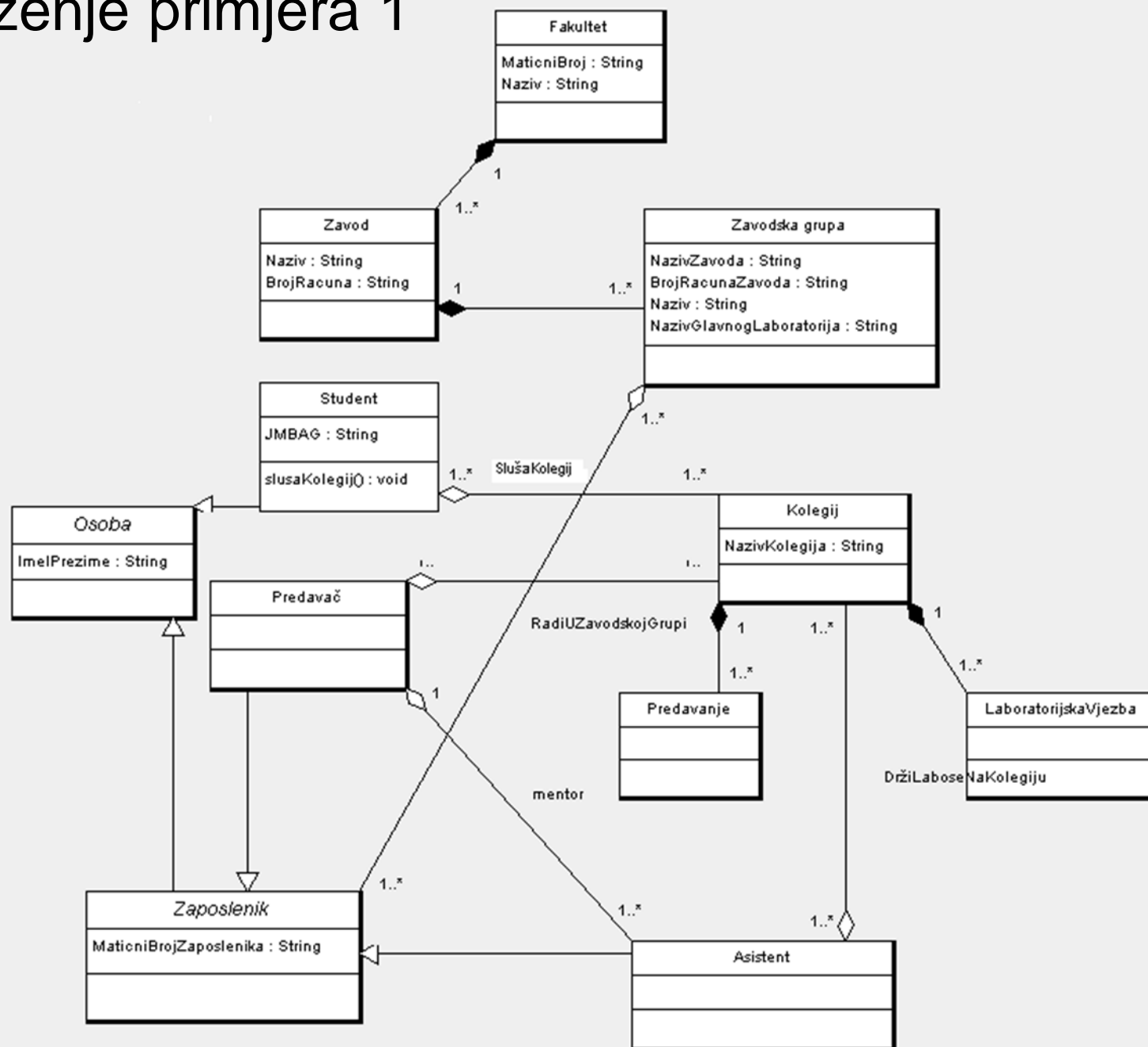
Komentari

- “ Unato formalnoj ekspresivnosti UML Class dijagrama 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 veze i nalaze na rubu crte koja obično u jednom od uglova dijagrama.

Primjer 1: Modeliranje organizacije fakulteta

Neki fakultet sastoji se od jednog ili više zavoda, a svaki zavod od jedne ili više zavodskih grupa. Zavodsku grupu čine zaposlenici. Zaposlenici mogu raditi i u nekoliko zavodskih grupa, 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 vodi 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 vode. Student je zasebna kategorija u organizaciji fakulteta i u ovom modelu pretpostavite samo da služi 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 razreda (String). Naziv i broj razreda zavoda nasljeđuju i zavodske grupe, s tim da one osim toga imaju i svoj naziv grupe te dodatno, naziv glavnog laboratorija (String).

