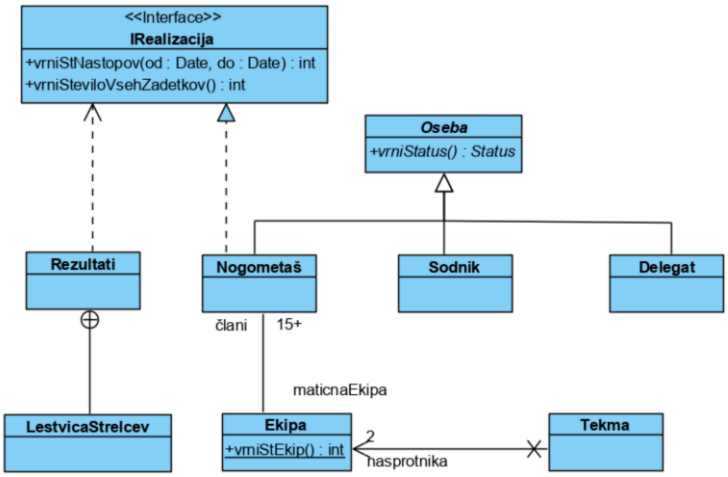
Kje smo zadnjič končali?



OO koncepti in UML

- Razred (“class”)
  - atributi
  - operacije/metode
- Objekt (“object”)
- Ograjevanje (“encapsulation”)
  public, protected, private, package
- Generalizacija/specializacija
  oz. dedovanje kot impl. koncept (“inheritance”)
- Vmesnik (“interface”) in realizacija ("realization")
- Delegiranje (“delegation”)

Razred

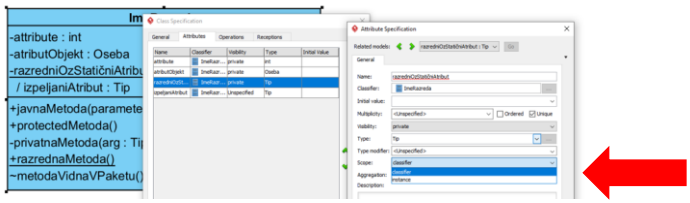
ImeRazreda
-attribute : int
-atributObjekt : Oseba
-razredniOzStatičniAtribut : Tip
/ izpeljaniAtribut : Tip
+javnaMetoda(parameter : TipParametra) : TipRezultata
+protectedMetoda()
-privatnaMetoda(arg : Tip) : int
+razrednaMetoda()
~metodaVidnaVPaketu() : void

AbstraktniRazred

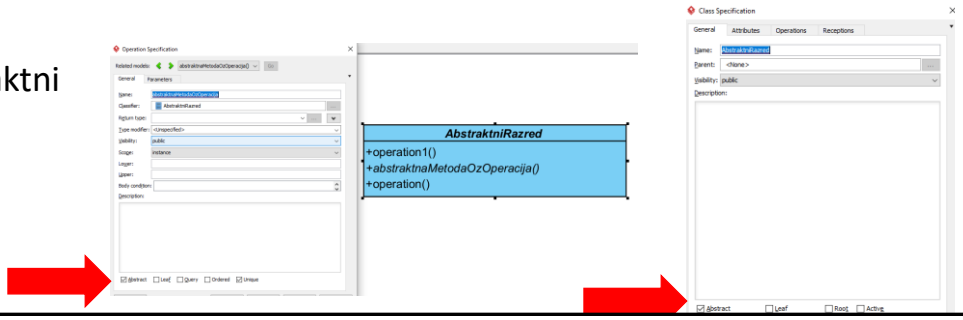
AbstraktniRazred
+operation1()
+abstraktnaMetodaOzOperacija()
+operation()

VP

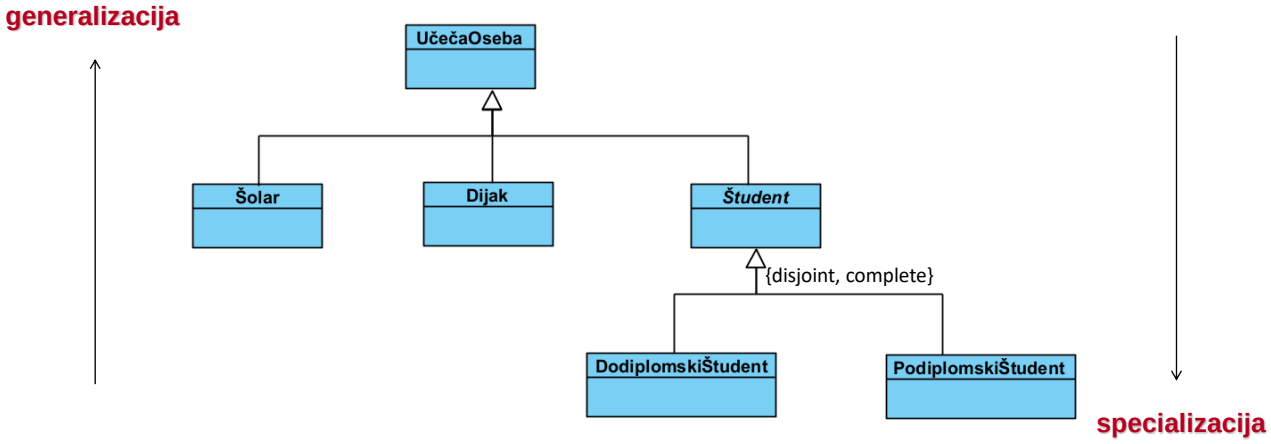
- Razredni (static)



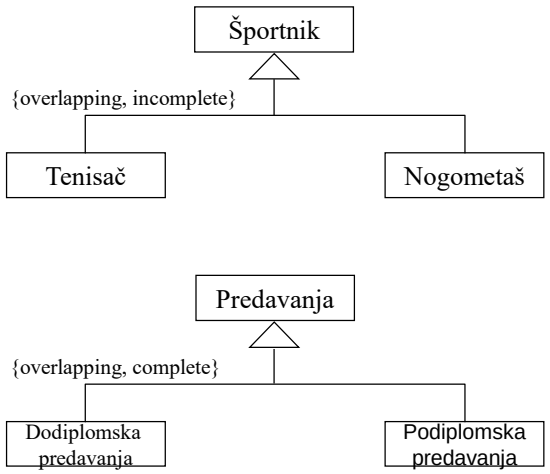
- Abstraktni



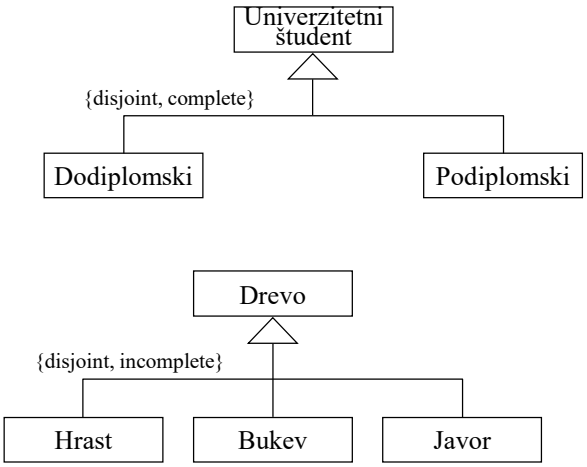
Generalizacija/specializacija



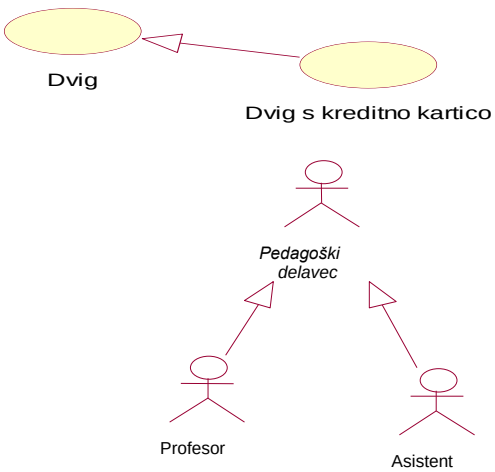
Generalizacija— omejitve



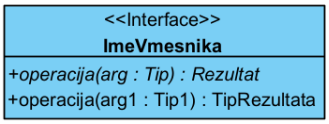
Generalizacija — omejitve

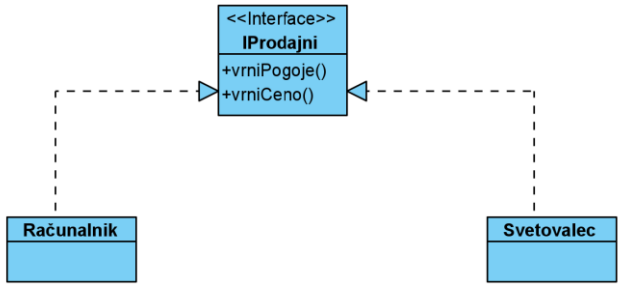


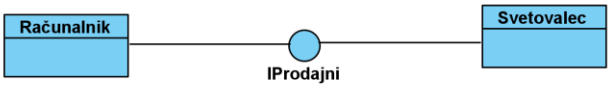
Gen/spec tudi med akterji, primeri uporabe, vmesniki …