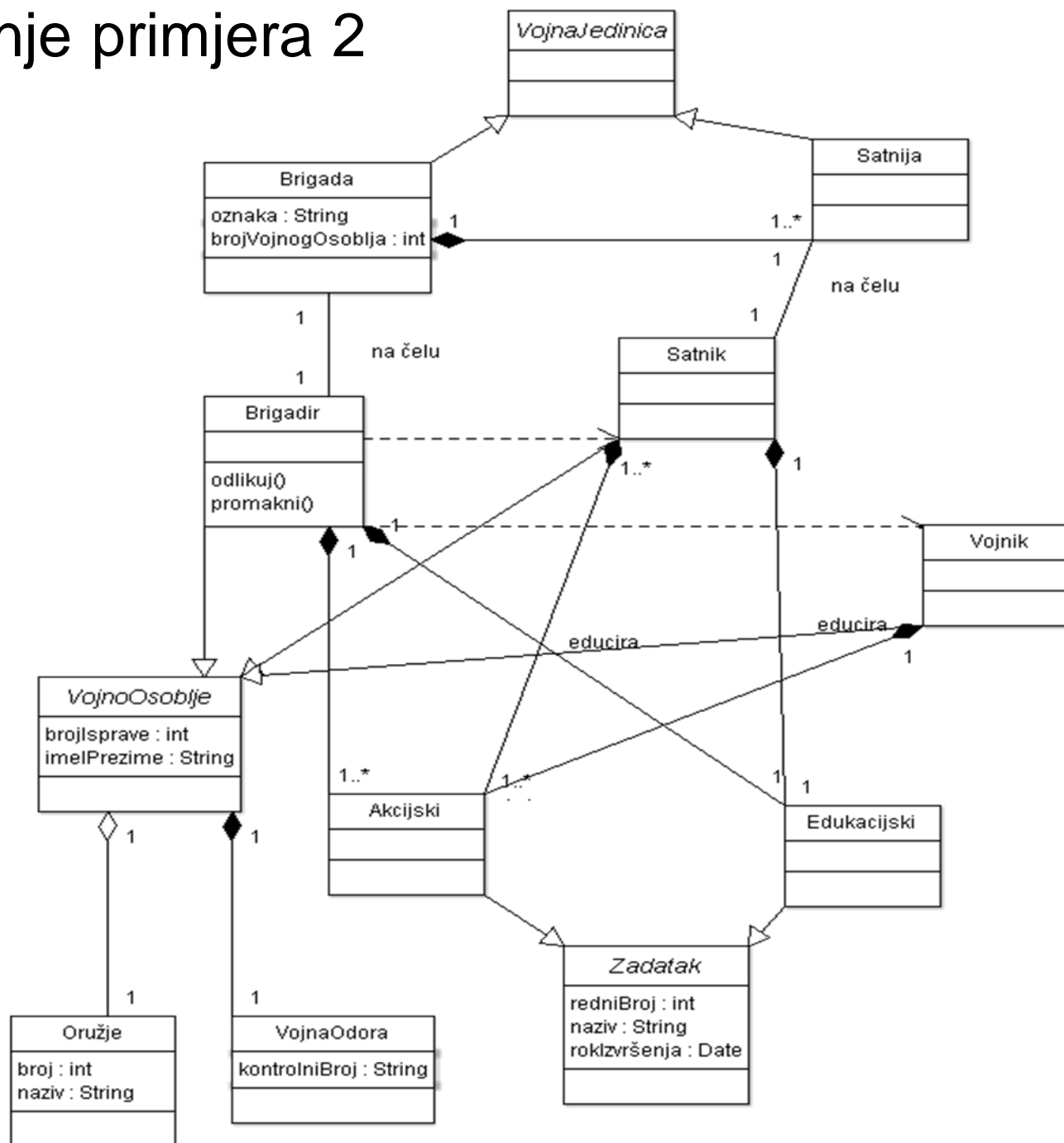
Rjezenje primjera 1



Primjer 2: Modeliranje organizacije vojne jedinice

Prepostavite da neka vojna jedinica može biti brigada ili satnija. Svaka brigada sadrži jednu ili više satnija. Na čelu brigade nalazi se brigadir, a na čelu satnije satnik. Brigadir i satnik su vojno osoblje, kao i vojnik. Brigadir 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 brigadir i satnik. Oni mogu imati najviše jedan edukacijski zadatak, a svaki edukacijski zadatak drži samo jedan brigadir ili satnik. Svaki član vojnog osoblja može obavljati jedan ili više akcijskih poslova, a jedan akcijski posao može obavljati više 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 jedinicu, vojna odora se uniztava. Oružje nosi svaki član vojnog osoblja, ali ono ostaje na raspolaganju čak i ako pojedinac napusti vojnu jedinicu. Svaki član vojnog osoblja ima svoje ime i prezime (String) i broj vojne isprave (int). Svaka brigada 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).

Rjezenje primjera 2



Nekoliko savjeta

- ” Class dijagrami 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 sa 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 dijelova sustava, nego zastarjeli dijagram cijelog sustava.

U enje

” Izvori:

- . Zbirka zadataka iz UML-a:

<http://www.fer.unizg.hr/predmet/opp>

- . Predavanja:

http://www.zemris.fer.hr/predmeti/opp/opp_predavanja.html

- . Allen Holub's UML Quick Reference:

<http://www.holub.com/goodies/uml>

- . ArgoUML manual:

<http://argouml.tigris.org>

- . Booch G., Jacobson I., Rumbaugh J. %UML Distilled+