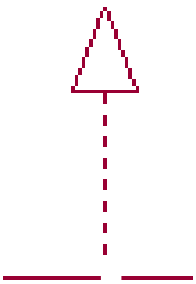
Vmesnik







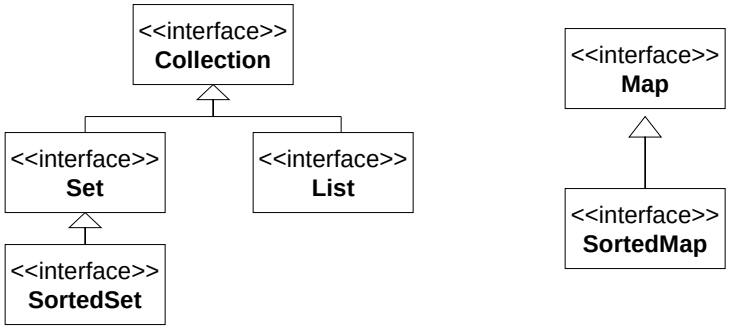
Povezava realizira



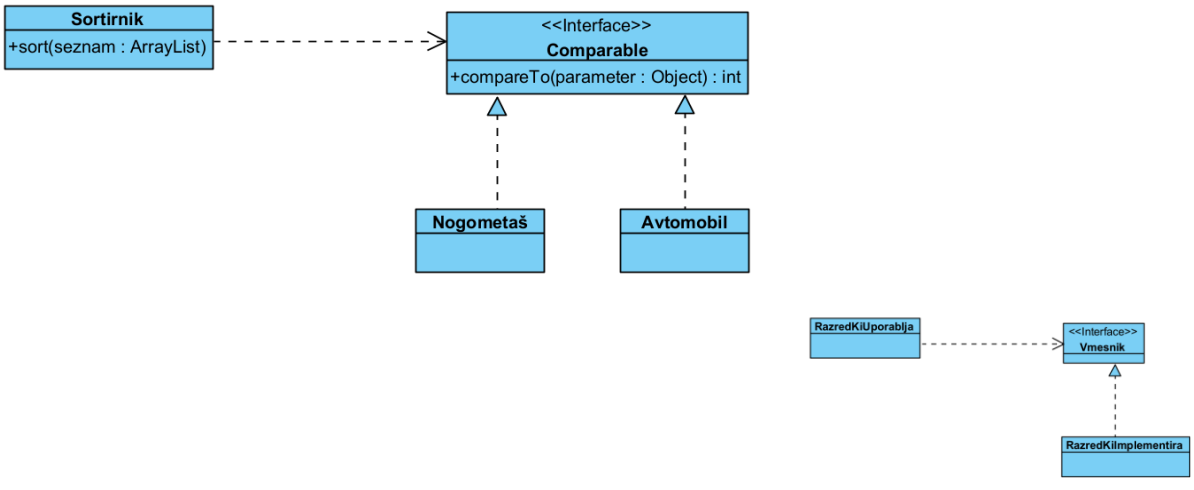
- uporabimo lahko za
  - Razred realizira vmesnik
  - Komponenta realizira vmesnik
  - Paket realizira vmesnik
  - Primer uporabe realizira primer uporabe
  - Paket realizira razred

  - PAZIMO NA GRAFIČNO PREDSTAVITEV

Vmesniki – generalizacija/specializacija

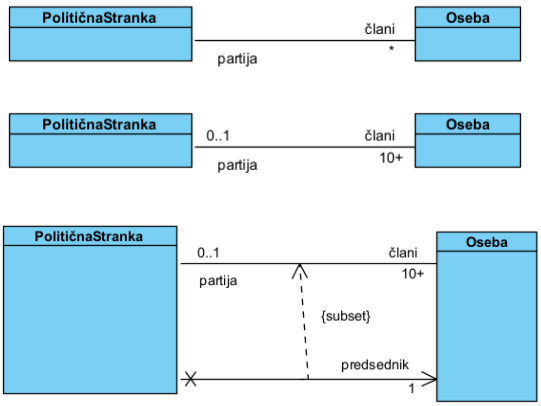


Odvisnost vs. realizacija



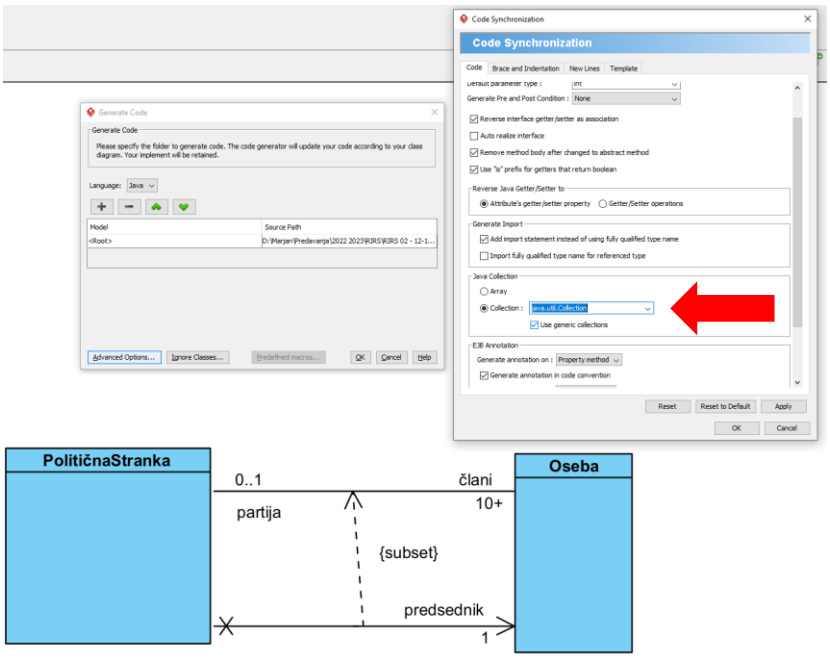
Asociacije med razredi

- ime, številnost, vloge, usmerjenost

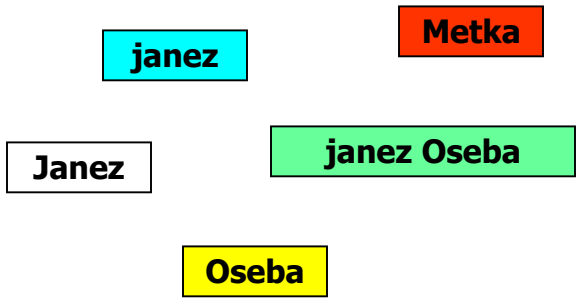


Vloge pomembnejše kot ime asociacije

```
public class Oseba {

        PolitičnaStranka partija;

}



public class PolitičnaStranka {

        Collection<Oseba> člani;
        Oseba predsednik;

}
```



Koliko objektov



je na sliki, če sem uporabil notacijo jezika UML?

Razred vs. objekt vs. vloge

razred

Točka
-x:Real -y:Real
+postavi(x:Real, y:Real) +premakni (dx:Real, dy:Real)

objekt

t1:Točka
x=12.5 y=5.28

vloga primerka razreda

izhodišče:Točka

objekt v vlogi

t1/izhodišče:Točka
x=12.5 y=5.28

Vloga: vmes - med tipi in primerki

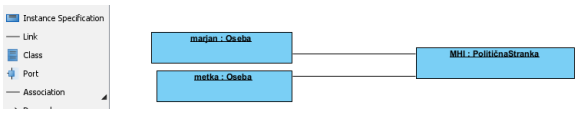
Klasifikacija (objektov v razrede)

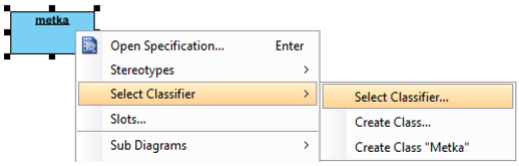
- Izberemo skupne lastnosti in zanemarimo unikatne lastnosti
- Skupino podobnih primerkov objektov klasificiramo v nek razred

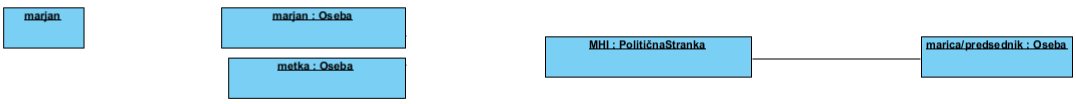
Zakaj pomembno?

Vemo, kaj lahko od vsakega objekta v tem razredu pričakujemo / zahtevamo!

Objektni diagram

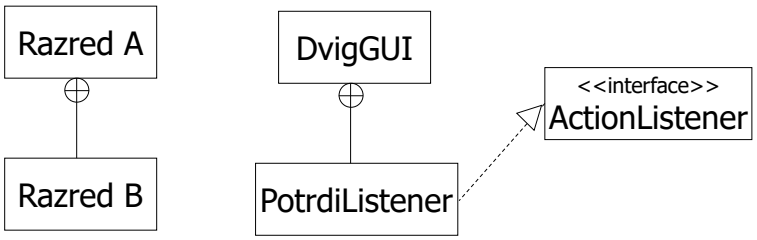






Uporaba gnezdenih razredov

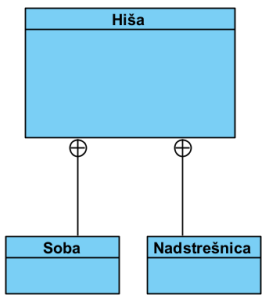
- razred je lahko deklariran znotraj drugega razreda (“nested”, “inner”)



Gnezdeni razredi

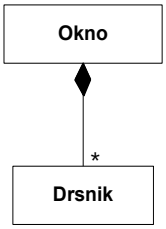
```
public class Hisa {

    static class Nadstrešnica {
        @Override
        public String toString() {
            return "sem nadstresnica";
        }
    }

    class Soba {

        String ime;

        public Soba(String katera) {
            ime = katera;
        }
```

```
System.out.println(new Hisa("Vila na Lentu").new Soba("Otroška"));

Hisa d = new Hisa("Družinska v Kamnici");
System.out.println(d.new Soba(" Spalnica"));
System.out.println(d.new Soba(" Dnevna soba"));

System.out.println(new Hisa.Nadstrešnica());
```

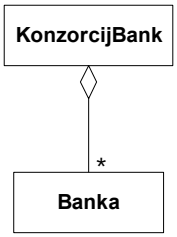


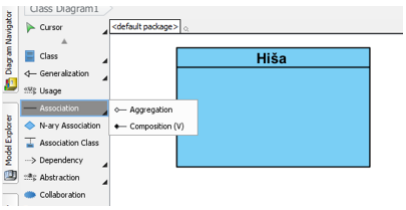
Agregacija in kompozicija

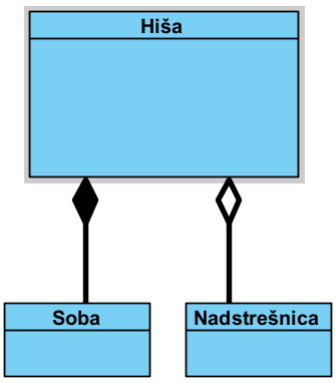
- kompozicija
- “by value”



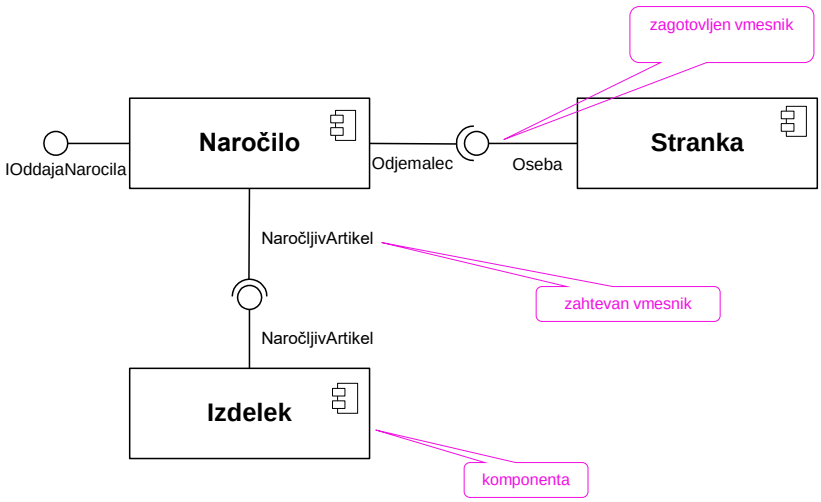
- agregacija
- “by reference”

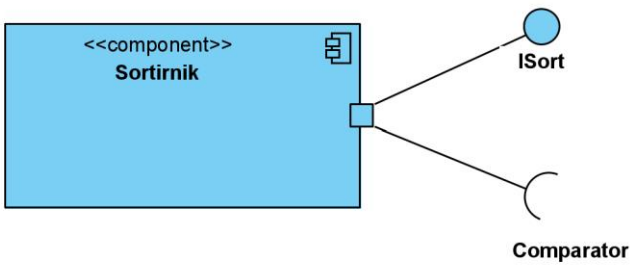






Zahtevan vmesnik (UML 2.0)





Omejitev na parametru

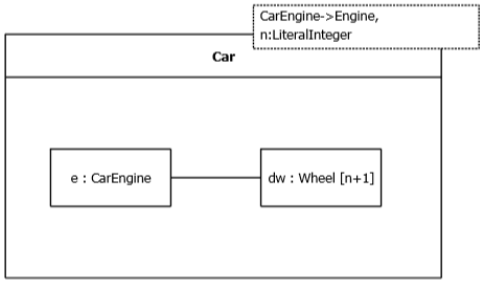
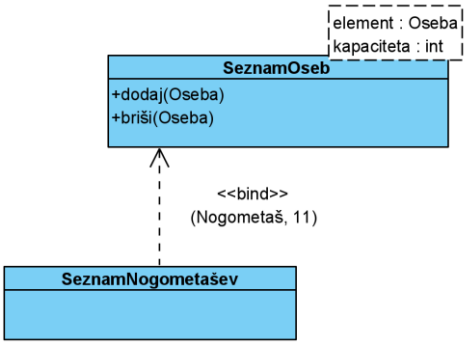
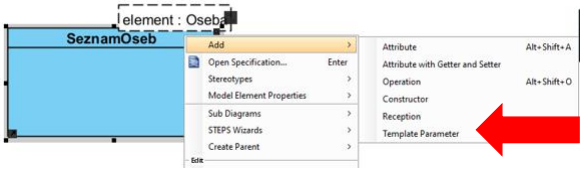


Figure 9.7 Template Class with constrained Class parameter

Parametrizirani razredi/vmesniki



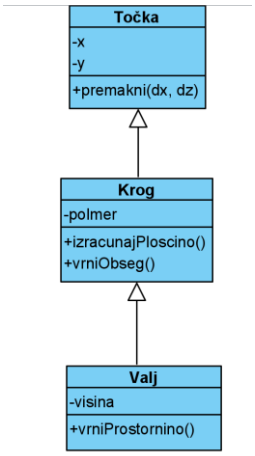


Delegiranje

- **Dedovanje med razredi, delegiranje med objekti**
- Objekt za izvedbo operacije zadolži (preloži odgovornost) drug objekt
- Dogodkovni modeli z delegiranjem
- Delegiranje je pogosto primernejši koncept kot dedovanje

Konceptualna pravilnost

- Preberimo!!!
  Krog je točka
  Valj je krog
  "is-a"
- Številnost (kardinalnost) **1:1**



Diagramske tehnike jezika UML

UML 1.X	UML 2.X
• diagrami primerov uporabe (use case diagrams), razredni diagrami (class diagrams), objektni diagrami (object diagrams), diagrami zaporedja (sequence diagrams)	
▪ diagrami sodelovanja (collaboration diagrams) ▪ diagrami stanj (statechart diagrams)	▪ Diagrami komunikacije (Communication diagrams) ▪ Pregledni diagram interakcije (Interaction overview Diagram) ▪ Diagram sestavljenih struktur (Composite Structure Diagram) ▪ Diagrami stanj (State machines) ▪ Časovni diagrami (timing diagram)
▪ diagrami aktivnosti (activity diagrams), komponentni diagrami (component diagrams), diagrami namestitve (deployment diagrams)	

29

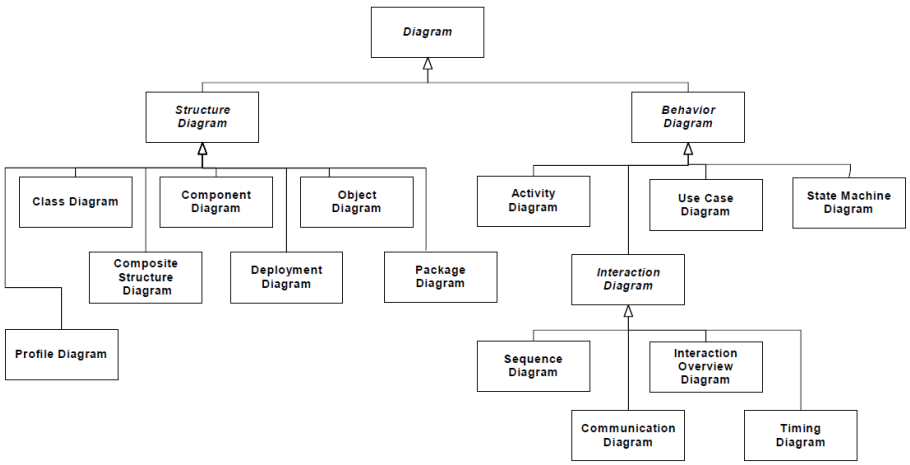
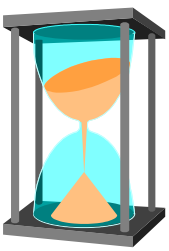


	Diagram	Purpose
Structure	Class	Properties and relationships of classes
	Component	Structure and connection of components
	Composite structure	Runtime decomposition of a class
	Deployment	Deployment of artifacts to nodes
	Object	Example configuration of instances
	Package	Compile-time hierarchic structure
Behavior	Activity	Procedural and parallel behavior
	Communication	Interaction between objects with emphases on links
	Interaction overview	Mix of sequence and activity diagrams
	Sequence	Interaction between objects with emphases on sequence
	State	Event changes of an object over its life
	Timing	Interaction between objects with emphases on time
	Use case	User interactions with a system

Table 1. UML 2.0 Diagram Types (Adapted from: Fowler (2004), p.11)



32

UML (ni metoda)

OMG - formalno potrjene specifikacije 2.5.1 (december 2017)

ISO release (ISO/IEC 19501)

The **Unified Modeling Language** (**UML**) is a general-purpose modeling language in the field of software engineering, which is designed to provide a standard way to visualize the design of a system.

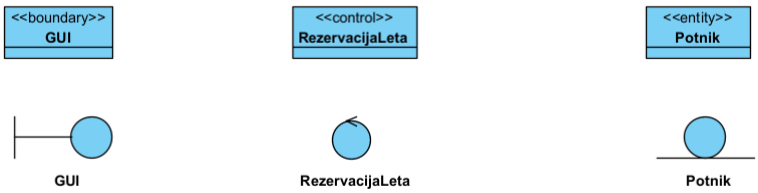
www.omg.org/spec/UML

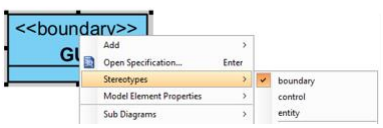
33

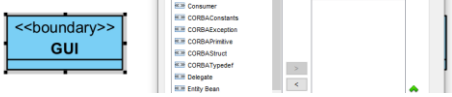
Zakaj temeljiti na scenarijih?

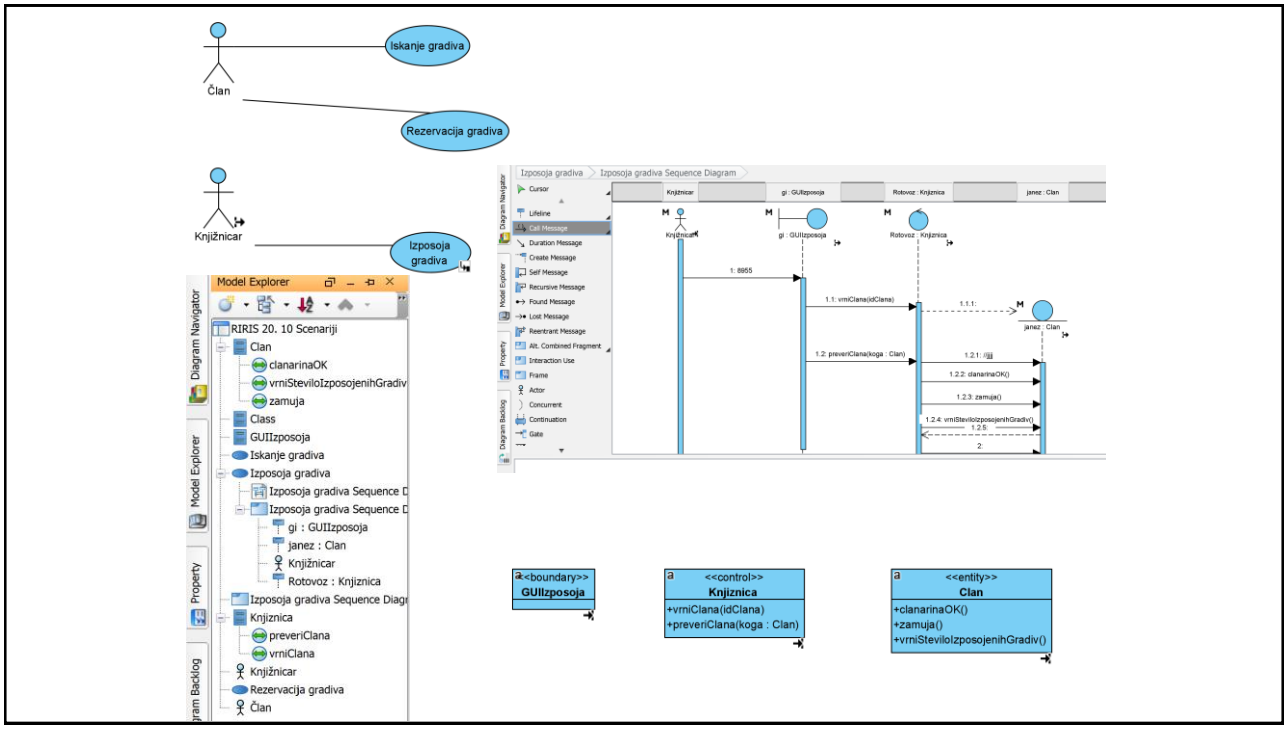
Stereotipi

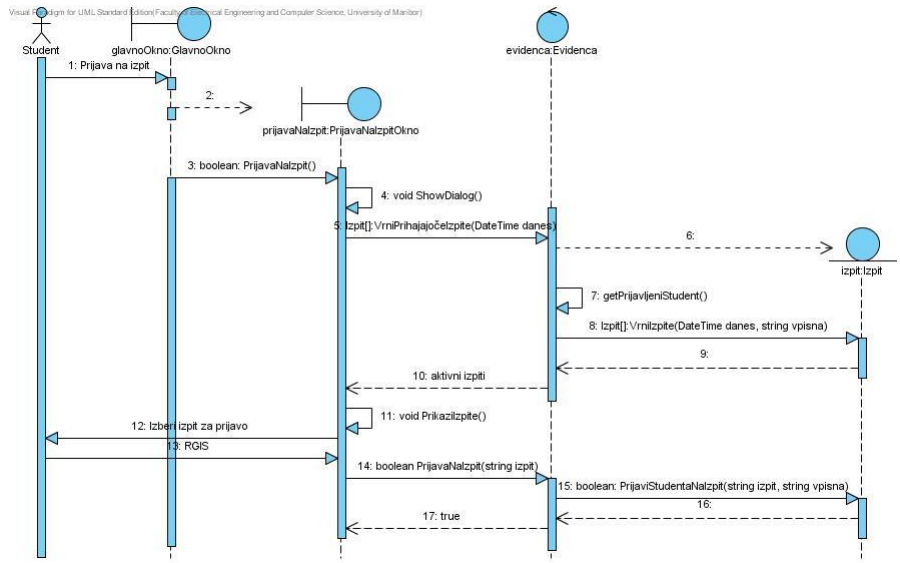
- Robustness analysis (Jacobson analysis model)
  - predstavlja preliminarni načrt
  - lahko vodi k identifikaciji dodatnih razredov, ki so potrebni
  - opredeli sodelovanje glede tega, kaj mora kdo znati/vedeti
  - zagotavlja celovitost in t.i. „sanity check“ preden izdelamo celoten načrt

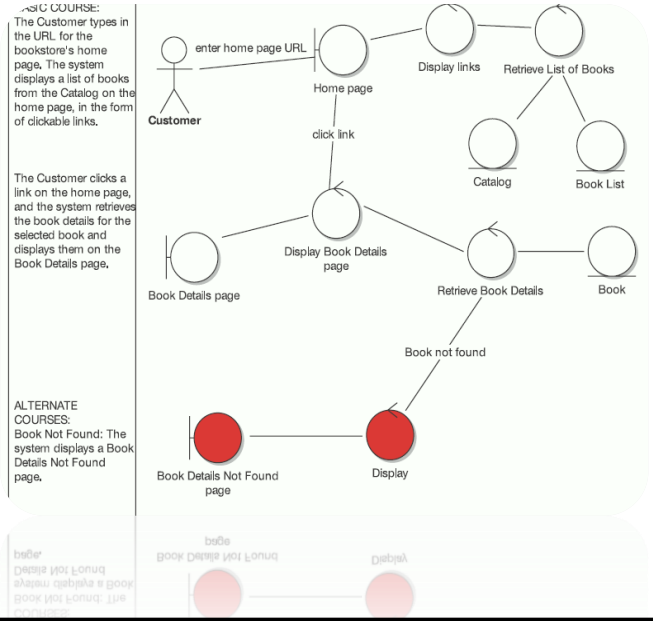







Član
Iskanje gradiva
Rezervacija gradiva
Knjižnicar
Izposoja gradiva
Model Explorer
RIRIS 20. 10 Scenariji
Clan
clanarinaOK
vrniSteviloIzposojenihGradiv
zamuja
Class
GUIIzposoja
Iskanje gradiva
Izposoja gradiva
Izposoja gradiva Sequence D
Izposoja gradiva Sequence D
gi : GUIIzposoja
janez : Clan
Knjižnicar
Rotovoz : Knjiznica
Izposoja gradiva Sequence Diagr
Knjiznica
preveriClana
vrniClana
Knjižnicar
Rezervacija gradiva
Član
Izposoja gradiva
Izposoja gradiva Sequence Diagram
1: 8955
1.1: vrniClana(idClana)
1.2: preveriClana(koga : Clan)
1.2.1: //jjj
1.2.2: clanarinaOK()
1.2.3: zamuja()
1.2.4: vrniSteviloIzposojenihGradiv()
2:
<<boundary>>
GUIIzposoja
<<control>>
Knjiznica
+vrniClana(idClana)
+preveriClana(koga : Clan)
<<entity>>
Clan
+clanarinaOK()
+zamuja()
+vrniSteviloIzposojenihGradiv()

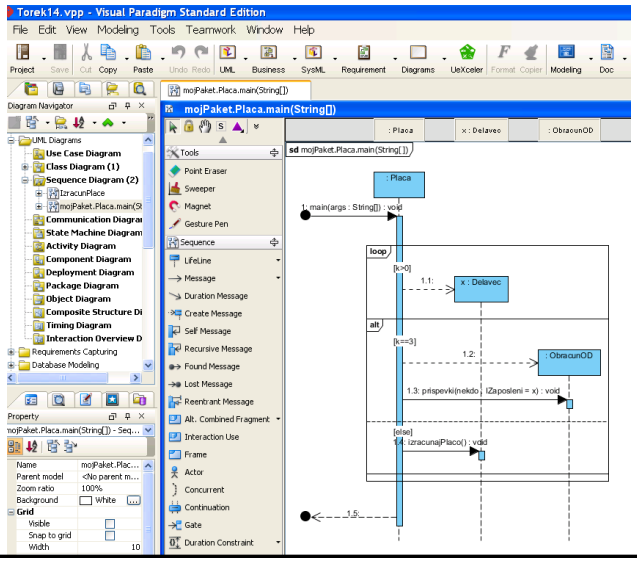

Student
glavnoOkno:GlavnoOkno
evidenca:Evidenca
1: Prijava na izpit
2:
prijavaNalzpit:PrijavaNalzpitOkno
3: boolean: PrijavaNalzpit()
4: void ShowDialog()
5: Izpit[]:VrniPrihajajočeIzpite(DateTime danes)
6:
izpit:Izpit
7: getPrijavljeniStudent()
8: Izpit[]:VrniIzpite(DateTime danes, string vpisna)
9:
10: aktivni izpiti
11: void PrikaziIzpite()
12: Izberi izpit za prijavo
13: RGIS
14: boolean PrijavaNalzpit(string izpit)
15: boolean: PrijaviStudentaNalzpit(string izpit, string vpisna)
16:
17: true



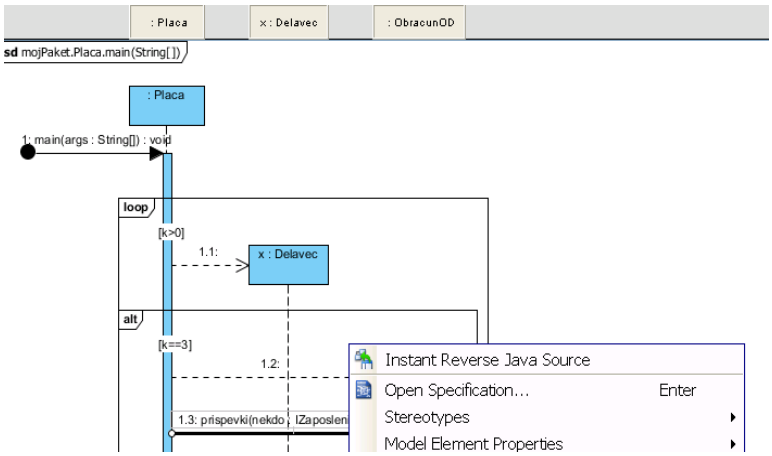
Forward vs. reverse engineering

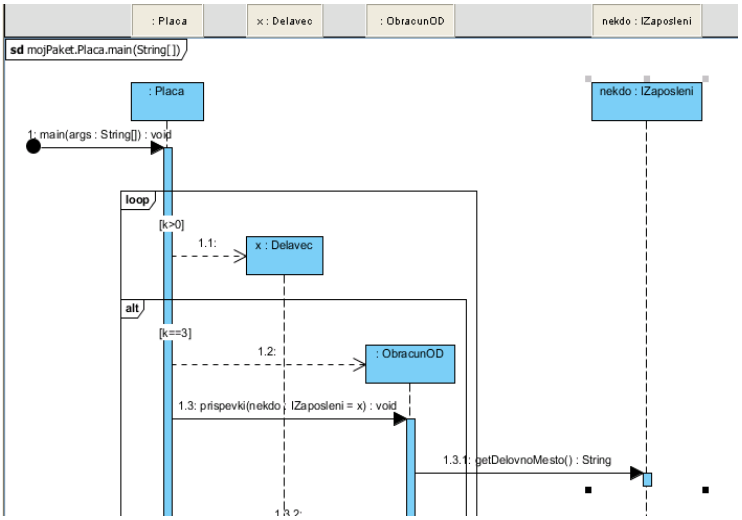
- Povratno inženirstvo (Visual Paradigm)
  - statični vidik
  - dinamični vidik

Diagrami zaporedja



Večja globina





Model vs. diagram

- Ali diagram=model?
- Model je množica, skupek **medsebojno povezanih diagramov in drugih izdelkov** (tudi besedila!)